



EBook Gratis

APRENDIZAJE iOS

Free unaffiliated eBook created from
Stack Overflow contributors.

#iOS

Tabla de contenido

Acerca de.....	1
Capítulo 1: Empezando con iOS.....	2
Observaciones.....	2
Notas.....	2
Etiquetas relacionadas de desbordamiento de pila.....	2
Versiones.....	2
Examples.....	3
Creación de una aplicación de vista única predeterminada.....	3
Hola Mundo.....	11
Comenzando un nuevo proyecto.....	11
Añadiendo una etiqueta.....	15
Agregando Código.....	17
Ejecutando la aplicación en el simulador.....	17
Continuando.....	18
Interfaz Xcode.....	18
Área del navegador.....	21
Los editores.....	21
Recursos y elementos en el área de servicios públicos.....	24
Administrar tareas con la barra de herramientas del área de trabajo.....	27
Crea tu primer programa en Swift 3.....	28
Crea tu primer programa.....	28
Capítulo 2: Abrazar contenido / Compresión de contenido en Autolayout.....	34
Observaciones.....	34
Examples.....	34
Definición: Tamaño del contenido intrínseco.....	34
Capítulo 3: Accesibilidad.....	36
Introducción.....	36
Examples.....	36
Hacer una vista accesible.....	36

Marco de accesibilidad.....	36
Cambio de pantalla.....	36
Cambio de diseño.....	37
Anuncio.....	37
Elementos de pedido.....	37
Contenedor de Accesibilidad.....	38
Vista modal.....	38
Elementos ocultos.....	38
Capítulo 4: Actualizando dinámicamente un UINavigationController.....	40
Examples.....	40
Conecte el UISwitch a una acción que podamos animar cambiando entre un diseño horizontal o.....	40
Capítulo 5: Alamofire.....	42
Sintaxis.....	42
Parámetros.....	42
Examples.....	42
Hacer una solicitud.....	42
Validación automática.....	42
Manejo de respuesta.....	42
Validación manual.....	43
Manejador de respuestas.....	43
Manejadores de respuesta encadenados.....	43
Capítulo 6: AÑADIR A UN LÍDER DE SWIFT BRIDGING.....	44
Examples.....	44
Cómo crear un encabezado de puente Swift manualmente.....	44
Xcode crea automáticamente.....	44
Capítulo 7: Aparición de la UIA.....	46
Examples.....	46
Establecer apariencia de todas las instancias de la clase.....	46
Apariencia para la clase cuando está contenida en la clase de contenedor.....	47
Capítulo 8: API de Google Places para iOS.....	49
Examples.....	49
Cómo llegar a lugares cercanos desde la ubicación actual.....	49

Capítulo 9: API de reconocimiento de voz de iOS 10	51
Examples	51
Discurso a texto: reconoce el habla de un paquete que contiene grabación de audio	51
Capítulo 10: Aplicación de Seguridad de Transporte (ATS)	53
Parámetros	53
Observaciones	54
Examples	54
Cargar todo el contenido HTTP	54
Cargar selectivamente contenido HTTP	55
Los puntos finales requieren SSL	55
Capítulo 11: Aplicación en la compra	57
Examples	57
IAP individual en Swift 2	57
Configuración en iTunesConnect	59
Pasos más básicos para comprar / suscribir un usuario a un IAP	61
Capítulo 12: AppDelegate	63
Introducción	63
Examples	63
Todos los estados de aplicación a través de métodos AppDelegate	63
Roles de AppDelegate:	64
Abrir un recurso especificado por URL	64
Manejo de notificaciones locales y remotas	65
Capítulo 13: ARC (Conteo Automático de Referencia)	67
Examples	67
Habilitar / deshabilitar ARC en un archivo	67
Capítulo 14: Archivo de texto básico de E / S	69
Examples	69
Lee y escribe desde la carpeta Documentos	69
Capítulo 15: Arquitectura MVP	71
Introducción	71
Observaciones	71

Examples.....	72
Cambio de perro.....	72
DoggyView.swift.....	72
DoggyService.swift.....	73
DoggyPresenter.swift.....	73
DoggyListViewController.swift.....	74
Capítulo 16: atribuidoTexto en UILabel.....	76
Introducción.....	76
Examples.....	76
Texto HTML en UILabel.....	76
Establecer diferentes propiedades para texto en solo UILabel.....	76
Capítulo 17: AVPlayer y AVPlayerViewController.....	78
Observaciones.....	78
Examples.....	78
Reproducción de medios utilizando AVPlayerViewController.....	78
C objetivo.....	78
Rápido.....	78
Reproducción de medios utilizando AVPlayer y AVPlayerLayer.....	78
C objetivo.....	78
Rápido.....	79
Ejemplo de AVPlayer.....	79
Capítulo 18: AWS SDK.....	80
Examples.....	80
Suba una imagen o un video a S3 usando AWS SDK.....	80
Capítulo 19: Barra de navegación.....	83
Examples.....	83
Personaliza la apariencia predeterminada de la barra de navegación.....	83
Ejemplo de SWIFT.....	83
Capítulo 20: Bloquear.....	84
Sintaxis.....	84
Examples.....	84
Animaciones UIView.....	84

Bloque de finalización personalizado para métodos personalizados.....	84
Modificar la variable capturada.....	85
Capítulo 21: Bloques de cadena en una cola (con MKBlockQueue).....	86
Introducción.....	86
Examples.....	86
Código de ejemplo.....	86
Capítulo 22: CAAanimación.....	88
Observaciones.....	88
Examples.....	88
Animar una vista de una posición a otra.....	88
C objetivo.....	88
Rápido.....	88
Vista animada - Lanzamiento.....	88
C OBJETIVO.....	88
RÁPIDO.....	89
Vista de revolución.....	89
Agitar vista.....	89
Animación Push View.....	90
C objetivo.....	90
Rápido.....	90
Capítulo 23: CAGradientLayer.....	91
Sintaxis.....	91
Parámetros.....	91
Observaciones.....	91
Examples.....	91
Creando un CAGradientLayer.....	91
Creando un CGGradientLayer con múltiples colores.....	92
Creando un CAGradientLayer horizontal.....	93
Creando un CAGradientLayer horizontal con múltiples colores.....	94
Animando un cambio de color en CAGradientLayer.....	95
Capítulo 24: CALayer.....	97

Examples.....	97
Creando un CALayer.....	97
Creando partículas con CAEmitterLayer.....	97
Vista del emisor con imagen personalizada.....	98
Cómo agregar un UIImage a un CALayer.....	99
Modificando la apariencia.....	99
Relacionado.....	103
Notas.....	103
Agregar transformaciones a un CALayer (traducir, rotar, escalar).....	103
Lo esencial.....	103
Preparar.....	104
Traducir.....	105
Escala.....	106
Girar.....	106
Transformaciones múltiples.....	107
Una nota sobre el punto de anclaje y la posición.....	108
Ver también.....	109
Deshabilitar animaciones.....	109
Esquinas redondeadas.....	109
Oscuridad.....	109
Capítulo 25: Calificación de solicitud / solicitud de revisión.....	111
Introducción.....	111
Examples.....	111
Calificar / Revisar la aplicación iOS.....	111
Capítulo 26: Cambiar color de barra de estado.....	112
Examples.....	112
Para barras de estado que no sean de UINavigationController.....	112
Para las barras de estado de UINavigationController.....	112
Si no puede cambiar el código de ViewController.....	113
Para la contención de ViewController.....	113
Cambiando el estilo de la barra de estado para toda la aplicación.....	114

RÁPIDO:	114
Paso 1:	114
Paso 2:	114
C OBJETIVO:	115
Capítulo 27: Cambiar el tamaño de UIImage	116
Parámetros	116
Examples	116
Redimensionar cualquier imagen por tamaño y calidad.	116
Capítulo 28: Cargar imagenes async	117
Examples	117
La manera más fácil.	117
Comprueba que la celda sigue siendo visible después de la descarga.	117
Capítulo 29: Carril rápido	119
Examples	119
herramientas de plano rápido.	119
Instalar fastlane	119
herramientas de iOS	119
Herramientas de TestFlight para iOS	119
Herramientas de android.	120
Capítulo 30: CAShapeLayer	121
Sintaxis	121
Observaciones	121
Examples	121
Operación básica de CAShapeLayer	121
Dibujar rectángulo.	125
Dibujar círculo.	126
Animación de CAShapeLayer	126
Capítulo 31: Categorías	128
Observaciones	128
Examples	128
Crear una categoría.	128

Capítulo 32: Clases de tamaño y adaptabilidad	132
Observaciones	132
Examples	132
Colecciones de rasgos	132
Actualización del diseño automático con cambios de colección de rasgos	133
Compatibilidad con la multitarea de iOS en iPad	134
Capítulo 33: Clases de tamaño y adaptabilidad	135
Observaciones	135
Examples	135
Clases de tamaño y adaptabilidad a través de storyboard	135
Capítulo 34: CLLocation	140
Examples	140
Filtro de distancia utilizando	140
Obtener la ubicación del usuario usando CLLocationManager	140
Capítulo 35: CloudKit	143
Observaciones	143
Tipos soportados	143
Examples	143
Registro de la aplicación para su uso con CloudKit	143
Utilizando CloudKit Dashboard	144
Tipos de registro	144
Guardando datos en CloudKit	144
Haciendo una clave de registro	144
Rápido	145
Haciendo el disco	145
Rápido	145
C objetivo	145
Nota	145
Accediendo al contenedor	145
Rápido	145
Guardando los registros en la base de datos de CloudKit	145

Rápido.....	146
Capítulo 36: Codificable.....	147
Introducción.....	147
Examples.....	147
Uso de Codable con JSONEncoder y JSONDecoder en Swift 4.....	147
Capítulo 37: Codificación del valor clave-Observación del valor clave.....	149
Observaciones.....	149
Examples.....	149
Uso del contexto para la observación KVO.....	149
Observando una propiedad de una subclase NSObject.....	150
Capítulo 38: Comprobando la conectividad de la red.....	151
Observaciones.....	151
Advertencias.....	151
Examples.....	151
Creando un oyente de Alcance.....	151
Agregar observador a los cambios de red.....	151
Alerta cuando la red deja de estar disponible.....	152
Alerta cuando la conexión se convierte en una red WIFI o celular.....	152
Verificar si está conectado a la red.....	152
Capítulo 39: Comprobando la versión de iOS.....	154
Examples.....	154
iOS 8 y versiones posteriores.....	154
Comparar versiones.....	154
C objetivo.....	154
Swift 2.0 y versiones posteriores.....	154
Versión del dispositivo iOS.....	155
C objetivo.....	155
Rápido.....	155
Swift 3.....	155
Capítulo 40: Concurrencia.....	156
Introducción.....	156

Sintaxis.....	156
Parámetros.....	156
Observaciones.....	156
Examples.....	157
Ejecutar código simultáneamente: ejecutar código mientras se ejecuta otro código.....	157
Ejecutando en el hilo principal.....	157
Grupo de despacho - esperando otros hilos completados.....	158
Capítulo 41: Configuración de iOS de Cartago.....	159
Examples.....	159
Instalación de Cartago Mac.....	159
Capítulo 42: Configurar balizas con CoreBluetooth.....	160
Introducción.....	160
Observaciones.....	160
Algunos puntos importantes.....	160
Escanear para UUID DE SERVICIO.....	160
Cómo descubrir el UUID de SERVICIO sin documentación.....	160
Convertir datos a UInt16 y contrario.....	161
Examples.....	161
Mostrando nombres de todos los Bluetooth de baja energía (BLE).....	161
Conectar y leer valor mayor.....	163
Escribe valor mayor.....	164
Capítulo 43: Control UISegmentado.....	166
Introducción.....	166
Examples.....	166
Creación de UISegmentedControl mediante código.....	166
Capítulo 44: Convertir HTML a cadena NSAttributedString y viceversa.....	167
Examples.....	167
Código de Objective C para convertir la cadena HTML a NSAttributedString y Vice Versa.....	167
Capítulo 45: Convertir NSAttributedString a UIImage.....	168
Examples.....	168
Conversión de NSAttributedString a UIImage.....	168

Capítulo 46: Core Graphics	169
Examples.....	169
Creando un Contexto de Core Graphics.....	169
Contexto de Core Graphics	169
Haciendo un contexto	169
Rápido.....	169
C objetivo.....	169
Presentando el Lienzo Dibujado al Usuario.....	170
Rápido.....	170
C objetivo.....	170
Capítulo 47: Core Motion	171
Examples.....	171
Accediendo al barómetro para obtener la altitud relativa.....	171
Capítulo 48: Core SpotLight en iOS	172
Examples.....	172
Core-Spotlight.....	172
Capítulo 49: Cortar un UIImage en un círculo	182
Examples.....	182
Cortar una imagen en un círculo - Objetivo C.....	182
Ejemplo de SWIFT 3.....	183
Capítulo 50: Creación de PDF en iOS	185
Examples.....	185
Crea PDF.....	185
Mostrar PDF.....	186
PDF de varias páginas.....	187
Crear PDF desde cualquier documento de Microsoft cargado en UIWebView.....	187
Capítulo 51: Creación de una ID de aplicación	189
Examples.....	189
Creación de productos de compra en la aplicación.....	189
Creando un usuario de Sandbox.....	191
Capítulo 52: Crear un marco personalizado en iOS	192

Examples.....	192
Crear Framework en Swift.....	192
Capítulo 53: Crear un video a partir de imágenes.....	193
Introducción.....	193
Examples.....	193
Crear video desde UIImage.....	193
Capítulo 54: Cree un archivo .ipa para cargar en la tienda de aplicaciones con Application.....	196
Examples.....	196
cree el archivo .ipa para cargar la aplicación en appstore con Application Loader.....	196
Capítulo 55: CTCallCenter.....	202
Examples.....	202
Interceptando llamadas desde tu aplicación incluso desde el fondo.....	202
Kit de llamadas - ios 10.....	203
Capítulo 56: CydiaSubstrate tweak.....	205
Introducción.....	205
Observaciones.....	205
Instalando theos.....	205
Examples.....	205
Crear nuevo tweak usando Theos.....	205
Usa nic para crear un nuevo proyecto.....	205
Anular método de guardar capturas de pantalla de iOS.....	206
Capítulo 57: Datos básicos.....	207
Introducción.....	207
Examples.....	207
Operaciones sobre datos básicos.....	207
Capítulo 58: Delegados de multidifusión.....	208
Introducción.....	208
Examples.....	208
Delegados multicast para cualquier control.....	208
Capítulo 59: Depuración se bloquea.....	213
Examples.....	213

Encontrar información sobre un accidente.....	213
La flecha roja.....	214
La consola de depuración.....	214
El rastro de pila.....	214
La depuración de SIGABRT y EXC_BAD_INSTRUCTION se bloquea.....	215
Depurando EXC_BAD_ACCESS.....	215
Capítulo 60: Detección de rostro utilizando CoreImage / OpenCV.....	219
Examples.....	219
Detección de rostro y rasgos.....	219
Capítulo 61: Diseño automático.....	222
Introducción.....	222
Sintaxis.....	222
Examples.....	222
Establecer restricciones programáticamente.....	222
Fijación.....	222
Anchura y altura.....	223
Centro en contenedor.....	223
Cómo usar Auto Layout.....	224
Restricciones del centro.....	224
Vistas espaciales uniformemente.....	227
UILabel tamaño intrínseco.....	229
Cómo animar con diseño automático.....	240
Resolver el conflicto de prioridad de UILabel.....	242
Tamaño de UILabel y Parentview según el texto en UILabel.....	245
Conceptos básicos del lenguaje de formato visual: restricciones en el código!.....	252
Uso mixto de diseño automático con diseño no automático.....	254
Diseño proporcional.....	254
NSLayoutConstraint: Constraints en código!.....	256
Capítulo 62: Enlace profundo en iOS.....	259
Observaciones.....	259
Examples.....	259

Abrir una aplicación basada en su esquema de URL.....	259
Agregando un esquema de URL a tu propia aplicación.....	259
Paso uno: Registrar un esquema de URL en Info.plist:	260
Paso dos: Manejar la URL en UIApplicationDelegate	260
Paso tres: realizar una tarea en función de la URL	261
Configuración de Deeplink para su aplicación.....	261
Capítulo 63: Enlaces universales	264
Observaciones.....	264
Examples.....	264
Servidor de configuración.....	264
Apoyo a múltiples dominios.....	265
Firma del archivo App-Site-Association.....	265
Instalar la aplicación iOS (habilitando los enlaces universales).....	266
C objetivo.....	269
Swift:.....	270
Código de aplicación de iOS	270
Capítulo 64: Entrega por paracaídas	271
Examples.....	271
Entrega por paracaídas.....	271
Capítulo 65: Escáner de códigos QR	272
Introducción.....	272
Examples.....	272
UIViewController busca QR y muestra la entrada de video.....	272
Escanear el código QR con el marco de AVFoudation.....	273
Paso 1	273
Paso 2	274
Paso 3	274
Capítulo 66: Establecer fondo de vista	276
Examples.....	276
Establecer fondo de vista.....	276
Rellene la imagen de fondo de una vista.....	276

Establecer vista de fondo con imagen.....	276
Creación de una vista de fondo degradado.....	276
Capítulo 67: EventKit.....	278
Examples.....	278
Solicitando Permiso.....	278
Rápido.....	278
C objetivo.....	278
Haciendo un EKEventStore.....	278
Rápido.....	278
C objetivo.....	278
Nota.....	278
Comprobando disponibilidad.....	278
Rápido.....	279
C objetivo.....	279
Solicitando Permiso.....	279
Rápido.....	279
Accediendo a diferentes tipos de calendarios.....	279
Accediendo a la gama de calendarios.....	279
Rápido.....	280
Iterando a través de calendarios.....	280
Rápido.....	280
Accediendo al título y color del calendario.....	280
Rápido.....	280
C objetivo.....	280
Añadiendo un evento.....	280
Creando el objeto evento.....	280
Rápido.....	280
C objetivo.....	280
Configuración de calendario relacionado, título y fechas.....	280
Rápido.....	281
Añadiendo evento al calendario.....	281

Rápido.....	281
C objetivo.....	281
Capítulo 68: Extensión para notificaciones push enriquecidas - iOS 10.....	282
Introducción.....	282
Examples.....	282
Extensión de contenido de notificación.....	282
Implementación.....	282
Capítulo 69: FacebookSDK.....	286
Examples.....	286
Integración de FacebookSDK.....	286
Creando su propio botón "Iniciar sesión con Facebook" personalizado.....	288
Obteniendo los datos de usuario de facebook.....	289
Capítulo 70: FileHandle.....	291
Introducción.....	291
Examples.....	291
Lea el archivo del directorio de documentos en trozos.....	291
Capítulo 71: Filtros de CoreImage.....	293
Examples.....	293
Ejemplo de filtro de imagen central.....	293
Capítulo 72: Firma de código.....	299
Examples.....	299
Perfiles de aprovisionamiento.....	299
Tipos de perfil de aprovisionamiento.....	299
Desarrollo.....	299
Distribución.....	299
Capítulo 73: Fuentes personalizadas.....	300
Examples.....	300
Incrustar fuentes personalizadas.....	300
Fuentes personalizadas con guión gráfico.....	301
UIKit + IBExtensions.h.....	301
UIKit + IBExtensions.m.....	302

Aplicar fuentes personalizadas a los controles dentro de un Storyboard.....	303
Notas (advertencias).....	304
Gotchas (deux).....	304
Resultado.....	305
Muestras.....	305
Manejo de fuentes personalizadas.....	305
Solución de fuente personalizada.....	305
Capítulo 74: GCD (Grand Central Dispatch).....	307
Introducción.....	307
Examples.....	307
Crear una cola de envío.....	307
Conseguir la cola principal.....	307
Grupo de despacho.....	308
Despido semáforo.....	309
Serial vs Colas de despacho concurrentes.....	310
Capítulo 75: Gráfico (Coreplot).....	312
Examples.....	312
Haciendo gráficos con CorePlot.....	312
Capítulo 76: Grupo de despacho.....	315
Introducción.....	315
Examples.....	315
Introducción.....	315
Capítulo 77: Guía para elegir los mejores patrones de arquitectura iOS.....	319
Introducción.....	319
Examples.....	319
Patrón MVC.....	319
Patrones de MVP.....	319
Patrón MVVM.....	320
Patrón VIPER.....	321
Capítulo 78: Guión gráfico.....	324
Introducción.....	324

Examples.....	324
Inicializar.....	324
Fetch Initial ViewController.....	324
Fetch ViewController.....	324
Capítulo 79: Hacer selectivas esquinas UIView redondeadas.....	325
Examples.....	325
Código de objetivo C para hacer la esquina seleccionada de un UIView redondeado.....	325
Capítulo 80: iBeacon.....	326
Parámetros.....	326
Observaciones.....	326
Examples.....	326
Operación básica iBeacon.....	326
Escaneando balizas específicas.....	327
Alineando iBeacons.....	327
Capítulo 81: IBOutlets.....	328
Observaciones.....	328
Examples.....	328
Usando un IBOutlet en un elemento UI.....	328
Capítulo 82: Imágenes de caché en línea.....	330
Examples.....	330
AlamofireImage.....	330
Capítulo 83: Instantánea de UIView.....	331
Examples.....	331
Obtención de la instantánea.....	331
Instantánea con subvista con otro marcado y texto.....	331
Capítulo 84: Integración SqlCipher.....	333
Introducción.....	333
Observaciones.....	333
Examples.....	335
Integración de código:.....	335
Capítulo 85: Interoperabilidad rápida y objetiva-C.....	337

Examples.....	337
Usando clases de Objective-C en Swift.....	337
Paso 1: Agregar implementación de Objective-C - .m.....	337
Paso 2: Añadir encabezado puente.....	337
Paso 3: Añadir encabezado Objective-C - .h.....	339
Paso 4: Construye tu clase de Objective-C.....	339
Paso 5: Agregar clase a Bridging-Header.....	339
Paso 6: Usa tu objeto.....	339
Usando clases rápidas en Objective-C.....	339
Paso 1: Crear nueva clase Swift.....	339
Paso 2: Importar archivos Swift a la clase ObjC.....	340
Paso 3: Usa tu clase.....	340
Nota:.....	340
Capítulo 86: iOS - Implementación de XMPP con el framework Robbie Hanson.....	342
Examples.....	342
Ejemplo de iOS XMPP Robbie Hanson con Openfire.....	342
SRXMPPDemo.....	342
Descargue el ejemplo y todas las clases aquí - https://github.com/SahebRoy92/SRXMPPDemo	342
Pasos a seguir.....	342
Capítulo 87: iOS TTS.....	346
Introducción.....	346
Examples.....	346
Texto a voz.....	346
C objetivo.....	346
Rápido.....	346
Métodos útiles.....	346
Capítulo 88: Kit de juego.....	348
Examples.....	348
Generando numeros al azar.....	348
Generacion.....	348
Rápido.....	348

Encontrar una contraseña en el llavero.....	358
Rápido.....	358
Rápido.....	358
Rápido.....	359
Rápido.....	359
Actualizando una contraseña en el llavero.....	359
Rápido.....	359
Rápido.....	359
Rápido.....	360
Rápido.....	360
Eliminar una contraseña del llavero.....	360
Rápido.....	360
Rápido.....	361
Llavero Añadir, actualizar, eliminar y buscar operaciones utilizando un archivo.....	361
Llavero de control de acceso (TouchID con contraseña de retorno).....	363
Rápido.....	363
Rápido.....	364
Rápido.....	364
Rápido.....	364
Rápido.....	365
Capítulo 91: Localización.....	366
Introducción.....	366
Examples.....	366
Localización en iOS.....	366
Capítulo 92: Manejando el teclado.....	367
Examples.....	367
Desplazando un UIScrollView / UITableView al visualizar el teclado.....	367
Descartar un teclado con toque en la vista.....	368
Crear un teclado personalizado en la aplicación.....	369
Crear el archivo de diseño de teclado .xib.....	369

Cree el archivo de teclado de subclase <code>.swift UIView</code>	370
Configurar el controlador de vista.....	372
Error común.....	373
Notas.....	374
Administrar el teclado usando un delegado de Singleton +.....	374
Mover la vista hacia arriba o hacia abajo cuando el teclado está presente.....	377
Nota: esto solo funciona para el teclado incorporado provisto por iOS.....	377
RÁPIDO:.....	377
C OBJETIVO:.....	378
Capítulo 93: Manejar múltiples entornos usando macro	379
Examples.....	379
Manejar múltiples entornos usando múltiples objetivos y macro.....	379
Capítulo 94: Manejo de esquemas de URL	390
Sintaxis.....	390
Parámetros.....	390
Observaciones.....	390
Examples.....	391
Usando el esquema de URL incorporado para abrir la aplicación Mail.....	391
Rápido:	391
C objetivo:	391
Esquemas de URL de Apple.....	391
Capítulo 95: Marco de contactos	395
Observaciones.....	395
Enlaces útiles	395
Examples.....	395
Autorizar el acceso de contacto.....	395
Importando el framework	395
Rápido.....	395
C objetivo.....	395
Comprobando la accesibilidad	395

Rápido.....	395
C objetivo.....	395
Solicitando Permiso.....	396
Rápido.....	396
Acceso a los contactos.....	396
Aplicando un filtro.....	396
Rápido.....	396
C objetivo.....	396
Especificando claves para buscar.....	396
Rápido.....	397
Trayendo contactos.....	397
Rápido.....	397
Accediendo a los datos de contacto.....	397
Rápido.....	397
Agregar un contacto.....	397
Rápido.....	397
Capítulo 96: Mensajería FCM en Swift.....	399
Observaciones.....	399
Examples.....	399
Inicializar FCM en Swift.....	399
Capítulo 97: Métodos personalizados de selección de UITableViewCells.....	401
Introducción.....	401
Examples.....	401
Distinción entre selección simple y doble en fila.....	401
Capítulo 98: Métodos personalizados de selección de UITableViewCells.....	402
Examples.....	402
Distinción entre selección simple y doble en fila.....	402
Capítulo 99: MKDistanceFormatter.....	403
Examples.....	403
Cadena de distancia.....	403
Unidades de distancia.....	403

Estilo de unidad.....	403
Capítulo 100: MKMapView.....	405
Examples.....	405
Añadir MKMapView.....	405
Cambiar tipo de mapa.....	405
.estándar.....	405
Swift 2.....	405
Swift 3.....	405
C objetivo.....	405
.satélite.....	406
Swift 2.....	406
Swift 3.....	406
C objetivo.....	407
.satelliteFlyover.....	407
Swift 2.....	408
Swift 3.....	408
C objetivo.....	408
.híbrido.....	408
Swift 2.....	408
Swift 3.....	408
C objetivo.....	408
.hybridFlyover.....	409
Swift 2.....	409
Swift 3.....	409
C objetivo.....	410
Establecer zoom / región para el mapa.....	410
Implementación de búsqueda local utilizando MKLocalSearch.....	410
OpenStreetMap Tile-Overlay.....	410
Mostrar ejemplo de UserLocation y UserTracking.....	413
C objetivo.....	413
Rápido.....	413

C objetivo.....	414
Rápido.....	414
Añadiendo Pin / punto de anotación en el mapa.....	414
Simular una ubicación personalizada.....	414
Desplácese hasta la coordenada y el nivel de zoom.....	414
Trabajando Con Anotación.....	416
Ajuste el rectángulo visible de la vista del mapa para mostrar todas las anotaciones.....	417
Capítulo 101: ModeloPresentaciónEstilos.....	418
Introducción.....	418
Observaciones.....	418
Examples.....	418
Explorando ModalPresentationStyle usando Interface Builder.....	418
Capítulo 102: Modismos de inicialización.....	429
Examples.....	429
Ajuste a tuplas para evitar la repetición de código.....	429
Inicializar con constantes posicionales.....	429
Inicializar atributos en didSet.....	429
Agrupar puntos de venta en un objeto NSO personalizado.....	430
Inicializar con entonces.....	430
Método de fábrica con bloque.....	431
Capítulo 103: Modos de fondo.....	432
Introducción.....	432
Examples.....	432
Activar la capacidad de los modos de fondo.....	432
Búsqueda de fondo.....	433
Rápido.....	433
C objetivo.....	434
Rápido.....	434
Prueba de búsqueda de fondo.....	434
Audio de fondo.....	435
Capítulo 104: Modos de fondo y eventos.....	436

Examples.....	436
Reproducir audio en segundo plano.....	436
Capítulo 105: MPMediaPickerDelegate.....	438
Observaciones.....	438
Examples.....	438
Cargue música con MPMediaPickerControllerDelegate y reprodúzcalo con AVAudioPlayer.....	438
Capítulo 106: MPVolumeView.....	440
Introducción.....	440
Observaciones.....	440
Examples.....	440
Añadiendo un MPVolumeView.....	440
Capítulo 107: MVVM.....	441
Examples.....	441
MVVM sin programación reactiva.....	441
Capítulo 108: MyLayout.....	445
Introducción.....	445
Examples.....	445
Una demostración simple para usar MyLayout.....	445
Capítulo 109: Notificaciones enriquecidas.....	447
Introducción.....	447
Examples.....	447
Creando una extensión simple UNNotificationContentExtension.....	447
Capítulo 110: Notificaciones push.....	456
Sintaxis.....	456
Parámetros.....	456
Examples.....	456
Dispositivo de registro para notificaciones push.....	456
Rápido.....	456
C objetivo.....	457
Rápido.....	458
C objetivo.....	459

Rápido.....	459
C objetivo.....	459
Rápido.....	460
C objetivo.....	460
Nota.....	460
Comprobando si su aplicación ya está registrada para la Notificación Push.....	460
Rápido.....	460
Registro para notificaciones push (no interactivas).....	460
Manejo de notificaciones push.....	461
Registro de ID de aplicación para usar con notificaciones automáticas.....	463
Cosas que necesitas.....	463
Habilitar el acceso de APN para ID de aplicación en el Centro de desarrolladores de Apple....	463
Habilitando el acceso APNs en Xcode.....	464
Anular el registro de notificaciones push.....	465
C objetivo.....	465
Rápido.....	465
Configuración del icono de la aplicación número de placa.....	465
Prueba de notificaciones push.....	465
Generar un certificado .pem de su archivo .cer, para pasarlo al desarrollador del servidor.....	467
Capítulo 111: NSArray.....	469
Introducción.....	469
Observaciones.....	469
Examples.....	469
Convertir Array en cadena json.....	469
Capítulo 112: NSAttributedString.....	470
Observaciones.....	470
Examples.....	470
Creación de una cadena que tiene un kerning personalizado (espaciado entre letras).....	470
Crear una cadena con texto tachado.....	470
Anexando cadenas atribuidas y texto en negrita en Swift.....	471
Cambiar el color de una palabra o cadena.....	471

Eliminando todos los atributos	472
Capítulo 113: NSBundle	473
Examples	473
Obtener el paquete principal	473
Obtener paquete por ruta	473
Capítulo 114: NSData	475
Observaciones	475
Recursos utiles	475
Examples	475
Creando objetos NSData	475
Usando un archivo	475
Rápido	475
C objetivo	475
Usando un objeto String	475
Rápido	475
C objetivo	475
Convertir NSData a otros tipos	476
Encadenar	476
Rápido	476
C objetivo	476
A Array	476
Rápido	476
C objetivo	476
A la matriz de bytes	476
Rápido	476
C objetivo	476
Convertir NSData a cadena HEX	476
Rápido	477
C objetivo	477
Capítulo 115: NSDate	478
Sintaxis	478

Observaciones.....	478
Examples.....	479
Obtener fecha actual.....	479
Rápido.....	480
Swift 3.....	480
C objetivo.....	480
Obtenga NSDate Object N segundos desde la fecha actual.....	480
Rápido.....	480
Swift 3.....	480
C objetivo.....	481
Comparación de fechas.....	481
Rápido.....	481
C objetivo.....	481
Rápido.....	481
C objetivo.....	481
Rápido.....	481
C objetivo.....	482
Swift 3.....	482
Obtener tiempo de época de Unix.....	483
Rápido.....	483
C objetivo.....	483
NSDateFormatter.....	483
1. Crea un objeto NSDateFormatter.....	483
Rápido.....	484
Swift 3.....	484
C objetivo.....	484
2. Establece el formato de fecha en el que quieres tu cadena.....	484
Rápido.....	484
C objetivo.....	484
3. Obtener la cadena con formato.....	484
Rápido.....	484

Swift 3.....	484
C objetivo.....	485
Nota.....	485
Extensión útil para convertir la fecha en cadena.....	485
Convierta NSDate que se compone de hora y minuto (solo) a un NSDate completo.....	485
C objetivo.....	485
UTC Time offset from NSDate with TimeZone.....	486
Obtener el tipo de ciclo de tiempo (12 horas o 24 horas).....	486
Comprobando si la fecha actual contiene el símbolo para AM o PM.....	486
C objetivo.....	486
Solicitando el tipo de ciclo de tiempo de NSDateFormatter.....	486
C objetivo.....	487
Referencia.....	487
Obtenga NSDate del formato de fecha JSON "/ Date (1268123281843) /".....	487
C objetivo.....	487
Obtenga tiempo histórico de NSDate (por ejemplo: hace 5s, hace 2m, hace 3h).....	488
C objetivo.....	488
Capítulo 116: NSHTTPCookieStorage.....	489
Examples.....	489
Almacena y lee las cookies de NSUserDefaults.....	489
Capítulo 117: NSInvocación.....	491
Examples.....	491
NSInvocation Objective-C.....	491
Capítulo 118: NSNotificationCenter.....	493
Introducción.....	493
Parámetros.....	493
Observaciones.....	493
Examples.....	493
Añadiendo un observador.....	494
Convenio de denominación.....	494
Swift 2.3.....	494

Swift 3	494
C objetivo	494
Removiendo observadores.....	495
Swift 2.3	495
Swift 3	495
C objetivo	495
Publicar una notificación.....	495
Rápido	495
C objetivo	495
Publicar una notificación con datos.....	495
Rápido	495
C objetivo	496
Observando una Notificación.....	496
Rápido	496
C objetivo	496
Agregar / eliminar un observador con un bloque.....	496
Añadir y eliminar el observador por nombre.....	497
Capítulo 119: NSPredicate	498
Sintaxis.....	498
Examples.....	498
Creando un NSPredicate usando predicateWithBlock.....	498
C objetivo	498
Rápido	499
Creando un NSPredicate usando predicateWithFormat.....	499
C objetivo	499
Rápido	499
Creando un NSPredicate con Variables de Sustitución.....	499
C objetivo	499
Rápido	499
Usando NSPredicate para filtrar una matriz.....	500
C objetivo	500

Rápido.....	500
Validación de formularios utilizando NSPredicate.....	500
NSPredicate con la condición `AND`,` OR` y `NOT`.....	502
C objetivo.....	502
Y - Condición.....	502
O - Condición.....	502
NO - Condición.....	502
Capítulo 120: NSTimer.....	504
Parámetros.....	504
Observaciones.....	504
Examples.....	504
Creando un temporizador.....	504
Manualmente disparando un temporizador.....	505
Invalidando un temporizador.....	505
Opciones de frecuencia del temporizador.....	506
Evento de temporizador repetido.....	506
Evento temporizador retardado no repetido.....	506
Paso de datos utilizando el temporizador.....	507
Capítulo 121: NSURConexión.....	508
Examples.....	508
Métodos de delegado.....	508
Solicitud sincrónica.....	508
Solicitud asincrónica.....	509
Capítulo 122: NSURL.....	510
Examples.....	510
Cómo obtener el último componente de cadena de NSURL String.....	510
Cómo obtener el último componente de cadena de la URL (NSURL) en Swift.....	510
Capítulo 123: NSUserActivity.....	511
Introducción.....	511
Observaciones.....	511
Tipos de actividad.....	511

Convertirse / renunciar a la actividad actual	511
Indexación de búsqueda	511
Examples.....	511
Creando un NSUserActivity.....	511
Capítulo 124: NSUserDefaults	513
Sintaxis.....	513
Observaciones.....	513
Examples.....	513
Estableciendo valores.....	513
Vencejo <3.....	513
Swift 3.....	513
C objetivo.....	513
Vencejo <3.....	514
Swift 3.....	514
C objetivo.....	514
Objetos personalizados	514
Rápido.....	514
C objetivo.....	514
Obtención de valores predeterminados.....	515
Rápido.....	515
C objetivo.....	515
Rápido.....	515
C objetivo.....	515
Valores de ahorro.....	515
Rápido.....	516
C objetivo.....	516
Utilice a los gerentes para guardar y leer datos.....	516
Rápido.....	516
C objetivo.....	517
Nota	517
Borrar NSUserDefaults.....	517

Rápido.....	518
C objetivo.....	518
Usos de UserDefaults en Swift 3.....	518
Capítulo 125: Objetos asociados a Objective-C.....	520
Introducción.....	520
Sintaxis.....	520
Parámetros.....	520
Observaciones.....	520
Examples.....	521
Ejemplo de objeto asociado básico.....	521
Capítulo 126: OpenGL.....	522
Introducción.....	522
Examples.....	522
Proyecto de ejemplo.....	522
Capítulo 127: Operaciones de toda la aplicación.....	523
Examples.....	523
Obtén lo mejor de UIViewController.....	523
Interceptar eventos del sistema.....	523
Capítulo 128: Pasando datos entre los controladores de vista.....	524
Examples.....	524
Usando Segues (pasando datos hacia adelante).....	524
Usando el patrón delegado (pasando los datos de vuelta).....	525
Rápido.....	526
C objetivo.....	527
Rápido.....	527
C objetivo.....	528
Pasando los datos hacia atrás usando desenrollar para segue.....	528
Pasar datos utilizando cierres (devolver datos).....	529
Usando el cierre de devolución de llamada (bloque) pasando los datos de vuelta.....	530
Asignando propiedad (Pasando datos adelante).....	531
Capítulo 129: Pasar datos entre los controladores de vista (con MessageBox-Concept).....	533

Introducción.....	533
Examples.....	533
Ejemplo de uso simple.....	533
Capítulo 130: Perfil con instrumentos.....	534
Introducción.....	534
Examples.....	534
Perfilador de tiempo.....	534
Capítulo 131: plist iOS.....	547
Introducción.....	547
Examples.....	547
Ejemplo:.....	547
Guardar y editar / borrar datos de Plist.....	552
Capítulo 132: Proceso de envío de aplicaciones.....	554
Introducción.....	554
Examples.....	554
Configurar perfiles de aprovisionamiento.....	554
Archivar el código.....	554
Exportar archivo IPA.....	556
Cargar archivo IPA utilizando el cargador de aplicaciones.....	557
Capítulo 133: Pruebas de interfaz de usuario.....	559
Sintaxis.....	559
Examples.....	559
Agregando archivos de prueba a Xcode Project.....	559
Al crear el proyecto.....	559
Después de crear el proyecto.....	559
Identificador de accesibilidad.....	559
Cuando la accesibilidad habilitada en Utilidades.....	560
Cuando la accesibilidad está deshabilitada en Utilidades.....	560
Configuración en el archivo UITest.....	561
UIView, UIImageView, UIScrollView.....	561
UILabel.....	562

UIStackView.....	562
UITableView.....	562
UITableViewCell.....	562
Elementos de UITableViewCell.....	562
UICollectionView.....	562
UIButton, UIBarButtonItem.....	562
UITextField.....	562
UITextView.....	563
UISwitch.....	563
Las alertas.....	563
Deshabilitar animaciones durante la prueba de interfaz de usuario.....	563
Almuerzo y finalización de la aplicación mientras se ejecuta.....	563
Aplicación de almuerzo para pruebas.....	563
Aplicación de terminación.....	563
Rotar dispositivos.....	563
Capítulo 134: Red de redes.....	565
Examples.....	565
Despacho de bloque de finalización en un hilo personalizado.....	565
Capítulo 135: Referencia CGContext.....	566
Observaciones.....	566
Examples.....	566
Dibujar línea.....	566
Dibujar texto.....	566
Capítulo 136: Reino.....	568
Observaciones.....	568
Examples.....	568
Clase de modelo base de RLMObject con clave principal - Objective-C.....	568
Capítulo 137: Segues.....	569
Examples.....	569
Una visión general.....	569
Preparando su controlador de vista antes de disparar un Segue.....	569

PrepareForSegue :	569
Parámetros.....	569
Ejemplo en Swift.....	570
Decidir si se debe realizar un Segue invocado.....	570
ShouldPerformSegueWithIdentifier :	570
Parámetros.....	570
Ejemplo en Swift.....	570
Usando Segues para navegar hacia atrás en la pila de navegación.....	570
Segue de disparo programáticamente.....	571
PerformSegueWithIdentifier:.....	571
Parámetros.....	571
Ejemplo en Swift.....	571
Capítulo 138: Seguridad	572
Introducción.....	572
Examples.....	572
Seguridad de transporte utilizando SSL.....	572
Protección de datos en las copias de seguridad de iTunes.....	573
Capítulo 139: Servicios de safari	575
Examples.....	575
Implementar SFSafariViewControllerDelegate.....	575
Agregar artículos a la lista de lectura de Safari.....	575
Abre una URL con SafariViewController.....	576
Capítulo 140: SESIÓN	577
Observaciones.....	577
Examples.....	578
Solicitud GET simple.....	578
Objective-C Crear una sesión y tarea de datos.....	579
Configuración de la configuración de fondo.....	579
Enviar una solicitud POST con argumentos utilizando NSURLSession en Objective-C.....	580
Capítulo 141: Simulador	586
Introducción.....	586

Observaciones.....	586
Diferentes tipos de simuladores.....	586
Obteniendo ayuda.....	586
Examples.....	587
Simulador de lanzamiento.....	587
Simulación 3D / Force Touch.....	587
Cambiar modelo de dispositivo.....	587
Simulador de navegación.....	587
Botón de inicio.....	587
Bloquear.....	587
Rotación.....	587
Capítulo 142: Simulador construye.....	588
Introducción.....	588
Examples.....	588
Instalando la compilación manualmente en el simulador.....	588
Capítulo 143: Simulando Ubicación Usando archivos GPX iOS.....	589
Examples.....	589
Su archivo .gpx: MPS_HQ.gpx.....	589
Para establecer esta ubicación:.....	589
Capítulo 144: Sintetizador AVSpeech.....	591
Sintaxis.....	591
Parámetros.....	591
Examples.....	591
Creación de un texto básico a voz.....	591
Capítulo 145: SiriKit.....	592
Observaciones.....	592
Diferentes tipos de peticiones Siri.....	592
Examples.....	592
Añadiendo la extensión de Siri a la aplicación.....	592
Añadiendo capacidad.....	592
Añadiendo la extensión.....	592

Según Apple:.....	593
Nota	593
Nota	593
Capítulo 146: SLComposeViewController	595
Examples.....	595
SLComposeViewController para Twitter, facebook, SinaWeibo y TencentWeibo.....	595
Capítulo 147: StoreKit	597
Examples.....	597
Obtenga información sobre productos localizados en la App Store.....	597
Capítulo 148: Swift: cambiando el control rootViewController en AppDelegate para presentar ..	598
Introducción.....	598
Observaciones.....	598
Enfoques:	598
Examples.....	599
Opción 1: intercambiar el controlador de vista de raíz (bueno).....	599
Opción 2: Presentar el flujo alternativo de manera modal (mejor).....	599
Capítulo 149: SWRevealViewController	601
Observaciones.....	601
Examples.....	601
Configurando una aplicación básica con SWRevealViewController.....	601
Capítulo 150: Tablas de clasificación de GameCenter	606
Examples.....	606
Tablas de clasificación de GameCenter.....	606
Capítulo 151: Teclado personalizado	609
Examples.....	609
Ejemplo de teclado personalizado.....	609
Capítulo 152: Tiempo de ejecución en Objective-C	617
Examples.....	617
Usando objetos asociados.....	617
Capítulo 153: Tipo dinámico	619
Observaciones.....	619

Examples.....	619
Obtener el tamaño del contenido actual.....	619
Rápido.....	619
C objetivo.....	619
Notificación de cambio de tamaño de texto.....	619
Rápido.....	619
C objetivo.....	620
Coincidencia de tamaño de fuente de tipo dinámico en WKWebView.....	620
Rápido.....	620
Manejo del cambio de tamaño de texto preferido sin notificaciones en iOS 10.....	621
Rápido.....	621
Capítulo 154: Toque 3D.....	622
Examples.....	622
Toque 3D con Swift.....	622
3 D Touch Objective-C Ejemplo.....	623
Capítulo 155: Tutorial de AirPrint en iOS.....	625
Examples.....	625
Impresión de AirPrint Banner Texto.....	625
Capítulo 156: Ubicación del núcleo.....	627
Sintaxis.....	627
Observaciones.....	627
Simular una ubicación en tiempo de ejecución.....	627
Examples.....	628
Link CoreLocation Framework.....	628
Solicitar permiso para usar los servicios de localización.....	629
Obtención del permiso de servicio de ubicación mientras la aplicación está en uso.....	629
Obtener permiso de servicio de ubicación siempre.....	630
Agrega tu propia ubicación personalizada usando el archivo GPX.....	632
Servicios de localización en el fondo.....	633
Capítulo 157: UIActivityViewController.....	635
Parámetros.....	635

Examples.....	635
Inicializando el Controlador de Vista de Actividades.....	635
C objetivo.....	635
Rápido.....	635
Capítulo 158: UIAlertController.....	636
Observaciones.....	636
Examples.....	636
AlertViews con UIAlertController.....	636
Pop-up temporal como un brindis.....	638
Rápido.....	638
Agregar campo de texto en UIAlertController como un cuadro de solicitud.....	638
Rápido.....	638
C objetivo.....	638
Hojas de acción con UIAlertController.....	639
Hoja de acción simple con dos botones.....	639
Rápido.....	639
C objetivo.....	639
Rápido.....	639
C objetivo.....	640
Rápido.....	640
C objetivo.....	640
Rápido.....	640
C objetivo.....	640
Hoja de acción con botón destructivo.....	641
Rápido.....	641
C objetivo.....	642
Visualización y manejo de alertas.....	642
Un botón.....	642
Rápido.....	642
Dos botones.....	643
Rápido.....	643

Tres botones	644
Rápido.....	644
Manipulación de botones	645
Rápido.....	645
Notas	645
Resaltando un botón de acción.....	645
Capítulo 159: UIBarButtonItem	647
Parámetros.....	647
Observaciones.....	647
Examples.....	647
Creando un UIBarButtonItem.....	647
Creando un UIBarButtonItem en el Interface Builder.....	647
Agrega un controlador de navegación a tu guión gráfico	647
Añadir un elemento de botón de barra	648
Establecer los atributos	649
Añadir una acción IB	650
Notas	650
Elemento de botón de barra Imagen original sin color de tinte.....	650
Capítulo 160: UIBezierPath	651
Examples.....	651
Cómo aplicar el radio de la esquina a los rectángulos dibujados por UIBezierPath.....	651
Cómo crear formas simples usando UIBezierPath.....	653
UIBezierPath + AutoLayout.....	655
Cómo aplicar sombras a UIBezierPath.....	656
Diseñando y dibujando un camino Bézier.....	657
Cómo dibujar una ruta Bézier en una vista personalizada	657
Diseño de contorno de la forma	658
Divide el camino en segmentos	658
Construye el camino programáticamente	659
Dibujar el camino	661

Estudio adicional	663
Notas	664
Vista circular y vista de columna con UIBezierPath.....	664
Capítulo 161: UIButton	667
Introducción.....	667
Observaciones.....	667
Tipos de botones	667
Examples.....	668
Creando un UIButton.....	668
Establecer título.....	668
Establecer el color del título.....	669
Alineación horizontal de contenidos.....	669
Obtener la etiqueta del título.....	670
Deshabilitando un UIButton.....	670
Agregar una acción a un UIButton a través del Código (programáticamente).....	670
Fuente de ajuste.....	671
Adjuntar un método a un botón.....	671
Obtenga el tamaño de UIButton basado estrictamente en su texto y fuente.....	672
Establecer imagen.....	672
Rápido.....	672
C objetivo.....	672
Estados de control múltiples	672
Rápido.....	673
C objetivo.....	673
Capítulo 162: UICollectionView	674
Examples.....	674
Crear una vista de colección programáticamente.....	674
Swift - UICollectionViewDelegateFlowLayout.....	674
Crear un UICollectionView.....	674
UICollectionView - Fuente de datos.....	675
Ejemplo Swift básico de una vista de colección.....	676
Crear un nuevo proyecto	676

Agrega el código	676
Configurar el guión gráfico	677
Conectar los puntos de venta	679
Terminado	679
Haciendo mejoras	680
Estudio adicional	681
Realización de actualizaciones por lotes	681
UICollectionViewDelegate configuración y selección de elementos	682
Administrar la vista de colección múltiple con DataSource y Flowlayout	683
Capítulo 163: UIColor	686
Examples	686
Creando un UIColor	686
Métodos no documentados	687
styleString	687
_systemDestructiveTintColor()	688
Color con componente alfa	689
Rápido	689
Swift 3	689
C objetivo	689
Hacer que los atributos definidos por el usuario apliquen el tipo de datos CGColor	689
El nuevo atributo definido por el usuario (borderUIColor) será reconocido y aplicado sin p	689
Creando un UIColor a partir de un número hexadecimal o cadena	690
Brillo de color ajustado de UIColor	692
UIColor de un patrón de imagen	693
Tono más claro y oscuro de un UIColor dado	694
Capítulo 164: UIControl - Manejo de eventos con bloques	696
Examples	696
Introducción	696
Capítulo 165: UIDatePicker	700
Observaciones	700
Examples	700

Crear un selector de fecha.....	700
Rápido.....	700
C objetivo.....	700
Configuración de fecha mínima-máxima.....	700
Fecha minima.....	700
Fecha maxima.....	700
Modos.....	700
Ajuste de intervalo de minutos.....	701
Duración de la cuenta regresiva.....	701
Capítulo 166: UIDevice.....	702
Parámetros.....	702
Observaciones.....	702
Examples.....	702
Obtener el nombre del modelo del dispositivo iOS.....	702
Obtención del estado de la batería y del nivel de la batería.....	704
Identificando el dispositivo y operando.....	704
Obtención de la orientación del dispositivo.....	705
Obtención del estado de la batería del dispositivo.....	706
Usando el sensor de proximidad.....	707
Capítulo 167: UIFeedbackGenerator.....	708
Introducción.....	708
Examples.....	708
Impacto de gatillo háptico.....	708
Rápido.....	708
C objetivo.....	709
Capítulo 168: UIFont.....	710
Introducción.....	710
Examples.....	710
Declarar e inicializar UIFont.....	710
Cambiando la fuente de una etiqueta.....	710
Capítulo 169: UIGestureRecognizer.....	711
Examples.....	711

UITapGestureRecognizer.....	711
UIPanGestureRecognizer.....	712
UITapGestureRecognizer (Doble toque).....	713
Notas.....	713
UILongPressGestureRecognizer.....	713
Notas.....	714
UISwipeGestureRecognizer.....	714
Notas.....	715
UIPinchGestureRecognizer.....	715
Notas.....	716
UIRotationGestureRecognizer.....	716
Notas.....	716
Añadiendo un reconocedor de gestos en el Interface Builder.....	716
Notas.....	718
Capítulo 170: UIImage.....	719
Observaciones.....	719
Examples.....	719
Creando UIImage.....	719
Con imagen local.....	719
Rápido.....	719
C objetivo.....	719
Nota.....	719
Con NSData.....	719
Rápido.....	719
Con UIColor.....	719
Rápido.....	720
C objetivo.....	720
Con contenido de archivo.....	720
C objetivo.....	720
Creando e inicializando objetos de imagen con el contenido del archivo.....	721

Imagen redimensionable con gorras.....	721
Comparando imagenes.....	722
Rápido.....	722
C objetivo.....	722
Crea UIImage con UIColor.....	723
Rápido.....	723
Swift 3.....	723
C objetivo:.....	723
Imagen degradada con colores.....	723
Gradiente de fondo de capa para límites.....	724
Convertir UIImage a / desde la codificación base64.....	724
Toma una instantánea de una vista.....	725
Aplicar UIColor a UIImage.....	725
Cambiar UIImage Color.....	725
Capítulo 171: UIImagePickerController.....	727
Introducción.....	727
Examples.....	727
Uso genérico de UIImagePickerController.....	727
Capítulo 172: UIImageView.....	729
Examples.....	729
Crear un UIImageView.....	729
Asignando una imagen a un UIImageView.....	729
Animando un UIImageView.....	730
Haciendo una imagen en un círculo o redondeado.....	730
C objetivo.....	731
Rápido.....	731
UIImage enmascarado con etiqueta.....	732
C objetivo.....	732
Swift 3.....	732
Cambiar color de una imagen.....	732
Cómo afecta la propiedad Modo a una imagen.....	732

Escala para llenar	733
Ajuste de aspecto	734
Relleno de aspecto	734
Redibujar	735
Centrar	735
Parte superior	736
Fondo	736
Izquierda	737
Derecha	737
Arriba a la izquierda	738
Parte superior derecha	738
Abajo a la izquierda	738
Abajo a la derecha	739
Notas	739
Capítulo 173: UIKit Dynamics	741
Introducción	741
Observaciones	741
Rápido	741
C objetivo	741
Examples	742
La plaza que cae	742
Vista de película basada en la velocidad del gesto	743
Rápido	743
C objetivo	745
Efecto de las "esquinas pegajosas" usando UITextFieldBehaviors	747
Rápido	747
C objetivo	749
UIDynamicBehavior Driven Custom Transition	752
Rápido	752
C objetivo	753

Rápido.....	754
C objetivo.....	754
Rápido.....	756
C objetivo.....	759
Transición de la sombra con la física del mundo real utilizando comportamientos UIDynamic.....	763
Rápido.....	764
C objetivo.....	765
Rápido.....	766
C objetivo.....	767
Rápido.....	767
C objetivo.....	771
Animación dinámica del mapa Cambios de posición a límites.....	776
Rápido.....	777
C objetivo.....	777
Rápido.....	777
C objetivo.....	778
Rápido.....	779
C objetivo.....	780
Capítulo 174: UIKit Dynamics con UICollectionView.....	782
Introducción.....	782
Examples.....	782
Creación de un comportamiento de arrastre personalizado con UIDynamicAnimator.....	782
Rápido.....	783
C objetivo.....	784
Rápido.....	785
C objetivo.....	786
Rápido.....	786
C objetivo.....	788
Rápido.....	789
C objetivo.....	791
Capítulo 175: UILabel.....	794

Introducción.....	794
Sintaxis.....	794
Observaciones.....	794
Examples.....	794
Cambio de texto en una etiqueta existente.....	794
Configuración del texto con literales de String.....	794
Configurando el texto con una variable.....	795
Color de texto.....	795
Aplicando color de texto a una porción del texto.....	796
Alineación del texto.....	796
Crea una UILabel.....	797
Con un marco.....	797
Rápido.....	797
C objetivo.....	797
Con diseño automático.....	797
Rápido.....	797
C objetivo.....	798
Con Objective-c + Visual Format Language (VFL).....	798
Con Interface Builder.....	798
Enlace entre Interface Builder y View Controller.....	799
Rápido.....	799
C objetivo.....	800
Establecer fuente.....	800
Rápido.....	800
C objetivo.....	800
Cambiar el tamaño de la fuente por defecto.....	800
Rápido.....	800
Swift 3.....	800
C objetivo.....	800
Use un peso de fuente específico.....	800
Rápido.....	800

Swift3.....	801
C objetivo.....	801
Rápido.....	801
Swift3.....	801
C objetivo.....	801
Utilice un estilo de texto de tipo dinámico.....	801
Rápido.....	801
Swift 3.....	801
C objetivo.....	801
Use una fuente diferente por completo.....	801
Rápido.....	802
C objetivo.....	802
Anular tamaño de fuente.....	802
Rápido.....	802
Swift 3.....	802
C objetivo.....	802
Usar fuente personalizada Swift.....	802
Número de líneas.....	802
Configurando el valor programáticamente.....	803
Rápido.....	803
C objetivo.....	803
Nota.....	803
Rápido.....	803
C objetivo.....	803
Nota.....	803
Nota.....	803
Estableciendo el valor en el Interface Builder.....	803
Tamaño para caber.....	804
Color de fondo.....	806
Añadir sombras al texto.....	807

Altura variable utilizando restricciones.....	807
Rápido.....	807
Rápido.....	808
LineBreakMode.....	808
Usando código.....	808
Rápido.....	808
Swift 3.....	808
C objetivo.....	809
Usando el guión gráfico.....	809
Constantes.....	809
Calcular límites de contenido (por ejemplo, alturas de celda dinámicas).....	810
Etiqueta clickeable.....	812
Rápido.....	812
C objetivo.....	812
Configuración de "userInteractionEnabled" en el inspector de atributos del guión gráfico.....	812
Marco de etiqueta dinámico de longitud de texto desconocido.....	813
C objetivo.....	813
Rápido.....	813
Etiqueta de texto atribuido.....	813
Justifica el texto.....	821
Etiqueta de tamaño automático para ajustar el texto.....	822
Pin los bordes izquierdo y superior.....	822
Notas.....	822
Obtenga el tamaño de UILabel basado estrictamente en su texto y fuente.....	823
Color de texto resaltado y resaltado.....	824
Capítulo 176: UILabel texto subrayado.....	825
Examples.....	825
Subrayando un texto en un UILabel usando Objective C.....	825
Subrayando un texto en UILabel usando Swift.....	825
Capítulo 177: UILocalNotification.....	826
Introducción.....	826

Observaciones.....	826
Examples.....	826
Programación de una notificación local.....	826
Registro para notificaciones locales.....	827
Respondiendo a la notificación local recibida.....	828
Gestionando notificaciones locales utilizando UUID.....	828
Rastrear una notificación.....	828
Cancelar una notificación.....	829
Presentando una notificación local de inmediato.....	829
Sonido de notificación.....	830
Regístrese y programe notificaciones locales en Swift 3.0 (iOS 10).....	830
Novedades en UILocalNotification con iOS10.....	831
Capítulo 178: UINavigationController.....	835
Observaciones.....	835
Examples.....	835
Apareciendo en un controlador de navegación.....	835
Al controlador de vista anterior.....	835
Al controlador de vista raíz.....	835
Creación de un controlador de navegación.....	835
Incrustar un controlador de vista en un controlador de navegación mediante programación.....	836
Presionando un controlador de vista en la pila de navegación.....	836
Propósito.....	837
Capítulo 179: UIPageViewController.....	838
Introducción.....	838
Sintaxis.....	838
Observaciones.....	838
Examples.....	838
Crear una paginación horizontal UIPageViewController programáticamente.....	838
Una forma sencilla de crear controladores de vista de página horizontales (páginas infinitas).....	840
Capítulo 180: UIPheonix: marco de IU fácil, flexible, dinámico y altamente escalable.....	844
Introducción.....	844

Observaciones.....	844
Examples.....	844
Ejemplo de componentes de interfaz de usuario.....	844
Ejemplo de uso.....	845
Capítulo 181: UIPickerView.....	847
Examples.....	847
Ejemplo basico.....	847
Rápido.....	847
C objetivo.....	847
Cambiando pickerView Color de fondo y color de texto.....	848
Capítulo 182: UIRefreshControl TableView.....	849
Introducción.....	849
Examples.....	849
Objective-C Ejemplo.....	849
Configurar refreshControl en tableView:.....	850
Capítulo 183: UIScrollView.....	851
Examples.....	851
Crear un UIScrollView.....	851
Tamaño de contenido de la vista de desplazamiento.....	851
ScrollView con AutoLayout.....	851
Desplazar contenido con diseño automático habilitado.....	857
Conceptos clave.....	858
Iniciar un nuevo proyecto.....	858
Guión gráfico.....	858
Terminado.....	861
Estudio adicional.....	861
Activar / Desactivar desplazamiento.....	861
Zoom In / Out UIImageView.....	862
Ahora crea la instancia de UIImageView.....	862
Detectar cuándo UIScrollView terminó de desplazarse con métodos delegados.....	863
C objetivo:.....	863

Rápido:.....	863
Restringir la dirección de desplazamiento.....	864
Capítulo 184: UIScrollView AutoLayout.....	865
Examples.....	865
Controlador de desplazamiento.....	865
Tamaño de contenido dinámico de UIScrollView a través de Storyboard.....	868
Capítulo 185: UIScrollView con niño StackView.....	871
Examples.....	871
Un ejemplo de StackView complejo dentro de Scrollview.....	871
Prevenir el diseño ambiguo.....	872
Desplazarse hasta el contenido dentro de StackViews anidados.....	873
Capítulo 186: UISearchController.....	874
Sintaxis.....	874
Parámetros.....	874
Observaciones.....	875
Examples.....	875
Barra de búsqueda en el título de la barra de navegación.....	875
Barra de búsqueda en el encabezado de vista de tabla.....	878
Implementación.....	879
UISerachController en Objective-C.....	880
Capítulo 187: UISlider.....	881
Examples.....	881
UISlider.....	881
Ejemplo de SWIFT.....	881
Añadiendo una imagen de pulgar personalizada.....	882
Capítulo 188: UISplitViewController.....	883
Observaciones.....	883
Examples.....	883
Interacción de la vista maestra y de detalle usando delegados en el objetivo C.....	883
Capítulo 189: UISplitViewController.....	892
Observaciones.....	892
Examples.....	892

Interactuando entre la vista maestra y la vista detallada usando delegados en el Objetivo	892
Capítulo 190: UIStackView	896
Examples	896
Crear una vista de pila horizontal programáticamente	896
Crear una vista de pila vertical programáticamente	896
Botones centrales con UIStackview	897
Capítulo 191: UIStoryboard	905
Introducción	905
Examples	905
Obteniendo una instancia de UIStoryboard programáticamente	905
RÁPIDO:	905
C OBJETIVO:	905
Abre otro guión gráfico	906
Capítulo 192: UISwitch	907
Sintaxis	907
Observaciones	907
1. Referencia de UISwitch: Documentación de Apple	907
2. Otra referencia dada por: Enoch Huang	907
Examples	907
Activar / desactivar	907
Establecer color de fondo	908
Establecer color de tinte	908
Establecer imagen para estado activado / desactivado	908
Capítulo 193: UITabBarController	910
Examples	910
Crear una instancia	910
Cambio del título de la barra de pestañas y el icono	910
C objetivo:	911
Rápido:	911
Controlador de navegación con TabBar	911
Personalización del color de la barra de pestañas	913
UITabBarController con selección de color personalizada	913

Elegir imagen para la barra de pestañas y configurar el título de la pestaña aquí	914
Selección de otra pestaña	915
Crear controlador de barra de pestañas programáticamente sin guión gráfico	915
Capítulo 194: UITableView	917
Introducción	917
Sintaxis	917
Observaciones	918
Examples	919
Células de auto-tamaño	919
Creando un UITableView	920
Agrega un UITableView a tu Guión Gráfico	920
Poblando tu tabla con datos	920
Creando una fuente de datos simple	921
Configurando su fuente de datos en su View Controller	921
Conectando la fuente de datos de la vista de tabla a su controlador de vista	922
Manejo de selecciones de filas	922
La solución definitiva	923
Rápido	923
C objetivo	924
Delegado y fuente de datos	925
UITableViewDataSource	925
UITableViewDelegate	929
Celdas personalizadas	931
Creación de su celda personalizada	932
Expandiendo y colapsando UITableViewCells	934
Deslizar para eliminar filas	937
Agrega el código	937
Guión gráfico	939
Terminado	939
Notas	939

Otras lecturas	939
Líneas separadoras	939
Edición del ancho de las líneas de separación	939
Cambio de las líneas de separación para celdas específicas	939
Eliminar todas las líneas de separación	940
Ocultar el exceso de líneas de separación	940
Capítulo 195: UITableViewCell	943
Introducción	943
Examples	943
Archivo Xib de UITableViewCell	943
Capítulo 196: UITableViewController	945
Introducción	945
Examples	945
TableView con propiedades dinámicas con tableViewCellStyle basic	945
TableView con celda personalizada	946
Capítulo 197: UITextField	948
Introducción	948
Sintaxis	948
Examples	949
Inicializar campo de texto	949
Rápido	949
C objetivo	949
Generador de interfaz	949
Entrada de vista de accesorios (barra de herramientas)	949
Rápido	949
C objetivo	950
Auto capitalización	950
Rápido	950
C objetivo	950
Descartar teclado	950
Rápido	951

C objetivo.....	953
Establecer alineación.....	953
Rápido.....	953
C objetivo.....	953
Tipo de teclado.....	953
Desplazando el desplazamiento cuando UITextView se convierte en el primer respondedor.....	954
Obtener el enfoque del teclado y ocultar el teclado.....	956
Rápido.....	956
C objetivo.....	956
Rápido.....	956
C objetivo.....	957
Reemplace el teclado con UIPickerView.....	957
Descartar teclado cuando el usuario presiona el botón de retorno.....	960
Obtención y configuración de la posición del cursor.....	961
Información útil.....	961
Obtener posición del cursor.....	961
Establecer la posición del cursor.....	961
Relacionado.....	962
Notas.....	963
Relacionado.....	963
Ocultar carete parpadeante.....	963
Swift 2.3 <.....	963
Swift 3.....	963
C objetivo.....	964
Cambiar el marcador de posición de color y fuente.....	964
Crear un UITextField.....	964
Rápido.....	964
C objetivo.....	964
Capítulo 198: UITextField Delegate.....	966
Examples.....	966
UITextField - Restringir el campo de texto a ciertos caracteres.....	966

Buscar siguiente etiqueta y administrar teclado.....	967
Acciones cuando un usuario ha comenzado / terminado interactuando con un campo de texto.....	967
Capítulo 199: UITextField personalizado.....	969
Introducción.....	969
Examples.....	969
UITextField personalizado para filtrar el texto de entrada.....	969
Personalizar campo de campo para no permitir todas las acciones como copiar, pegar, etc.....	970
Capítulo 200: UITextView.....	971
Examples.....	971
Cambiar texto.....	971
Establecer texto atribuido.....	971
Cambiar la alineación del texto.....	971
UITextViewDelegate métodos.....	971
Cambiar fuente.....	972
Cambiar el color del texto.....	972
UITextView con texto HTML.....	972
Detección automática de enlaces, direcciones, fechas y más.....	973
Habilitando la autodetección.....	973
Datos seleccionables.....	973
Compruebe si está vacío o nulo.....	973
Obtención y configuración de la posición del cursor.....	974
Información útil.....	974
Obtener posición del cursor.....	974
Establecer la posición del cursor.....	974
Relacionado.....	975
Notas.....	976
Relacionado.....	976
Elimine los rellenos adicionales para ajustar a un texto medido con precisión.....	976
Capítulo 201: UIViewController.....	977
Examples.....	977
Subclases.....	977

Crear una instancia.....	979
Establecer la vista programáticamente.....	979
Instancia de un guión gráfico.....	979
Accede al controlador de vista de contenedor.....	980
Agregar / quitar un controlador de vista hijo.....	981
Capítulo 202: UIViews personalizados de archivos XIB.....	982
Observaciones.....	982
Examples.....	982
Elementos de cableado.....	982
Cómo hacer UIView reutilizable personalizado utilizando XIB.....	1003
Capítulo 203: UIWebView.....	1005
Observaciones.....	1005
Examples.....	1005
Crear una instancia de UIWebView.....	1005
Hacer una solicitud de URL.....	1005
Dejar de cargar contenido web.....	1006
Actualizar contenido web actual.....	1006
Determinar el tamaño del contenido.....	1006
Cargar cadena HTML.....	1007
Cargar JavaScript.....	1007
Cargue archivos de documentos como .pdf, .txt, .doc, etc.....	1008
Hacer enlaces que dentro de UIWebview haga clic en.....	1008
Cargar archivo HTML local en webView.....	1009
Capítulo 204: Usando Aseets de Imagen.....	1011
Introducción.....	1011
Examples.....	1011
Icono de la aplicación utilizando los activos de imagen.....	1011
LaunchImage usando los recursos de imagen.....	1014
Notas:.....	1018
Capítulo 205: UUID (identificador único universal).....	1019
Observaciones.....	1019
Examples.....	1019

Generando UUID.....	1019
UUID aleatorio.....	1019
Rápido.....	1019
C objetivo.....	1019
Identificador para el vendedor.....	1019
Rápido.....	1019
C objetivo.....	1020
IFA de Apple frente a IFV (Identificador de Apple para anunciantes frente a Identificador.....)	1020
Crear una cadena UUID para dispositivos iOS.....	1020
Swift 3.0.....	1020
Capítulo 206: Vista.....	1022
Sintaxis.....	1022
Observaciones.....	1022
Examples.....	1022
Crear una vista UIV.....	1022
Hacer la vista redondeada.....	1023
Programáticamente.....	1023
Configuración de Storyboard.....	1024
Extensión rápida.....	1025
Tomando una instantánea.....	1025
Usando IBInspectable y IBDesignable.....	1025
Animando una vista.....	1028
Extensión UIView para atributos de tamaño y marco.....	1029
Administre mediante programación la inserción y eliminación de UIView en y desde otra UIVI.....	1030
Crear UIView utilizando Autolayout.....	1031
Utilizando Tamaño de Contenido Intrínseco.....	1033
Sacude una vista.....	1036
Capítulo 207: WCSSessionDelegate.....	1038
Introducción.....	1038
Examples.....	1038
Controlador del kit de reloj (WKInterfaceController).....	1038
Capítulo 208: WKWebView.....	1039

Introducción.....	1039
Examples.....	1039
Creando un WebBrowser simple.....	1039
Agregar script de usuario personalizado cargado desde el paquete de aplicaciones.....	1045
Envía mensajes desde JavaScript y los maneja desde el lado nativo.....	1046
Capítulo 209: Xcode Build & Archive desde la línea de comandos.....	1047
Sintaxis.....	1047
Parámetros.....	1047
Observaciones.....	1047
Examples.....	1047
Construir y archivar.....	1047
Capítulo 210: XCTest framework - Unit Testing.....	1049
Examples.....	1049
Agregar archivos de prueba a Xcode Project.....	1049
Al crear el proyecto.....	1049
Después de crear el proyecto.....	1049
Rápido.....	1050
C objetivo.....	1050
Agregar Storyboard y View Controller como instancias al archivo de prueba.....	1051
Definiendo el controlador de vista.....	1051
Rápido.....	1051
Presentando el Storyboard e inicializando el controlador de vista.....	1051
Rápido.....	1051
C objetivo.....	1051
Añadiendo métodos de prueba.....	1052
Métodos de prueba.....	1052
Rápido.....	1052
C objetivo.....	1052
Rápido.....	1052
C objetivo.....	1052
Nota.....	1053

Empezar a probar.....	1053
Probando un método específico.....	1053
Probando todos los métodos.....	1053
Ver el resultado de la prueba.....	1053
Ejecutando todas las pruebas.....	1053
Importar un módulo que pueda ser probado.....	1054
Vista de gatillo de carga y apariencia.....	1054
Ver cargando.....	1054
Ver apariencia.....	1054
Escribiendo una clase de prueba.....	1054
Creditos.....	1056

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [ios](#)

It is an unofficial and free iOS ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official iOS.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con iOS

Observaciones

Notas

1- No necesita una cuenta de desarrollador de Apple para comenzar a desarrollar aplicaciones iOS. La documentación y las herramientas se pueden descargar gratis con su ID de Apple. También puede firmar e instalar aplicaciones en *sus dispositivos personales* utilizando la misma ID de Apple. Si desea distribuir o vender aplicaciones en la [App Store](#), debe inscribirse en el Programa para desarrolladores de Apple a partir de 99 USD (este es el precio en el momento de la redacción y puede cambiar). Esto también agregará incidentes de soporte a nivel de código y pruebas beta para sus aplicaciones a través de TestFlight.

2- Crear un ID de Apple sin una tarjeta de crédito [requiere un proceso corto](#). Si no le importa asociar un método de pago como parte de la inscripción, vaya a <https://appleid.apple.com/>

- [Empezar a desarrollar aplicaciones iOS \(Swift\)](#)
- [Ayuda de Xcode \(incluyendo Primeros pasos\)](#)
- [Descargas \(incluido Xcode si no desea pasar por la AppStore\)](#)

Etiquetas relacionadas de desbordamiento de pila

- IDE (entorno de desarrollo integrado) de [xcode](#) Apple para desarrollar aplicaciones iOS y macOS
- [swift-language](#) Uno de los principales idiomas que puede utilizar para desarrollar en iOS.
- [object-c-language](#) Uno de los principales idiomas que puede utilizar para desarrollar en iOS.
- [cacao](#) Una API de Apple para desarrollar en iOS y macOS.
- [sprite-kit](#) para gráficos animados en 2D.
- [Core-Data](#) Para almacenar y recuperar datos relacionales.

Versiones

Versión	Fecha de lanzamiento
iPhone OS 2	2008-07-11
iPhone OS 3	2009-06-17

Versión	Fecha de lanzamiento
iOS 4	2010-06-08
iOS 5	2011-10-12
ios 6	2012-09-19
ios 7	2013-09-18
iOS 8	2014-09-17
iOS 8.1	2014-10-20
iOS 8.2	2015-03-09
iOS 8.3	2015-04-08
iOS 8.4	2015-06-30
iOS 9	2015-09-16
iOS 9.1	2015-10-22
iOS 9.2	2015-12-08
iOS 9.3	2016-03-21
iOS 10.0.1	2016-09-13
iOS 10.1	2016-10-24
iOS 10.2	2016-12-12
iOS 10.2.1	2017-01-23
iOS 10.3	2017-03-27
iOS 10.3.3	2017-07-19

Examples

Creación de una aplicación de vista única predeterminada

Para desarrollar una aplicación para iOS, debes comenzar con una aplicación llamada Xcode. Hay otras herramientas alternativas que puedes usar, pero Xcode es la herramienta oficial de Apple. Tenga en cuenta, sin embargo, que solo se ejecuta en macOS. La última versión oficial es Xcode 8.3.3 con Xcode 9 (actualmente en versión beta) que se lanzará más adelante este año.

1. Inicia tu Mac e instala [Xcode desde la App Store](#) si aún no está instalado.

(Si prefiere no usar la App Store o tiene problemas, también puede [descargar Xcode del sitio web de Apple Developer](#) , pero asegúrese de seleccionar la última versión y **no** una versión beta).



2. Abrir Xcode. Se abrirá la siguiente ventana:



Welcome to Xcode

Version 8.0 (8A218a)



Get started with a playground

Explore new ideas quickly and easily.



Create a new Xcode project

Create an app for iPhone, iPad, Mac, Apple TV, or watchOS.



Check out an existing project

Start working on something from an SCM repository.

La ventana te presenta las siguientes opciones:

- **Comenzando con un campo de juego:** Esto se introdujo con el lenguaje Swift y Xcode 6. Es un área interactiva que se puede usar para escribir pequeños fragmentos de código para verificar los cambios en el tiempo de ejecución. Es una excelente manera para que los estudiantes de Swift sean introducidos a las nuevas funciones de Swift.

Esto se introdujo con el lenguaje Swift y Xcode 6. Es un área interactiva que se puede usar para escribir pequeños fragmentos de código para verificar los cambios en el tiempo de ejecución. Es una excelente manera para que los estudiantes de Swift sean introducidos a las nuevas funciones de Swift.

- **Crear un nuevo proyecto de Xcode:** *elija esta opción* , que crea un nuevo proyecto con la configuración predeterminada.
- **Revisar un proyecto existente:** esto se usa para desproteger un proyecto desde una ubicación de repositorio, por ejemplo, retirar un proyecto de SVN.

3. Seleccione la segunda opción **Crear un nuevo proyecto de Xcode** y Xcode le pedirá que realice una configuración inicial del proyecto:

Choose a template for your new project:

ios

watchOS

tvOS

macOS

Cross-platform

Application



Single View
Application



Game



Master
App



Sticker Pack
Application



iMessage
Application

Framework & Library



Cocoa Touch
Framework



Cocoa Touch
Static Library



Metal

Cancel

Este asistente se utiliza para seleccionar su plantilla de proyecto. Hay 5 opciones:

- **iOS:** se utiliza para crear aplicaciones, bibliotecas y marcos de iOS
- **watchOS:** se utiliza para crear aplicaciones, bibliotecas y marcos de watchOS
- **tvOS:** se utiliza para crear aplicaciones, bibliotecas y marcos de tvOS
- **macOS:** se utiliza para crear aplicaciones macOS, bibliotecas, marcos, paquetes, AppleScripts, etc.
utiliza para crear aplicaciones macOS, bibliotecas, marcos, paquetes, AppleScripts, etc.
- **Multiplataforma:** se utiliza para crear aplicaciones multiplataforma, plantillas y contenidos de compra dentro de la aplicación

Puedes ver que hay muchas plantillas diferentes para tu aplicación. Estas plantillas son útiles para impulsar su desarrollo; están pre-construidos con algunas configuraciones básicas de proyectos como interfaces de UI y archivos de clase.

Aquí, usaremos la primera opción, **iOS** .

1. Aplicación Master-Detail:

Esta plantilla contiene una interfaz maestra y de detalles combinada: el maestro contiene objetos que están relacionados con la interfaz de detalles. La selección de objetos en el maestro cambiará la interfaz de detalles. Puede ver este tipo de interfaz de usuario en las aplicaciones de configuración, notas y contactos en el iPad.

2. Aplicación basada en la página:

Esta plantilla se utiliza para crear la aplicación basada en la página. Las páginas son vistas diferentes mantenidas por un contenedor.

3. Aplicación de vista única:

Esta es una plantilla normal de desarrollo de aplicaciones. Esto es bueno para que los principiantes aprendan el flujo de aplicaciones.

4. Aplicación tabulada:

Esta plantilla crea pestañas en la parte inferior de una aplicación. Cada pestaña tiene una IU diferente y un flujo de navegación diferente. Puede ver esta plantilla utilizada en aplicaciones como Clock, iTunes Store, iBooks y App Store.

5. Juego:

Este es un punto de partida para el desarrollo del juego. Puede ir más lejos con tecnologías de juegos como SceneKit, SpriteKit, OpenGL ES y Metal.

4. En este ejemplo, comenzaremos con la **aplicación de vista única**

Choose options for your new project:

Product Name:

Team:

None

Organization Name:

StackOver

Organization Identifier:

com.stacko

Bundle Identifier:

com.stacko

Language:

Swift

Devices:

Universal

Use Core

Include U

Include U

Cancel

El asistente le ayuda a definir las propiedades del proyecto:

- **Nombre del producto:** El nombre del proyecto / aplicación.
- **Nombre** de la organización : el nombre de la organización en la que está involucrado
- **Identificador de organización:** el identificador de organización único que se utiliza en el identificador de paquete. Se recomienda seguir la notación inversa del servicio de nombres de dominio.
el identificador de organización único que se utiliza en el identificador de paquete. Se recomienda seguir la notación inversa del servicio de nombres de dominio.
- **Identificador de paquete:** *este campo es muy importante*. Se basa en su nombre de proyecto y el identificador de la organización, elija sabiamente. El identificador del paquete se usará en el futuro para instalar la aplicación en un dispositivo y cargarla en iTunes Connect (que es el lugar donde subimos las aplicaciones para publicarlas en la App Store). Es una clave única para identificar su aplicación.
- **Idioma:** el lenguaje de programación que le gustaría usar. Aquí puede cambiar Objective-C a Swift si no está seleccionado.
- **Dispositivos:** dispositivos compatibles para su aplicación que se pueden cambiar más adelante. Muestra iPhone, iPad y Universal. Las aplicaciones universales admiten dispositivos iPhone y iPad, y se recomienda seleccionar esta opción cuando no es necesario ejecutar la aplicación en un solo tipo de dispositivo.
- **Use Core Data:** si desea usar Core Data Model en su proyecto, `.xcdatamodel` como seleccionado y creará un archivo para el `.xcdatamodel` . También puede agregar este archivo más adelante si no sabe de antemano.
- **Incluir pruebas unitarias:** esto configura el objetivo de la prueba unitaria y crea clases para la prueba unitaria
- **Incluir prueba de interfaz de usuario:** esto configura el objetivo de la prueba de interfaz de usuario y crea clases para las pruebas de interfaz de usuario

Haga clic en **Siguiente** y le pedirá una ubicación donde desea crear el directorio del proyecto.

Haga clic en **Crear** y verá la interfaz de usuario de Xcode con una configuración de proyecto ya definida. Puedes ver algunas clases y archivos de Storyboard.

Esta es una plantilla básica para una aplicación de vista única.

En la parte superior izquierda de la ventana, verifique que se haya seleccionado un simulador (por ejemplo, "iPhone 6" como se muestra aquí) y luego presione el botón RUN triangular.



5. Se abrirá una nueva aplicación: Simulador (esto puede llevar algo de tiempo la primera vez que lo ejecute y es posible que deba intentarlo dos veces si ve un error la primera vez). Esta aplicación nos proporciona la simulación de dispositivos para las aplicaciones creadas. ¡Casi parece un dispositivo real! Contiene algunas aplicaciones como un dispositivo real. Puede simular orientaciones, ubicación, movimiento de agitación, advertencias de memoria,

barra de estado durante una llamada, toque con el dedo, bloqueo, reinicio, inicio, etc.

Verá una aplicación en blanco normal porque todavía no hemos realizado ningún cambio en la plantilla.

Así que empieza tu propia. Es un largo plazo y hay muchas nuevas oportunidades esperándote.

Si no estás seguro de a dónde ir a continuación, prueba el tutorial de Apple ' [Jump Right In](#) '. Ya has realizado los primeros pasos, así que estás listo para comenzar.

Hola Mundo

Después de configurar Xcode, no es difícil poner en funcionamiento su primer iOS.

En el siguiente ejemplo:

- Iniciar un nuevo proyecto
- Agregar una etiqueta
- Imprimiendo mensaje a consola.
- Correr en el simulador

Comenzando un nuevo proyecto

Cuando aparezca la pantalla de bienvenida de Xcode, elija **Crear un nuevo proyecto de Xcode** . Alternativamente, puede hacer **Archivo> Nuevo> Proyecto ...** desde el menú de Xcode si ya lo tiene abierto.



Welcome to Xcode

Version ...



Get started with a playground

Explore new ideas quickly and easily.



Create a new Xcode project

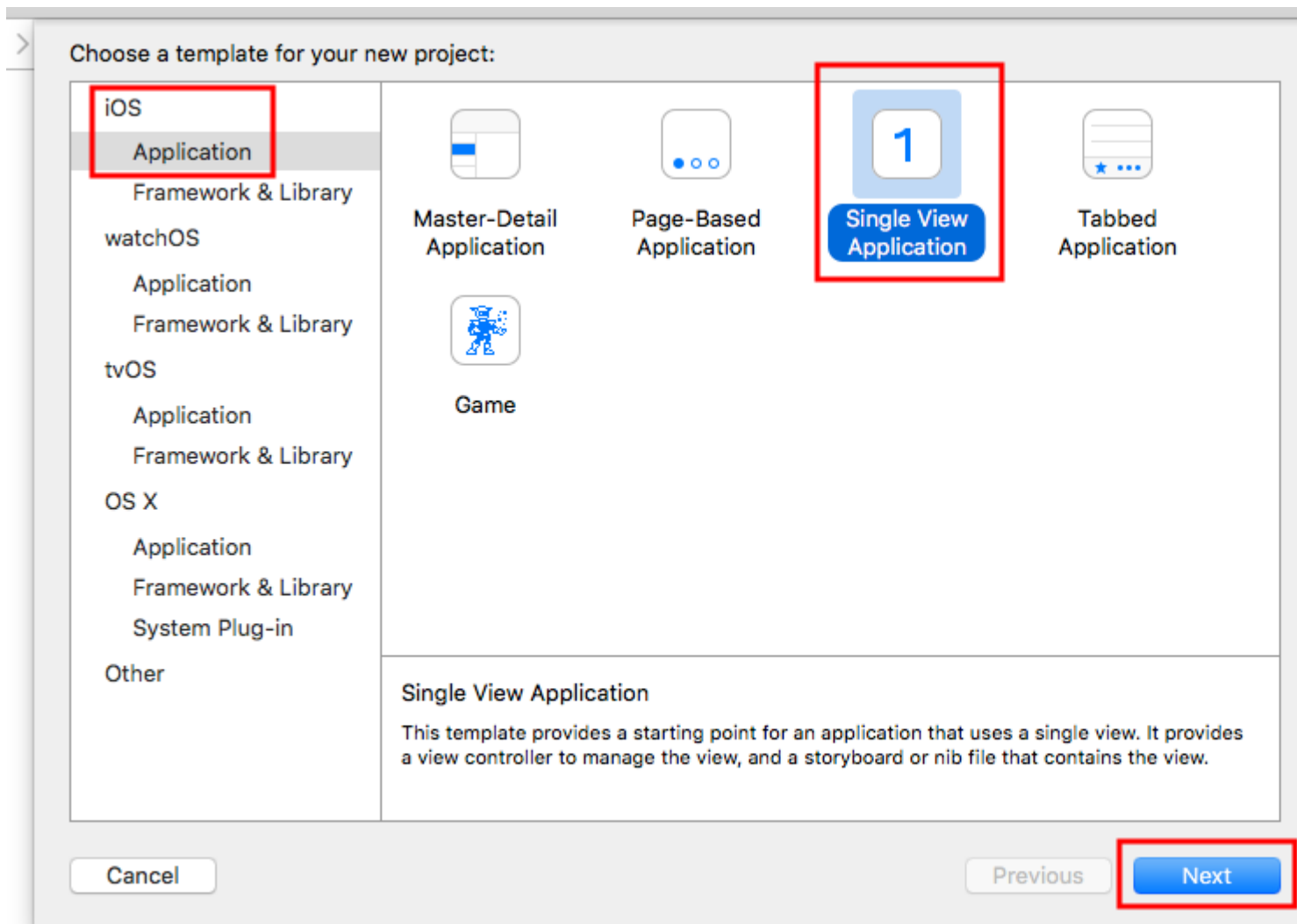
Start building a new iPhone, iPad or Mac application.



Check out an existing project

Start working on something from an SCM repository.

Elija una **aplicación de vista única** y haga clic en **Siguiente** .



Escribe "HelloWorld" para el **nombre del producto** (o lo que realmente quieras) y en **Idioma** , asegúrate de que **Swift** esté seleccionado.

- **Universal** significa que su aplicación se ejecutará tanto en el iPhone como en el iPad.
- **Use Core Data se** refiere al almacenamiento de datos persistentes, que no es necesario en nuestra aplicación Hello World.
- No haremos **Pruebas Unitarias** o **Pruebas de UI** en este ejemplo, pero no está de más hacer un hábito de agregarlas.

> Choose options for your new project:

Product Name: HelloWorld

Organization Name: Me

Organization Identifier: com.example

Bundle Identifier: com.example.HelloWorld

Language: Swift

Devices: Universal

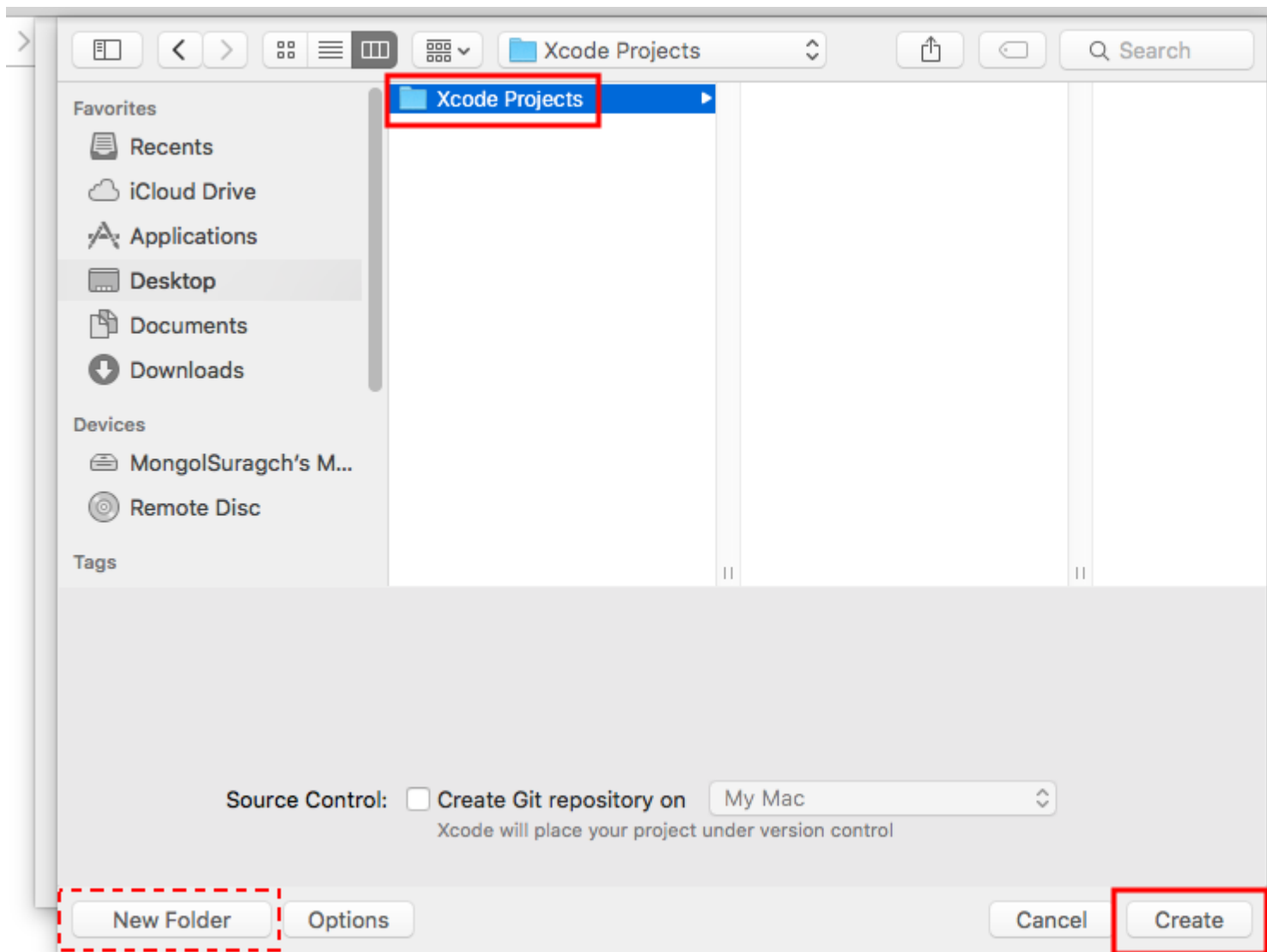
Use Core Data

Include Unit Tests

Include UI Tests

Cancel Previous Next

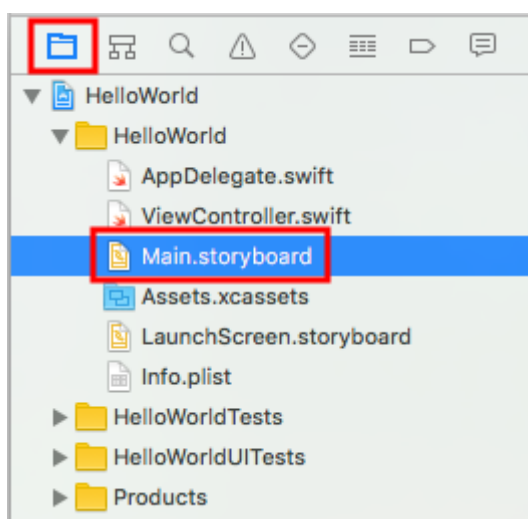
Elija una carpeta existente o cree una nueva donde guardará sus proyectos de Xcode. Este será el predeterminado en el futuro. Creamos uno aquí llamado "Proyectos Xcode". Luego haga clic en **Crear** . Puede seleccionar Source Control si lo desea (usado cuando se sincroniza con sitios como [GitHub](https://github.com)), pero no lo necesitaremos en este ejemplo.



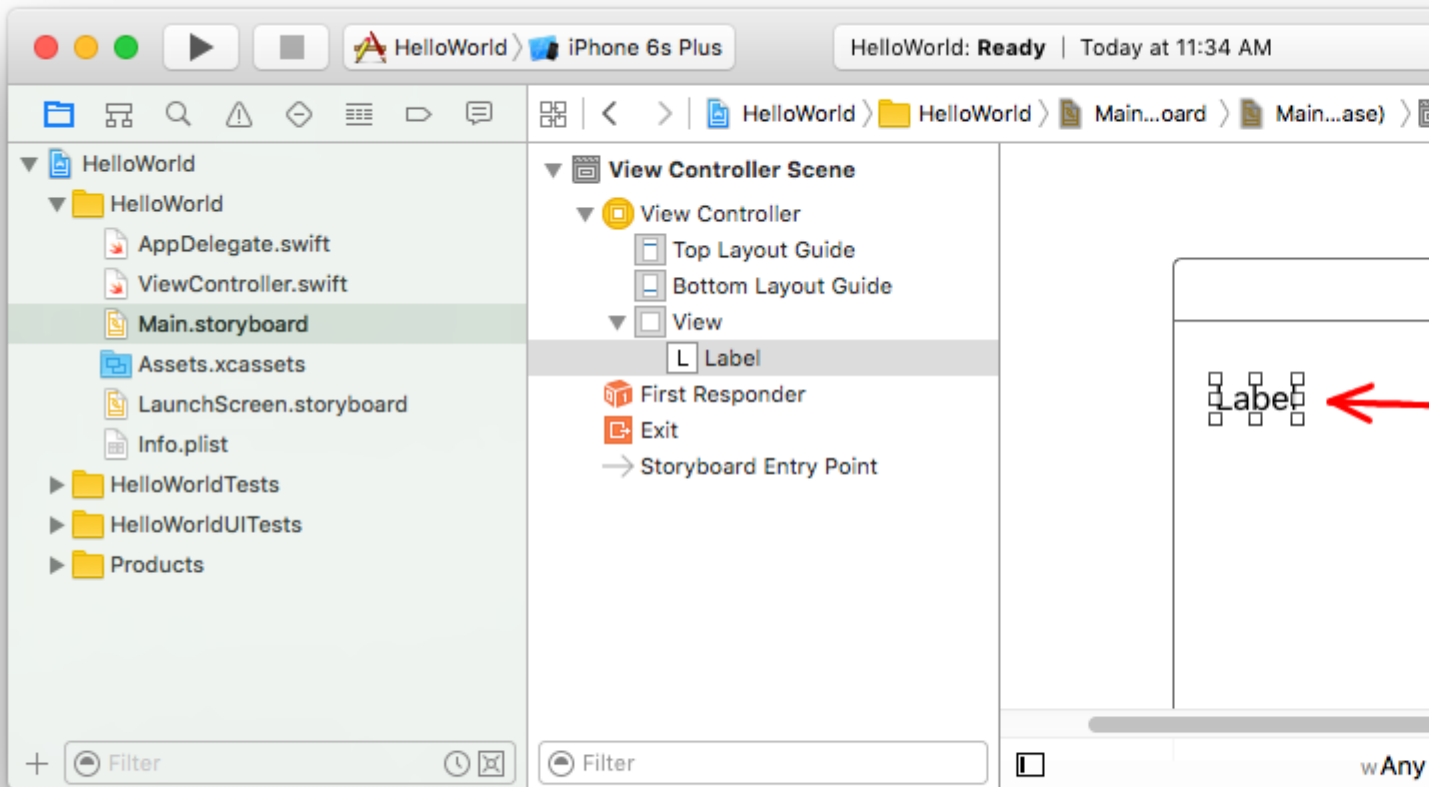
Añadiendo una etiqueta

Esta es la estructura de archivos de un proyecto Xcode.

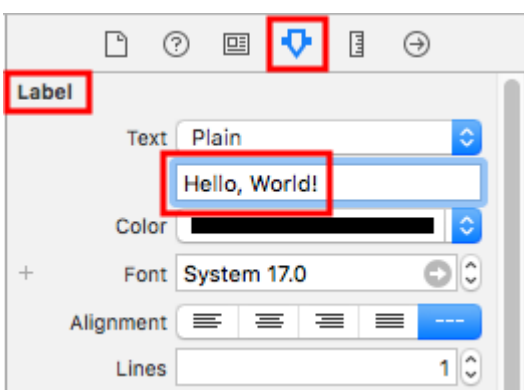
Seleccione *Main.storyboard* en el Project Navigator.



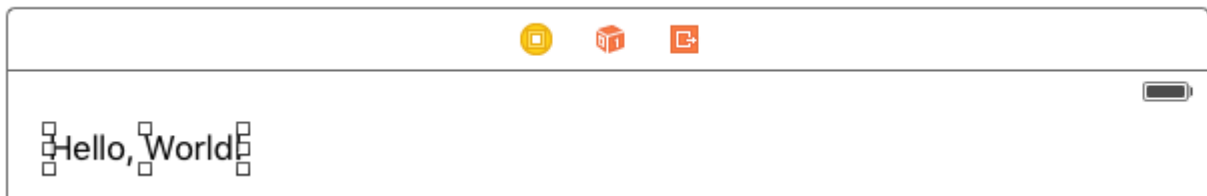
Escriba "etiqueta" en el campo de búsqueda de la biblioteca de objetos en la parte inferior derecha de Xcode. A continuación, arrastre la UILabel al controlador de vista del guión gráfico. Colóquelo generalmente en la región de la esquina superior izquierda.



Asegúrese de que la etiqueta esté seleccionada en el guión gráfico y luego en el **Inspector de atributos**, cambie el texto a "¡Hola, mundo!". A continuación, tendrá que cambiar el tamaño y la posición de la etiqueta en el guión gráfico, ya que la longitud del texto es más larga ahora.

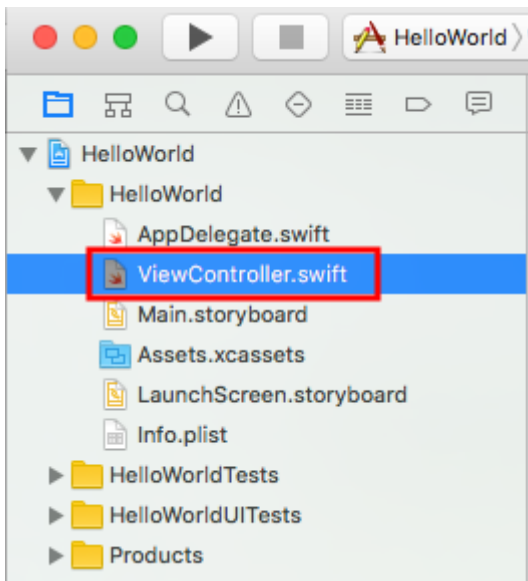


Alternativamente, haga doble clic en la etiqueta en el guión gráfico para editarla para que sea "¡Hola, mundo!". En cualquier caso, el guión gráfico debería verse algo así:



Agregando Código

Seleccione *ViewController.swift* en el Project Navigator.



Agregue `print("Successfully created my first iOS application.")` `viewDidLoad()` `print("Successfully created my first iOS application.")` al método `viewDidLoad()` . Debería verse algo como esto.

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        // print to the console when app is run
        print("Successfully created my first iOS application.")
    }

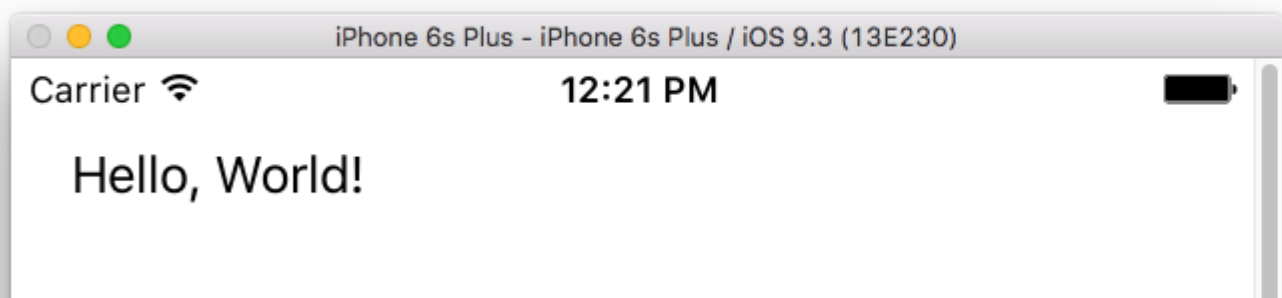
    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }
}
```

Ejecutando la aplicación en el simulador



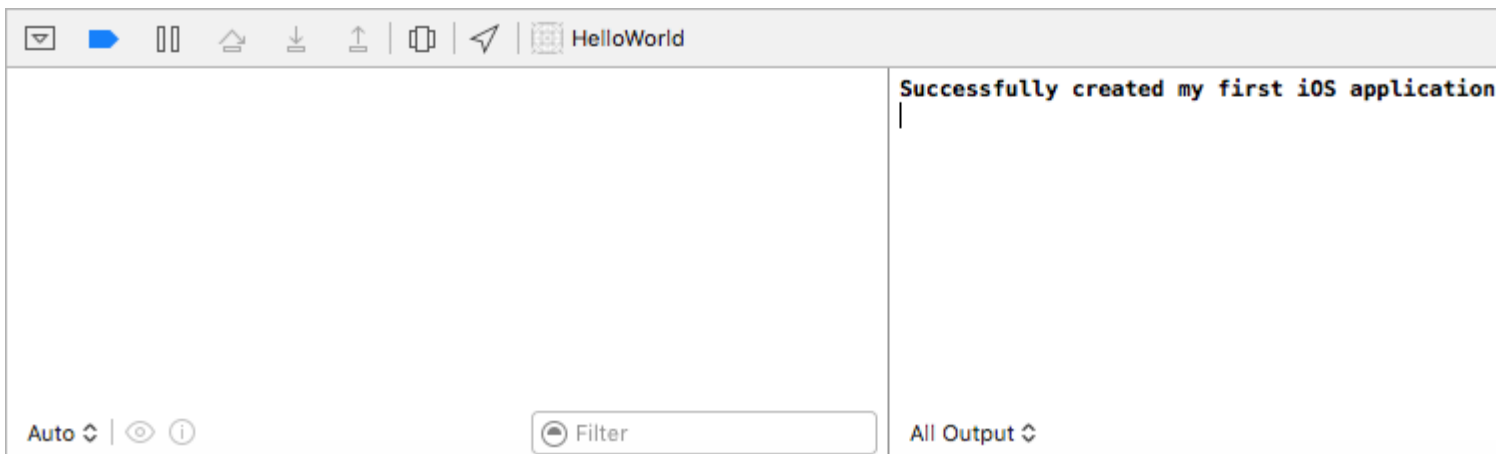
Presiona el botón Ejecutar para construir y ejecutar la aplicación. En este ejemplo, el dispositivo simulador actual (denominado "esquema") se asignó de manera predeterminada al iPhone 6s Plus. Las versiones más recientes de Xcode predeterminarán a los esquemas más nuevos. También puede elegir otros esquemas haciendo clic en el nombre. Simplemente nos quedaremos con el valor predeterminado.

El simulador tardará algún tiempo en iniciarse en la primera ejecución. Una vez en ejecución, debe verse así:



En el menú del simulador, puede seleccionar **Ventana > Escala** para hacerlo más pequeño, o presione `cmd + 1/2/3/4/5` para la escala 100% / 75% / 50% / 33% / 25% respectivamente.

El área de depuración de Xcode (en la parte inferior) también debería haber impreso "He creado con éxito mi primera aplicación iOS". a la consola. "Creé con éxito mi primera aplicación de iOS". mensaje es la cadena que imprimiste mediante programación en la parte **Agregar código** .

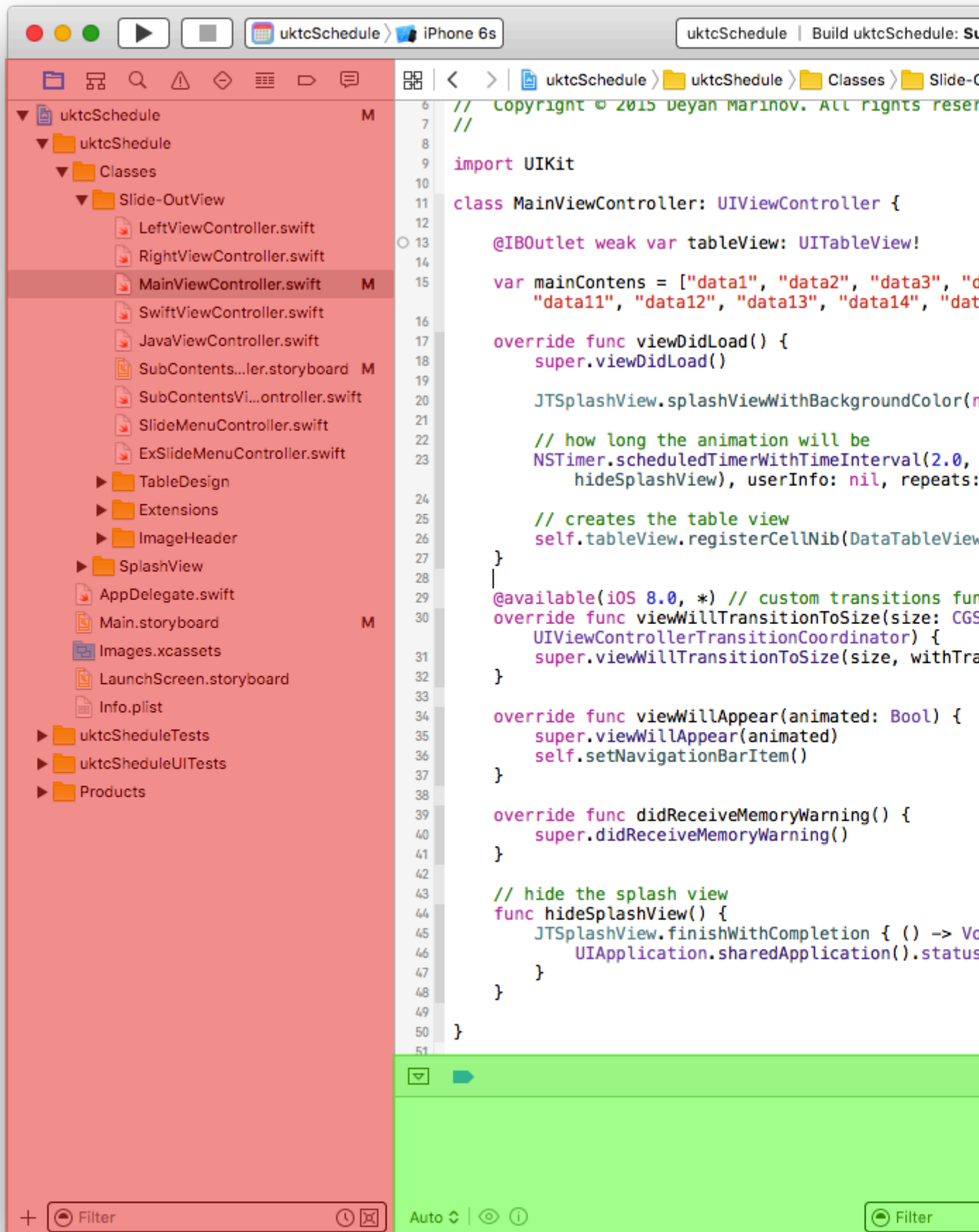


Continuando

Debería aprender acerca de las restricciones de diseño automático a continuación. Estos le ayudan a colocar sus controles en el guión gráfico para que se vean bien en cualquier tamaño y orientación del dispositivo.

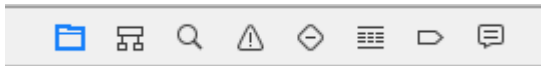
Interfaz Xcode

En Xcode, tiene tres áreas de trabajo distintas: Navegadores (en rojo), Área de depuración (en verde) y Utilidades (en azul).



La ventana del área de trabajo siempre incluye el área del editor. Cuando selecciona un archivo en su proyecto, su contenido aparece en el área del editor, donde Xcode abre el archivo en un editor apropiado. Por ejemplo, en la imagen de arriba, el área del editor MainViewController.swift, un archivo de código Swift que se selecciona en el área del navegador a la izquierda de la ventana del área de trabajo.

Área del navegador



La ventana del navegador contiene las siguientes ocho opciones:

- **Navegador de proyectos.** Agregue, elimine, agrupe y administre los archivos en su proyecto, o elija un archivo para ver o editar su contenido en el área del editor.
- **Símbolo del navegador.** Examine los símbolos en su proyecto como una lista o jerarquía. Los botones a la izquierda de la barra de filtro le permiten limitar los símbolos mostrados a una combinación de solo clases y protocolos, solo símbolos en su proyecto o solo contenedores.
- **Buscar navegador** Use las opciones de búsqueda y los filtros para encontrar rápidamente cualquier cadena dentro de su proyecto.
- **Navegador de incidencias.** Vea problemas como diagnósticos, advertencias y errores encontrados al abrir, analizar y construir su proyecto.
- **Navegador de prueba.** Crea, gestiona, ejecuta y revisa pruebas unitarias.
- **Navegador de depuración.** Examine los subprocesos en ejecución y la información de la pila asociada en un punto o tiempo específico durante la ejecución del programa.
- **Navegador de punto de interrupción.** Ajuste los puntos de interrupción especificando características como las condiciones de activación.
- **Informe del navegador.** Vea el historial de sus tareas de compilación, ejecución, depuración, integración continua y control de fuente.

Los editores

La mayoría del trabajo de desarrollo en Xcode ocurre en el área del editor, el área principal que siempre está visible dentro de la ventana del área de trabajo. Los editores que más utilizas son:

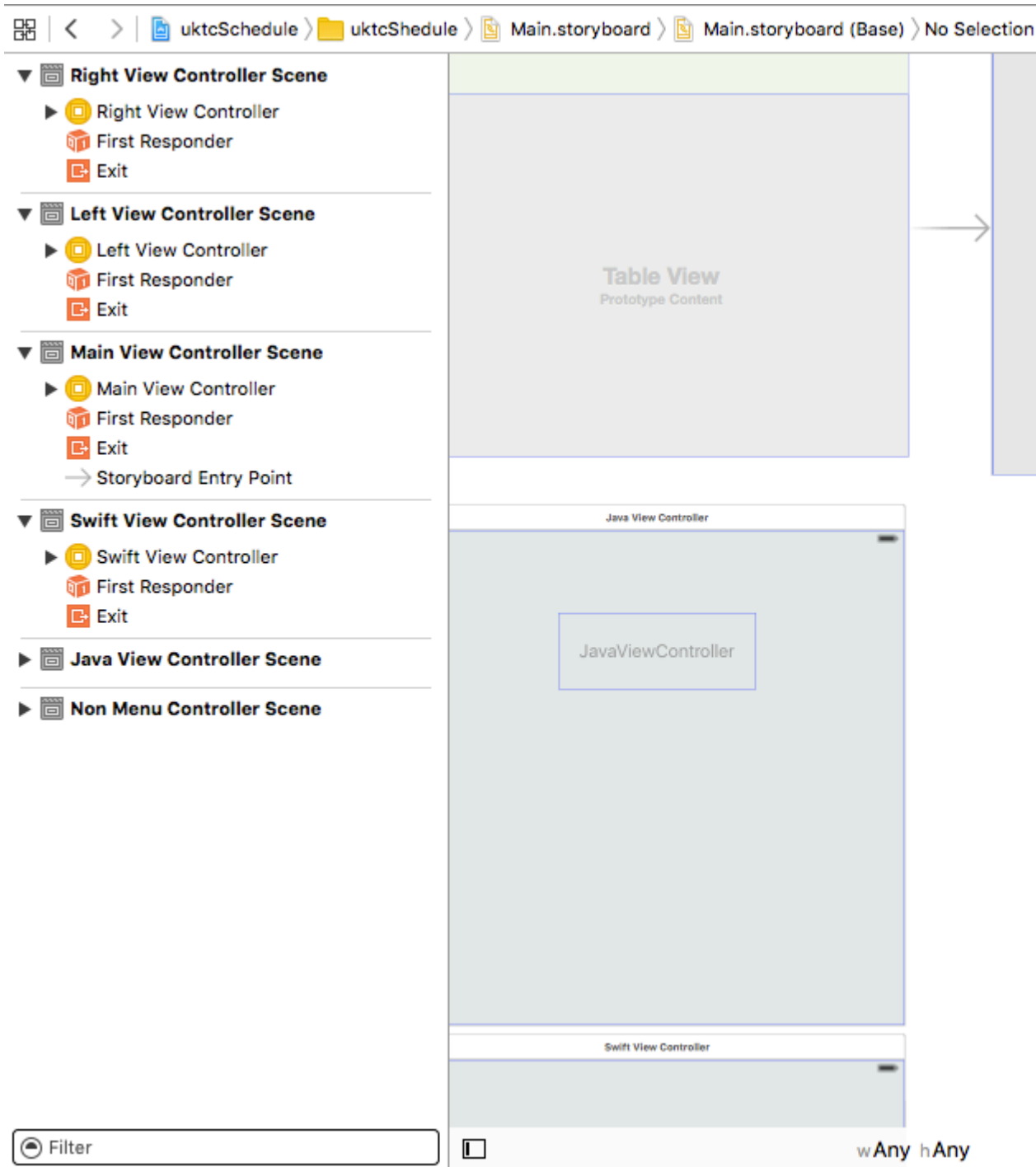
- **Editor de fuente** Escribir y editar código fuente.

```

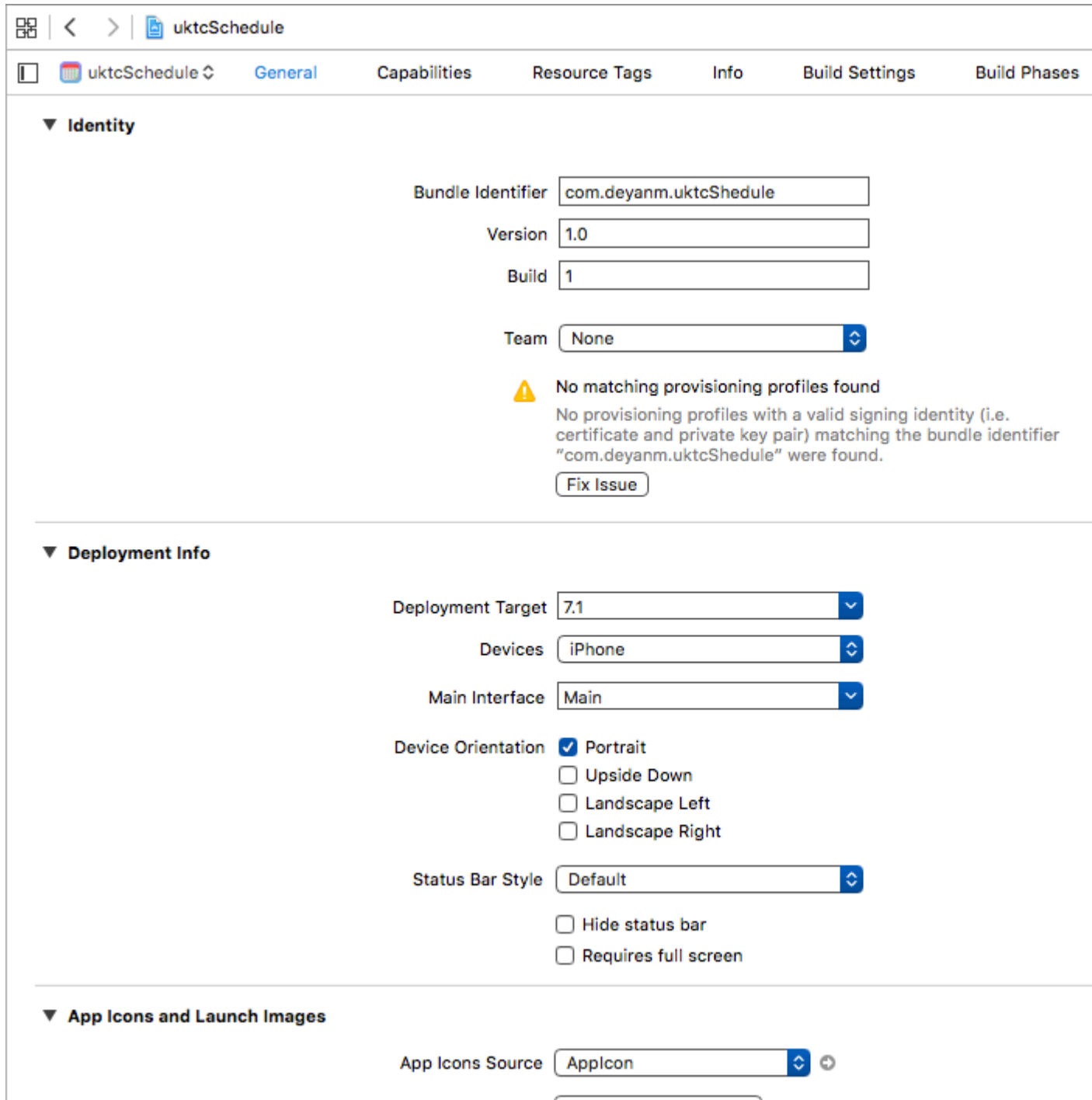
uktcSchedule > uktcShedule > Classes > Slide-OutView > LeftViewController.swift > No Selection
1 //
2 // LeftViewController.swift
3 // uktcShedule
4 //
5 // Created by Deyan Marinov on 10/9/15.
6 // Copyright © 2015 Deyan Marinov. All rights reserved.
7 //
8
9 import UIKit
10
11 enum LeftMenu: Int {
12     case Main = 0
13     case Swift
14     case Java
15 }
16
17 protocol LeftMenuProtocol : class {
18     func changeViewController(menu: LeftMenu)
19 }
20
21 class LeftViewController : UIViewController, LeftMenuProtocol {
22
23     @IBOutlet weak var tableView: UITableView!
24     var menus = ["Main", "Swift", "Java"]
25     var mainViewController: UIViewController!
26     var swiftViewController: UIViewController!
27     var javaViewController: UIViewController!
28     var goViewController: UIViewController!
29     var nonMenuViewController: UIViewController!
30     var imageHeaderView: ImageHeaderView!
31
32     required init?(coder aDecoder: NSCoder) {
33         super.init(coder: aDecoder)
34     }
35
36     override func viewDidLoad() {
37         super.viewDidLoad()
38         self.tableView.separatorColor = UIColor(red: 224/255, green: 224/255, blue: 224/255,
39
40         let storyboard = UIStoryboard(name: "Main", bundle: nil)
41         let swiftViewController = storyboard.instantiateViewControllerWithIdentifier("SwiftV
42         self.swiftViewController = UINavigationController(rootViewController: swiftViewContr
43
44         let javaViewController = storyboard.instantiateViewControllerWithIdentifier("JavaVie
45         self.javaViewController = UINavigationController(rootViewController: javaViewControl
46
47         self.tableView.registerClass(RecursiveTableViewCell.self)

```

- **Generador de interfaz.** Crea y edita gráficamente archivos de interfaz de usuario.



- **Editor de proyectos.** Vea y edite cómo deben construirse sus aplicaciones, por ejemplo, especificando las opciones de construcción, las arquitecturas de destino y los derechos de las aplicaciones.



Configure el área del editor para una tarea determinada con los botones de configuración del editor en el lado derecho de la barra de herramientas:

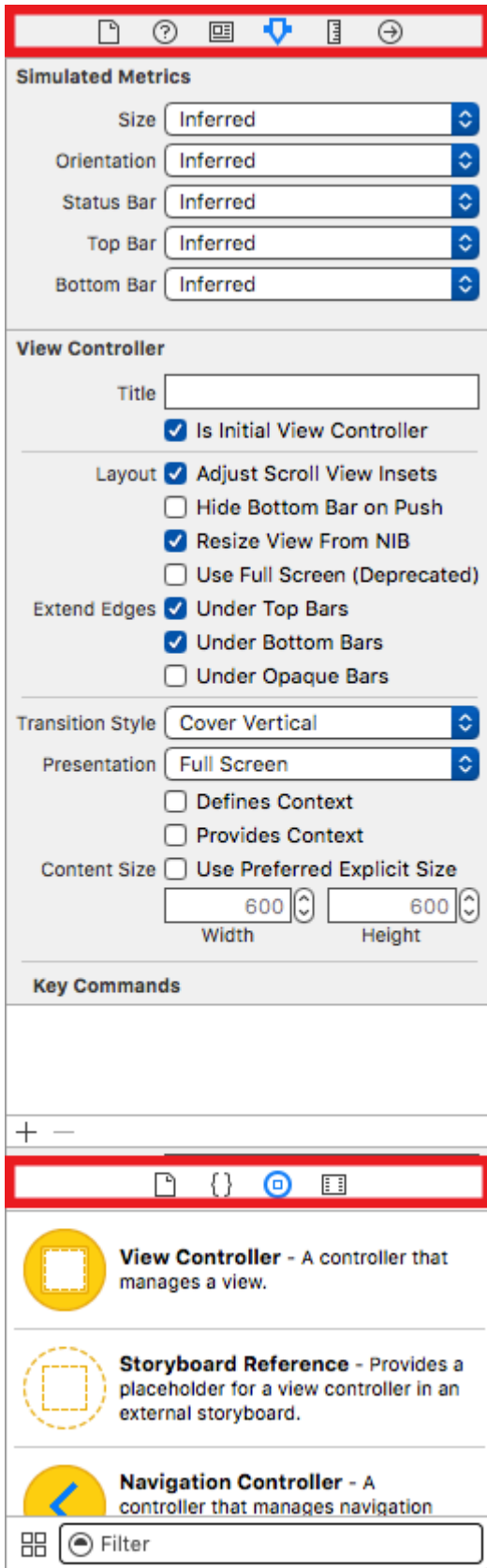


- **Editor estándar.** Rellena el área del editor con el contenido del archivo seleccionado.
- **Editor asistente.** Presenta un panel de editor separado con contenido relacionado lógicamente con el contenido en el panel de editor estándar. También puede cambiar el contenido.
- **Editor de versiones.** Muestra las diferencias entre el archivo seleccionado en un panel y otra versión de ese mismo archivo en un segundo panel. Este editor funciona solo cuando su proyecto está bajo control de código fuente.

Recursos y elementos en el área de servicios públicos.

El área de utilidades en el extremo derecho de la ventana del área de trabajo le brinda acceso rápido a estos recursos: Inspectores, para ver y modificar las características del archivo abierto en un editor Bibliotecas de recursos listos para usar en su proyecto

El panel superior del área de utilidades muestra los inspectores. El panel inferior le da acceso a las bibliotecas.



El primer panel (resaltado en rojo) es la **barra de inspección**, utilízcela para elegir el inspector que mejor se adapte a su tarea actual. Dos inspectores siempre están visibles en la barra de inspectores (hay inspectores adicionales disponibles en algunos editores):

- **Inspector de archivos.** Ver y administrar los metadatos para el archivo seleccionado. Normalmente, localizará guiones gráficos y otros archivos multimedia y cambiará la

configuración de los archivos de interfaz de usuario.

- **Ayuda rápida.** Ver detalles sobre un símbolo, un elemento de interfaz o una configuración de compilación en el archivo. Por ejemplo, la Ayuda rápida muestra una descripción concisa de un método, dónde y cómo se declara el método, su alcance, los parámetros que toma y la disponibilidad de su plataforma y arquitectura.

Use la **barra de la Biblioteca** (la segunda resaltada en rojo) para acceder a las bibliotecas de recursos listas para usar para su proyecto:

- **Plantillas de archivo.** Plantillas para tipos comunes de archivos y construcciones de código.
- **Fragmentos de código.** Piezas cortas de código fuente para usar en su software, como declaraciones de clase, flujos de control, declaraciones de bloque y plantillas para tecnologías de Apple de uso común.
- **Objetos.** Elementos para la interfaz de usuario de su aplicación.
- **Medios de comunicación.** Archivos que contienen gráficos, íconos, archivos de sonido y similares.

Para usar una biblioteca, arrástrela directamente al área apropiada. Por ejemplo, para usar un fragmento de código, arrástrelo desde la biblioteca al editor de origen; para crear un archivo de origen a partir de una plantilla de archivo, arrastre su plantilla al navegador del proyecto.

Para restringir los elementos que se muestran en una biblioteca seleccionada, escriba el texto relevante en el campo de texto en la **barra de filtros** (el panel inferior). Por ejemplo, escriba "botón" en el campo de texto para mostrar todos los botones en la biblioteca de Objetos.

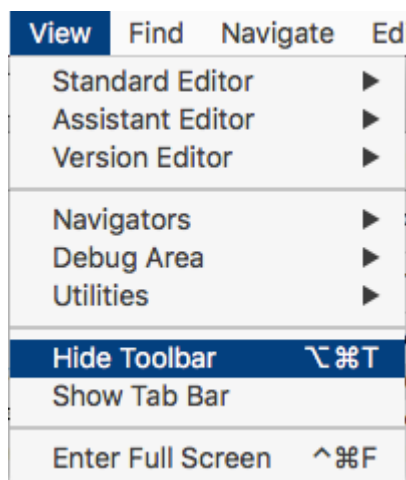
Administrar tareas con la barra de herramientas del área de trabajo

La barra de herramientas en la parte superior de la ventana del área de trabajo proporciona un acceso rápido a los comandos de uso frecuente. El **botón Ejecutar** construye y ejecuta sus productos. El **botón Detener** termina su código de ejecución. El **menú Esquema** le permite configurar los productos que desea compilar y ejecutar. El **visor de actividad** muestra el progreso de las tareas que se están ejecutando actualmente mostrando mensajes de estado, progreso de compilación y otra información sobre su proyecto.

Los **botones de configuración del editor** (el primer grupo de tres botones) le permiten configurar el área del editor, y los **botones de configuración del área de trabajo** (el segundo grupo de tres botones) ocultan o muestran las áreas opcionales de navegador, depuración y utilidades.



El **menú Ver** incluye comandos para ocultar o mostrar la barra de herramientas.



Crea tu primer programa en Swift 3

Aquí les presento cómo crear el primer programa básico en el lenguaje Swift 3. Primero necesita tener conocimientos básicos de lenguaje de programación o no tener que estar listo para aprenderlo desde el principio.

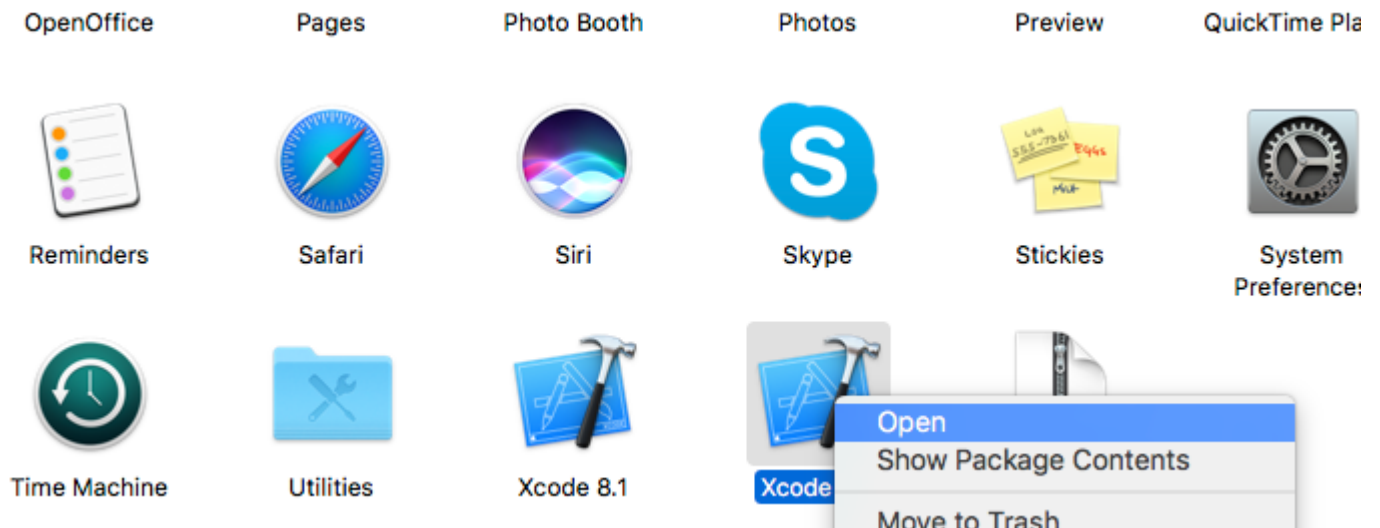
Requisitos para desarrollos:

1. MAC OS - Versión 10.11.6 o posterior para el nuevo Xcode 8.2
2. Xcode - Versión 8.2 [Documento de Apple para la introducción de Xcode.](#)

Xcode 8.2 tiene nuevas funciones de lenguaje Swift 3 con nuevas API compatibles con iOS 10.

Crea tu primer programa

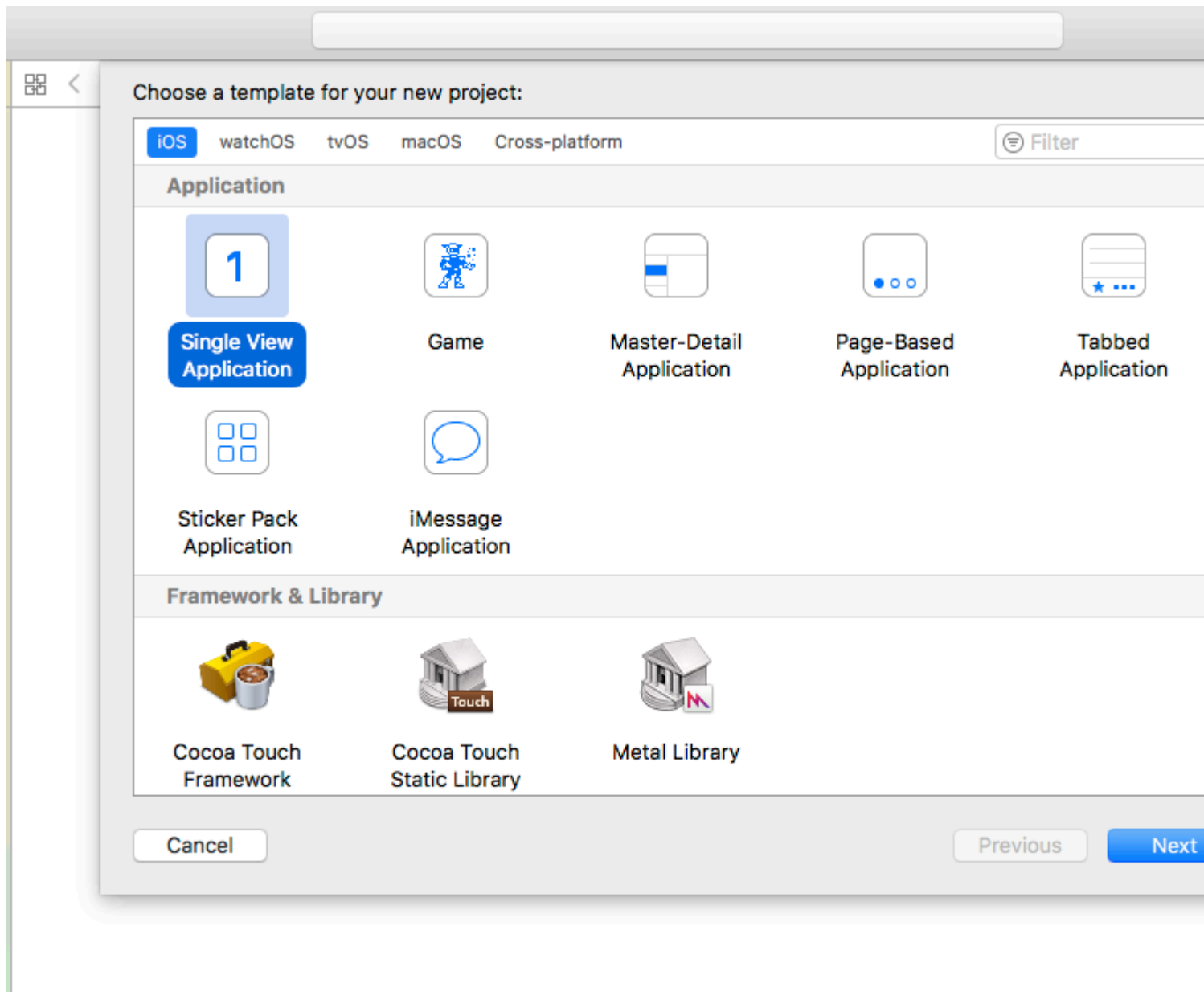
Primero ve a Aplicación y abre tu Xcode 8.2.



Después de eso verás la pantalla.



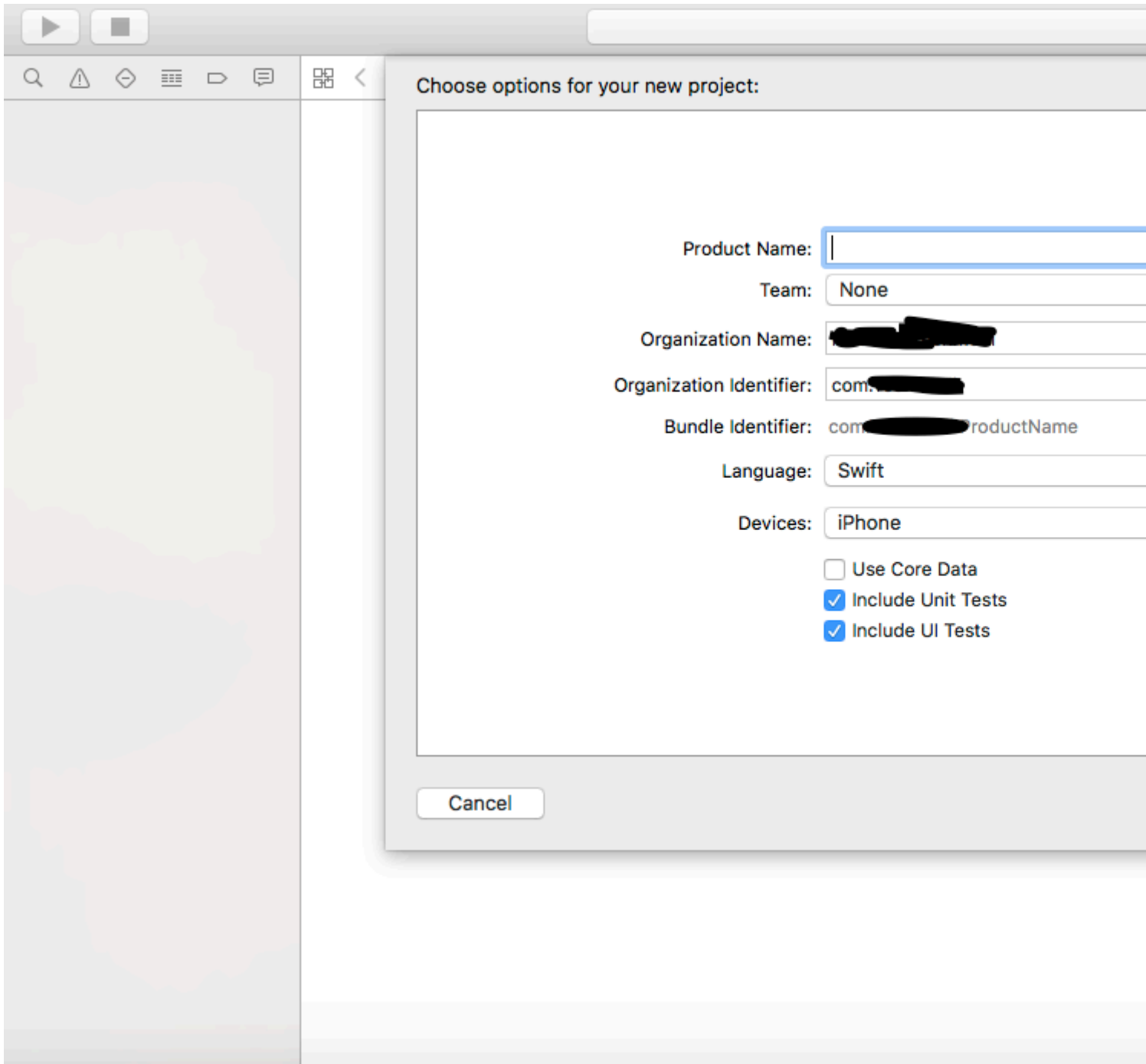
Luego elija Crear nuevo proyecto y después verá la siguiente pantalla.



Esta es también una parte muy importante dentro de Xcode para seleccionar nuestro tipo de proyecto. Tenemos que elegir nuestro proyecto de acuerdo a los tipos de sistema operativo. Hay cinco tipos de opciones disponibles en la parte superior:

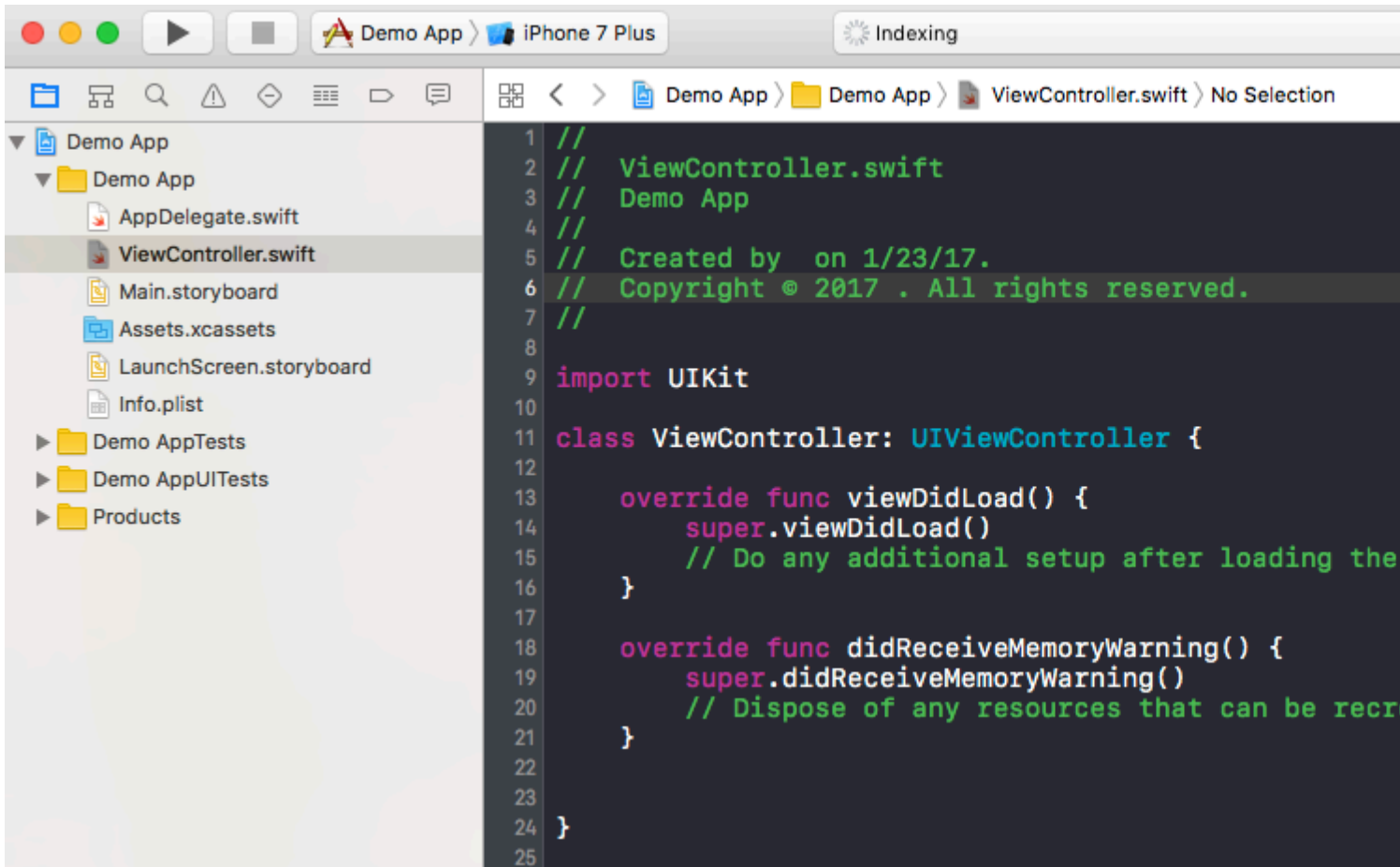
1. [iOS](#)
2. [watchOS](#)
3. [Mac OS](#)
4. Multiplataforma

Ahora estamos eligiendo la plataforma iOS para el desarrollo y creando un proyecto muy básico con la opción de aplicación de vista única:



Luego debemos proporcionar el Nombre del producto, que representará el nombre de su paquete y el nombre de la aplicación.

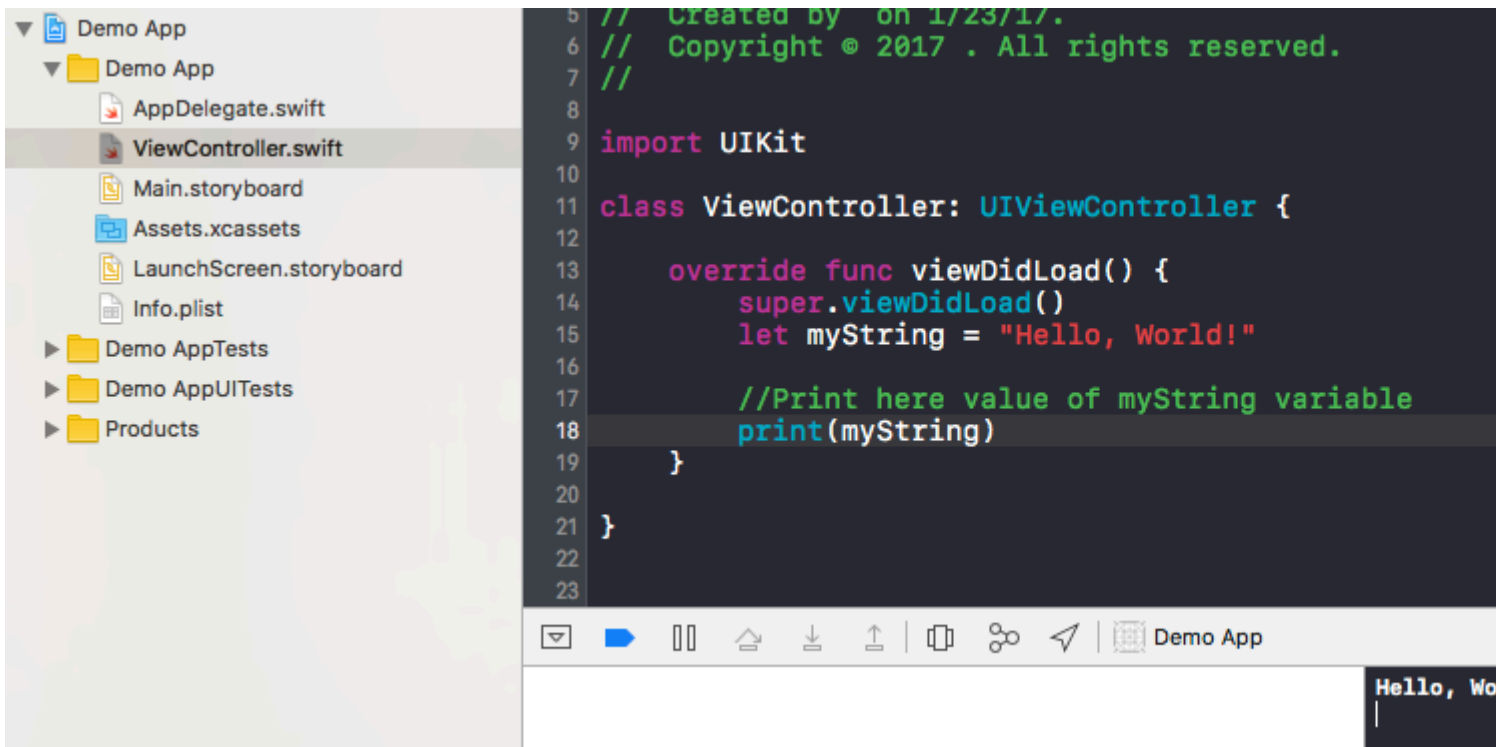
Nombre de la aplicación que puede cambiar más tarde según sus requisitos. Luego debemos hacer clic en "Crear" y después de eso, su pantalla se verá así:



Dentro de esta clase, puede ver que el nombre del archivo es ViewController.swift y dentro de la clase el nombre también es ViewController, que es la herencia de la súper clase UIViewController y finalmente estamos creando nuestra primera variable cuyo nombre es **myString** del tipo 'String'. Agregue lo siguiente en 'super.viewDidLoad ()'

```
let myString = "Hello, World!"
```

Vamos a imprimir el contenido de esta variable. Primero, seleccione su tipo de simulador en la parte superior izquierda de la pantalla y luego haga clic en el botón "Ejecutar".



Después de eso, su salida se mostrará en el terminal que se encuentra en la parte inferior derecha. Enhorabuena, este es tu primer programa Hello World dentro de Xcode.

Lea Empezando con iOS en línea: <https://riptutorial.com/es/ios/topic/191/empezando-con-ios>

Capítulo 2: Abrazar contenido / Compresión de contenido en Autolayout

Observaciones

Resistencia a la compresión de contenido Prioridad

Este valor determina qué tan resistente es una vista a ser comprimida o reducida. Un valor más alto aquí significa que la vista será menos probable que se comprima y que se mantenga igual.

Contenido que abraza la prioridad

Este valor determina la resistencia de una vista a expandirse. Puede imaginar que "abrazar" aquí significa "tamaño para adaptarse": los límites de la vista se "abrazarán" o estarán cerca del tamaño del contenido intrínseco. Un valor más alto aquí significa que la vista tendrá menos probabilidades de crecer y más probabilidades de permanecer igual.

Examples

Definición: Tamaño del contenido intrínseco

Antes de Auto Layout, siempre tenía que decirle a los botones y otros controles qué tan grandes debían ser, ya sea configurando sus propiedades de marco o límites o redimensionándolos en Interface Builder. Pero resulta que la mayoría de los controles son perfectamente capaces de determinar cuánto espacio necesitan, en función de su contenido.

Una **etiqueta** sabe qué ancho y alto es porque sabe la longitud del texto que se ha establecido en ella, así como el tamaño de la fuente para ese texto. Del mismo modo para un **botón**, que podría combinar el texto con una imagen de fondo y algunos rellenos.

Lo mismo ocurre con los controles segmentados, las barras de progreso y la mayoría de los otros controles, aunque algunos pueden tener solo una altura predeterminada pero un ancho desconocido.

Esto se conoce como el tamaño del contenido intrínseco, y es un concepto importante en el diseño automático. Auto Layout le pregunta a sus controles qué tan grandes deben ser y coloca la pantalla en función de esa información.

Por lo general, desea utilizar el `intrinsic content size`, pero hay algunos casos en los que tal vez no quiera hacerlo. Puede evitar esto estableciendo una restricción de Ancho o Altura explícita en un control.

Imagine lo que sucede cuando configura una imagen en un `UIImageView` si esa imagen es mucho

más grande que la pantalla. Por lo general, desea asignar a las vistas de la imagen un ancho y una altura fijos y escalar el contenido, a menos que desee que la vista cambie de tamaño a las dimensiones de la imagen.

Referencia: <https://www.raywenderlich.com/115444/auto-layout-tutorial-in-ios-9-part-2-constraints>

Lea [Abrazar contenido / Compresión de contenido en Autolayout en línea](https://riptutorial.com/es/ios/topic/6899/abrazar-contenido---compresion-de-contenido-en-autolayout):

<https://riptutorial.com/es/ios/topic/6899/abrazar-contenido---compresion-de-contenido-en-autolayout>

Capítulo 3: Accesibilidad

Introducción

La accesibilidad en iOS permite a los usuarios con discapacidades auditivas y discapacidades visuales acceder a iOS y a su aplicación al admitir varias funciones como VoiceOver, Control de voz, Blanco sobre negro, Audio mono, Discurso a texto, etc. Proporcionar accesibilidad en la aplicación iOS significa hacer que la aplicación sea utilizable para todos.

Examples

Hacer una vista accesible

Marque su subclase `UIView` como un elemento accesible para que sea visible para VoiceOver.

```
myView.isAccessibilityElement = YES;
```

Asegúrese de que la vista exprese una etiqueta, un valor y una sugerencia significativos. Apple proporciona más detalles sobre cómo elegir buenas descripciones en la [Guía de programación de accesibilidad](#).

Marco de accesibilidad

El marco de accesibilidad es utilizado por VoiceOver para los toques de prueba de impacto, dibujando el cursor de VoiceOver y calculando en qué parte del elemento enfocado simula un toque cuando el usuario toca la pantalla dos veces. Tenga en cuenta que el marco está en coordenadas de pantalla!

```
myElement.accessibilityFrame = frameInScreenCoordinates;
```

Si sus elementos o diseños de pantalla cambian a menudo, considere anular `accessibilityFrame` para proporcionar siempre un `rect` actualizado. El cálculo del marco relativo a la pantalla de las subvistas de vista de desplazamiento puede ser propenso a errores y tedioso. iOS 10 introduce una nueva API para hacer esto más fácil: `accessibilityFrameInContainerSpace`.

Cambio de pantalla

VoiceOver funciona muy bien la mayor parte del tiempo, leyendo en voz alta pantallas llenas de contenido y siguiendo intuitivamente al usuario. Por desgracia, ninguna solución general es perfecta. A veces solo usted, el desarrollador de la aplicación, sabe dónde debe enfocarse VoiceOver para una experiencia de usuario óptima. Afortunadamente, VoiceOver escucha las notificaciones de accesibilidad del sistema para obtener pistas sobre dónde pertenece el foco. Para mover el cursor de VoiceOver manualmente, publique una notificación de cambio en la pantalla de accesibilidad:

```
UIAccessibilityPostNotification(UIAccessibilityScreenChangedNotification, firstElement);
```

Cuando se publica esta notificación, una breve serie de tonos notifica a los usuarios del cambio. El segundo parámetro puede ser el siguiente elemento para enfocar o una cadena que anuncia el cambio. Solo publique una notificación de cambio de pantalla si la experiencia de VoiceOver es mala sin ella y no existe ninguna otra solución. Mover el cursor de VoiceOver es como tocar la pantalla de un usuario vidente. Puede ser molesto y desorientador ser guiado en esa dirección.

Cambio de diseño

En muchos casos, el contenido dentro de una sola pantalla se actualizará con contenido nuevo o diferente. Por ejemplo, imagine un formulario que revela opciones adicionales basadas en la respuesta del usuario a una pregunta anterior. En este caso, una notificación de "cambio de diseño" le permite anunciar el cambio o enfocarse en un nuevo elemento. Esta notificación acepta los mismos parámetros que la notificación de cambio de pantalla.

```
UIAccessibilityPostNotification(UIAccessibilityLayoutChangedNotification, firstElement);
```

Anuncio

Los anuncios son útiles para alertar a los usuarios sobre eventos que no requieren ninguna interacción, como "pantalla bloqueada" o "carga finalizada". Use un anuncio más específico para notificar a los usuarios sobre cambios en la pantalla o cambios menores en el diseño.

```
UIAccessibilityPostNotification(UIAccessibilityAnnouncementNotification, @"The thing happened!");
```

Elementos de pedido

VoiceOver navega desde la parte superior izquierda a la inferior derecha, independientemente de la jerarquía de vistas. Esta es generalmente la forma en que el contenido se organiza en los idiomas de izquierda a derecha, ya que las personas videntes tienden a escanear la pantalla en un "patrón en forma de F". Los usuarios de VoiceOver esperarán navegar de la misma manera que los usuarios típicos. La previsibilidad y la consistencia son muy importantes para la accesibilidad. Evite realizar personalizaciones que "mejoren" el comportamiento predeterminado (por ejemplo, ordenar primero la barra de pestañas en el orden de deslizamiento). Dicho esto, si ha recibido comentarios de que el orden de los elementos en su aplicación es sorprendente, hay un par de maneras en que puede mejorar la experiencia.

Si VoiceOver debería leer las subvistas de una vista una después de la siguiente, pero no es así, es posible que deba sugerir a VoiceOver que los elementos contenidos en una sola vista están relacionados. Puede hacerlo configurando `shouldGroupAccessibilityChildren`:

```
myView.shouldGroupAccessibilityChildren = YES;
```

Para admitir estructuras de navegación complejas que abarcan varios contenedores o incluyen

interfaces representadas sin UIKit, considere implementar el protocolo de contenedor en la vista principal.

Contenedor de Accesibilidad

VoiceOver puede navegar muchas aplicaciones en iOS porque la mayoría de UIKit clases de `UIAccessibilityProtocol` implementan `UIAccessibilityProtocol`. Las características que no representan elementos en pantalla utilizando `UIView`, incluidas las aplicaciones que aprovechan Core Graphics o Metal para realizar dibujos, deben describir estos elementos para la accesibilidad. A partir de iOS 8.0, esto puede hacerse asignando una propiedad en `UIView` contenga elementos inaccesibles:

```
myInaccessibleContainerView.accessibilityElements = @[elements, that, should, be, accessible];
```

Cada objeto en la matriz puede ser una instancia de `UIAccessibilityElement` o cualquier otra clase que se adhiera a `UIAccessibilityProtocol`. Los elementos secundarios deben devolverse en el orden en que el usuario debe navegarlos. Como autor de la aplicación, puede usar contenedores de accesibilidad para anular el ordenamiento predeterminado de arriba a abajo a la derecha de la navegación por deslizamiento de VoiceOver. Dado que `UIView` implementa `UIAccessibilityProtocol`, puede combinar instancias de `UIAccessibilityElement` y `UIView` en la misma matriz de elementos de accesibilidad secundarios. Tenga en cuenta que si asigna elementos manualmente, no necesita implementar ningún método de protocolo de accesibilidad dinámico, aunque es posible que deba emitir una notificación de cambio de pantalla para que VoiceOver detecte los elementos.

Vista modal

Las vistas modales captan por completo la atención del usuario hasta que se completa una tarea. iOS aclara esto a los usuarios atenuando y deshabilitando el resto del contenido cuando está visible una vista modal, como una alerta o ventana emergente. Una aplicación que implementa una interfaz modal personalizada debe sugerir a VoiceOver que esta vista merece la atención del usuario al configurar `accessibilityViewIsModal`. Tenga en cuenta que esta propiedad solo debe establecerse en la vista que contenga contenido modal, no elementos contenidos dentro de una vista modal.

```
myModalView.accessibilityViewIsModal = YES;
```

Etiquetar una vista como modal anima a VoiceOver a ignorar las vistas de hermanos. Si, después de configurar esta propiedad, encuentra que VoiceOver aún navega por otros elementos en su aplicación, intente ocultar las vistas de problemas hasta que se elimine el modal.

Elementos ocultos

La mayoría de las clases de UIKit, incluida `UIView`, se adhieren a `UIAccessibilityProtocol` y devuelven los valores correctos de forma predeterminada. Es fácil dar por sentado que un conjunto `UIView` en oculto también está ausente de la jerarquía de accesibilidad y no será

navegado por VoiceOver. Si bien este comportamiento predeterminado suele ser suficiente, hay ocasiones en que una vista estará presente en la jerarquía de vistas pero no será visible o navegable. Por ejemplo, una colección de botones puede ser superpuesta por otra vista, haciéndolos invisibles para un usuario vidente. Sin embargo, VoiceOver todavía intentará navegarlos ya que técnicamente no están ocultos de `UIKit` y, por lo tanto, aún están presentes en la jerarquía de accesibilidad. En tales casos, debe sugerir a VoiceOver que la vista principal no es accesible. Puede hacerlo ocultando explícitamente la vista de `UIKit` configurando la opción `oculta` cuando la vista se sale de la pantalla:

```
myViewFullofButtons.hidden = YES;
```

Alternativamente, puede dejar la vista principal visible y simplemente ocultar sus elementos secundarios de la jerarquía de accesibilidad:

```
myViewFullofButtons.accessibilityElementsHidden = YES;
```

Las vistas temporales son otro lugar donde querrá ocultar elementos de la jerarquía de accesibilidad y dejarlos visibles para los usuarios. Por ejemplo, la vista que aparece cuando presiona el botón de volumen es visible para los usuarios videntes, pero no exige atención como lo hace una alerta normal. No querrá que VoiceOver interrumpa al usuario y mueva el cursor lejos de lo que estaba haciendo para anunciar el nuevo volumen, especialmente dado que el ajuste del volumen ya proporciona retroalimentación auditiva a través del sonido de clic. En casos como este, querrá ocultar la vista usando `accessibilityElementsHidden`.

Lea **Accesibilidad en línea**: <https://riptutorial.com/es/ios/topic/773/accesibilidad>

Capítulo 4: Actualizando dinámicamente un UIViewStackView

Examples

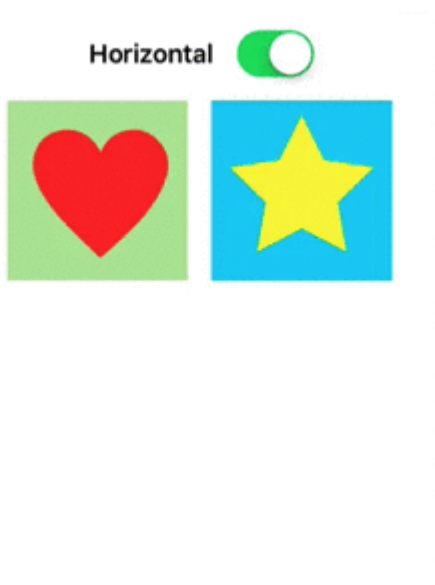
Conecte el UISwitch a una acción que podamos animar cambiando entre un diseño horizontal o vertical de las vistas de la imagen.

```
@IBAction func axisChange(sender: UISwitch) {
    UIView.animateWithDuration(1.0) {
        self.updateConstraintsForAxis()
    }
}
```

La función updateConstraintForAxis solo establece el eje de la vista de pila que contiene las dos vistas de imagen:

```
private func updateConstraintsForAxis() {
    if (axisSwitch.on) {
        stackView.axis = .Horizontal
    } else {
        stackView.axis = .Vertical
    }
}
```

El siguiente gif animado te da una idea de cómo aparece esto:



Lea [Actualizando dinámicamente un UIViewStackView en línea](#):

Capítulo 5: Alamofire

Sintaxis

- respuesta()
- responseData ()
- responseString (codificación: NSStringEncoding)
- responseJSON (opciones: NSJSONReadingOptions)
- responsePropertyList (opciones: NSPropertyListReadOptions)

Parámetros

Parámetro	Detalles
Método	.OPTIONS, .GET, .HEAD, .POST, .PUT, .PATCH, .DELETE, .TRACE, .CONNECT
URLString	URLStringConvertible
parámetros	[String: AnyObject]?
codificación	ParameterEncoding
encabezados	[String: String]?

Examples

Hacer una solicitud

```
import Alamofire

Alamofire.request(.GET, "https://httpbin.org/get")
```

Validacion automatica

```
Alamofire.request("https://httpbin.org/get").validate().responseJSON { response in
switch response.result {
case .success:
    print("Validation Successful")
case .failure(let error):
    print(error)
}
}
```

Manejo de respuesta

```

Alamofire.request(.GET, "https://httpbin.org/get", parameters: ["foo": "bar"])
    .responseJSON { response in
        print(response.request) // original URL request
        print(response.response) // URL response
        print(response.data) // server data
        print(response.result) // result of response serialization

        if let JSON = response.result.value {
            print("JSON: \(JSON)")
        }
    }

```

Validacion manual

```

Alamofire.request(.GET, "https://httpbin.org/get", parameters: ["foo": "bar"])
    .validate(statusCode: 200..<300)
    .validate(contentType: ["application/json"])
    .response { response in
        print(response)
    }

```

Manejador de respuestas

```

Alamofire.request(.GET, "https://httpbin.org/get", parameters: ["foo": "bar"])
    .validate()
    .response { request, response, data, error in
        print(request)
        print(response)
        print(data)
        print(error)
    }

```

Manejadores de respuesta encadenados

```

Alamofire.request(.GET, "https://httpbin.org/get")
    .validate()
    .responseString { response in
        print("Response String: \(response.result.value)")
    }
    .responseJSON { response in
        print("Response JSON: \(response.result.value)")
    }

```

Lea Alamofire en línea: <https://riptutorial.com/es/ios/topic/1823/alamofire>

Capítulo 6: AÑADIR A UN LÍDER DE SWIFT BRIDGING

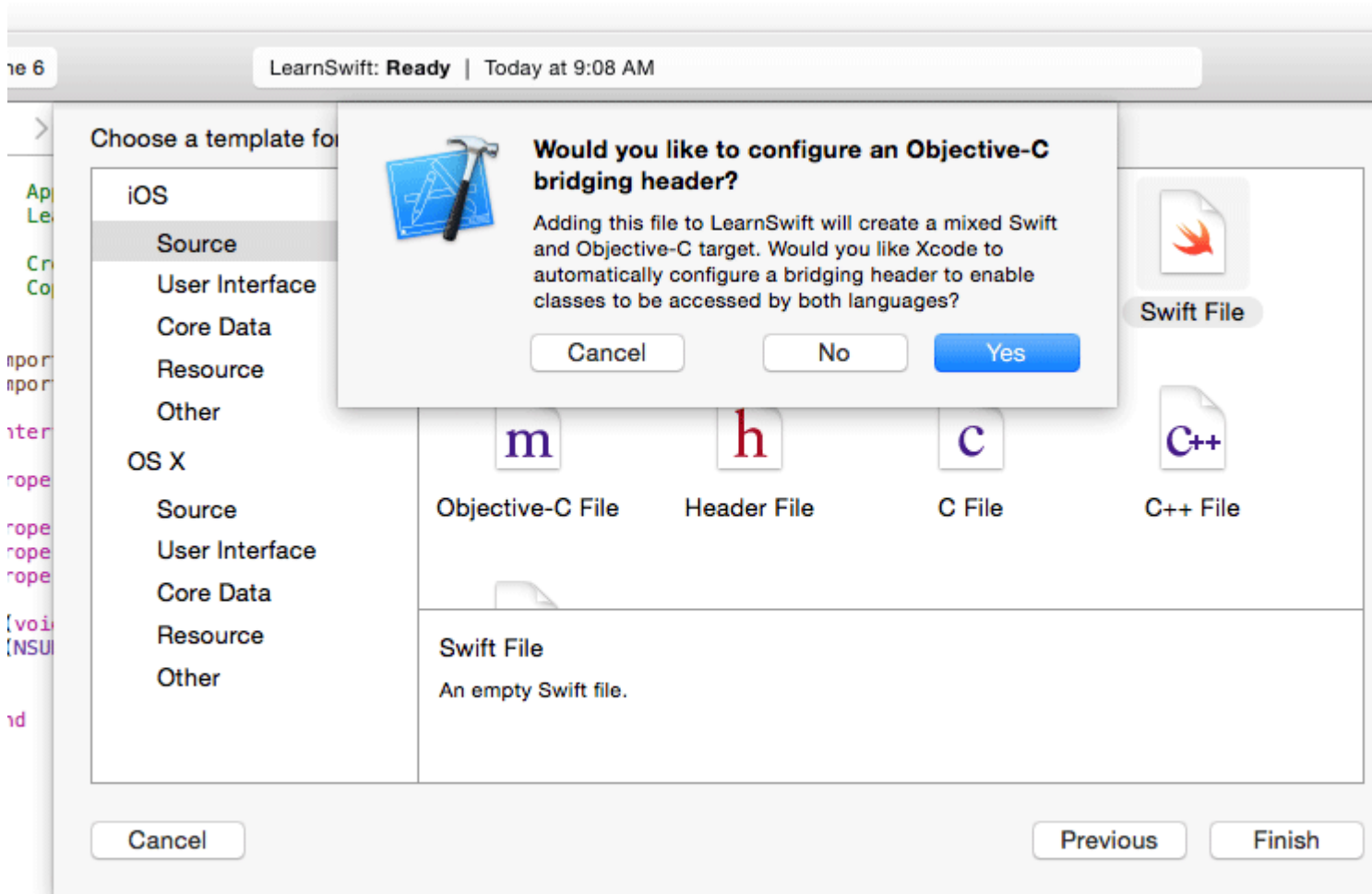
Examples

Cómo crear un encabezado de puente Swift manualmente

- Agregue un nuevo archivo a Xcode (Archivo> Nuevo> Archivo), luego seleccione "Fuente" y haga clic en "Archivo de encabezado".
- Asigne un nombre a su archivo "YourProjectName-Bridging-Header.h". Ejemplo: en mi aplicación Station, el archivo se denomina "Station-Bridging-Header".
- Crea el archivo.
- Vaya a la configuración de compilación de su proyecto y busque la sección "Compilador de Swift - Generación de código". Es posible que encuentre más rápido escribir "Swift Compiler" en el cuadro de búsqueda para reducir los resultados. Nota: Si no tiene una sección de "Compilación de Swift - Generación de código", esto significa que probablemente todavía no haya agregado ninguna clase de Swift a su proyecto. Agrega un archivo Swift, luego intenta de nuevo.
- Junto a "Objective-C Bridging Header" deberá agregar el nombre / ruta de su archivo de encabezado. Si su archivo reside en la carpeta raíz de su proyecto, simplemente coloque el nombre del archivo de encabezado allí. Ejemplos: "ProjectName / ProjectName-Bridging-Header.h" o simplemente "ProjectName-Bridging-Header.h".
- Abra su encabezado de puente recién creado e importe sus clases de Objective-C usando las declaraciones #import. Se podrá acceder a cualquier clase listada en este archivo desde sus clases de swift.

Xcode crea automáticamente

Agrega un nuevo archivo Swift a tu proyecto Xcode. Nómbralo como desee y debería obtener un cuadro de alerta que le pregunta si desea crear un encabezado puente. Nota: Si no recibe un mensaje para agregar un encabezado puente, probablemente rechazó este mensaje una vez antes y tendrá que agregarlo manualmente (ver más abajo)



Lea **AÑADIR A UN LÍDER DE SWIFT BRIDGING** en línea:
<https://riptutorial.com/es/ios/topic/10851/anadir-a-un-lider-de-swift-bridging>

Capítulo 7: Aparición de la UIA

Examples

Establecer apariencia de todas las instancias de la clase.

Para personalizar la apariencia de todas las instancias de una clase, acceda al proxy de apariencia de la clase deseada. Por ejemplo:

Establecer color de tinte UIButton

Rápido:

```
UIButton.appearance().tintColor = UIColor.greenColor()
```

C objetivo:

```
[UIButton appearance].tintColor = [UIColor greenColor];
```

Establecer color de fondo UIButton

Rápido:

```
UIButton.appearance().backgroundColor = UIColor.blueColor()
```

C objetivo:

```
[UIButton appearance].backgroundColor = [UIColor blueColor];
```

Establecer color de texto UILabel

Rápido:

```
UILabel.appearance().textColor = UIColor.redColor()
```

C objetivo:

```
[UILabel appearance].textColor = [UIColor redColor];
```

Establecer color de fondo UILabel

Rápido:

```
UILabel.appearance().backgroundColor = UIColor.greenColor()
```

C objetivo:

```
[UILabel appearance].backgroundColor = [UIColor greenColor];
```

Establecer color de tinte UINavigationController

Rápido:

```
UINavigationController.appearance().tintColor = UIColor.cyanColor()
```

C objetivo:

```
[UINavigationController appearance].tintColor = [UIColor cyanColor];
```

Establecer color de fondo UINavigationController

Rápido:

```
UINavigationController.appearance().backgroundColor = UIColor.redColor()
```

C objetivo:

```
[UINavigationController appearance].backgroundColor = [UIColor redColor];
```

Apariencia para la clase cuando está contenida en la clase de contenedor

Use `appearanceWhenContainedInInstancesOfClasses:` para personalizar la apariencia para la instancia de una clase cuando está contenida dentro de una instancia de clase de contenedor. Por ejemplo, la personalización de `UILabel` y `backgroundColor` `textColor` dentro de la clase `ViewController` se verá así:

Establecer color de texto UILabel

Rápido:

```
UILabel.appearanceWhenContainedInInstancesOfClasses([ViewController.self]).textColor =  
UIColor.whiteColor()
```

C objetivo:

```
[UILabel appearanceWhenContainedInInstancesOfClasses:@[[ViewController class]]].textColor =  
[UIColor whiteColor];
```

Establecer color de fondo UILabel

Rápido:

```
UILabel.appearanceWhenContainedInInstancesOfClasses([ViewController.self]).backgroundColor =  
UIColor.blueColor()
```


C objetivo:

```
[UILabel appearanceWhenContainedInInstancesOfClasses:@[[ViewController  
class]].backgroundColor = [UIColor blueColor];
```

Lea Aparición de la UIA en línea: <https://riptutorial.com/es/ios/topic/3422/aparicion-de-la-uia>

Capítulo 8: API de Google Places para iOS

Examples

Cómo llegar a lugares cercanos desde la ubicación actual

Prerrequisitos

1. Instala pods en tu proyecto.
2. Instala el SDK de GooglePlaces
3. Servicio de localización activado

Primero necesitamos obtener la ubicación de los usuarios obteniendo su longitud y latitud actuales.

1. Importar GooglePlaces y GooglePlacePicker

```
import GooglePlaces
import GooglePlacePicker
```

2. Agregue el protocolo CLLocationManagerDelegate

```
class ViewController: UIViewController, CLLocationManagerDelegate {
}
```

3. crea tu CLLocationManager ()

```
var currentLocation = CLLocationManager()
```

4. Solicitar autorización

```
currentLocation = CLLocationManager()
currentLocation.requestAlwaysAuthorization()
```

5. Crea un botón para llamar al método GooglePlacePicker

@IBAction func placePickerAction (remitente: AnyObject) {

```
if CLLocationManager.authorizationStatuses() == .AuthorizedAlways {
    let center =
    CLLocationCoordinate2DMake((currentLocation.location?.coordinate.latitude)!,
    (currentLocation.location?.coordinate.longitude)!)
    let northEast = CLLocationCoordinate2DMake(center.latitude + 0.001, center.longitude +
    0.001)
    let southWest = CLLocationCoordinate2DMake(center.latitude - 0.001, center.longitude -
    0.001)
    let viewport = GMSCoordinateBounds(coordinate: northEast, coordinate: southWest)
```

```
let config = GMSPickerConfig(viewport: viewport)
placePicker = GMSPicker(config: config)

placePicker?.pickPlaceWithCallback({ (place: GMSPickerPlace?, error: NSError?) -> Void in
    if let error = error {
        print("Pick Place error: \(error.localizedDescription)")
        return
    }

    if let place = place {
        print("Place name: \(place.name)")
        print("Address: \(place.formattedAddress)")
    } else {
        print("Place name: nil")
        print("Address: nil")
    }
})
}
```

Lea API de Google Places para iOS en línea: <https://riptutorial.com/es/ios/topic/6908/api-de-google-places-para-ios>

Capítulo 9: API de reconocimiento de voz de iOS 10

Examples

Discurso a texto: reconoce el habla de un paquete que contiene grabación de audio

```
//import Speech
//import AVFoundation

// create a text field to show speech output
@IBOutlet weak var transcriptionTextField: UITextView!
// we need this audio player to play audio
var audioPlayer: AVAudioPlayer!

override func viewDidLoad()
{
    super.viewDidLoad()
}

// this function is required to stop audio on audio completion otherwise it will play same
audio again and again
func audioPlayerDidFinishPlaying(_ player: AVAudioPlayer, successfully flag: Bool)
{
    player.stop()
}

// this function is required to get a speech recognizer and after that make and request to
speech recognizer
func requestSpeechAuth()
{
    SFSpeechRecognizer.requestAuthorization { authStatus in
        if authStatus == SFSpeechRecognizerAuthorizationStatus.authorized {
            if let path = Bundle.main.url(forResource: "mpthreetest", withExtension: "m4a") {
                do {
                    let sound = try AVAudioPlayer(contentsOf: path)
                    self.audioPlayer = sound
                    self.audioPlayer.delegate = self
                    sound.play()
                } catch {
                    print("error")
                }
            }

            let recognizer = SFSpeechRecognizer()
            let request = SFSpeechURLRecognitionRequest(url:path)
            recognizer?.recognitionTask(with: request) { (result, error) in
                if let error = error {
                    print("there is a error\(error)")
                } else {
                    // here you are printing out the audio output basically showing it on uitext field
                    self.transcriptionTextField.text =
                    result?.bestTranscription.formattedString
                }
            }
        }
    }
}
```

```
        }
    }
}

// here you are calling requestSpeechAuth function on UIButton press
@IBAction func playButtonPress(_ sender: AnyObject)
{
    requestSpeechAuth()
}
```

Lea API de reconocimiento de voz de iOS 10 en línea: <https://riptutorial.com/es/ios/topic/5986/api-de-reconocimiento-de-voz-de-ios-10>

Capítulo 10: Aplicación de Seguridad de Transporte (ATS)

Parámetros

Parámetro	Detalles
<code>NSAppTransportSecurity</code>	Configurar ATS
<code>NSAllowsArbitraryLoads</code>	Establézcalo en <code>YES</code> para deshabilitar ATS en todas partes. En iOS 10 y versiones posteriores, y macOS 10.12 y versiones posteriores, el valor de esta clave se ignora si alguna de las siguientes claves está presente en el archivo Info.plist de su aplicación: <code>NSAllowsArbitraryLoadsInMedia</code> , <code>NSAllowsArbitraryLoadsInWebContent</code> , <code>NSAllowsLocalNetworking</code>
<code>NSAllowsArbitraryLoadsInMedia</code>	Configúrelo en <code>YES</code> para deshabilitar ATS para los medios cargados mediante las API del marco de AV Foundation. (iOS 10+ , macOS 10.12+)
<code>NSAllowsArbitraryLoadsInWebContent</code>	Establézcalo en <code>YES</code> para deshabilitar ATS en las vistas web de su aplicación (<code>WKWebView</code> , <code>UIWebView</code> , <code>WebView</code>) sin afectar sus conexiones <code>NSURLSession</code> . (iOS 10+ , macOS 10.12+)
<code>NSAllowsLocalNetworking</code>	Establézcalo en <code>YES</code> para deshabilitar las conexiones a dominios no calificados y a dominios locales. (iOS 10+ , macOS 10.12+)
<code>NSExceptionDomains</code>	Configurar excepciones para dominios específicos
<code>NSIncludesSubdomains</code>	Establézcalo en <code>YES</code> para aplicar las excepciones a todos los subdominios del dominio seleccionado.
<code>NSRequiresCertificateTransparency</code>	Establézcalo en <code>YES</code> para requerir que las marcas de tiempo de Transparencia de certificado (CT) válidas, firmadas, de registros de CT conocidos, se presenten para

Parámetro	Detalles
	certificados de servidor (X.509) en un dominio. (iOS 10+ , macOS 10.12+)
<code>NSExceptionAllowsInsecureHTTPLoads</code>	Establézcalo en YES para permitir HTTP en el dominio seleccionado.
<code>NSExceptionRequiresForwardSecrecy</code>	El valor predeterminado es YES ; Establézcalo en NO para deshabilitar el secreto de reenvío y acepte más cifrados.
<code>NSExceptionMinimumTLSVersion</code>	El valor predeterminado es <code>TLSv1.2</code> ; Los valores posibles son: <code>TLSv1.0</code> , <code>TLSv1.1</code> , <code>TLSv1.2</code>
<code>NSThirdPartyExceptionAllowsInsecureHTTPLoads</code>	Similar a <code>NSExceptionAllowsInsecureHTTPLoads</code> , pero para dominios sobre los que no tiene control
<code>NSThirdPartyExceptionRequiresForwardSecrecy</code>	Similar a <code>NSExceptionRequiresForwardSecrecy</code> , pero para dominios sobre los que no tiene control
<code>NSThirdPartyExceptionMinimumTLSVersion</code>	Similar a <code>NSExceptionMinimumTLSVersion</code> , pero para dominios sobre los que no tiene control

Observaciones

La [App Transport Security](#) es una característica de seguridad en iOS y macOS. Evita que las aplicaciones establezcan conexiones no seguras. Por defecto, las aplicaciones solo pueden usar conexiones seguras HTTPS.

Si una aplicación necesita conectarse a un servidor a través de HTTP, las excepciones deben definirse en la `Info.plist` . (ver los ejemplos para más información sobre eso)

Nota: En 2017, Apple hará cumplir ATS. Eso significa que ya no puede cargar aplicaciones que tengan excepciones ATS definidas en la `Info.plist` . Si puede proporcionar buenos argumentos, por qué tiene que usar HTTP, puede ponerse en contacto con Apple y es posible que le permitan definir excepciones. (Fuente: [WWDC 2016 - Sesión 706](#))

Puede encontrar más información sobre la configuración de la Seguridad de transporte de la aplicación en la [Documentación de CocoaKeys](#) .

Examples

Cargar todo el contenido HTTP

Apple presentó ATS con iOS 9 como una nueva característica de seguridad para mejorar la privacidad y la seguridad entre las aplicaciones y los servicios web. ATS por defecto falla todas las solicitudes que no son HTTPS. Si bien esto puede ser realmente bueno para los entornos de producción, puede ser una molestia durante las pruebas.

ATS se configura en el archivo `Info.plist` del `Info.plist` con el diccionario `NSAppTransportSecurity` (`App Transport Security Settings` en el editor de Xcode `Info.plist`). Para permitir todo el contenido HTTP, agregue el booleano `Allow Arbitrary Loads` (`NSAllowsArbitraryLoads`) y `NSAllowsArbitraryLoads` en `YES`. Esto no se recomienda para aplicaciones de producción, y si se requiere contenido HTTP, se recomienda que se habilite selectivamente en su lugar.

Cargar selectivamente contenido HTTP

Al igual que para habilitar todo el contenido HTTP, toda la configuración se realiza bajo la `App Transport Security Settings` la `App Transport Security Settings`. Agregue el diccionario de `Exception Domains` (`NSExceptionDomains`) a la configuración ATS de nivel superior.

Para cada dominio, agregue un elemento del diccionario a los Dominios de excepción, donde la clave es el dominio en cuestión. Establezca `NSExceptionAllowsInsecureHTTPLoads` en `YES` para deshabilitar el requisito de HTTPS para ese dominio.

Los puntos finales requieren SSL

Introducido en iOS 9, todos los puntos finales deben cumplir con la especificación HTTPS. Cualquier punto final que no use SSL fallará con una advertencia en el registro de la consola. A su aplicación le parecerá que la conexión a internet falló.

Para configurar excepciones: coloque lo siguiente en su archivo `Info.plist`:

1. Permitir dominio particular (testdomain.com) **solamente**:

```
<key>NSAppTransportSecurity</key>
<dict>
<key>NSExceptionDomains</key>
<dict>
  <key>testdomain.com</key>
  <dict>
    <key>NSIncludesSubdomains</key>
    <true/>
    <key>NSExceptionAllowsInsecureHTTPLoads</key>
    <true/>
  </dict>
</dict>
</dict>
```

La clave que permite tal comportamiento es `NSExceptionAllowsInsecureHTTPLoads`. En este caso, la aplicación solo permitirá la conexión HTTP al dominio mencionado (testdomain.com) y bloqueará todas las demás conexiones HTTP.

La clave `NSIncludesSubdomains` especifica que también se debe permitir cualquier **subdominio** del dominio mencionado (testdomain.com).

2. Permitir cualquier dominio:

```
<key>NSAppTransportSecurity</key>
<dict>
  <key>NSAllowsArbitraryLoads</key>
  <true/>
</dict>
```

En este caso, la aplicación permitirá la conexión HTTP a **cualquier** dominio. A partir del 1 de enero de 2017, el uso de este indicador causará una revisión exhaustiva de la App Store y los desarrolladores de aplicaciones tendrán que explicar por qué necesitan usar esta excepción en primer lugar. Las posibles explicaciones incluyen:

- Una aplicación que carga contenido multimedia encriptado que no contiene información personalizada.
- Conexiones a dispositivos que no pueden actualizarse para usar conexiones seguras.
- Conexión a un servidor que es administrado por otra entidad y no admite conexiones seguras.

Lea Aplicación de Seguridad de Transporte (ATS) en línea:

<https://riptutorial.com/es/ios/topic/5435/aplicacion-de-seguridad-de-transporte--ats->

Capítulo 11: Aplicación en la compra

Examples

IAP individual en Swift 2

Después de crear un IAP en iTunesConnect:

En el controlador de vista que desea comprar

```
import StoreKit
```

y agregar los delegados relevantes

```
class ViewController: UIViewController, SKProductsRequestDelegate, SKPaymentTransactionObserver {
```

declarar una variable con el ID de producto de iTunesConnect

```
var product_id: NSString?

override func viewDidLoad() {

    product_id = "YOUR_PRODUCT_ID"
    super.viewDidLoad()
    SKPaymentQueue.defaultQueue().addTransactionObserver(self)

    //Check if product is purchased
    if (NSUserDefaults.standardUserDefaults().boolForKey("purchased")){

        // Hide ads
        adView.hidden = true

    } else {
        print("Should show ads...")
    }

}
```

cablear un botón a una función para comprar el IAP

```
@IBAction func unlockAction(sender: AnyObject) {

    print("About to fetch the product...")

    // Can make payments
    if (SKPaymentQueue.canMakePayments())
    {
        let productID:NSSet = NSSet(object: self.product_id!);
        let productsRequest:SKProductsRequest = SKProductsRequest(productIdentifiers:
```

```

productID as! Set<NSString>);
    productsRequest.delegate = self;
    productsRequest.start();
    println("Fetching Products");
} else {
    print("Can't make purchases");
}
}

```

Y aquí hay algunos métodos de ayuda.

```

func buyProduct(product: SKProduct) {
    println("Sending the Payment Request to Apple");
    let payment = SKPayment(product: product)
    SKPaymentQueue.defaultQueue().addPayment(payment);
}

```

Los métodos de delegado que deben ser declarados.

```

func productsRequest (request: SKProductsRequest, didReceiveResponse response:
SKProductsResponse) {

    let count : Int = response.products.count
    if (count>0) {
        var validProduct: SKProduct = response.products[0] as SKProduct
        if (validProduct.productIdentifier == self.product_id) {
            print(validProduct.localizedTitle)
            print(validProduct.localizedDescription)
            print(validProduct.price)
            buyProduct(validProduct);
        } else {
            print(validProduct.productIdentifier)
        }
    } else {
        print("nothing")
    }
}

func request(request: SKRequest!, didFailWithError error: NSError!) {
    print("Error Fetching product information");
}

func paymentQueue(_ queue: SKPaymentQueue,
updatedTransactions transactions: [SKPaymentTransaction])
{
    print("Received Payment Transaction Response from Apple");

    for transaction:AnyObject in transactions {
        if let trans:SKPaymentTransaction = transaction as? SKPaymentTransaction{
            switch trans.transactionState {
                case .Purchased:
                    print("Product Purchased");
                    SKPaymentQueue.defaultQueue().finishTransaction(transaction as!
SKPaymentTransaction)
                    // Handle the purchase

```

```

        NSUserDefaults.standardUserDefaults().setBool(true , forKey: "purchased")
        adView.hidden = true
        break;
    case .Failed:
        print("Purchased Failed");
        SKPaymentQueue.defaultQueue().finishTransaction(transaction as!
SKPaymentTransaction)
        break;

    case .Restored:
        print("Already Purchased");
        SKPaymentQueue.defaultQueue().restoreCompletedTransactions()

        // Handle the purchase
        NSUserDefaults.standardUserDefaults().setBool(true , forKey: "purchased")
        adView.hidden = true
        break;
    default:
        break;
    }
}
}
}
}

```

Y luego el código para restaurar un no consumible en la compra de la aplicación.

```

if (SKPaymentQueue.canMakePayments()) {
    SKPaymentQueue.defaultQueue().restoreCompletedTransactions()
}

```

Configuración en iTunesConnect

En [iTunesConnect](#) , seleccione la aplicación a la que desea agregar un IAP.

Haga clic en las características y verá esto:

In-App Purchases (0)

Clic

Haga clic en el signo más. A continuación, deberá seleccionar el tipo de IAP que desea realizar.

Luego deberá completar toda la información de su IAP.

In-App Purchase Summary

Enter a reference name and a product ID for this In-App Purchase.

Reference Name

Product ID

Pricing and Availability

Enter the pricing and availability details for this In-App Purchase.

Cleared for Sale

Price

<https://riptutorial.com/es/ios/topic/2549/aplicacion-en-la-compra>

Capítulo 12: AppDelegate

Introducción

AppDelegate es un protocolo que define métodos que son llamados por el objeto de UIA singleton UIA en respuesta a eventos importantes en la vida de una aplicación.

Normalmente se utiliza para realizar tareas en el inicio de la aplicación (configuración del entorno de la aplicación, analytics (ej .: Mixpanel / GoogleAnalytics / Crashlitics), DB stack, etc.) y cierre (ej .: guardar el contexto DB), manejo de solicitudes abiertas de URL y aplicaciones similares Tareas.

Examples

Todos los estados de aplicación a través de métodos AppDelegate

Para recibir actualizaciones o para hacer algo antes de que la aplicación se convierta en un usuario, puede utilizar el método que se encuentra debajo.

AppDidFinishLaunching

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {  
    // Write your code before app launch  
    return YES;  
}
```

Mientras la aplicación entra en primer plano:

```
- (void)applicationWillEnterForeground:(UIApplication *)application {  
    // Called as part of the transition from the background to the active state; here you can  
    undo many of the changes made on entering the background.  
}
```

Al iniciar la aplicación y también al fondo para el método de golpe de primer plano:

```
- (void)applicationDidBecomeActive:(UIApplication *)application {  
    // Restart any tasks that were paused (or not yet started) while the application was  
    inactive. If the application was previously in the background, optionally refresh the user  
    interface.  
}
```

Mientras la aplicación ingrese en segundo plano:

```
- (void)applicationDidEnterBackground:(UIApplication *)application {  
    // Use this method to release shared resources, save user data, invalidate timers, and  
    store enough application state information to restore your application to its current state in  
    case it is terminated later.
```



```
// If your application supports background execution, this method is called instead of
applicationWillTerminate: when the user quits.
}
```

Mientras que la aplicación renuncia activa

```
- (void)applicationWillResignActive:(UIApplication *)application {
    // Sent when the application is about to move from active to inactive state. This can
    occur for certain types of temporary interruptions (such as an incoming phone call or SMS
    message) or when the user quits the application and it begins the transition to the background
    state.
    // Use this method to pause ongoing tasks, disable timers, and invalidate graphics
    rendering callbacks. Games should use this method to pause the game.
}
```

Mientras que la aplicación termina:

```
- (void)applicationWillTerminate:(UIApplication *)application {
    // Called when the application is about to terminate. Save data if appropriate. See also
    applicationDidEnterBackground:.
}
```

Roles de AppDelegate:

- AppDelegate contiene el `startup code` su aplicación.
- Responde a `key changes` en el `state` de su aplicación. Específicamente, responde tanto a las interrupciones temporales como a los cambios en el estado de ejecución de la aplicación, como cuando la aplicación pasa del primer plano al fondo.
- `responds to notifications` originan desde fuera de la aplicación, como notificaciones remotas (también conocidas como notificaciones push), advertencias de poca memoria, notificaciones de finalización de descargas y más.
- Se `determines si la state preservation` y de `restoration` deben producirse y ayuda en el proceso de conservación y restauración, según sea necesario.
- `responds to events` que se dirigen a la aplicación en sí y no son específicos de las vistas o controladores de su aplicación. Puede usarlo para almacenar los objetos de datos centrales de su aplicación o cualquier contenido que no tenga un controlador de vista propietario.

Abrir un recurso especificado por URL

Pide al delegado que abra un recurso especificado por una URL y proporciona un diccionario de opciones de inicio.

Ejemplo de uso:

```
func application(_ app: UIApplication, open url: URL, options: [UIApplicationOpenURLOptionsKey
: Any] = [:]) -> Bool {
    return SomeManager.shared.handle(
        url,
        sourceApplication: options[.sourceApplication] as? String,
        annotation: options[.annotation]
    )
}
```

```
}
```

Manejo de notificaciones locales y remotas

Ejemplo de uso:

```
/* Instance of your custom APNs/local notification manager */  
private var pushManager: AppleNotificationManager!
```

Registro:

```
func application(application: UIApplication, didRegisterUserNotificationSettings  
notificationSettings: UIUserNotificationSettings) {  
    // Called to tell the delegate the types of notifications that can be used to get the  
    user's attention  
    pushManager.didRegisterSettings(notificationSettings)  
}  
  
func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken  
deviceToken: NSData) {  
    // Tells the delegate that the app successfully registered with Apple Push Notification  
    service (APNs)  
    pushManager.didRegisterDeviceToken(deviceToken)  
}  
  
func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError  
error: NSError) {  
    // Sent to the delegate when Apple Push Notification service cannot successfully complete  
    the registration process.  
    pushManager.didFailToRegisterDeviceToken(error)  
}
```

Manejo remoto de notificaciones:

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject  
: AnyObject]) {  
    // Remote notification arrived, there is data to be fetched  
    // Handling it  
    pushManager.handleNotification(userInfo,  
                                   background: application.applicationState == .Background  
    )  
}
```

Manejo de notificaciones locales:

```
func application(application: UIApplication, didReceiveLocalNotification notification:  
UILocalNotification) {  
    pushManager.handleLocalNotification(notification, background: false)  
}
```

Acción de manejo (en desuso):

```
func application(application: UIApplication, handleActionWithIdentifier identifier: String?,  
forRemoteNotification userInfo: [NSObject : AnyObject],
```

```
        completionHandler: () -> Void) {  
    pushManager.handleInteractiveRemoteNotification(userInfo, actionIdentifier: identifier,  
completion: completionHandler)  
}
```

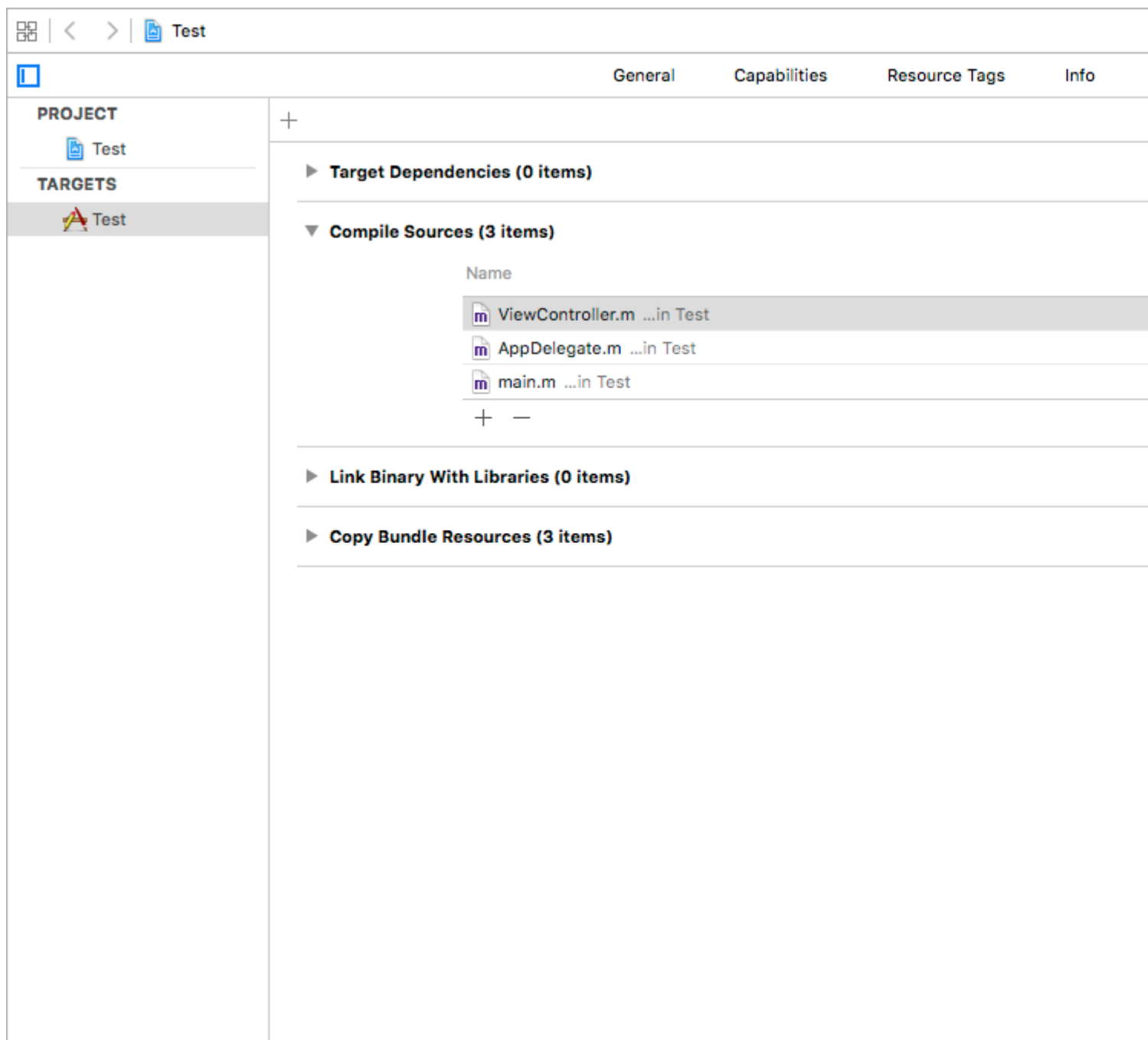
Lea AppDelegate en línea: <https://riptutorial.com/es/ios/topic/8740/appdelegate>

Capítulo 13: ARC (Conteo Automático de Referencia)

Examples

Habilitar / deshabilitar ARC en un archivo

ARC se puede deshabilitar para archivos individuales agregando el `-fno-objc-arc` compilador `-fno-objc-arc` para cada archivo. A la inversa, se puede agregar en *Objetivos* ▶ Fases de compilación Sources Compilar orígenes



Lea ARC (Conteo Automático de Referencia) en línea: [https://riptutorial.com/es/ios/topic/4150/arc-conteo-automatico-de-referencia-](https://riptutorial.com/es/ios/topic/4150/arc-conteo-automatico-de-referencia)

Capítulo 14: Archivo de texto básico de E / S

Examples

Lee y escribe desde la carpeta Documentos

Swift 3

```
import UIKit

// Save String to file
let fileName = "TextFile"
let documentDirectory = try FileManager.default.urlForDirectory(.documentDirectory, in:
.userDomainMask, appropriateFor: nil, create: true)

var fileURL = try
documentDirectory.appendingPathComponent(fileName).appendingPathExtension("txt")

print("FilePath: \(fileURL.path)")

var toFileString = "Text to write"
do {
    // Write to file
    try toFileString.writeToURL(fileURL, atomically: true, encoding: NSUTF8StringEncoding)
} catch let error as NSError {
    print("Failed writing to URL: \(fileURL), Error:\(error.localizedDescription)")
}

// Reading
var fromFileString = ""
do {
    fromFileString = try String(contentsOfURL: fileURL)
} catch let error as NSError {
    print("Failed reading from URL: \(fileURL), Error: " + error.localizedDescription)
}
print("Text input from file: \(fromFileString)")
```

Swift 2

```
import UIKit

// Save String to file
let fileName = "TextFile"
let DocumentDirectoryURL = try!
NSFileManager.defaultManager().URLForDirectory(.DocumentDirectory, inDomain: .UserDomainMask,
appropriateForURL: nil, create: true)

let fileURL =
DocumentDirectoryURL.URLByAppendingPathComponent(fileName).URLByAppendingPathExtension("txt")
print("FilePath: \(fileURL.path)")

var toFileString = "Text to write"
do {
    // Write to file
    try toFileString.writeToURL(fileURL, atomically: true, encoding: NSUTF8StringEncoding)
```

```
} catch let error as NSError {
    print("Failed writing to URL: \(fileURL), Error:\(error.localizedDescription)")
}

// Reading
var fromFileString = ""
do {
    fromFileString = try String(contentsOfURL: fileURL)
} catch let error as NSError {
    print("Failed reading from URL: \(fileURL), Error: " + error.localizedDescription)
}
print("Text input from file: \(fromFileString)")
```

Lea Archivo de texto básico de E / S en línea: <https://riptutorial.com/es/ios/topic/8892/archivo-de-texto-basico-de-e---s>

Capítulo 15: Arquitectura MVP

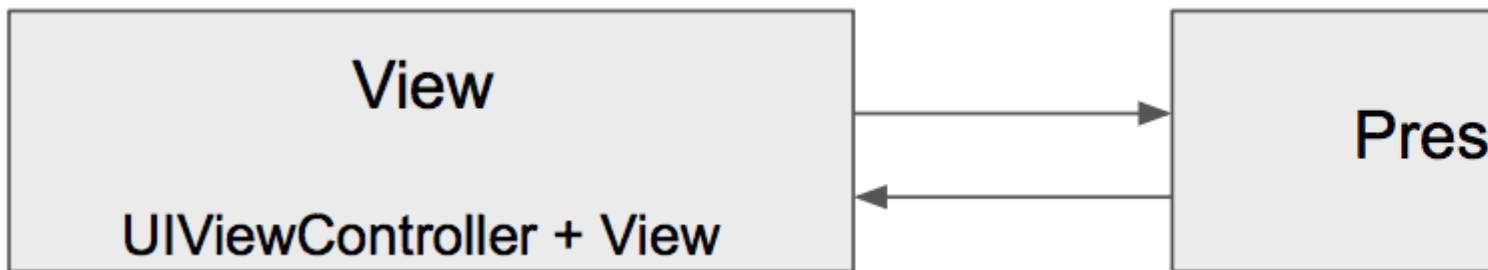
Introducción

MVP es un patrón arquitectónico, una derivación del Modelo – Vista – Controlador. Está representado por tres componentes distintos: Modelo, Vista y el Presentador. Fue diseñado para facilitar las pruebas de unidades automatizadas y mejorar la separación de inquietudes en la lógica de presentación.

En los ejemplos, encontrará un proyecto simple construido con el patrón MVP en mente.

Observaciones

Componentes:



- **El modelo** es una interfaz responsable de los datos del dominio (que se mostrarán o se ejecutarán en la GUI)
- **View** es responsable de la capa de presentación (GUI)
- **Presenter** es el "hombre medio" entre Model y View. Reacciona a las acciones del usuario realizadas en la Vista, recupera datos del Modelo y los formatea para mostrarlos en la Vista.

Deberes componentes:

Modelo	Ver	Presentador
Se comunica con la capa DB	Renders datos	Realiza consultas al modelo.
Levantando eventos apropiados	Recibe eventos	Formatos de datos del modelo
	Lógica de validación muy básica.	Envía datos formateados a la Vista
		Lógica de validación compleja

Diferencias entre MVC y MVP :

- La vista en MVC está estrechamente unida al controlador, la parte de vista del MVP consta de UIViews y UIViewController
- MVP View es lo más tonto posible y casi no contiene lógica (como en MVVM), MVC View tiene cierta lógica de negocios y puede consultar el Modelo
- MVP View maneja los gestos de los usuarios y delega la interacción con el Presentador, en MVC el Controlador maneja los gestos y comandos.
- El patrón MVP es altamente compatible con Unit Testing, MVC tiene soporte limitado
- MVC Controller tiene muchas dependencias de UIKit, MVP Presenter no tiene ninguna

Pros:

- MVP hace que UIViewController sea parte del componente View, es tonto, pasivo y ... menos masivo;]
- La mayor parte de la lógica empresarial está encapsulada debido a las vistas tontas, lo que proporciona una excelente capacidad de prueba. Se pueden introducir objetos simulados para probar la parte del dominio.
- Las entidades separadas son más fáciles de mantener en mente, las responsabilidades están claramente divididas.

Contras

- Escribirás más código.
- Barrera para desarrolladores sin experiencia o para aquellos que aún no trabajan con el patrón.

Examples

Cambio de perro

```
import Foundation

enum Breed: String {
    case bulldog = "Bulldog"
    case doberman = "Doberman"
    case labrador = "Labrador"
}

struct Dog {
    let name: String
    let breed: String
    let age: Int
}
```

DoggyView.swift

```
import Foundation

protocol DoggyView: NSObjectProtocol {
    func startLoading()
    func finishLoading()
}
```

```

func setDoggies(_ doggies: [DoggyViewData])
func setEmpty()
}

```

DoggyService.swift

```

import Foundation

typealias Result = ([Dog]) -> Void

class DoggyService {

    func deliverDoggies(_ result: @escaping Result) {

        let firstDoggy = Dog(name: "Alfred", breed: Breed.labrador.rawValue, age: 1)
        let secondDoggy = Dog(name: "Vinny", breed: Breed.doberman.rawValue, age: 5)
        let thirdDoggy = Dog(name: "Lucky", breed: Breed.labrador.rawValue, age: 3)

        let delay = DispatchTime.now() + Double(Int64(Double(NSEC_PER_SEC)*2)) /
Double(NSEC_PER_SEC)

        DispatchQueue.main.asyncAfter(deadline: delay) {
            result([firstDoggy,
                    secondDoggy,
                    thirdDoggy])
        }
    }
}

```

DoggyPresenter.swift

```

import Foundation

class DoggyPresenter {

    // MARK: - Private
    fileprivate let dogService: DoggyService
    weak fileprivate var dogView: DoggyView?

    init(dogService: DoggyService) {
        self.dogService = dogService
    }

    func attachView(_ attach: Bool, view: DoggyView?) {
        if attach {
            dogView = nil
        } else {
            if let view = view { dogView = view }
        }
    }

    func getDogs() {
        self.dogView?.startLoading()

        dogService.deliverDoggies { [weak self] doggies in
            self?.dogView?.finishLoading()
        }
    }
}

```

```

        if doggies.count == 0 {
            self?.dogView?.setEmpty()
        } else {
            self?.dogView?.setDoggies(doggies.map {
                return DoggyViewData(name: "\($0.name) \($0.breed)",
                                     age: "\($0.age)")
            })
        }
    }
}

struct DoggyViewData {
    let name: String
    let age: String
}

```

DoggyListViewController.swift

```

import UIKit

class DoggyListViewController: UIViewController, UITableViewDataSource {

    @IBOutlet weak var emptyView: UIView?
    @IBOutlet weak var tableView: UITableView?
    @IBOutlet weak var spinner: UIActivityIndicatorView?

    fileprivate let dogPresenter = DoggyPresenter(dogService: DoggyService())
    fileprivate var dogsToDisplay = [DoggyViewData]()

    override func viewDidLoad() {
        super.viewDidLoad()

        tableView?.dataSource = self
        spinner?.hidesWhenStopped = true
        dogPresenter.attachView(true, view: self)
        dogPresenter.getDogs()
    }

    // MARK: DataSource
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return dogsToDisplay.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
    UITableViewCell {
        let cell = UITableViewCell(style: .subtitle, reuseIdentifier: "Cell")
        let userViewData = dogsToDisplay[indexPath.row]
        cell.textLabel?.text = userViewData.name
        cell.detailTextLabel?.text = userViewData.age
        return cell
    }
}

extension DoggyListViewController: DoggyView {

    func startLoading() {
        spinner?.startAnimating()
    }
}

```

```
func finishLoading() {
    spinner?.stopAnimating()
}

func setDoggies(_ doggies: [DoggyViewData]) {
    dogsToDisplay = doggies
    tableView?.isHidden = false
    emptyView?.isHidden = true;
    tableView?.reloadData()
}

func setEmpty() {
    tableView?.isHidden = true
    emptyView?.isHidden = false;
}
}
```

Lea Arquitectura MVP en línea: <https://riptutorial.com/es/ios/topic/9467/arquitectura-mvp>

Capítulo 16: atribuidoTexto en UILabel

Introducción

El texto con estilo actual que se muestra en la etiqueta.

Puede agregar texto HTML en UILabel usando la propiedad attributedText o texto único personalizado de UILabel con diferentes propiedades

Examples

Texto HTML en UILabel

```
NSString * htmlString = @"<html><body> <b> Example bold text in HTML </b> </body></html>";
NSAttributedString * attrStr = [[NSAttributedString alloc] initWithData:[htmlString
dataUsingEncoding:NSUTF8StringEncoding] options:@{ NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType } documentAttributes:nil error:nil];

UILabel * yourLabel = [[UILabel alloc] init];
yourLabel.attributedText = attrStr;
```

Establecer diferentes propiedades para texto en solo UILabel

El primer paso que debe realizar es crear un objeto NSMutableAttributedString . La razón por la que creamos una NSMutableAttributedString lugar de NSAttributedString es porque nos permite NSMutableAttributedString una cadena.

```
NSString *fullStr = @"Hello World!";
NSMutableAttributedString *attString =[[NSMutableAttributedString
alloc]initWithString:fullStr];

// Finding the range of text.
NSRange rangeHello = [fullStr rangeOfString:@"Hello"];
NSRange rangeWorld = [fullStr rangeOfString:@"World!"];

// Add font style for Hello
[attString addAttribute: NSFontAttributeName
                 value: [UIFont fontWithName:@"Copperplate" size:14]
                 range: rangeHello];
// Add text color for Hello
[attString addAttribute: NSForegroundColorAttributeName
                 value: [UIColor blueColor]
                 range: rangeHello];

// Add font style for World!
[attString addAttribute: NSFontAttributeName
                 value: [UIFont fontWithName:@"Chalkduster" size:20]
                 range: rangeWorld];
// Add text color for World!
[attString addAttribute: NSForegroundColorAttributeName
                 value: [UIColor colorWithRed:(66.0/255.0) green:(244.0/255.0)
```

```
blue:(197.0/255.0) alpha:1]
        range: rangeWorld];

// Set it to UILabel as attributedText
UILabel * yourLabel = [[UILabel alloc] initWithFrame:CGRectMake(10, 150, 200, 100)];
yourLabel.attributedText = attString;
[self.view addSubview:yourLabel];
```

Salida:



HELLO World!

Lea atribuidoTexto en UILabel en línea: <https://riptutorial.com/es/ios/topic/10927/atribuidotexto-en-UILabel>

Capítulo 17: AVPlayer y AVPlayerViewController

Observaciones

importar AVKit, importar AVFoundation.

Examples

Reproducción de medios utilizando AVPlayerViewController

C objetivo

```
NSURL *url = [[NSURL alloc] initWithString:@"YOUR URL"]; // url can be remote or local

AVPlayer *player = [AVPlayer playerWithURL:url];
// create a player view controller

AVPlayerViewController *controller = [[AVPlayerViewController alloc] init];
[self presentViewController:controller animated:YES completion:nil];
controller.player = player;
[player play];
```

Rápido

```
let player = AVPlayer(URL: url) // url can be remote or local

let playerViewController = AVPlayerViewController()
// creating a player view controller
playerViewController.player = player
self.presentViewController(playerViewController, animated: true) {

    playerViewController.player!.play()
}
```

Reproducción de medios utilizando AVPlayer y AVPlayerLayer

C objetivo

```
NSURL *url = [NSURL URLWithString:@"YOUR URL"];
AVPlayer *player = [AVPlayer playerWithURL:videoURL];
AVPlayerLayer *playerLayer = [AVPlayerLayer playerLayerWithPlayer:player];
playerLayer.frame = self.view.bounds;
[self.view.layer addSublayer:playerLayer];
[player play];
```

Rápido

```
let url = NSURL(string: "YOUR URL")
let player = AVPlayer(URL: videoURL!)
let playerLayer = AVPlayerLayer(player: player)
playerLayer.frame = self.view.bounds
self.view.layer.addSublayer(playerLayer)
player.play()
```

Ejemplo de AVPlayer

`AVPlayer * avPlayer = [AVPlayer playerWithURL: [NSURL URLWithString: @"YOUR URL"]];`

```
AVPlayerViewController *avPlayerCtrl = [[AVPlayerViewController alloc] init];
avPlayerCtrl.view.frame = self.view.frame;
avPlayerCtrl.player = avPlayer;
avPlayerCtrl.delegate = self;
[avPlayer play];
[self presentViewController:avPlayerCtrl animated:YES completion:nil
```

Lea [AVPlayer y AVPlayerViewController](https://riptutorial.com/es/ios/topic/5092/avplayer-y-avplayerviewController) en línea:

<https://riptutorial.com/es/ios/topic/5092/avplayer-y-avplayerviewController>

Capítulo 18: AWS SDK

Examples

Suba una imagen o un video a S3 usando AWS SDK

Antes de comenzar con el ejemplo, recomendaría crear un Singleton con un miembro de la clase delegado para poder utilizar un archivo en segundo plano y dejar que el usuario siga usando su aplicación mientras los archivos se cargan, incluso cuando la aplicación es el fondo

Comencemos, primero, debemos crear una enumeración que represente la configuración de S3:

```
enum S3Configuration : String
{
    case IDENTITY_POOL_ID = "YourIdentityPoolId"
    case BUCKET_NAME     = "YourBucketName"
    case CALLBACK_KEY    = "YourCustomStringForCallBackWhenUploadingInTheBackground"
    case CONTENT_TYPE_IMAGE = "image/png"
    case CONTENT_TYPE_VIDEO = "video/mp4"
}
```

Ahora, deberíamos establecer las credenciales cuando su aplicación se inicie por primera vez, por lo tanto, debemos configurarlas dentro de `AppDelegate` en el método `didFinishLaunchingWithOptions` (preste atención a que debe establecer su región en el `regionType`):

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool
{
    let credentialProvider = AWSCognitoCredentialsProvider(regionType: .EUWest1, identityPoolId:
S3Configuration.IDENTITY_POOL_ID.rawValue)
    let configuration = AWSServiceConfiguration(region: .EUWest1, credentialsProvider:
credentialProvider)
    AWSS3TransferUtility.registerS3TransferUtilityWithConfiguration(configuration, forKey:
S3Configuration.CALLBACK_KEY.rawValue)
}
```

Como ya estamos dentro de `AppDelegate`, deberíamos implementar la devolución de llamada en segundo plano que maneja el SDK de AWS:

```
func application(application: UIApplication, handleEventsForBackgroundURLSession identifier:
String, completionHandler: () -> Void)
{
    // Will print the identifier you have set at the enum: .CALLBACK_KEY
    print("Identifier: " + identifier)
    // Stores the completion handler.
    AWSS3TransferUtility.interceptApplication(application,
                                             handleEventsForBackgroundURLSession: identifier,
                                             completionHandler: completionHandler)
}
```

Ahora, cuando el usuario mueva la aplicación al fondo, su carga continuará con la carga real.

Para cargar el archivo con el SDK de AWS, tendremos que escribir el archivo en el dispositivo y darle a la SDK la ruta real. Por el bien del ejemplo, imagine que tenemos un UIImage (también podría ser un video ...) y lo escribiremos en una carpeta temporal:

```
// Some image....
let image = UIImage()
let fileURL = NSURL(fileURLWithPath:
NSTemporaryDirectory()).URLByAppendingPathComponent(fileName)
let filePath = fileURL.path!
let imageData = UIImageJPEGRepresentation(image, 1.0)
imageData!.writeToFile(filePath, atomically: true)
```

FileURL y fileName se utilizarán para la carga real más adelante.

Hay 2 cierres que tendremos que definir que son proporcionados por el SDK de AWS,

1. `AWSS3TransferUtilityUploadCompletionHandlerBlock` : un cierre que notifica cuando se realiza la carga (o no)
2. `AWSS3TransferUtilityUploadProgressBlock` : un cierre que notifica cada byte enviado

Si planea tener un Singleton debe definir esos tipos como miembros de la clase. La implementación debería verse así:

```
var completionHandler : AWSS3TransferUtilityUploadCompletionHandlerBlock? =
    { (task, error) -> Void in

        if ((error) != nil)
        {
            print("Upload failed")
        }
        else
        {
            print("File uploaded successfully")
        }
    }

var progressBlock : AWSS3TransferUtilityUploadProgressBlock? =
    { [unowned self] (task, bytesSent: Int64, totalBytesSent: Int64,
totalBytesExpectedToSend: Int64) -> Void in

        let progressInPercentage = Float(Double(totalBytesSent) /
Double(totalBytesExpectedToSend)) * 100
        print(progressInPercentage)
    }
```

NOTA: Si está utilizando un Singleton, es posible que desee definir un delegado que informará sobre el progreso o cuando se complete el archivo. Si no está utilizando un Singleton, puede crear un método estático que tenga los tipos relevantes:

```
static func uploadImageToS3(fileURL : NSURL,
                            fileName : String,
                            progressFunctionUpdater : Float -> Void,
                            resultBlock : (NSError?) -> Void)
{
    // Actual implementation .....
```

```
// ...  
// ...  
}
```

1. `progressFunctionUpdater` : informará a una función con progreso.
2. `resultBlock` : si devuelve nil, la carga se realizó con éxito, envía el objeto de error

Señoras y señores, la carga real:

```
let fileData = NSData(contentsOfFile: fileURL.relativePath!)  
  
let expression = AWSS3TransferUtilityUploadExpression()  
expression.uploadProgress = progressBlock  
  
let transferUtility =  
AWSS3TransferUtility.S3TransferUtilityForKey(S3Configuration.CALLBACK_KEY.rawValue)  
  
transferUtility?.uploadData(fileData!,  
    bucket: S3Configuration.BUCKET_NAME.rawValue,  
    key: fileName,  
    contentType: S3Configuration.CONTENT_TYPE_IMAGE.rawValue,  
    expression: expression,  
    completionHandler: completionHandler).continueWithBlock  
    { (task : AWSTask) -> AnyObject? in  
  
        if let error = task.error  
        {  
            print(error)  
        }  
        if let exception = task.exception  
        {  
            print("Exception: " + exception.description)  
        }  
        if let uploadTask = task.result as? AWSS3TransferUtilityUploadTask  
        {  
            print("Upload started...")  
        }  
  
        return nil  
    }  
}
```

Feliz S3 subiendo :)

Lea AWS SDK en línea: <https://riptutorial.com/es/ios/topic/4734/aws-sdk>

Capítulo 19: Barra de navegación

Examples

Personaliza la apariencia predeterminada de la barra de navegación.

```
// Default UINavigationController appearance throughout the app
[[UINavigationController appearance] setTitleTextAttributes:@{NSForegroundColorAttributeName:
[UIColor whiteColor],
                                                            NSFontAttributeName : [UIFont
fontWithName:@"HelveticaNeue-CondensedBold" size:17],
                                                            }
};

[[UINavigationController appearance] setTintColor:[UIColor whiteColor]];
[[UINavigationController appearance] setBarTintColor:[UIColor KNGRed]];
[[UINavigationController appearance] setTranslucent:NO];
[[UINavigationController appearance] setBarStyle:UIBarStyleBlack];
[[UIBarButtonItem appearanceWhenContainedIn: [UISearchBar class], nil] setTintColor:[UIColor
KNGGray]];
```

Ejemplo de SWIFT

```
navigationController?.navigationBar.titleTextAttributes = [NSForegroundColorAttributeName:
UIColor.white, NSFontAttributeName:UIFont(name: "HelveticaNeue-CondensedBold", size: 17)!,]
navigationController?.navigationBar.tintColor = .white
navigationController?.navigationBar.barTintColor = .red
navigationController?.navigationBar.isTranslucent = false
navigationController?.navigationBar.barStyle = .black
```

Lea Barra de navegación en línea: <https://riptutorial.com/es/ios/topic/7066/barra-de-navegacion>

Capítulo 20: Bloquear

Sintaxis

- Como variable:

```
returnType (^ blockName) (parametersTypes) = ^ returnType (parámetros) {...};
```

- Como propiedad:

```
@property (nonatomic, copy) returnType (^ blockName) (parametersTypes);
```

- Como parámetro del método:

```
-(void) methodWithBlock: (returnType (^) (parametersTypes)) blockName;
```

- Como typedef:

```
typedef returnType (^ TypeName) (parametersTypes);
```

```
TypeName blockName = ^ returnType (parámetros) {...};
```

Examples

Animaciones UIView

```
[UIView animateWithDuration:1.0
 animations:^(
     someView.alpha = 0;
     otherView.alpha = 1;
 )
 completion:^(BOOL finished) {
     [someView removeFromSuperview];
 }];
```

El carácter quilate "^" define un bloque. Por ejemplo, `^{ ... }` es un bloque. Más específicamente, es un bloque que devuelve "vacío" y no acepta argumentos. Es equivalente a un método como "-(anular) algo", pero no hay un nombre inherente asociado con el bloque de código.

Definir un bloque que pueda aceptar argumentos que funcionen de manera muy similar. Para proporcionar un argumento a un bloque, defina el bloque de la siguiente manera: `^(BOOL someArg, NSString someStr) {...}*`. Cuando use llamadas a la API que admiten bloques, escribirá bloques que se parecen a esto, especialmente para bloques de animación o bloques de conexión NSURLConnection, como se muestra en el ejemplo anterior.

Bloque de finalización personalizado para métodos personalizados

1- Define tu propio bloque personalizado

```
typedef void(^myCustomCompletion) (BOOL);
```

2- Crear un método personalizado que tome su bloque de finalización personalizado como un parámetro.

```
-(void) customMethodName:(myCustomCompletion) compblock{  
    //do stuff  
    // check if completion block exist; if we do not check it will throw an exception  
    if(compblock)  
        compblock(YES);  
}
```

3- Cómo usar el bloque en tu método.

```
[self customMethodName:^(BOOL finished) {  
    if(finished){  
        NSLog(@"success");  
    }  
}];
```

Modificar la variable capturada

El bloque capturará variables que aparecieron en el mismo ámbito léxico. Normalmente estas variables se capturan como valor "const":

```
int val = 10;  
void (^blk)(void) = ^{  
    val = 20; // Error! val is a constant value and cannot be modified!  
};
```

Para modificar la variable, debe utilizar el modificador de tipo de almacenamiento `__block`.

```
__block int val = 10;  
void (^blk)(void) = ^{  
    val = 20; // Correct! val now can be modified as an ordinary variable.  
};
```

Lea Bloquear en línea: <https://riptutorial.com/es/ios/topic/6888/bloquear>

Capítulo 21: Bloques de cadena en una cola (con MKBlockQueue)

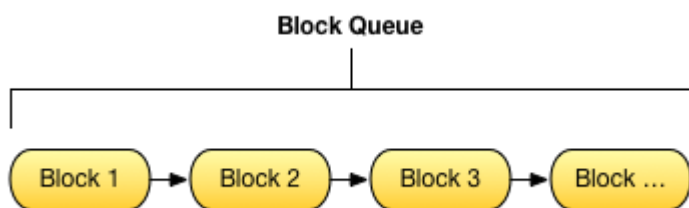
Introducción

MKBlockQueue le permite crear una cadena de bloques y ejecutarlos uno tras otro en una cola. Comparado con NSOperation, con MKBlockQueue usted mismo decide cuándo se completa un bloqueo y cuándo desea que la cola continúe. También puede pasar datos de un bloque al siguiente.

<https://github.com/MKGitHub/MKBlockQueue>

Examples

Código de ejemplo



```
// create the dictionary that will be sent to the blocks
var myDictionary:Dictionary<String, Any> = Dictionary<String, Any>()
myDictionary["InitialKey"] = "InitialValue"

// create block queue
let myBlockQueue:MKBlockQueue = MKBlockQueue()

// block 1
let b1:MKBlockQueueBlockType =
{
    (blockQueueObserver:MKBlockQueueObserver, dictionary:inout Dictionary<String, Any>) in

    print("Block 1 started with dictionary: \(dictionary)")
    dictionary["Block1Key"] = "Block1Value"

    // tell this block is now completed
    blockQueueObserver.blockCompleted(with:&dictionary)
}

// block 2
let b2:MKBlockQueueBlockType =
{
    (blockQueueObserver:MKBlockQueueObserver, dictionary:inout Dictionary<String, Any>) in

    var copyOfDictionary:Dictionary<String, Any> = dictionary
```

```

// test calling on main thread, async, with delay
DispatchQueue.main.asyncAfter(deadline:(.now() + .seconds(1)), execute:
{
    print("Block 2 started with dictionary: \(copyOfDictionary)")

    copyOfDictionary["Block2Key"] = "Block2Value"

    // tell this block is now completed
    blockQueueObserver.blockCompleted(with:&copyOfDictionary)
})
}

// block 3
let b3:MKBlockQueueBlockType =
{
    (blockQueueObserver:MKBlockQueueObserver, dictionary:inout Dictionary<String, Any>) in

    var copyOfDictionary:Dictionary<String, Any> = dictionary

    // test calling on global background queue, async, with delay
    DispatchQueue.global(qos:.background).asyncAfter(deadline:(.now() + .seconds(1)), execute:
    {
        print("Block 3 started with dictionary: \(copyOfDictionary)")

        copyOfDictionary["Block3Key"] = "Block3Value"

        // tell this block is now completed
        blockQueueObserver.blockCompleted(with:&copyOfDictionary)
    })
}

// add blocks to the queue
myBlockQueue.addBlock(b1)
myBlockQueue.addBlock(b2)
myBlockQueue.addBlock(b3)

// add queue completion block for the queue
myBlockQueue.queueCompletedBlock(
{
    (dictionary:Dictionary<String, Any>) in
    print("Queue completed with dictionary: \(dictionary)")
})

// run queue
print("Queue starting with dictionary: \(myDictionary)")
myBlockQueue.run(with:&myDictionary)

```

Lea Bloques de cadena en una cola (con MKBlockQueue) en línea:

<https://riptutorial.com/es/ios/topic/9122/bloques-de-cadena-en-una-cola--con-mkblockqueue->

Capítulo 22: CAAnimación

Observaciones

CAAnimation es una clase de animación abstracta. Proporciona el soporte básico para los protocolos **CAMediaTiming** y **CAAction** . Para animar las capas Core Animation u objetos de Scene Kit, cree instancias de las subclases concretas **CABasicAnimation** , **CAKeyframeAnimation** , **CAAnimationGroup** o **CATransition** .

Examples

Animar una vista de una posición a otra.

C objetivo

```
CABasicAnimation *animation = [CABasicAnimation animationWithKeyPath:@"position.x"];
animation.fromValue = @0;
animation.toValue = @320;
animation.duration = 1;

[_label.layer addAnimation:animation forKey:@"basic"];
```

Rápido

```
let animation = CABasicAnimation(keyPath: "position.x")
animation.fromValue = NSNumber(value: 0.0)
animation.toValue = NSNumber(value: 320.0)

_label.layer.addAnimation(animation, forKey: "basic")
```

La vista se moverá de 0 a 320 horizontalmente. Si desea mover la vista a Vertical, solo reemplace la ruta de acceso de esta manera:

```
"position.y"
```

Vista animada - Lanzamiento

C OBJETIVO

```
CATransition* transition = [CATransition animation];
transition.startProgress = 0;
transition.endProgress = 1.0;
transition.type = @"flip";
```

```
transition.subtype = @"fromLeft";
transition.duration = 0.8;
transition.repeatCount = 5;
[_label.layer addAnimation:transition forKey:@"transition"];
```

RÁPIDO

```
var transition = CATransition()
transition.startProgress = 0
transition.endProgress = 1.0
transition.type = "flip"
transition.subtype = "fromLeft"
transition.duration = 0.8
transition.repeatCount = 5
label.layer.addAnimation(transition, forKey: "transition")
```

Vista de revolución

```
CGRect boundingRect = CGRectMake(-150, -150, 300, 300);

CAKeyframeAnimation *orbit = [CAKeyframeAnimation animation];
orbit.keyPath = @"position";
orbit.path = CFAutorelease(CGPathCreateWithEllipseInRect(boundingRect, NULL));
orbit.duration = 4;
orbit.additive = YES;
orbit.repeatCount = HUGE_VALF;
orbit.calculationMode = kCAAnimationPaced;
orbit.rotationMode = kCAAnimationRotateAuto;

[_label.layer addAnimation:orbit forKey:@"orbit"];
```

Agitar vista

C objetivo

```
CAKeyframeAnimation *animation = [CAKeyframeAnimation animationWithKeyPath:@"position.x"];
animation.values = @[ @0, @10, @-10, @10, @0 ];
animation.keyTimes = @[ @0, @(1 / 6.0), @(3 / 6.0), @(5 / 6.0), @1 ];
animation.duration = 0.4;
animation.additive = YES;
[_label.layer addAnimation:animation forKey:@"shake"];
```

Swift 3

```
let animation = CAKeyframeAnimation(keyPath: "position.x")
animation.values = [ 0, 10, -10, 10, 0 ]
animation.keyTimes = [ 0, NSNumber(value: (1 / 6.0)), NSNumber(value: (3 / 6.0)),
NSNumber(value: (5 / 6.0)), 1 ]
animation.duration = 0.4
animation.isAdditive = true
label.layer.add(animation, forKey: "shake")
```

Animación Push View

C objetivo

```
CATransition *animation = [CATransition animation];  
[animation setSubtype:kCATransitionFromRight];//kCATransitionFromLeft  
[animation setDuration:0.5];  
[animation setType:kCATransitionPush];  
[animation setTimingFunction:[CAMediaTimingFunction  
functionWithName:kCAMediaTimingFunctionEaseInEaseOut]];  
[[yourView layer] addAnimation:animation forKey:@"SwitchToView1"];
```

Rápido

```
let animation = CATransition()  
animation.subtype = kCATransitionFromRight//kCATransitionFromLeft  
animation.duration = 0.5  
animation.type = kCATransitionPush  
animation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionEaseInEaseOut)  
yourView.layer.addAnimation(animation, forKey: "SwitchToView1")
```

Lea CAAnimación en línea: <https://riptutorial.com/es/ios/topic/981/caanimacion>

Capítulo 23: CAGradientLayer

Sintaxis

- `CAGradientLayer ()` // Devuelve un objeto `CALayer` inicializado.
- `CAGradientLayer (layer: layer)` // Override para copiar o inicializar campos personalizados de la capa especificada.

Parámetros

Parámetro	Detalles
color	Una matriz de objetos <code>CGColorRef</code> que definen el color de cada parada de degradado. Animable
localizaciones	Una matriz opcional de objetos <code>NSNumber</code> que definen la ubicación de cada parada de degradado. Animable
punto final	El punto final del degradado cuando se dibuja en el espacio de coordenadas de la capa. Animable
punto de partida	El punto de inicio del degradado cuando se dibuja en el espacio de coordenadas de la capa. Animable
tipo	Estilo de gradiente dibujado por la capa. El valor predeterminado es <code>kCAGradientLayerAxial</code> .

Observaciones

- Utilice `startPoint` y `endPoint` para cambiar la orientación de `CAGradientLayer`.
- Usa las `locations` para afectar la extensión / posiciones de los colores.

Examples

Creando un CAGradientLayer

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds
```

```
// Color at the top of the gradient.
let topColor: CGColor = UIColor.red.cgColor

// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.yellow.cgColor

// Set colors.
gradientLayer.colors = [topColor, bottomColor]

// Set locations of the colors.
gradientLayer.locations = [0.0, 1.0]

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, at: 0)
```

Resultado:



Creando un CAGradientLayer con múltiples colores.

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds

// Color at the top of the gradient.
let topColor: CGColor = UIColor.blue.cgColor

// Color at the middle of the gradient.
let middleColor: CGColor = UIColor.yellow.cgColor

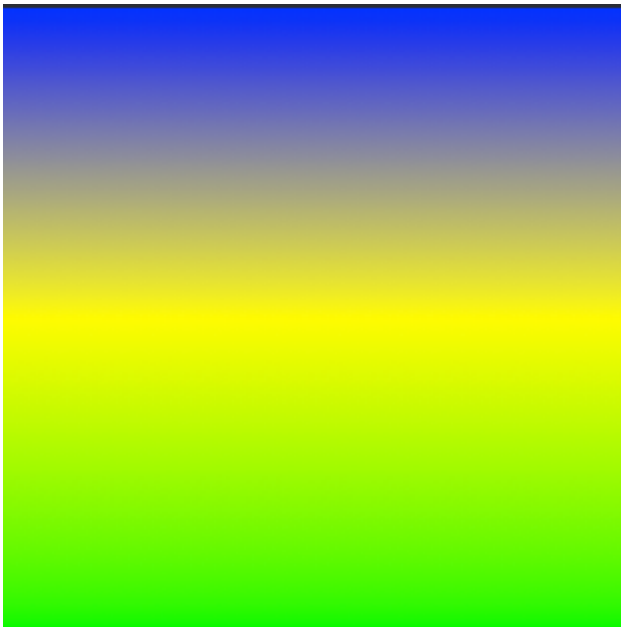
// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.green.cgColor

// Set colors.
gradientLayer.colors = [topColor, middleColor, bottomColor]
```

```
// Set locations of the colors.
gradientLayer.locations = [0.0, 0.5, 1.0]

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, at: 0)
```

Resultado:



Creando un CAGradientLayer horizontal.

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds

// Color at the top of the gradient.
let topColor: CGColor = UIColor.redColor().CGColor

// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.yellowColor().CGColor

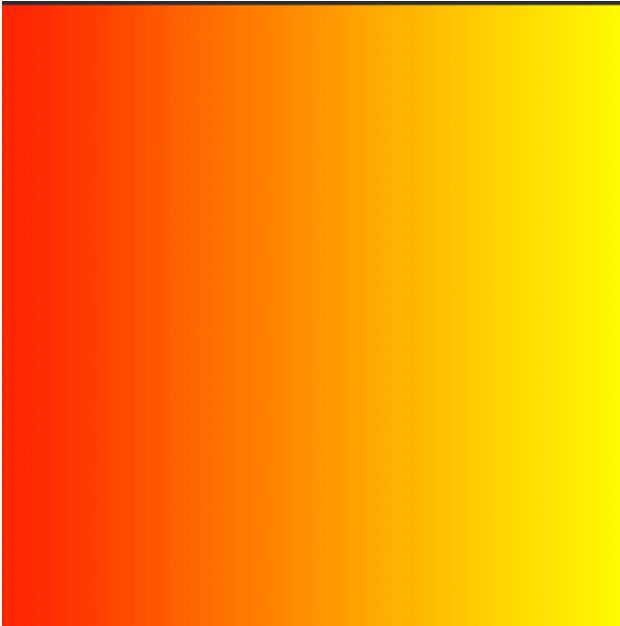
// Set colors.
gradientLayer.colors = [topColor, bottomColor]

// Set start point.
gradientLayer.startPoint = CGPoint(x: 0.0, y: 0.5)

// Set end point.
gradientLayer.endPoint = CGPoint(x: 1.0, y: 0.5)

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, atIndex: 0)
```

Resultado:



Creando un CAGradientLayer horizontal con múltiples colores.

```
// View to hold the CAGradientLayer.
let view: UIView = UIView(frame: CGRect(x: 0, y: 0, width: 320, height: 320))

// Initialize gradient layer.
let gradientLayer: CAGradientLayer = CAGradientLayer()

// Set frame of gradient layer.
gradientLayer.frame = view.bounds

// Color at the top of the gradient.
let topColor: CGColor = UIColor.greenColor().CGColor

// Color at the middle of the gradient.
let middleColor: CGColor = UIColor.blueColor().CGColor

// Color at the bottom of the gradient.
let bottomColor: CGColor = UIColor.blackColor().CGColor

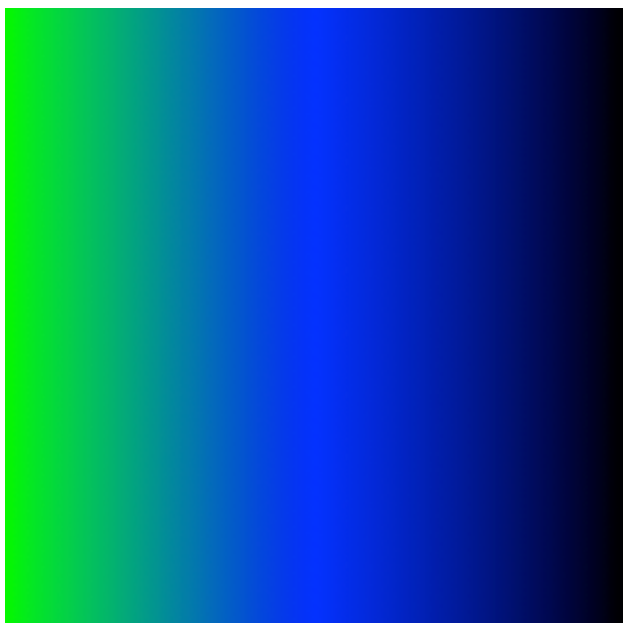
// Set colors.
gradientLayer.colors = [topColor, middleColor, bottomColor]

// Set start point.
gradientLayer.startPoint = CGPoint(x: 0.0, y: 0.5)

// Set end point.
gradientLayer.endPoint = CGPoint(x: 1.0, y: 0.5)

// Insert gradient layer into view's layer heirarchy.
view.layer.insertSublayer(gradientLayer, atIndex: 0)
```

Resultado:



Animando un cambio de color en CAGradientLayer.

```
// Get the current colors of the gradient.
let oldColors = self.gradientLayer.colors

// Define the new colors for the gradient.
let newColors = [UIColor.red.cgColor, UIColor.yellow.cgColor]

// Set the new colors of the gradient.
self.gradientLayer.colors = newColors

// Initialize new animation for changing the colors of the gradient.
let animation: CABasicAnimation = CABasicAnimation(keyPath: "colors")

// Set current color value.
animation.fromValue = oldColors

// Set new color value.
animation.toValue = newColors

// Set duration of animation.
animation.duration = 0.3

// Set animation to remove once its completed.
animation.isRemovedOnCompletion = true

// Set receiver to remain visible in its final state when the animation is completed.
animation.fillMode = kCAFillModeForwards

// Set linear pacing, which causes an animation to occur evenly over its duration.
animation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionLinear)

// Set delegate of animation.
animation.delegate = self

// Add the animation.
self.gradientLayer.addAnimation(animation, forKey: "animateGradientColorChange")
```

Resultado:

Animate



Lea CAGradientLayer en línea: <https://riptutorial.com/es/ios/topic/1190/cagradientlayer>

Capítulo 24: CALayer

Examples

Creando un CALayer

Puedes crear un CALayer y establecer su marco de esta manera:

Rápido:

```
let layer = CALayer()
layer.frame = CGRect(x: 0, y: 0, width: 60, height: 80)
```

C objetivo:

```
CALayer *layer = [[CALayer alloc] init];
layer.frame = CGRectMake(0, 0, 60, 80);
```

Luego puede agregarlo como una subcapa a un CALayer existente:

Rápido:

```
existingLayer.addSublayer(layer)
```

C objetivo:

```
[existingLayer addSublayer:layer];
```

Nota:

Para hacer esto necesitas incluir el framework QuartzCore.

Rápido:

```
@import QuartzCore
```

C objetivo

```
#import <QuartzCore/QuartzCore.h>
```

Creando partículas con CAEmitterLayer

La clase **CAEmitterLayer** proporciona un sistema de emisor de partículas para Core Animation. Las partículas están definidas por instancias de **CAEmitterCell**.

Las partículas se dibujan sobre el color de fondo y el borde de la capa.

```

var emitter = CAEmitterLayer()

emitter.emitterPosition = CGPoint(x: frame.size.width / 2.0, y: -20)
emitter.emitterShape = kCAEmitterLayerLine
emitter.emitterSize = CGSize(width: frame.size.width, height: 1)

emitter.emitterCells = cells
layer.addSublayer(emitter)

```

Vista del emisor con imagen personalizada.

Por ejemplo, crearemos una vista que contenga una capa emisora y anime las partículas.

```

import QuartzCore

class ConfettiView: UIView {
    // main emitter layer
    var emitter: CAEmitterLayer!

    // array of color to emit
    var colors: [UIColor]!

    // intensity of appearance
    var intensity: Float!

    private var active :Bool!

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setup()
    }

    override init(frame: CGRect) {
        super.init(frame: frame)
        setup()
    }

    func setup() {
        // initialization
        colors = [UIColor.redColor(),
                 UIColor.greenColor(),
                 UIColor.blueColor()
                ]
        intensity = 0.2

        active = false
    }

    func startConfetti() {
        emitter = CAEmitterLayer()

        emitter.emitterPosition = CGPoint(x: frame.size.width / 2.0, y: -20)
        emitter.emitterShape = kCAEmitterLayerLine
        emitter.emitterSize = CGSize(width: frame.size.width, height: 1)

        var cells = [CAEmitterCell]()
        for color in colors {
            cells.append(confettiWithColor(color))
        }
    }
}

```

```

    emitter.emitterCells = cells
    layer.addSublayer(emitter)
    active = true
}

func stopConfetti() {
    emitter?.birthRate = 0
    active = false
}

func confettiWithColor(color: UIColor) -> CAEmitterCell {
    let confetti = CAEmitterCell()

    confetti.birthRate = 10.0 * intensity
    confetti.lifetime = 180.0 * intensity
    confetti.lifetimeRange = 0
    confetti.color = color.CGColor
    confetti.velocity = CGFloat(350.0 * intensity)
    confetti.velocityRange = CGFloat(40.0 * intensity)
    confetti.emissionLongitude = CGFloat(M_PI)
    confetti.emissionRange = CGFloat(M_PI_4)
    confetti.spin = CGFloat(3.5 * intensity)
    confetti.spinRange = CGFloat(4.0 * intensity)

    // WARNING: A layer can set this property to a CGImageRef to display the image as its
    contents.
    confetti.contents = UIImage(named: "confetti")?.CGImage
    return confetti
}

internal func isActive() -> Bool {
    return self.active
}
}

```

Debe agregar una imagen de "confeti" o definir rect con **confetti.contentsRect**

Cómo agregar un UIImage a un CALayer

Puede agregar una imagen a la `layer` una vista simplemente usando su propiedad de `contents` :

```
myView.layer.contents = UIImage(named: "star")?.CGImage
```

- Tenga en cuenta que el `UIImage` necesita ser convertido a `CGImage` .

Si desea agregar la imagen en su propia capa, puede hacerlo de la siguiente manera:

```

let myLayer = CALayer()
let myImage = UIImage(named: "star")?.CGImage
myLayer.frame = myView.bounds
myLayer.contents = myImage
myView.layer.addSublayer(myLayer)

```

Modificando la apariencia

El código anterior produce una vista como esta. El azul claro es el `UIView` y la estrella azul oscuro es el `UIImage`.



Como puedes ver, sin embargo, parece pixelado. Esto se debe a que el `UIImage` es más pequeño que el `UIView` por lo que se está escalando para completar la vista, que es el valor predeterminado, no se especifica nada más.

Los siguientes ejemplos muestran variaciones en la propiedad `contentsGravity` la capa. El código se ve así:

```
myView.layer.contents = UIImage(named: "star")?.CGImage
myView.layer.contentsGravity = kCAGravityTop
myView.layer.geometryFlipped = true
```

En iOS, es posible que desee establecer la [propiedad `geometryFlipped`](#) en `true` si está haciendo algo con gravedad superior o inferior, de lo contrario será lo contrario de lo que espera. (Solo la gravedad se invierte verticalmente, no la representación del contenido. Si tiene problemas con el contenido que se invierte, consulte [esta respuesta de Desbordamiento de pila](#)).

Hay dos ejemplos de `UIView` continuación para cada configuración de `contentsGravity` `UIView`, una vista es más grande que el `UIImage` y la otra es más pequeña. De esta manera puedes ver los efectos de la escala y la gravedad.

`kCAGravityResize`

Este es el valor predeterminado.



`kCAGravityResizeAspect`



`kCAGravityResizeAspectFill`



`kCAGravityCenter`



`kCAGravityTop`



`kCAGravityBottom`



`kCAGravityLeft`



`kCAGravityRight`



`kCAGravityTopLeft`



`kCAGravityTopRight`



`kCAGravityBottomLeft`



`kCAGravityBottomRight`



Relacionado

- [Propiedad de modo de contenido de una vista](#)
- [Dibujando un UIImage en drawRect con CGContextDrawImage](#)
- [Tutorial de CALayer: Primeros pasos](#)

Notas

- Este ejemplo proviene originalmente de [esta respuesta de desbordamiento de pila](#) .

Agregar transformaciones a un CALayer (traducir, rotar, escalar)

Lo esencial

Hay una serie de transformaciones diferentes que puedes hacer en una capa, pero las básicas son

- traducir
- escala
- girar



Para realizar transformaciones en un `CALayer`, establezca la propiedad de `transform` la capa en un tipo `CATransform3D`. Por ejemplo, para traducir una capa, harías algo como esto:

```
myLayer.transform = CATransform3DMakeTranslation(20, 30, 0)
```

La palabra `Make` se usa en el nombre para crear la transformación inicial: `CATransform3D Make Translation`. Las transformaciones subsiguientes que se aplican omiten la `Make`. Vea, por ejemplo, esta rotación seguida de una traducción:

```
let rotation = CATransform3DMakeRotation(CGFloat(30.0 * M_PI / 180.0), 20, 20, 0)
myLayer.transform = CATransform3DTranslate(rotation, 20, 30, 0)
```

Ahora que tenemos la base de cómo hacer una transformación, veamos algunos ejemplos de cómo hacer cada uno. Primero, sin embargo, mostraré cómo configuro el proyecto en caso de que también quieras jugar con él.

Preparar

Para los siguientes ejemplos, configuro una aplicación de vista única y agregué una vista `UIView` con un fondo azul claro al guión gráfico. Conecté la vista al controlador de vista con el siguiente código:

```
import UIKit

class ViewController: UIViewController {

    var myLayer = CATextLayer()
    @IBOutlet weak var myView: UIView!

    override func viewDidLoad() {
        super.viewDidLoad()
    }
}
```

```

// setup the sublayer
addSubLayer()

// do the transform
transformExample()
}

func addSubLayer() {
    myLayer.frame = CGRect(x: 0, y: 0, width: 100, height: 40)
    myLayer.backgroundColor = UIColor.blueColor().CGColor
    myLayer.string = "Hello"
    myView.layer.addSublayer(myLayer)
}

//***** Replace this function with the examples below *****/

func transformExample() {

    // add transform code here ...

}
}

```

Hay muchos tipos diferentes de `CALayer`, pero elegí usar `CATextLayer` para que las transformaciones sean más claras visualmente.

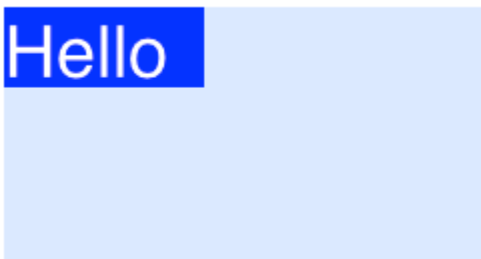
Traducir

La transformación de la traducción mueve la capa. La sintaxis básica es

```
CATransform3DMakeTranslation(tx: CGFloat, ty: CGFloat, tz: CGFloat)
```

donde `tx` es el cambio en las coordenadas x, `ty` es el cambio en y, y `tz` es el cambio en z.

Ejemplo



En iOS, el origen del sistema de coordenadas está en la parte superior izquierda, por lo que si quisiéramos mover la capa 90 puntos a la derecha y 50 puntos hacia abajo, haríamos lo siguiente:

```
myLayer.transform = CATransform3DMakeTranslation(90, 50, 0)
```

Notas

- Recuerde que puede pegar esto en el método `transformExample()` en el código del proyecto anterior.
- Ya que vamos a tratar dos dimensiones aquí, `tz` se establece en `0`.
- La línea roja en la imagen de arriba va desde el centro de la ubicación original hasta el centro de la nueva ubicación. Esto se debe a que las transformaciones se realizan en relación con el punto de anclaje y el punto de anclaje por defecto está en el centro de la capa.

Escala

La transformación a escala estira o aplasta la capa. La sintaxis básica es

```
CATransform3DMakeScale(sx: CGFloat, sy: CGFloat, sz: CGFloat)
```

donde `sx`, `sy` y `sz` son los números por los cuales escalar (multiplicar) las coordenadas x, y y z, respectivamente.

Ejemplo



Si quisiéramos la mitad del ancho y el triple de la altura, haríamos lo siguiente

```
myLayer.transform = CATransform3DMakeScale(0.5, 3.0, 1.0)
```

Notas

- Como solo trabajamos en dos dimensiones, solo multiplicamos las coordenadas z por `1.0` para no afectarlas.
- El punto rojo en la imagen de arriba representa el punto de anclaje. Observe cómo se realiza el escalado en relación con el punto de anclaje. Es decir, todo se estira hacia o desde el punto de anclaje.

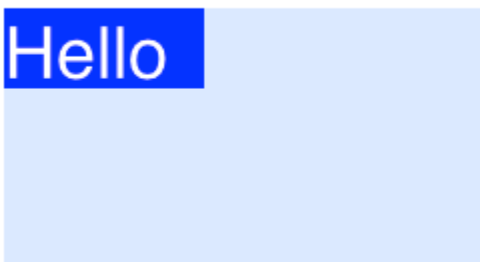
Girar

La transformación de rotación gira la capa alrededor del punto de anclaje (el centro de la capa de forma predeterminada). La sintaxis básica es

```
CATransform3DMakeRotation(angle: CGFloat, x: CGFloat, y: CGFloat, z: CGFloat)
```

donde `angle` es el ángulo en radianes en que se debe rotar la capa y `x`, `y` y `z` son los ejes sobre los cuales rotar. Establecer un eje en 0 cancela una rotación alrededor de ese eje en particular.

Ejemplo



Si quisiéramos rotar una capa en sentido horario 30 grados, haríamos lo siguiente:

```
let degrees = 30.0
let radians = CGFloat(degrees * M_PI / 180)
myLayer.transform = CATransform3DMakeRotation(radians, 0.0, 0.0, 1.0)
```

Notas

- Como estamos trabajando en dos dimensiones, solo queremos que el plano xy gire alrededor del eje z. Así establecimos `x` e `y` a `0.0` y establecimos `z` a `1.0`.
- Esto hizo girar la capa en el sentido de las agujas del reloj. Podríamos haber girado en sentido contrario a las manecillas del reloj estableciendo `z` en `-1.0`.
- El punto rojo muestra dónde está el punto de anclaje. La rotación se realiza alrededor del punto de anclaje.

Transformaciones múltiples

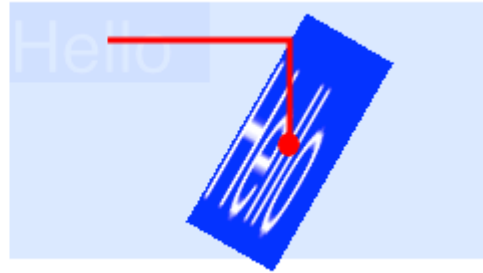
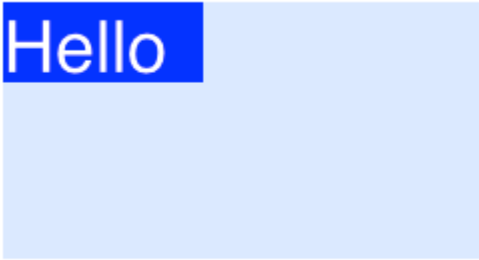
Para combinar múltiples transformaciones podríamos usar concatenación como esta

```
CATransform3DConcat(a: CATransform3D, b: CATransform3D)
```

Sin embargo, vamos a hacer uno tras otro. La primera transformación utilizará la `Make` en su nombre. Las siguientes transformaciones no utilizarán `Make`, pero tomarán la transformación

anterior como un parámetro.

Ejemplo



Esta vez combinamos las tres transformaciones anteriores.

```
let degrees = 30.0
let radians = CGFloat(degrees * M_PI / 180)

// translate
var transform = CATransform3DMakeTranslation(90, 50, 0)

// rotate
transform = CATransform3DRotate(transform, radians, 0.0, 0.0, 1.0)

// scale
transform = CATransform3DScale(transform, 0.5, 3.0, 1.0)

// apply the transforms
myLayer.transform = transform
```

Notas

- El orden en que se realizan las transformaciones en materia.
- Todo se hizo en relación con el punto de anclaje (punto rojo).

Una nota sobre el punto de anclaje y la posición

Hicimos todas nuestras transformaciones anteriores sin cambiar el punto de anclaje. Sin embargo, a veces es necesario cambiarlo, como si desea girar alrededor de algún otro punto además del centro. Sin embargo, esto puede ser un poco complicado.

El punto de anclaje y la posición están en el mismo lugar. El punto de anclaje se expresa como una unidad del sistema de coordenadas de la capa (el valor predeterminado es `0.5, 0.5`) y la posición se expresa en el sistema de coordenadas de la superapa. Se pueden configurar de esta manera

```
myLayer.anchorPoint = CGPoint(x: 0.0, y: 1.0)
```

```
myLayer.position = CGPoint(x: 50, y: 50)
```

Si solo establece el punto de anclaje sin cambiar la posición, entonces el marco cambia para que la posición esté en el lugar correcto. O más precisamente, el marco se recalcula basándose en el nuevo punto de anclaje y la posición anterior. Esto suele dar resultados inesperados. Los siguientes dos artículos tienen una excelente discusión de esto.

- [Acerca del punto de anclaje](#)
- [Traducir rotar traducir?](#)

Ver también

- [Borde, esquinas redondeadas y sombra en un CALayer](#)
- [Usando un borde con una ruta Bezier para una capa](#)

Este ejemplo proviene originalmente de [este ejemplo de desbordamiento de pila](#).

Deshabilitar animaciones

CALayer animaciones de propiedades de CALayer están habilitadas por defecto. Cuando esto no es deseable, se pueden desactivar de la siguiente manera.

Rápido

```
CATransaction.begin()
CATransaction.setDisableActions(true)

// change layer properties that you don't want to animate

CATransaction.commit()
```

C objetivo

```
[CATransaction begin];
[CATransaction setDisableActions:YES];

// change layer properties that you don't want to animate

[CATransaction commit];
```

Esquinas redondeadas

```
layer.masksToBounds = true;
layer.cornerRadius = 8;
```

Oscuridad

Puedes usar 5 propiedades en cada capa para configurar tus sombras:

- `shadowOffset` - esta propiedad mueve tu sombra a la izquierda / derecha o arriba / abajo

```
self.layer.shadowOffset = CGSizeMake(-1, -1); // 1px left and up  
self.layer.shadowOffset = CGSizeMake(1, 1); // 1px down and right
```

- `shadowColor` - esto establece el color de tu sombra

```
self.layer.shadowColor = [UIColor blackColor].CGColor;
```

- `shadowOpacity` - esta es la opacidad de la sombra, de 0 a 1

```
self.layer.shadowOpacity = 0.2;
```

- `shadowRadius` : este es el radio de desenfoque (equivalente a la propiedad de desenfoque en Sketch o Photoshop)

```
self.layer.shadowRadius = 6;
```

- `shadowPath` : esta es una propiedad importante para el rendimiento, cuando el iOS no configurado basa la sombra en el canal alfa de la vista, que puede ser intensivo en el rendimiento con un PNG complejo con alfa. Esta propiedad te permite forzar una forma para tu sombra y tener más rendimiento gracias a ella.

C objetivo

```
self.layer.shadowPath = [UIBezierPath bezierPathWithOvalInRect:CGRectMake(0,0,100,100)];  
//this does a circular shadow
```

Swift 3

```
self.layer.shadowPath = UIBezierPath(ovalIn: CGRect(x: 0, y: 0, width: 100, height:  
100)).cgPath
```

Lea CALayer en línea: <https://riptutorial.com/es/ios/topic/1462/calayer>

Capítulo 25: Calificación de solicitud / solicitud de revisión

Introducción

Ahora, desde iOS 10.3, no es necesario navegar la aplicación a Apple Store para calificar / revisar. Apple ha introducido la clase `SKStoreReviewController` en el marco de `storekit`. En qué desarrollador solo necesita llamar al método de clase `requestReview ()` de la clase `SKStoreReviewController` y el sistema maneja todo el proceso por usted.

Además, puede continuar incluyendo un enlace persistente en la configuración o en las pantallas de configuración de su aplicación que enlaza a la página de su tienda de productos. Para ope automáticamente

Examples

Calificar / Revisar la aplicación iOS

Simplemente escriba debajo del código de una línea desde donde desea que el usuario califique / revise su aplicación.

```
SKStoreReviewController.requestReview ()
```

Lea [Calificación de solicitud / solicitud de revisión en línea](https://riptutorial.com/es/ios/topic/9678/calificacion-de-solicitud---solicitud-de-revision):

<https://riptutorial.com/es/ios/topic/9678/calificacion-de-solicitud---solicitud-de-revision>

Capítulo 26: Cambiar color de barra de estado

Examples

Para barras de estado que no sean de UINavigationController

1. En info.plist configurar `View controller-based status bar appearance` en YES
2. En vista, los controladores no contenidos en `UINavigationController` implementan este método.

En Objective-C:

```
- (UIStatusBarStyle)preferredStatusBarStyle
{
    return UIStatusBarStyleLightContent;
}
```

En Swift:

```
override func preferredStatusBarStyle() -> UIStatusBarStyle {
    return UIStatusBarStyle.LightContent
}
```

Para las barras de estado de UINavigationController

Subclase `UINavigationController` y luego invalida estos métodos:

En Objective-C:

```
- (UIStatusBarStyle)preferredStatusBarStyle
{
    return UIStatusBarStyleLightContent;
}
```

En Swift:

```
override func preferredStatusBarStyle() -> UIStatusBarStyle {
    return .lightContent
}
```

Alternativamente, puede establecer `barStyle` en la instancia de `UINavigationController` :

C objetivo:

```
// e.g. in your view controller's viewDidLoad method:
self.navigationController.navigationBar.barStyle = UIBarStyleBlack; // this will give you a
```

```
white status bar
```

Rápido

```
// e.g. in your view controller's viewDidLoad method:  
navigationController?.navigationBar.barStyle = .black // this will give you a white status bar
```

`UIBarStyle` opciones de `UIBarStyle` son `default` , `black` , `blackOpaque` , `blackTranslucent` . Los últimos 3 deberían darle una barra de estado con texto blanco, solo los dos últimos especifican la opacidad de la barra.

Nota: todavía puedes cambiar el aspecto de tu barra de navegación como quieras.

Si no puede cambiar el código de ViewController

Si está utilizando una biblioteca que contiene (por ejemplo) `AwesomeViewController` con un color de barra de estado incorrecto, puede intentar esto:

```
let awesomeViewController = AwesomeViewController()  
awesomeViewController.navigationBar.barStyle = .blackTranslucent // or other style
```

Para la contención de ViewController

Si está utilizando `UINavigationController` hay algunos otros métodos que vale la pena analizar.

Cuando desea que un controlador de la vista infantil controle la presentación de la barra de estado (es decir, si el niño está colocado en la parte superior de la pantalla).

en Swift

```
class RootViewController: UIViewController {  
  
    private let messageBarController = MessageBarController()  
  
    override func childViewControllerForStatusBarStyle() -> UIViewController? {  
        return messageBarController  
    }  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
  
        //add child vc code here...  
  
        setNeedsStatusBarAppearanceUpdate()  
    }  
}  
  
class MessageBarController: UIViewController {  
  
    override func preferredStatusBarStyle() -> UIStatusBarStyle {  
        return .Default  
    }  
}
```

```
}
```

Cambiando el estilo de la barra de estado para toda la aplicación.

RÁPIDO:

Paso 1:

En su **Info.plist** agregue el siguiente atributo:

```
View controller-based status bar appearance
```

y establece su valor en

```
NO
```

como se describe en la siguiente imagen:

Key	Type	Value
▼ Information Property List	Dictionary	(15 iter
View controller-based status...	Boolean	NO

Paso 2:

En su archivo **AppDelegate.swift**, en el método `didFinishLaunchingWithOptions`, agregue este código:

```
UIApplication.shared.statusBarStyle = .lightContent
```

o

```
UIApplication.shared.statusBarStyle = .default
```

- La opción **.lightContent** establecerá el color de la **barra de estado** en blanco para toda la aplicación.
- La opción **.default** establecerá el color de la **barra de estado** en el color negro original, para toda la aplicación.

C OBJETIVO:

Sigue el primer paso de la sección **SWIFT** . Luego agregue este código al archivo **AppDelegate.m** :

```
[[UIApplication sharedApplication] setStatusBarStyle:UIStatusBarStyleLightContent];
```

o

```
[[UIApplication sharedApplication] setStatusBarStyle:UIStatusBarStyleDefault];
```

Lea **Cambiar color de barra de estado en línea**: <https://riptutorial.com/es/ios/topic/378/cambiar-color-de-barra-de-estado>

Capítulo 27: Cambiar el tamaño de UIImage

Parámetros

CGInterpolaciónCalidad	Niveles de calidad de interpolación para renderizar una imagen.
La calidad de interpolación es un parámetro de estado gráfico.	<pre>typedef enum CGInterpolationQuality CGInterpolationQuality;</pre>

Examples

Redimensionar cualquier imagen por tamaño y calidad.

```
- (UIImage *)drawImageBySize:(CGSize)size quality:(CGInterpolationQuality)quality  
{  
    UIGraphicsBeginImageContextWithOptions(size, NO, 0.0);  
    CGContextRef context = UIGraphicsGetCurrentContext();  
    CGContextSetInterpolationQuality(context, quality);  
    [self drawInRect: CGRectMake (0, 0, size.width, size.height)];  
    UIImage *resizedImage = UIGraphicsGetImageFromCurrentImageContext();  
    UIGraphicsEndImageContext();  
    return resizedImage;  
}
```

Lea Cambiar el tamaño de UIImage en línea: <https://riptutorial.com/es/ios/topic/6422/cambiar-el-tamano-de-uiimage>

Capítulo 28: Cargar imagenes async

Examples

La manera más fácil

La forma más sencilla de crear esto es usar [Alamofire](#) y su [UIImageViewExtension](#) . Lo que necesitamos es una vista de tabla con una celda que tenga una imageView en ella y la llamemos `imageView` .

En la función `cellForRowAt:` de `tableView`, descargamos la imagen y la configuramos de la siguiente manera:

```
let url = URL(string: "https://httpbin.org/image/png")!
let placeholderImage = UIImage(named: "placeholder")!

imageView.af_setImage(withURL: url, placeholderImage: placeholderImage)
```

La URL debe apuntar a la imagen que desea descargar y la imagen `placeHolder` debe ser una imagen almacenada. Luego, llamamos al método `af_setImage` en el `imageView` que descarga la imagen en la url dada y durante la descarga se mostrará la imagen del marcador de posición. Tan pronto como se descarga la imagen, se muestra la imagen solicitada.

Comprueba que la celda sigue siendo visible después de la descarga.

A veces, la descarga tarda más de lo que se muestra la celda. En este caso, puede suceder que la imagen descargada se muestre en la celda incorrecta. Para solucionar esto no podemos usar la [extensión UIImageView](#) .

Seguiremos usando [Alamofire](#), sin embargo, usaremos el controlador de finalización para mostrar la imagen.

En este escenario, todavía necesitamos una `tableView` con una celda que tenga una `imageView` en ella. En el método `cellForRowAt:` descargaríamos la imagen con el siguiente código:

```
let placeholderImage = UIImage(named: "placeholder")!
imageView.image = placeholderImage

let url = URL(string: "https://httpbin.org/image/png")!

Alamofire.request(url!, method: .get).responseImage { response in
    guard let image = response.result.value else { return }

    if let updateCell = tableView.cellForRow(at: indexPath) {
        updateCell.imageView.image = image
    }
}
```

En este ejemplo, primero configuramos la imagen a la imagen de marcador de posición.

Posteriormente descargamos la imagen con el método de `request` de [Alamofire](#). Pasamos la url como primer argumento y, como solo queremos obtener la imagen, usaremos el método HTTP `.get`. Como estamos descargando una imagen, queremos que la respuesta sea una imagen, por lo tanto, usamos el método `.responseImage`.

Después de que se haya descargado la imagen, se llama al cierre y, en primer lugar, nos aseguramos de que la imagen descargada exista realmente. Luego, nos aseguramos de que la celda aún esté visible al verificar que `cellForRow` (at: `indexPath`) no devuelve `nil`. Si no hace nada, si no, asignamos la imagen descargada recientemente.

Esta última declaración `if` asegura que la celda aún estará visible si el usuario ya ha desplazado sobre la celda, el `updateCell` será nulo y la instrucción `if` devuelve `nil`. Esto nos ayuda a evitar mostrar la imagen incorrecta en una celda.

Lea [Cargar imagenes async en línea](https://riptutorial.com/es/ios/topic/10793/cargar-imagenes-async): <https://riptutorial.com/es/ios/topic/10793/cargar-imagenes-async>

Capítulo 29: Carril rápido

Examples

herramientas de plano rápido

[fastlane](#) es una herramienta de automatización de compilación de código abierto para Android e iOS para desarrolladores. Se reduce el tiempo de generación de su construcción. Es una herramienta de línea de comandos que utiliza [Ruby](#), por lo que necesitas Ruby en tu computadora. La mayoría de las Mac ya tienen Ruby instalado por defecto.

Instalar fastlane

1. Abre una terminal.
2. Ejecutar `sudo gem install fastlane --verbose`
3. Si aún no ha instalado las herramientas de línea de comandos de Xcode, ejecute `xcode-select --install` para instalarlas
4. Ahora, `cd` en la carpeta de su proyecto (escriba `cd` [con el espacio al final] y arrastre la carpeta del proyecto al terminal)
5. Ejecute `fastlane init` para obtener la configuración de fastlane.
6. Ahora puedes usar todas las herramientas de Fastlane:

herramientas de iOS

- [entregar](#) : cargue capturas de pantalla, metadatos y su aplicación en el App Store
- [instantánea](#) : automatice la toma de capturas de pantalla localizadas de su aplicación iOS en todos los dispositivos
- [frameit](#) : coloca rápidamente tus capturas de pantalla en los marcos correctos del dispositivo
- [pem](#) : genera y renueva automáticamente tus perfiles de notificaciones push
- [suspiro](#) : porque preferirías pasar tu tiempo construyendo cosas que luchando contra el aprovisionamiento
- [producir](#) : crear nuevas aplicaciones de iOS en iTunes Connect y Dev Portal usando la línea de comandos
- [cert](#) : crea y mantiene automáticamente certificados de firma de código iOS
- [gimnasio](#) : construir tus aplicaciones iOS nunca fue tan fácil
- [coincidencia](#) : sincronice fácilmente sus certificados y perfiles en todo su equipo con Git
- [scan](#) : la forma más fácil de ejecutar pruebas para sus aplicaciones iOS y Mac
- [nave espacial](#) : biblioteca Ruby para acceder al Centro de desarrollo de Apple y iTunes Connect

Herramientas de TestFlight para iOS

- [Piloto](#) : la mejor manera de administrar los probadores y compilaciones de TestFlight desde

su terminal.

- [embarque](#) : la forma más sencilla de invitar a los probadores beta de TestFlight

Herramientas de android

- [suministro](#) : sube tu aplicación de Android y sus metadatos a Google Play
- [screengrab](#) : automatice la toma de capturas de pantalla localizadas de su aplicación de Android en todos los dispositivos

Lea Carril rápido en línea: <https://riptutorial.com/es/ios/topic/3574/carril-rapido>

Capítulo 30: CAShapeLayer

Sintaxis

1. shapeLayer.fillColor
2. shapeLayer.fillRule
3. shapeLayer.lineCap
4. shapeLayer.lineDashPattern
5. shapeLayer.lineDashPhase
6. shapeLayer.lineJoin

Observaciones

La clase CAShapeLayer dibuja una spline Bezier cúbica en su espacio de coordenadas. La forma se compone entre los contenidos de la capa y su primera subcapa.

Examples

Operación básica de CAShapeLayer

UIBezierPath usando para crear una ruta circular ShapeLayer

```
CAShapeLayer *circleLayer = [CAShapeLayer layer];
[circleLayer setPath:[UIBezierPath bezierPathWithOvalInRect:
CGRectMake(50, 50, 100, 100)] CGPath]];
circleLayer.lineWidth = 2.0;
[circleLayer setStrokeColor:[UIColor redColor] CGColor]];
[circleLayer setFillColor:[UIColor clearColor] CGColor]];
circleLayer.lineJoin = kCALineJoinRound; //4 types are available to create a line style
circleLayer.lineDashPattern = [NSArray arrayWithObjects:
[NSNumber numberWithInt:2],[NSNumber numberWithInt:3 ], nil];
// self.origImage is parentView
[[self.view layer] addSublayer:circleLayer];
self.currentShapeLayer = circleLayer; // public value using to keep that reference of the
shape Layer
self.view.layer.borderWidth = 1.0f;
self.view.layer.borderColor = [[UIColor blueColor]CGColor]; // that will plotted in the
mainview
```

Eliminar ShapeLayer

Mantenga una referencia a esa capa de forma. Por ejemplo, podría tener una propiedad currentShapeLayer: ahora que tiene una referencia, puede eliminar fácilmente la capa:

Tipo 1:

```
[self.currentShapeLayer removeFromSuperlayer];
```

Tipo 2:

```
self.view.layer.sublayers = nil ; //removed all earlier shapes
```

Otra operacion

```
//Draw Square Shape

CAShapeLayer *squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(20, 20, 100, 100);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = nil;
squareLayer.strokeColor = [[UIColor redColor] CGColor];
squareLayer.path = [UIBezierPath bezierPathWithRect:squareLayer.bounds].CGPath;
[[self.view layer] addSublayer:squareLayer];

//Draw Circle Shape

CAShapeLayer *circleShape = [CAShapeLayer layer];
circleShape.frame = CGRectMake(160, 20, 120, 120);
circleShape.lineWidth = 2.0;
circleShape.fillColor = nil;
circleShape.strokeColor = [[UIColor redColor] CGColor];
circleShape.path = [UIBezierPath bezierPathWithOvalInRect:circleShape.bounds].CGPath;
[[self.view layer] addSublayer:circleShape];

//Subpaths
//UIBezierPath can have any number of "path segments" (or subpaths) so you can effectively
draw as many shapes or lines as you want in a single path object

CAShapeLayer *shapeLayer = [CAShapeLayer layer];
shapeLayer.frame = CGRectMake(20, 140, 200, 200);
shapeLayer.lineWidth = 2.0;
shapeLayer.fillColor = nil;
shapeLayer.strokeColor = [[UIColor redColor] CGColor];

CGMutablePathRef combinedPath= CGPathCreateMutableCopy(circleShape.path);
CGPathAddPath(combinedPath, NULL, squareLayer.path);

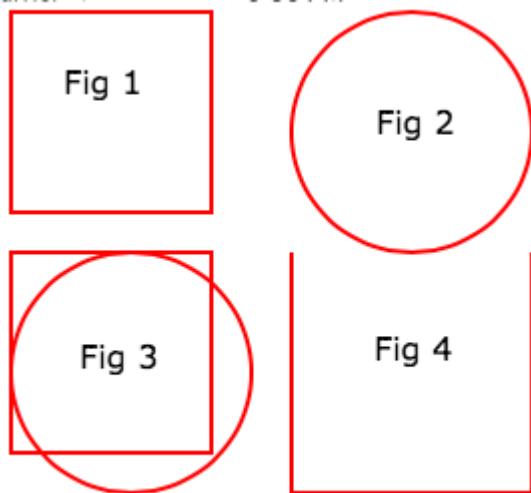
shapeLayer.path = combinedPath;
[[self.view layer] addSublayer:shapeLayer];

//Open Path
// Paths do not need to connect their end points back to their starting points. A path that
connects back to its starting point is called a closed path, and one that does not is called
an open path.

shapeLayer = [CAShapeLayer layer];
shapeLayer.frame = CGRectMake(160, 140, 300, 300);
shapeLayer.lineWidth = 2.0;
shapeLayer.fillColor = nil;
shapeLayer.strokeColor = [[UIColor redColor] CGColor];

UIBezierPath *linePath=[UIBezierPath bezierPath];
[linePath moveToPoint:CGPointZero];
[linePath addLineToPoint:CGPointMake(0 , 120)];
[linePath addLineToPoint:CGPointMake(120 , 120)];
```

```
[linePath addLineToPoint:CGPointMake(120 , 0)];
shapeLayer.path = linePath.CGPath;
[[self.view layer] addSublayer:shapeLayer];
```



Conceptos de relleno // Color de relleno

```
CAShapeLayer *squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(20, 30, 100, 100);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor yellowColor]CGColor];
squareLayer.strokeColor = [[UIColor redColor] CGColor];
squareLayer.path = [UIBezierPath bezierPathWithRect:squareLayer.bounds].CGPath;
[[self.view layer] addSublayer:squareLayer];

//Fill Pattern Color
//images.jpeg

squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(140, 30, 100, 100);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor colorWithPatternImage:[UIImage
imageName:@"images.jpeg"]]CGColor];
squareLayer.strokeColor = [[UIColor redColor] CGColor];
squareLayer.path = [UIBezierPath bezierPathWithRect:squareLayer.bounds].CGPath;
[[self.view layer] addSublayer:squareLayer];

//Fill Rule

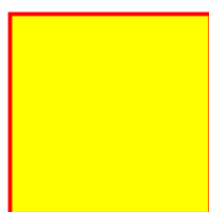
//Type 1: kCAFillRuleNonZero
squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(0, 140, 150, 150);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor yellowColor]CGColor];
squareLayer.fillRule = kCAFillRuleNonZero; // indicate the rule type
squareLayer.strokeColor = [[UIColor redColor] CGColor];
UIBezierPath *outerPath = [UIBezierPath bezierPathWithRect:CGRectInset (squareLayer.bounds,
20.0, 20.0)];
UIBezierPath *innerPath = [UIBezierPath bezierPathWithRect:CGRectInset (squareLayer.bounds,
50.0, 50.0)];
CGMutablePathRef combinedPath= CGPathCreateMutableCopy (outerPath.CGPath);
```

```

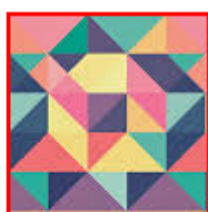
CGPathAddPath(combinedPath, NULL, innerPath.CGPath);
squareLayer.path = combinedPath;
[[self.view layer] addSublayer:squareLayer];

//Type 2: kCAFillRuleEvenOdd
squareLayer = [CAShapeLayer layer];
squareLayer.frame = CGRectMake(140, 140, 150, 150);
squareLayer.lineWidth = 2.0;
squareLayer.fillColor = [[UIColor yellowColor]CGColor];
squareLayer.fillRule = kCAFillRuleEvenOdd; // indicate the rule type
squareLayer.strokeColor = [[UIColor redColor] CGColor];
outerPath = [UIBezierPath bezierPathWithRect:CGRectMakeInset(squareLayer.bounds, 20.0, 20.0)];
innerPath = [UIBezierPath bezierPathWithRect:CGRectMakeInset(squareLayer.bounds, 50.0, 50.0)];
combinedPath= CGPathCreateMutableCopy(outerPath.CGPath);
CGPathAddPath(combinedPath, NULL, innerPath.CGPath);
squareLayer.path = combinedPath;
[[self.view layer] addSublayer:squareLayer];

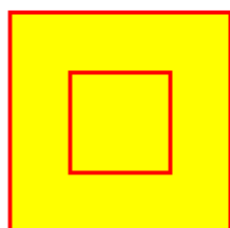
```



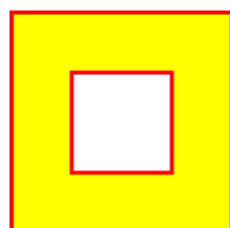
Fill color



Fill Pattern



Fill Rule Zero



Fill Rule Even

Listado las propiedades de estilo de acceso

```

fillColor
    Fill the color based on the drawn shape.

fillRule
    Fill Rule the there are two rule is applied to draw the shape.
    1. kCAFillRuleNonZero
    2. kCAFillRuleEvenOdd

lineCap
    Below type used to change the style of the line.
    1. kCALineCapButt
    2. kCALineCapRound
    3. kCALineCapSquare

lineDashPattern
    The dash pattern applied to the shape's path when stroked.
    Create DashStyle while you will stroke the line.

lineDashPhase
    The dash phase applied to the shape's path when stroked. Animatable.

```

`lineJoin`
Line join style for the shape path. Below style use to draw the line join style.

1. `kCALineJoinMiter`
2. `kCALineJoinRound`
3. `kCALineJoinBevel`

`lineWidth`
Which using to set the line width.

`miterLimit`
The miter limit used when stroking the shape's path. Animatable.

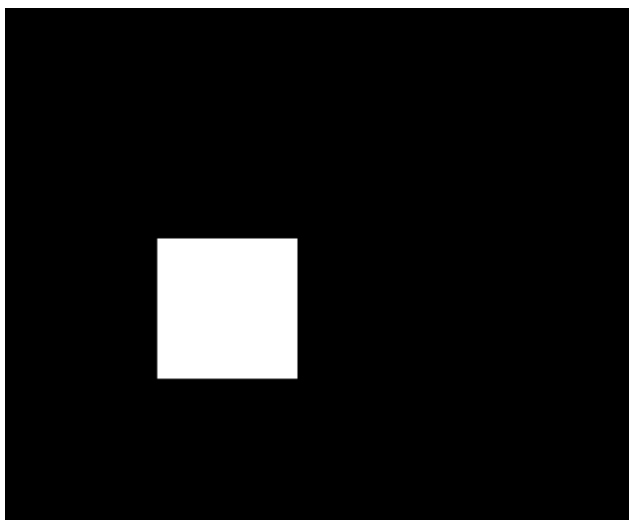
`strokeColor`
Set the stroke color based on the path of the line.

`strokeStart`
When the stroke will start.

`strokeEnd`
When the stroke will end.

Dibujar rectángulo

```
CAShapeLayer *mask = [[CAShapeLayer alloc] init];  
mask.frame = CGRectMake(50, 50, 100, 100);  
CGFloat width = 100;  
CGFloat height = 100;  
CGMutablePathRef path = CGPathCreateMutable();  
CGPathMoveToPoint(path, nil, 30, 30);  
CGPathAddLineToPoint(path, nil, width, 30);  
CGPathAddLineToPoint(path, nil, width, height);  
CGPathAddLineToPoint(path, nil, 30, height);  
CGPathAddLineToPoint(path, nil, 30, 30);  
CGPathCloseSubpath(path);  
  
mask.path = path;  
CGPathRelease(path);  
  
self.view.layer.mask = mask;
```



Dibujar círculo

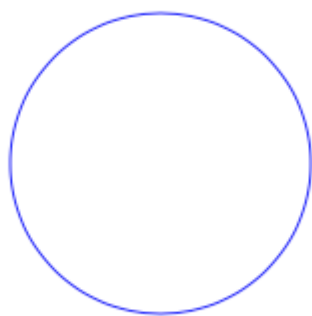
```
CAShapeLayer *circle = [CAShapeLayer layer];

[circle setPath:[UIBezierPath bezierPathWithOvalInRect:CGRectMake(100, 100, 150, 150)]
CGPath]];

[circle setStrokeColor:[UIColor blueColor] CGColor]];

[circle setFillColor:[UIColor clearColor] CGColor]];

[[self.view layer] addSublayer:circle];
```



Animación de CAShapeLayer

```
CAShapeLayer *circle = [CAShapeLayer layer];

[circle setPath:[UIBezierPath bezierPathWithOvalInRect:CGRectMake(100, 100, 150, 150)]
CGPath]];

[circle setStrokeColor:[UIColor blueColor] CGColor]];

[circle setFillColor:[UIColor clearColor] CGColor]];

[[self.view layer] addSublayer:circle];
```

```
CABasicAnimation *pathAnimation = [CABasicAnimation animationWithKeyPath:@"strokeEnd"];
pathAnimation.duration = 1.5f;
pathAnimation.fromValue = [NSNumber numberWithFloat:0.0f];
pathAnimation.toValue = [NSNumber numberWithFloat:1.0f];
pathAnimation.repeatCount = 10;
pathAnimation.autoreverses = YES;

[circle addAnimation:pathAnimation
      forKey:@"strokeEnd"];
```



Lea `CAShapeLayer` en línea: <https://riptutorial.com/es/ios/topic/3575/cashapelayer>

Capítulo 31: Categorías

Observaciones

Las categorías se pueden utilizar para anular los métodos de una clase. Incluso si el método es realmente privado. No se puede acceder al método anulado desde la categoría o desde cualquier otro lugar. Por lo tanto, es importante asegurarse de que al agregar métodos a una clase existente, esos métodos no existan ya.

Examples

Crear una categoría

Las categorías proporcionan la capacidad de agregar alguna funcionalidad adicional a un objeto sin subclasificar o cambiar el objeto real.

Por ejemplo, queremos establecer algunas fuentes personalizadas. Permite crear una categoría que agrega funcionalidad a la clase `UIFont`. Abra su proyecto Xcode, haga clic en Archivo -> Nuevo -> Archivo y elija Archivo Objective-C, haga clic en Siguiente, ingrese el nombre de su categoría, diga "CustomFont" elija el tipo de archivo como Categoría y Clase como UIFont, luego haga clic en "Siguiente" seguido de "Crear". "

Choose a template for your new file:

The dialog displays the following templates:

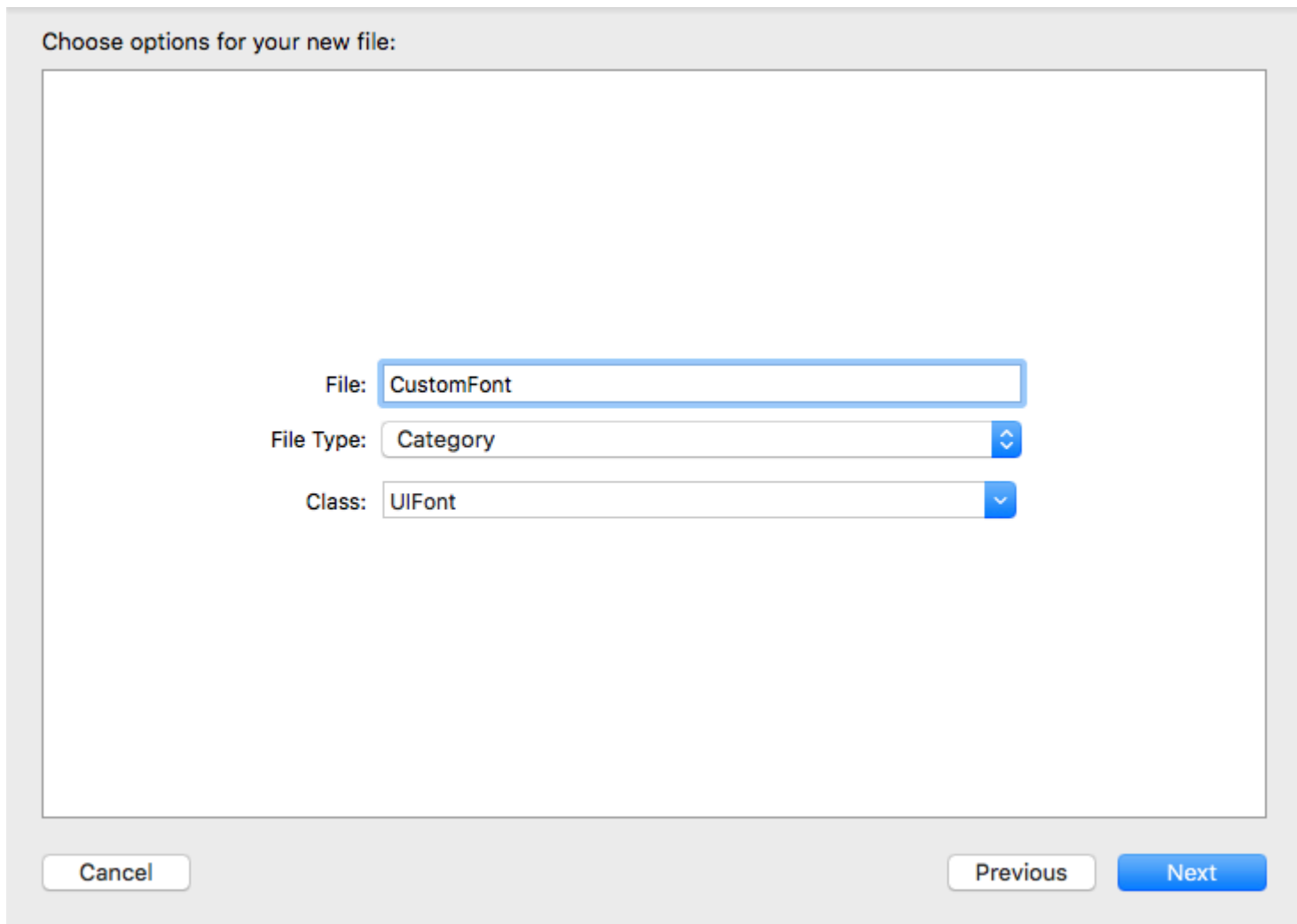
- iOS**
 - Source
 - User Interface
 - Core Data
 - Apple Watch
 - Resource
 - Other
- watchOS**
 - Source
 - User Interface
 - Core Data
 - Resource
 - Other
- tvOS**
 - Source
 - User Interface
 - Core Data
 - Resource

Available file templates:

- Cocoa Touch Class
- UI Test Case Class
- Unit Test Case Class
- Playground
- Swift File
- Objective-C File** (Selected)
- Header File
- C File
- C++ File
- Metal File

Objective-C File
An empty Objective-C file, category, protocol or extension.

Buttons: Cancel, Previous, Next



Declare el método de la categoría: -

Haga clic en "UIFont + CustomFonts.h" para ver el archivo de encabezado de la nueva categoría. Agregue el siguiente código a la interfaz para declarar el método.

```
@interface UIFont (CustomFonts)

+ (UIFont *)productSansRegularFontWithSize:(CGFloat) size;

@end
```

Ahora implemente el método de la categoría: -

Haga clic en "UIFont + CustomFonts.m" para ver el archivo de implementación de la categoría. Agregue el siguiente código para crear un método que establezca la fuente ProductSansRegular.

```
+ (UIFont *)productSansRegularFontWithSize:(CGFloat) size{

    return [UIFont fontWithName:@"ProductSans-Regular" size:size];

}
```

Importa tu categoría

```
#import "UIFont+CustomFonts.h"
```

Ahora establece la fuente de la etiqueta

```
[self.label setFont:[UIFont productSansRegularFontWithSize:16.0]];
```

Lea Categorías en línea: <https://riptutorial.com/es/ios/topic/3633/categorias>

Capítulo 32: Clases de tamaño y adaptabilidad

Observaciones

Al crear aplicaciones adaptables, tenga en cuenta las limitaciones de las clases de tamaño: son *generalizaciones*, no guías específicas para tamaños de píxeles o dispositivos exactos. Nunca intente determinar en qué dispositivo se está ejecutando su aplicación, o si está en un modo de pantalla dividida, según las clases de tamaño.

En su lugar, tome decisiones de diseño de alto nivel en la clase de tamaño y use Diseño automático para cambiar los marcos de vista precisos. (Vea también el método `viewWillTransition(to:with:)` para una notificación más precisa de qué tan grande será la vista de un controlador después de una transición).

Examples

Colecciones de rasgos

En una aplicación iOS, su interfaz de usuario puede adoptar una de varias formas y tamaños generales diferentes. Estos se definen utilizando **clases de tamaño**, que están disponibles a través de una vista o **colección de rasgos** del controlador de vista.

Apple define dos clases de tamaño: **regular** y **compacto**. Cada una de estas clases de tamaño está disponible en ambos ejes del dispositivo (**horizontal** y **vertical**). Su aplicación puede existir en cualquiera de estos cuatro estados durante su vida útil. Como abreviatura, los desarrolladores a menudo describen una combinación de clase de tamaño diciendo o escribiendo las dos clases de tamaño, con el eje horizontal primero: "Compacto / Regular" describe una interfaz que es horizontalmente compacta pero verticalmente regular.

En su aplicación, use métodos en el protocolo de Entorno del Medio Ambiente para verificar su clase de tamaño actual y responder a los cambios:

```
class MyViewController: UIViewController {
    override func viewDidLoad(_ animated: Bool) {
        super.viewDidLoad(animated)
        print("Horizontal size class: \(traitCollection.horizontalSizeClass)")
        print("Vertical size class: \(traitCollection.verticalSizeClass)")
    }

    override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
        super.traitCollectionDidChange(previousTraitCollection)
        print("Trait collection changed; size classes may be different.")
    }
}
```

Tanto `UIView` como `UIViewController` se ajustan a `UITraitEnvironment`, por lo que puede ver su

colección actual de rasgos y manejar los cambios en las subclases de cualquiera de ellos.

Actualización del diseño automático con cambios de colección de rasgos

Hacer que una aplicación sea **adaptable**, es decir, responder a los cambios de tamaño de la clase al cambiar su diseño, a menudo implica mucha ayuda del sistema de diseño automático. Una de las principales formas en que las aplicaciones se adaptan es mediante la actualización de las restricciones de diseño automático activas cuando cambia la clase de tamaño de una vista.

Por ejemplo, considere una aplicación que utiliza un `UIStackView` para organizar dos `UILabels`. Podríamos querer que estas etiquetas se apilen unas sobre otras en entornos horizontalmente compactos, pero nos sentamos uno al lado del otro cuando tengamos un poco más de espacio en entornos horizontalmente regulares.

```
class ViewController: UIViewController {
    var stackView: UIStackView!

    override func viewDidLoad() {
        super.viewDidLoad()

        stackView = UIStackView()
        for text in ["foo", "bar"] {
            let label = UILabel()
            label.translatesAutoresizingMaskIntoConstraints = false
            label.text = text
            stackView.addArrangedSubview(label)
        }

        view.addSubview(stackView)
        stackView.translatesAutoresizingMaskIntoConstraints = false
        stackView.centerXAnchor.constraint(equalTo: view.centerXAnchor).isActive = true
        stackView.centerYAnchor.constraint(equalTo: view.centerYAnchor).isActive = true
    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)
        updateAxis(forTraitCollection: traitCollection)
    }

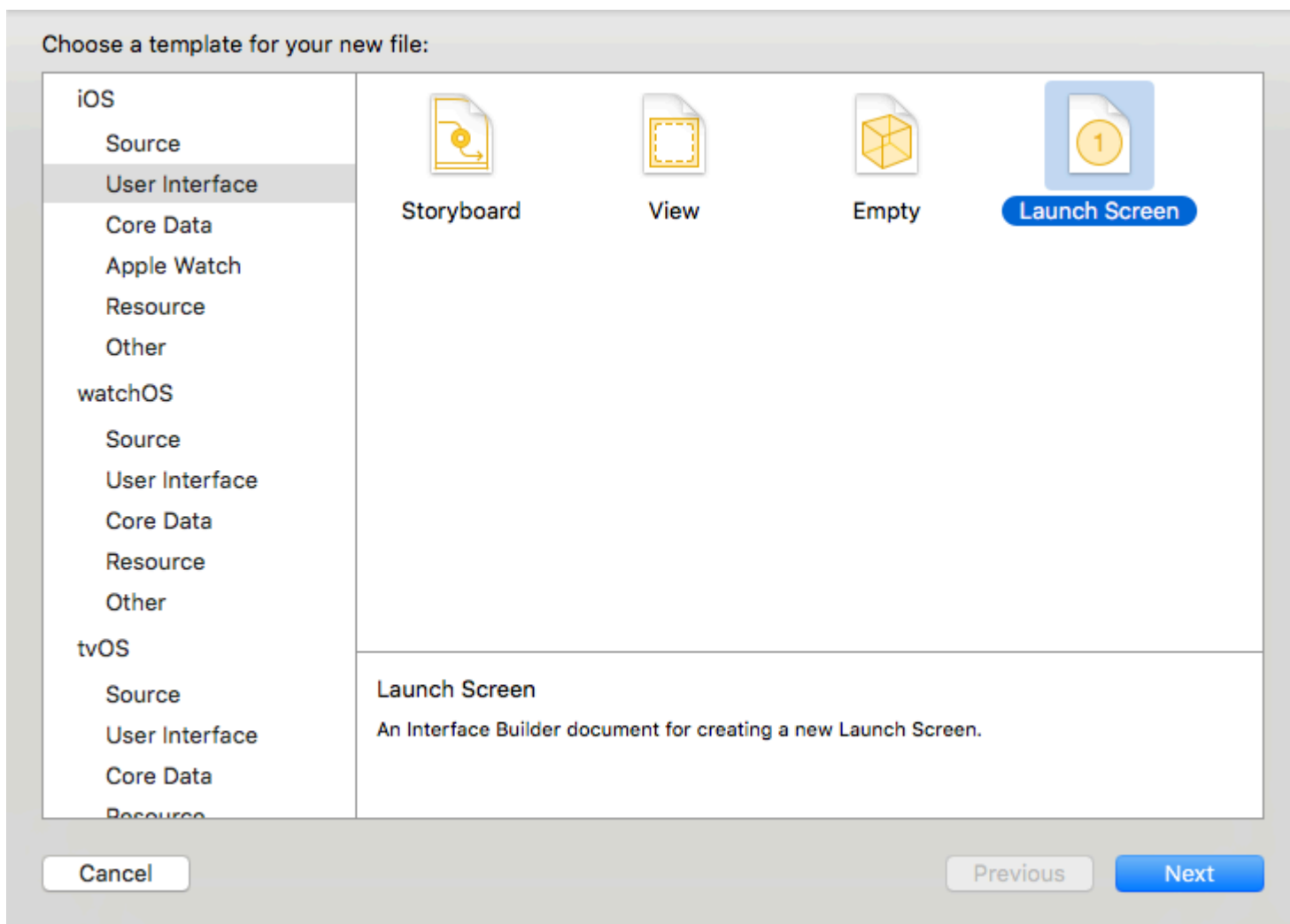
    override func traitCollectionDidChange(_ previousTraitCollection: UITraitCollection?) {
        super.traitCollectionDidChange(previousTraitCollection)
        updateAxis(forTraitCollection: traitCollection)
    }

    private func updateAxis(forTraitCollection traitCollection: UITraitCollection) {
        switch traitCollection.horizontalSizeClass {
        case .regular:
            stackView.axis = .horizontal
        case .compact:
            stackView.axis = .vertical
        case .unspecified:
            print("Unspecified size class!")
            stackView.axis = .horizontal
        }
    }
}
```

Compatibilidad con la multitarea de iOS en iPad

Una pieza clave de la adaptabilidad en una aplicación moderna de iOS es el soporte de multitarea en iPad. De forma predeterminada, las aplicaciones creadas en Xcode 7 y más nuevas se configurararán para admitir la multitarea: tendrán un archivo `LaunchScreen.storyboard` que utiliza el diseño automático.

La forma más fácil para que las aplicaciones existentes opten por la multitarea es crear un guión gráfico de ese tipo y luego configurarlo como la pantalla de inicio del proyecto:



App Icons Source

Launch Images Source

Launch Screen File

Una vez que su aplicación admita la multitarea del iPad, audite las vistas existentes y los controladores de vista para asegurarse de que utilicen el diseño automático y puedan admitir una variedad de combinaciones de clases de tamaño.

Lea Clases de tamaño y adaptabilidad en línea: <https://riptutorial.com/es/ios/topic/4628/clases-de-tamano-y-adaptabilidad>

Capítulo 33: Clases de tamaño y adaptabilidad

Observaciones

Para obtener más detalles (Clases de tamaño y Adaptividad a través de Storyboard) sobre el uso del diseño automático para la adaptabilidad en iOS, podemos seguir el [enlace del sitio del desarrollador de Apple](#) .

También podemos añadir restricciones **Programatically** utilizando **formato de Lenguaje Visual** tal como se describe [aquí en el sitio de desarrolladores de Apple](#) .

Examples

Clases de tamaño y adaptabilidad a través de storyboard

Podemos agregar adaptabilidad a cualquier subclase de `UIView` que agregamos en el controlador de vista en el archivo de plumilla.




Tomemos un ejemplo de agregar adaptabilidad usando clases de tamaño a una vista.

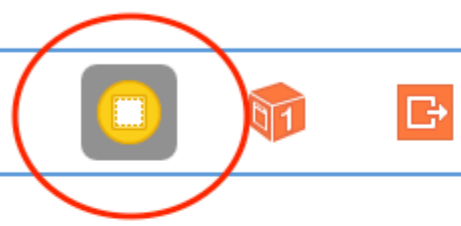
1. Agregue una vista en el controlador de vista como:



el Editor Asistente como;

[Redacted]: Succeeded

mai...se) >  View Controller Scene >  View Controller | <  >



<https://riptutorial.com/es/ios/topic/6424/clases-de-tamano-y-adaptabilidad>

Capítulo 34: CLLocation

Examples

Filtro de distancia utilizando

Ejemplo:

```
CLLocationManager *locationManager = [[CLLocationManager alloc] init];
locationManager.delegate = self;
locationManager.desiredAccuracy = kCLLocationAccuracyBest;
locationManager.distanceFilter = 5;
```

Por ejemplo, en el código de ejemplo anterior, los cambios de ubicación de menos de 5 metros no se enviarán a la devolución de llamada, sino que se ignorarán.

Obtener la ubicación del usuario usando CLLocationManager

1 - Incluya el CoreLocation.framework en su proyecto; Esto se logra haciendo clic en:

```
root directory -> build phases -> Link Binary With Libraries
```

Haga clic en el botón (+), busque CoreLocation.framework y haga clic en agregar.

2- Modifique el archivo info.plist para solicitar permiso para usar la ubicación del usuario abriéndolo como código fuente. Agregue cualquiera de los siguientes pares clave: valor debajo de la etiqueta para solicitar el uso de la ubicación del usuario mientras la aplicación está en uso:

```
<key>NSLocationWhenInUseUsageDescription</key>
<string>message to display when asking for permission</string>
```

3- Importar CoreLocation al ViewController que lo utilizará.

```
import CoreLocation
```

4- Asegúrese de que su ViewController cumpla con el protocolo CLLocationManagerDelegate

```
class ViewController: UIViewController, CLLocationManagerDelegate {}
```

Después de estos pasos, podemos crear un objeto CLLocationManager como variable de instancia y usarlo en ViewController.

```
var manager:CLLocationManager!
```

No usamos 'vamos' aquí porque modificaremos el administrador para especificar su delegado, la distancia mínima antes del evento de actualización y su precisión

```

//initialize the manager
manager = CLLocationManager()

//specify delegate
manager.delegate = self

//set the minimum distance the phone needs to move before an update event is triggered (for
example: 100 meters)
manager.distanceFilter = 100

//set Accuracy to any of the following depending on your use case

//let kCLLocationAccuracyBestForNavigation: CLLocationAccuracy
//let kCLLocationAccuracyBest: CLLocationAccuracy
//let kCLLocationAccuracyNearestTenMeters: CLLocationAccuracy
//let kCLLocationAccuracyHundredMeters: CLLocationAccuracy
//let kCLLocationAccuracyKilometer: CLLocationAccuracy
//let kCLLocationAccuracyThreeKilometers: CLLocationAccuracy

manager.desiredAccuracy = kCLLocationAccuracyBest

//ask the user for permission
manager.requestWhenInUseAuthorization()

//Start collecting location information
if #available(iOS 9.0, *) {
    manager.requestLocation()
} else {
    manager.startUpdatingLocation()
}

```

Ahora, para obtener acceso a las actualizaciones de ubicación, podemos implementar la función a continuación, que se denomina tiempo extra al que se alcanza el filtro de distancia.

```

func locationManager(manager: CLLocationManager, didUpdateLocations locations: [CLLocation])
{}

```

El parámetro de ubicación es una matriz de objetos CLLocation que representan la ubicación real del dispositivo. Desde estos objetos, uno puede obtener acceso a los siguientes atributos: `coordinate`, `altitude`, `floor`, `horizontalAccuracy`, `verticalAccuracy`, `timestamp`, `description`, `course`, `speed` **precisión** `coordinate`, `altitude`, `floor`, `horizontalAccuracy`, `verticalAccuracy`, `timestamp`, `description`, `course`, `speed` **precisión** `coordinate`, `altitude`, `floor`, `horizontalAccuracy`, `verticalAccuracy`, `timestamp`, `description`, `course`, `speed` **y una** `distance(from:)` **función** `distance(from:)` **que mide la distancia entre dos ubicaciones.**

Nota: al solicitar permiso para la ubicación, hay dos tipos diferentes de autorización.

La autorización "Cuando está en uso" solo otorga permiso a la aplicación para recibir su ubicación cuando la aplicación está en uso o en primer plano.

La autorización "Siempre" le otorga a la aplicación permisos de fondo que pueden reducir la duración de la batería en caso de que la aplicación esté cerrada.

Archivo de plist debe ajustarse según sea necesario.

Lea CLLocation en línea: <https://riptutorial.com/es/ios/topic/2002/cllocation>

Capítulo 35: CloudKit

Observaciones

Tipos soportados

- NSData
- NSDate (Fecha)
- NSNumber (Int / Double)
- Cadena NS (String)
- NSArray (Array)
- CLLocation
- CKReferencia
- CKAsset

[Más detalles](#)

[CloudKit Dashboard](#)

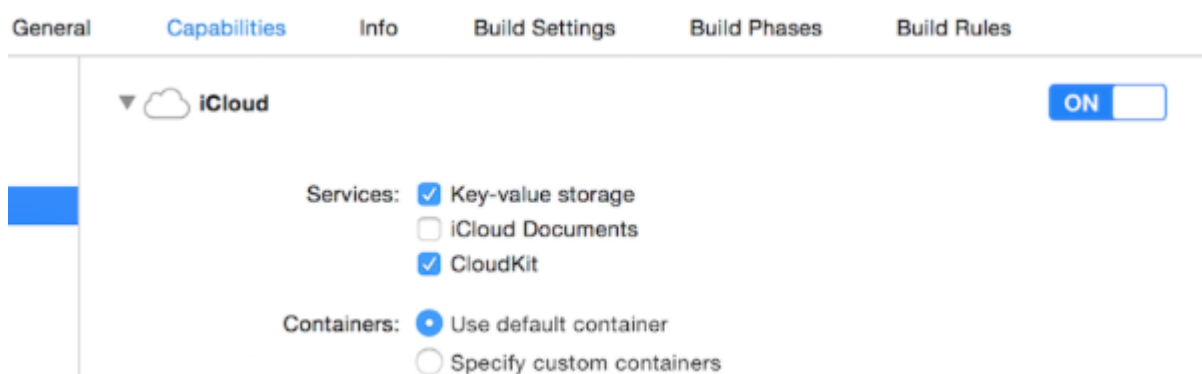
Examples

Registro de la aplicación para su uso con CloudKit

Lo que necesita es obtener un archivo de derechos para que la aplicación pueda acceder a su iCloud y escribir registros utilizando CloudKit.

Siga los pasos para otorgar acceso a iCloud desde su aplicación:

- 1- Seleccione el proyecto en el Project Navigator y luego abra la pestaña General.
- 2- En la sección Identidad, establezca su ID de desarrollador de Apple en el menú desplegable Equipo. (Si no está disponible, agréguelo al menú Xcode -> Preferencias -> Cuentas.
- 3- Ir a la pestaña Capacidades en las propiedades del proyecto y activar iCloud. Luego, seleccione "Key-Value Storage" y "CloudKit".



4- Asegúrate de que estos artículos estén marcados:

Steps: ✓ Add the "iCloud" entitlement to your App ID
✓ Add the "iCloud containers" entitlement to your App ID
✓ Add the "iCloud" entitlement to your entitlements file
✓ Link CloudKit.framework

Si todos los elementos están marcados, su aplicación está lista para usar CloudKit.

Utilizando CloudKit Dashboard

Todos los registros creados utilizando el código relacionado con CloudKit se pueden previsualizar, editar e incluso eliminar en CloudKit Dashboard. Para acceder a CloudKit Dashboard, vaya [aquí](#) .

Hay varias partes en el tablero de instrumentos:

- Tipos de registro (que se discutirán más adelante)
- Roles de seguridad (que es donde puede establecer bases de datos como públicas o privadas)
- Tipos de suscripción (que su aplicación podría registrar para [notificaciones automáticas de Apple \(APN\)](#) para notificarle cuando se modifica un registro)

Tipos de registro

Aquí, obtiene una lista de todos los tipos de registro existentes en la aplicación. Cuando abre CloudKit Dashboard por primera vez para una aplicación, hay un tipo de registro llamado Usuarios, que puede usar o simplemente borrarlo y usar el suyo propio.

En esta página, puede hacer que sus datos sean escritos manualmente. Por supuesto, en la mayoría de los casos esto no tiene sentido, porque el SDK de iOS puede manejarlo mucho mejor que el panel de control, pero la funcionalidad también está ahí si lo prefiere. El mayor uso de esta página es para previsualizar tipos.

Guardando datos en CloudKit

Para guardar la fecha en CloudKit, debemos hacer:

- A `CKRecordID` (la clave de su registro único)
- Un `CKRecord` (que incluye datos)

Haciendo una clave de registro

Para asegurarnos de que cada nuevo identificador de registro sea único, usamos la *marca de tiempo* actual, que es única. Obtenemos la marca de tiempo usando `NSDate`'s método `timeIntervalSinceReferenceDate()` . Está en la forma de `###.###` (# son números), que utilizaremos la parte entera. Para hacer esto, dividimos la cadena:

Rápido

```
let timestamp = String(format: "%f", NSDate.timeIntervalSinceReferenceDate())
let timestampParts = timestamp.componentsSeparatedByString(".")
let recordID = CKRecordID(recordName: timestampParts[0])
```

Haciendo el disco

Para hacer el registro, debemos especificar el tipo de registro (explicado en Uso de CloudKit Dashboard) como Usuarios, el ID como lo que hicimos en este momento y los datos. Aquí, agregaremos un texto de muestra, una imagen y la fecha actual al registro:

Rápido

```
let record = CKRecord(recordType: "Users", recordID: recordID)
record.setObject("Some Text", forKey: "text")
record.setObject(CKAsset(fileURL: someValidImageURL), forKey: "image")
record.setObject(NSDate(), forKey: "date")
```

C objetivo

```
CKRecord *record = [[CKRecord alloc] initWithRecordType: "Users" recordID: recordID];
[record setObject: "Some Text" forKey: "text"];
[record setObject: [CKAsset assetWithURL: someValidImageURL] forKey: "image"];
[record setObject: [[NSDate alloc] init] forKey: "date"];
```

Nota

Aquí, no agregamos el `UIImage` directamente al registro, porque, como se mencionó en Comentarios, el formato de la imagen no se admite directamente en CloudKit, por lo que hemos convertido `UIImage` en `CKAsset` .

Accediendo al contenedor

Rápido

```
let container = CKContainer.defaultContainer()
let database = container.privateCloudDatabase // or container.publicCloudDatabase
```

Guardando los registros en la base de datos

de CloudKit

Rápido

```
database.saveRecord(record, completionHandler: { (_, error) -> Void in
    print(error ?? "")
})
```

Lea CloudKit en línea: <https://riptutorial.com/es/ios/topic/4946/cloudkit>

Capítulo 36: Codificable

Introducción

Codable se agrega con Xcode 9, iOS 11 y Swift 4. Codable se usa para hacer que sus tipos de datos sean codificables y decodificables para la compatibilidad con representaciones externas como JSON.

Uso codificable para admitir tanto la codificación como la decodificación, declare la conformidad con Codable, que combina los protocolos Encodable y Decodable. Este proceso se conoce como hacer codificables tus tipos.

Examples

Uso de Codable con JSONEncoder y JSONDecoder en Swift 4

Tomemos un ejemplo con la estructura de la película, aquí hemos definido la estructura como codificable. Así, podemos codificarlo y decodificarlo fácilmente.

```
struct Movie: Codable {
    enum MovieGenre: String, Codable {
        case horror, skifi, comedy, adventure, animation
    }

    var name : String
    var moviesGenre : [MovieGenre]
    var rating : Int
}
```

Podemos crear un objeto a partir de la película como:

```
let upMovie = Movie(name: "Up", moviesGenre: [.comedy , .adventure, .animation], rating : 4)
```

El upMovie contiene el nombre "Up" y es movieGenre es comedia, aventura y animación que contiene 4 calificaciones de 5.

Codificar

JSONEncoder es un objeto que codifica instancias de un tipo de datos como objetos JSON. JSONEncoder soporta el objeto codificable.

```
// Encode data
let jsonEncoder = JSONEncoder()
do {
    let jsonData = try jsonEncoder.encode(upMovie)
    let jsonString = String(data: jsonData, encoding: .utf8)
    print("JSON String : " + jsonString!)
}
```

```
catch {  
}
```

JSONEncoder nos dará los datos JSON que se utilizan para recuperar la cadena JSON.

La cadena de salida será como:

```
{  
  "name": "Up",  
  "moviesGenre": [  
    "comedy",  
    "adventure",  
    "animation"  
  ],  
  "rating": 4  
}
```

Descodificar

JSONDecoder es un objeto que decodifica instancias de un tipo de datos de objetos JSON. Podemos recuperar el objeto de la cadena JSON.

```
do {  
  // Decode data to object  
  
  let jsonDecoder = JSONDecoder()  
  let upMovie = try jsonDecoder.decode(Movie.self, from: jsonData)  
  print("Rating : \(upMovie.name)")  
  print("Rating : \(upMovie.rating)")  
}  
catch {  
}
```

Al decodificar JSONData, recibiremos el objeto Movie nuevamente. Así podemos obtener todos los valores que se guardan en ese objeto.

La salida será como:

```
Name : Up  
Rating : 4
```

Lea Codificable en línea: <https://riptutorial.com/es/ios/topic/10639/codificable>

Capítulo 37: Codificación del valor clave- Observación del valor clave

Observaciones

KVC : - Codificación Clave-Valor

Normalmente se accede a las variables de instancia a través de propiedades o accesores, pero KVC ofrece otra forma de acceder a las variables en forma de cadenas. De esta manera, su clase actúa como un diccionario y el nombre de su propiedad, por ejemplo, "age" se convierte en clave y el valor que posee la propiedad se convierte en valor para esa clave.

```
For example, you have employee class with "age" property. Normally we access like this.  
emp.age = @"20";  
NSString age = emp.age;  
  
But KVC works like this:  
[emp valueForKey:@"age"];  
[emp setValue:@"25" forKey:@"age"];
```

KVO : - Key-Value Observer

El mecanismo a través del cual se notifica a los objetos cuando hay un cambio en cualquiera de las propiedades se llama KVO. Ex .: teclado de notificación

Por ejemplo, el objeto de persona está interesado en recibir una notificación cuando la propiedad accountBalance se cambia en el objeto BankAccount. Para lograr esto, Person Object debe registrarse como observador de la propiedad AccountBalance de BankAccount mediante el envío de un addObserver: forKeyPath: options: context: message.

Examples

Uso del contexto para la observación KVO

```
-(void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object  
change:(NSDictionary<NSString *,id> *)change context:(void *)context
```

El contexto es importante si envía su clase para que otros la usen. El contexto le permite a su observador de la clase verificar que es su observador el que está siendo llamado.

El problema de no pasar un observador es que, si alguna subclase de su clase registra un observador para el mismo objeto, la misma clave y no pasa un contexto, entonces se puede llamar al observador de la superclase varias veces.

Una variable que es única e interna para su uso es un buen contexto.

Para más información.

[importancia y buen contexto](#)

Observando una propiedad de una subclase NSObject

La mayoría de las funciones KVO y KVC ya están implementadas de forma predeterminada en todas `NSObject` subclases de `NSObject` .

Para comenzar a observar una propiedad llamada `firstName` de un objeto llamado `personObject` haga esto en la clase de observación:

```
[personObject addObserver:self
                forKeyPath:@"firstName"
                options:NSKeyValueObservingOptionNew
                context:nil];
```

El objeto al que `self` refiere el código anterior recibirá un `observeValueForKeyPath:ofObject:change:context:` cada vez que cambie la ruta clave observada.

```
- (void)observeValueForKeyPath:(NSString *)keyPath
    ofObject:(id)object
    change:(NSDictionary<NSString *,id> *)change
    context:(void *)context
{
    NSLog(@"new value of %@ is: %@", keyPath, change[NSKeyValueChangeNewKey]);
}
```

"Ruta clave" es un término KVC. `NSObject` subclases de `NSObject` implementan la funcionalidad KVC por defecto.

Se podrá acceder a una variable de instancia llamada `_firstName` mediante la ruta de la clave `@"firstName"` .

Cuando se acceda a la ruta de acceso `@"firstName"` , se `_firstName` un método getter llamado `firstName` , independientemente de que haya una variable de instancia `setFirstName` o `setFirstName` método `setFirstName` setter.

[Lea Codificación del valor clave-Observación del valor clave en línea:](#)

<https://riptutorial.com/es/ios/topic/3493/codificacion-del-valor-clave-observacion-del-valor-clave>

Capítulo 38: Comprobando la conectividad de la red

Observaciones

El código fuente de `Reachability.h` `Reachability.m` se puede encontrar en el [sitio de documentación](#) para desarrolladores de Apple.

Advertencias

A diferencia de otras plataformas, Apple aún debe proporcionar un conjunto estándar de API para determinar el estado de la red de un dispositivo iOS y ofrecer solo estos ejemplos de código vinculados anteriormente. El archivo de origen cambia con el tiempo, pero una vez importados en un proyecto de aplicación, los desarrolladores rara vez los actualizan.

Por esta razón, la mayoría de los desarrolladores de aplicaciones tienden a usar una de las muchas bibliotecas mantenidas por Github / [Cocoapod](#) para [lograr](#) accesibilidad.

Apple también recomienda, para las solicitudes realizadas a instancias del usuario, que [siempre intente una conexión primero](#), antes de usar `Reachability` / `SCNetworkReachability` para [diagnosticar la falla o esperar a que la conexión regrese](#).

Examples

Creando un oyente de Alcance

La clase de [Alcance](#) de Apple verifica periódicamente el estado de la red y alerta a los observadores sobre los cambios.

```
Reachability *internetReachability = [Reachability reachabilityForInternetConnection];
[internetReachability startNotifier];
```

Agregar observador a los cambios de red

`Reachability` utiliza los mensajes de `NSNotification` para alertar a los observadores cuando el estado de la red ha cambiado. Tu clase tendrá que convertirse en un observador.

```
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(reachabilityChanged:) name:kReachabilityChangedNotification object:nil];
```

En otro lugar de tu clase, implementa la firma del método

```
- (void) reachabilityChanged:(NSNotification *)note {
    //code which reacts to network changes
```



```
}
```

Alerta cuando la red deja de estar disponible

```
- (void)reachabilityChanged:(NSNotification *)note {
    Reachability* reachability = [note object];
    NetworkStatus netStatus = [reachability currentReachabilityStatus];

    if (netStatus == NotReachable) {
        NSLog(@"Network unavailable");
    }
}
```

Alerta cuando la conexión se convierte en una red WIFI o celular.

```
- (void)reachabilityChanged:(NSNotification *)note {
    Reachability* reachability = [note object];
    NetworkStatus netStatus = [reachability currentReachabilityStatus];

    switch (netStatus) {
        case NotReachable:
            NSLog(@"Network unavailable");
            break;
        case ReachableViaWWAN:
            NSLog(@"Network is cellular");
            break;
        case ReachableViaWiFi:
            NSLog(@"Network is WIFI");
            break;
    }
}
```

Verificar si está conectado a la red.

Rápido

```
import SystemConfiguration

/// Class helps to code reuse in handling internet network connections.
class NetworkHelper {

    /**
     Verify if the device is connected to internet network.
     - returns:         true if is connected to any internet network, false if is not
                       connected to any internet network.
     */
    class func isConnectedToNetwork() -> Bool {
        var zeroAddress = sockaddr_in()

        zeroAddress.sin_len = UInt8(sizeofValue(zeroAddress))
        zeroAddress.sin_family = sa_family_t(AF_INET)

        let defaultRouteReachability = withUnsafePointer(&zeroAddress) {
            SCNetworkReachabilityCreateWithAddress(nil, UnsafePointer($0))
        }
    }
}
```

```

var flags = SCNetworkReachabilityFlags()

if !SCNetworkReachabilityGetFlags(defaultRouteReachability!, &flags) {
    return false
}

let isReachable = (flags.rawValue & UInt32(kSCNetworkFlagsReachable)) != 0
let needsConnection = (flags.rawValue & UInt32(kSCNetworkFlagsConnectionRequired)) != 0

return (isReachable && !needsConnection)
}

}

if NetworkHelper.isConnectedToNetwork() {
    // Is connected to network
}

```

C objetivo:

Podemos verificar la conectividad de la red dentro de algunas líneas de código como:

```

-(BOOL)isConntectedToNetwork
{
    Reachability *networkReachability = [Reachability reachabilityForInternetConnection];
    NetworkStatus networkStatus = [networkReachability currentReachabilityStatus];
    if (networkStatus == NotReachable)
    {
        NSLog(@"There IS NO internet connection");
        return false;
    } else
    {
        NSLog(@"There IS internet connection");
        return true;
    }
}

```

Lea [Comprobando la conectividad de la red en línea:](https://riptutorial.com/es/ios/topic/704/comprobando-la-conectividad-de-la-red)

<https://riptutorial.com/es/ios/topic/704/comprobando-la-conectividad-de-la-red>

Capítulo 39: Comprobando la versión de iOS

Examples

iOS 8 y versiones posteriores

Swift 3:

```
let minimumVersion = OperatingSystemVersion(majorVersion: 8, minorVersion: 1, patchVersion: 2)
if ProcessInfo().isOperatingSystemAtLeast(minimumVersion) {
    //current version is >= (8.1.2)
} else {
    //current version is < (8.1.2)
}
```

Comparar versiones

```
let minimumVersionString = "3.1.3"
let versionComparison = UIDevice.current.systemVersion.compare(minimumVersionString, options:
.numeric)
switch versionComparison {
    case .orderedSame, .orderedDescending:
        //current version is >= (3.1.3)
        break
    case .orderedAscending:
        //current version is < (3.1.3)
        fallthrough
    default:
        break;
}
```

C objetivo

```
NSString *version = @"3.1.3";
NSString *currentVersion = @"3.1.1";
NSComparisonResult result = [currentVersion compare:version options:NSNumericSearch];
switch(result) {
    case: NSOrderedAscending:
        //less than the current version
        break;
    case: NSOrderedDescending:
    case: NSOrderedSame:
        // equal or greater than the current version
        break;
}
```

Swift 2.0 y versiones posteriores

```
if #available(iOS 9, *) {
```

```
// iOS 9
} else {
// iOS 8 or earlier
}
```

Versión del dispositivo iOS

Esto le dará la versión actual del sistema.

C objetivo

```
NSString *version = [[UIDevice currentDevice] systemVersion]
```

Rápido

```
let version = UIDevice.currentDevice().systemVersion
```

Swift 3

```
let version = UIDevice.current.systemVersion
```

Lea [Comprobando la versión de iOS en línea](https://riptutorial.com/es/ios/topic/2194/comprobando-la-version-de-ios):

<https://riptutorial.com/es/ios/topic/2194/comprobando-la-version-de-ios>

Capítulo 40: Concurrencia

Introducción

Tema relacionado: [Grand Central Dispatch](#)

Sintaxis

- `dispatch_async`: ejecuta un bloque de código en una cola separada y no detiene la cola actual. Si la cola está en un hilo diferente al que se llamó `dispatch_async`, el código en el bloque se ejecutará mientras que el código después de que `dispatch_async` también se ejecute
- `dispatch_sync` - Se ejecuta un bloque de código en una cola separada, y se detiene la cola actual. Si la cola está en un subproceso diferente al que se llamó `dispatch_async`, el código en el bloque se ejecutará, y la ejecución en el subproceso donde se llamó el método solo se reanudará después de que finalice

Parámetros

cola	<p>La cola en la que se ejecutará el código en el bloque de envío. Una <i>cola</i> es como (pero no exactamente igual que) un hilo; El código en diferentes colas puede ejecutarse en paralelo. Use <code>dispatch_get_main_queue</code> para obtener la cola para el hilo principal Para crear una nueva cola, que a su vez crea un nuevo hilo, use <code>dispatch_queue_create("QUEUE_NAME", DISPATCH_QUEUE_CONCURRENT)</code> . El primer parámetro es el nombre de la cola, que se muestra en el depurador si hace una pausa mientras el bloque todavía se está ejecutando. El segundo parámetro no importa a menos que desee utilizar la misma cola para múltiples llamadas <code>dispatch_async</code> o <code>dispatch_sync</code> . Describe lo que sucede cuando otro bloque se coloca en la misma cola; <code>DISPATCH_QUEUE_CONCURRENT</code> hará que ambos bloques se ejecuten al mismo tiempo, mientras que <code>DISPATCH_QUEUE_SERIAL</code> hará que el segundo bloque espere a que finalice el primer bloque.</p>
bloquear	<p>El código en este bloque se ejecutará en la cola de <code>queue</code> ; ponga el código que desea ejecutar en la cola separada aquí. Un consejo útil: si está escribiendo esto en Xcode y el argumento del bloque tiene el contorno azul a su alrededor, haga doble clic en el argumento y Xcode creará automáticamente un bloque vacío (esto se aplica a todos los argumentos del bloque en cualquier función o método)</p>

Observaciones

Siempre que haga algo en un subproceso separado, lo que ocurre cuando se usan colas, es importante mantener la seguridad de los subprocesos. Es posible que algunos métodos, en

particular los de `UIView` s, no funcionen y / o se bloqueen en hilos distintos al hilo principal. Además, asegúrese de no cambiar nada (variables, propiedades, etc.) que también se esté utilizando en el hilo principal, a menos que tenga en cuenta este cambio.

Examples

Ejecutar código simultáneamente: ejecutar código mientras se ejecuta otro código

Digamos que quieres actuar en acción (en este caso, registrar "Foo"), mientras haces otra cosa (registrar "Barra"). Normalmente, si no usa la concurrencia, una de estas acciones se ejecutará completamente, y la otra ejecución se ejecutará solo después de que haya finalizado por completo. Pero con la concurrencia, puede hacer que ambas acciones se ejecuten al mismo tiempo:

```
dispatch_async(dispatch_queue_create("Foo", DISPATCH_QUEUE_CONCURRENT), ^{
    for (int i = 0; i < 100; i++) {
        NSLog(@"Foo");
        usleep(100000);
    }
});

for (int i = 0; i < 100; i++) {
    NSLog(@"Bar");
    usleep(50000);
}
```

Esto registrará "Foo" 100 veces, haciendo una pausa de 100 ms cada vez que se registre, pero hará todo esto en un hilo separado. Mientras se está registrando `Foo`, "Barra" también se registrará en intervalos de 50 ms, al mismo tiempo. Idealmente debería ver una salida con "Foo" y "Barras" mezcladas

Ejecutando en el hilo principal

Cuando se realizan tareas de forma asíncrona, normalmente se convierte en una necesidad de garantizar que se ejecute un fragmento de código en el hilo principal. Por ejemplo, puede querer golpear una API REST de forma asíncrona, pero poner el resultado en una `UILabel` en la pantalla. Antes de actualizar `UILabel`, debe asegurarse de que su código se ejecute en el hilo principal:

```
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
    //Perform expensive tasks
    //...

    //Now before updating the UI, ensure we are back on the main thread
    dispatch_async(dispatch_get_main_queue(), ^{
        label.text = //....
    });
})
```

Siempre que actualice las vistas en la pantalla, asegúrese de hacerlo en el hilo principal, de lo

contrario podría ocurrir un comportamiento indefinido.

Grupo de despacho - esperando otros hilos completados.

```
dispatch_group_t preapreWaitingGroup = dispatch_group_create();

dispatch_group_enter(preapreWaitingGroup);
[self doAsynchronousTaskWithComplete:^(id someResults, NSError *error) {
    // Notify that this task has been completed.
    dispatch_group_leave(preapreWaitingGroup);
}]

dispatch_group_enter(preapreWaitingGroup);
[self doOtherAsynchronousTaskWithComplete:^(id someResults, NSError *error) {
    dispatch_group_leave(preapreWaitingGroup);
}]

dispatch_group_notify(preapreWaitingGroup, dispatch_get_main_queue(), ^{
    // This block will be executed once all above threads completed and call
    dispatch_group_leave
    NSLog(@"Prepare completed. I'm readyyyy");
});
```

Actualización 1. Swift 3 versión.

```
let prepareGroup = DispatchGroup()
prepareGroup.enter()
doAsynchronousTaskWithComplete() { (someResults, error) in
    // Notify that this task has been completed.
    prepareGroup.leave()
}

prepareGroup.enter()
doOtherAsynchronousTaskWithComplete() { (someResults, error) in
    // Notify that this task has been completed.
    prepareGroup.leave()
}

prepareGroup.notify(queue: DispatchQueue.main) {
    // This block will be executed once all above threads completed and call
    dispatch_group_leave
    print("Prepare completed. I'm readyyyy")
}
```

Lea Concurrency en línea: <https://riptutorial.com/es/ios/topic/1090/concurrency>

Capítulo 41: Configuración de iOS de Cartago

Examples

Instalación de Cartago Mac

Configuración de cartago

Descargue la última versión de Cartago desde el enlace dado Enlace de [descarga](#)

Abajo en la sección de descargas, descargue el archivo **Carthage.pkg** .

Una vez que se complete la descarga, instálela haciendo doble clic en el archivo pkg de descarga.

Para comprobar si la descarga se realizó correctamente, ejecute el siguiente comando en la versión de `0.18-19-g743fa0f` su Terminal. Esto debería `0.18-19-g743fa0f` la versión instalada como `0.18-19-g743fa0f`

Lea Configuración de iOS de Cartago en línea:

<https://riptutorial.com/es/ios/topic/7404/configuracion-de-ios-de-cartago>

Capítulo 42: Configurar balizas con CoreBluetooth

Introducción

Caliente para leer y escribir datos en un dispositivo bluetooth de baja energía.

Observaciones

Algunos puntos importantes

- No se necesitan capacidades.
- Los bytes de la tienda de iPhone en formato Little Endian, así que compruebe si el accesorio Bluetooth también usa Little Endian. Ejemplo:
 - Intel CPU usualmente usa little endian.
 - La arquitectura ARM era little-endian antes de la versión 3 cuando se convirtió en big-endian.
- Después de una operación única o por lotes, la conexión se perderá, por lo que debe volver a conectarse antes de continuar.

Escanear para UUID DE SERVICIO

```
func SearchBLE() {
    cb_manager.scanForPeripherals(withServices:[service_uuid], options: nil)
    StopSearchBLE()
}
```

Cómo descubrir el UUID de SERVICIO sin documentación.

```
func centralManager(_ central: CBCentralManager, didConnect peripheral:
CBPeripheral) {
    peripheral.delegate = self
    peripheral.discoverServices(nil)
}

func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
    for service in peripheral.services! {
        print("Service: \(service)\n error: \(error)")
    }
}
```

- `discoverServices (nil)` - NIL significa que se devolverán todos los servicios, lo que no es una buena opción (LEA Comentarios 3)

- Si no ha encontrado el UUID DE SERVICIO, ejecute su código y busque en la consola

```
Service: <CBService: 0x171e75280, isPrimary = YES, UUID = Battery>
error: nil
Service: <CBService: 0x171e74c40, isPrimary = YES, UUID = Device Information>
error: nil
Service: <CBService: 0x171e75300, isPrimary = YES, UUID = FFF0>
error: nil
```

- Encontré tener 3 servicios: Batería, información del dispositivo (Firmware) y FFF0
- Este servicio de uuid no es estándar, puede encontrar una lista de estándares [aquí](#)
- FFF0 es el UUID DE SERVICIO en este caso

Convertir datos a UInt16 y contrario.

Agrega estas extensiones a tu clase

```
protocol DataConvertible {
    init?(data: Data)
    var data: Data { get }
}

extension DataConvertible {

    init?(data: Data) {
        guard data.count == MemoryLayout<Self>.size else { return nil }
        self = data.withUnsafeBytes { $0.pointee }
    }

    var data: Data {
        var value = self
        return Data(buffer: UnsafeBufferPointer(start: &value, count: 1))
    }
}

extension UInt16 : DataConvertible {
    init?(data: Data) {
        guard data.count == MemoryLayout<UInt16>.size else { return nil }
        self = data.withUnsafeBytes { $0.pointee }
    }

    var data: Data {
        var value = CFSwapInt16HostToBig(self)
        return Data(buffer: UnsafeBufferPointer(start: &value, count: 1))
    }
}
```

Examples

Mostrando nombres de todos los Bluetooth de baja energía (BLE)

- Para este ejemplo, tengo una sala controlada con un solo dispositivo BLE habilitado.
- Su clase debe extender CBCentralManagerDelegate.
- Implementar el método: centralManagerDidUpdateState (_ central: CBCentralManager).
- Use la cola global para no congelar la pantalla mientras busca un dispositivo.

- Cree una instancia de CBCentralManager y espere la devolución de llamada centralManagerDidUpdateState respuesta.

```
class BLEController: CBCentralManagerDelegate{
var cb_manager: CBCentralManager!
var bles : [CBPeripheral] = []

    override func viewDidLoad() {
        super.viewDidLoad()
        cb_manager = CBCentralManager(delegate: self, queue: DispatchQueue.global())
    }

    func centralManagerDidUpdateState(_ central: CBCentralManager) {
        print("UPDATE STATE - \(central)")
    }
}
```

La devolución de llamada a centralManagerDidUpdateState indica que CoreBluetooth está listo, por lo que puede buscar BLE ahora. Actualice el código centralManagerDidUpdateState para buscar todos los dispositivos BLE cuando esté listo.

```
func centralManagerDidUpdateState(_ central: CBCentralManager) {
    print("UPDATE STATE - \(central)")
    SearchBLE()
}

func SearchBLE() {
    cb_manager.scanForPeripherals(withServices: nil, options: nil)
    StopSearchBLE()
}

func StopSearchBLE() {
    let when = DispatchTime.now() + 5 // change 5 to desired number of seconds
    DispatchQueue.main.asyncAfter(deadline: when) {
        self.cb_manager.stopScan()
    }
}
```

- SearchBLE () busca dispositivos BLE y deja de buscar después de 5s
- cb_manager.scanForPeripherals (withServices: nil, options: nil) busca todos los BLE que se encuentren en el rango con usted.
- StopSearchBLE () detendrá la búsqueda después de 5s.
- Cada BLE encontrado devolverá call a func centralManager (_ central: CBCentralManager, didDescubrir periférico: CBPeripheral, advertisementData: [String: Any], rssi RSSI: NSNumber)

```
func centralManager(_ central: CBCentralManager, didDiscover peripheral:
    CBPeripheral, advertisementData: [String : Any], rssi RSSI: NSNumber) {
    guard let name = peripheral.name else {
        return
    }
    print(name)
    bles.append(peripheral)
```

```
}
```

Conectar y leer valor mayor

- Estoy en una habitación controlada con una única baliza de mina que usa el protocolo IBEACON.
- BLEController necesita extender CBPeripheralDelegate
- Usaré el primer BLE para conectar después de que la búsqueda haya terminado.
- Modificar el método StopSearchBLE ()

```
class BLEController: CBCentralManagerDelegate, CBPeripheralDelegate{
//...
    func StopSearchMiniewBeacon() {
        let when = DispatchTime.now() + 5 // change 2 to desired number of seconds
        DispatchQueue.main.asyncAfter(deadline: when) {
            self.cb_manager.stopScan()
            self.cb_manager.connect(bles.first)
        }
    }
}
/...
}
```

- En la documentación de su dispositivo BLE, debe buscar el UUID DE SERVICIO y la CARACTERÍSTICA PRINCIPAL DE UUID

```
var service_uuid = CBUUID(string: "0000fff0-0000-1000-8000-00805f9b34fb")
var major_uuid = CBUUID(string: "0000fff2-0000-1000-8000-00805f9b34fb")
func centralManager(_ central: CBCentralManager, didConnect peripheral:
CBPeripheral) {
    peripheral.delegate = self
    peripheral.discoverServices([service_uuid])
}

func peripheral(_ peripheral: CBPeripheral, didDiscoverServices error: Error?) {
    print("Service: \(service)\n error: \(error)")
    peripheral.discoverCharacteristics([major_uuid], for: (peripheral.services?[0]!))
}
```

- Cree una variable como 'service_uuid' y 'major_uuid' como el código anterior. '-0000-1000-8000-00805f9b34fb' es parte de la norma. 'fff0' es mi UUID DE SERVICIO, 'fff2' es mi característica de UUID PRINCIPAL y se requiere '0000' para completar el bloque de 4 bytes uuid 1º.
- discoverCharacteristics ([major_uuid], for: (peripheral.services?[0]!) obtendrá la característica principal de mi servidor de dispositivos gatt y tendrá NIL como valor por ahora.
- (periferico.servicios? icono0)! - 0 porque se devolverá un solo valor una vez que hice peripheral.discoverServices ([service_uuid])

```
func peripheral(_ peripheral: CBPeripheral, didDiscoverCharacteristicsFor service: CBService,
error: Error?) {
    for characteristic in service.characteristics! {
        print("Characteristic: \(characteristic)\n error: \(error)")
        if(characteristic.uuid.uuidString == "FFF2"){
```

```

        peripheral.readValue(for: characteristic)
    }
}

func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBCharacteristic, error: Error?) {
    print("Characteristic read: \(characteristic)\n error: \(error)")
    let major = UInt16.init(bigEndian: UInt16(data: characteristic.value!))
    print("major: \(major)")
}

```

- El valor de la característica solo se podrá leer después de la llamada `peripheral.readValue` (para: característica)
- `readValue` dará como resultado una función periférica (`_ periférica: CBPeripheral`, `didUpdateValueFor feature: CBCaracterística`, `error: ¿Error?`) con valor en Tipo de datos.

Escribe valor mayor

- Necesitas descubrir los servicios y características.
- No necesita leer el valor de la característica antes de escribir sobre ella.
- continuará por, para este ejemplo, después de leer el valor. Modificar la función del periférico (`_ periférico: CBPeripheral`, `didUpdateValueFor feature: CBCaracterística`, `error: ¿Error?`)
- Agrega una variable `new_major` y `reset_characteristic`

```

var reset_characteristic : CBCharacteristic!
func peripheral(_ peripheral: CBPeripheral, didDiscoverCharacteristicsFor service: CBService,
error: Error?) {
    for characteristic in service.characteristics! {
        print("Characteristic: \(characteristic)\n error: \(error)")
        if(characteristic.uuid.uuidString == "FFF2"){
            peripheral.readValue(for: characteristic)
        }
        if(characteristic.uuid.uuidString == "FFFF"){
            reset_characteristic = characteristic
        }
    }
}

let new_major : UInt16 = 100
func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBCharacteristic, error: Error?) {
    print("Characteristic read: \(characteristic)\n error: \(error)")
    let major = UInt16.init(bigEndian: UInt16(data: characteristic.value!))
    print("major: \(major)")
    peripheral.writeValue(new_major.data, for: characteristic, type:
CBCharacteristicWriteType.withResponse)
}

```

- El iPhone de default enviará y recibirá bytes en formato Little Endian, pero mi dispositivo MINEW con el conjunto de chips NRF51822 tiene architecture ARM y necesita bytes en formato Big Endian, así que tengo que intercambiarlo.
- La documentación del dispositivo BLE le dirá qué tipo de entrada y salida tendrá cada característica y si puede leerla como arriba (`CBCharacteristicWriteType.withResponse`).

```

func peripheral(_ peripheral: CBPeripheral, didWriteValueFor characteristic: CBCharacteristic,
error: Error?) {
    print("Characteristic write: \(characteristic)\n error: \(error)")
    if(characteristic.uuid.uuidString == "FFF2"){
        print("Resetting")
        peripheral.writeValue("minew123".data(using: String.Encoding.utf8)!, for:
reset_characteristic, type: CBCharacteristicWriteType.withResponse)
    }
    if(characteristic.uuid.uuidString == "FFFF"){
        print("Reboot finish")
        cb_manager.cancelPeripheralConnection(peripheral)
    }
}
}

```

- Para actualizar la información de un servidor gatt, debe reiniciarlo mediante programación o guardar datos en él, apagarlo y encenderlo manualmente.
- Es característico el FFFF que lo hacen en este dispositivo.
- 'minew123' es la contraseña predeterminada para reiniciar o guardar información en este caso.
- ejecute su aplicación y observe su consola por cualquier error, espero que ninguno, pero aún no verá el nuevo valor.

```

func peripheral(_ peripheral: CBPeripheral, didUpdateValueFor characteristic:
CBCharacteristic, error: Error?) {
    print("Characteristic read: \(characteristic)\n error: \(error)")
    let major = UInt16.init(bigEndian: UInt16(data: characteristic.value!))
    print("major: \(major)")
    //peripheral.writeValue(new_major.data, for: characteristic, type:
CBCharacteristicWriteType.withResponse)
}
}

```

- El último paso es comentar la última línea del método didUpdateValueFor y volver a ejecutar la aplicación, ahora tendrá el nuevo valor.

Lea [Configurar balizas con CoreBluetooth en línea:](https://riptutorial.com/es/ios/topic/9488/configurar-balizas-con-corebluetooth)

<https://riptutorial.com/es/ios/topic/9488/configurar-balizas-con-corebluetooth>

Capítulo 43: Control UISegmentado

Introducción

Un objeto `UISegmentedControl` es un control horizontal formado por varios segmentos, cada uno de los cuales funciona como un botón discreto. Un control segmentado proporciona un medio compacto para agrupar varios controles.

Examples

Creación de `UISegmentedControl` mediante código

1. Cree una nueva instancia de `UISegmentedControl` con 3 elementos (segmentos):

```
let mySegmentedControl = UISegmentedControl (items: ["One", "Two", "Three"])
```

2. Marco de configuración;

```
mySegmentedControl.frame = CGRect(x: 0.0, y: 0.0, width: 300, height: 50)
```

3. Haga la selección por defecto (no que los segmentos estén indexados por 0):

```
mySegmentedControl.selectedSegmentIndex = 0
```

4. Configurar destino:

```
mySegmentedControl.addTarget(self, action: #selector(segmentedValueChanged(_:)), for: .valueChanged)
```

- 5 El valor de la manija cambió:

```
func segmentedValueChanged(_ sender:UISegmentedControl!) {  
    print("Selected Segment Index is : \(sender.selectedSegmentIndex)")  
}
```

6. Agregar `UISegmentedControl` a la jerarquía de vistas

```
yourView.addSubview(mySegmentedControl)
```

Lea `Control UISegmentado` en línea: <https://riptutorial.com/es/ios/topic/9963/control-uisegmentado>

Capítulo 44: Convertir HTML a cadena NSAttributedString y viceversa

Examples

Código de Objective C para convertir la cadena HTML a NSAttributedString y Vice Versa

Código de conversión de HTML a NSAttributedString: -

```
//HTML String
NSString *htmlString=[[NSString alloc]initWithFormat:@"<!DOCTYPE html><html><body><h1>My
First Heading</h1><p>My first paragraph.</p></body></html>"];
//Converting HTML string with UTF-8 encoding to NSAttributedString
NSAttributedString *attributedString = [[NSAttributedString alloc]
initWithData: [htmlString
dataUsingEncoding:NSUTF8StringEncoding]
options: @{ NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType }
documentAttributes: nil
error: nil ];
```

Conversión de NSAttributedString a HTML: -

```
//Dictionary to hold all the attributes of NSAttributedString
NSDictionary *documentAttributes = @{NSDocumentTypeDocumentAttribute:
NSHTMLTextDocumentType};
//Saving the NSAttributedString with all its attributes as a NSData Entity
NSData *htmlData = [attributedString dataFromRange:NSMakeRange(0, attributedString.length)
documentAttributes:documentAttributes error:NULL];
//Convert the NSData into HTML String with UTF-8 Encoding
NSString *htmlString = [[NSString alloc] initWithData:htmlData
encoding:NSUTF8StringEncoding];
```

Lea Convertir HTML a cadena NSAttributedString y viceversa en línea:

<https://riptutorial.com/es/ios/topic/7225/convertir-html-a-cadena-nsattributed-y-viceversa>

Capítulo 45: Convertir NSAttributedString a UIImage

Examples

Conversión de NSAttributedString a UIImage

C objetivo

```
NSMutableAttributedString *str = [[NSMutableAttributedString alloc] initWithString:@"Hello.
That is a test attributed string."];
[str addAttribute:NSBackgroundColorAttributeName value:[UIColor yellowColor]
range:NSMakeRange(3, 5)];
[str addAttribute:NSForegroundColorAttributeName value:[UIColor greenColor]
range:NSMakeRange(10, 7)];
[str addAttribute:NSFontAttributeName value:[UIFont fontWithName:@"HelveticaNeue-Bold"
size:20.0] range:NSMakeRange(20, 10)];
UIImage *customImage = [self imageFromAttributedString:str];
```

La función `imageFromAttributedString` es como se define a continuación:

```
- (UIImage *)imageFromAttributedString:(NSAttributedString *)text
{
    UIGraphicsBeginImageContextWithOptions(text.size, NO, 0.0);

    // draw in context
    [text drawAtPoint:CGPointMake(0.0, 0.0)];

    // transfer image
    UIImage *image = [UIGraphicsGetImageFromCurrentImageContext()
imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
    UIGraphicsEndImageContext();

    return image;
}
```

Lea [Convertir NSAttributedString a UIImage en línea](https://riptutorial.com/es/ios/topic/7242/convertir-nsattributedString-a-UIImage):

<https://riptutorial.com/es/ios/topic/7242/convertir-nsattributedString-a-UIImage>

Capítulo 46: Core Graphics

Examples

Creando un Contexto de Core Graphics

Contexto de Core Graphics

Un contexto de Core Graphics es un lienzo que podemos dibujar en él y establecer algunas propiedades como el grosor de línea.

Haciendo un contexto

Para hacer un contexto, usamos la función `UIGraphicsBeginImageContextWithOptions()`. Luego, cuando hayamos terminado con el dibujo, solo llamamos a `UIGraphicsEndImageContext()` para finalizar el contexto:

Rápido

```
let size = CGSize(width: 256, height: 256)

UIGraphicsBeginImageContextWithOptions(size, false, 0)

let context = UIGraphicsGetCurrentContext()

// drawing code here

UIGraphicsEndImageContext()
```

C objetivo

```
CGSize size = [CGSize width:256 height:256];

UIGraphicsBeginImageContextWithOptions(size, NO, 0);

CGContext *context = UIGraphicsGetCurrentContext();

// drawing code here

UIGraphicsEndImageContext();
```

En el código anterior, pasamos 3 parámetros a la función

`UIGraphicsBeginImageContextWithOptions()` :

1. Un objeto `CGSize` que almacena todo el tamaño del contexto (el lienzo)

2. Un valor booleano que si es verdadero, el contexto será opaco.
3. Un valor entero que establece la escala (1 para las pantallas sin retina, 2 para la retina y 3 para la retina HD). Si se establece en 0, el sistema maneja automáticamente la escala según el dispositivo de destino.

Presentando el Lienzo Dibujado al Usuario

Rápido

```
let image = UIGraphicsGetImageFromCurrentImageContext()  
imageView.image = image //assuming imageView is a valid UIImageView object
```

C objetivo

```
UIImage *image = UIGraphicsGetImageFromCurrentImageContext();  
imageView.image = image; //assuming imageView is a valid UIImageView object
```

Lea Core Graphics en línea: <https://riptutorial.com/es/ios/topic/5530/core-graphics>

Capítulo 47: Core Motion

Examples

Accediendo al barómetro para obtener la altitud relativa.

Rápido

Importe la biblioteca Core Motion:

```
import CoreMotion
```

A continuación, necesitamos crear un objeto `CMAltimeter`, pero un `CMAltimeter` común es crearlo en `viewDidLoad()`. Si se hace de esa manera, no se podrá acceder al altímetro cuando necesitamos llamar a un método. Sin embargo, siga adelante y cree su objeto `CMAltimeter` justo antes de `viewDidLoad()`:

```
let altimeter = CMAltimeter()
```

Ahora:

1. Necesitamos verificar si `relativeAltitude` está disponible incluso con el siguiente método:
`CMAltimeter.isRelativeAltitudeAvailable`.
2. Si eso se vuelve `true`, entonces puede comenzar a monitorear el cambio de altitud con `startRelativeAltitudeUpdatesToQueue`.
3. Si no hay errores, debería poder recuperar los datos de las propiedades `relativeAltitude` y `Presión`.

A continuación se muestra la definición de una acción de botón para comenzar a monitorear con nuestro barómetro.

```
@IBAction func start(sender: AnyObject){
    if CMAltimeter.isRelativeAltitudeAvailable() {
        // 2
        altimeter.startRelativeAltitudeUpdatesToQueue(NSOperationQueue.mainQueue(), withHandler: {
            data, error in
                // 3
                if (error == nil) {
                    println("Relative Altitude: \(data.relativeAltitude)")
                    println("Pressure: \(data.pressure)")
                }
            })
    }
}
```

Lea Core Motion en línea: <https://riptutorial.com/es/ios/topic/7636/core-motion>

Capítulo 48: Core Spotlight en iOS

Examples

Core-Spotlight

Objetivo

1. Cree un nuevo proyecto de iOS y agregue el marco *CoreSpotlight* y *MobileCoreServices* a su proyecto.



General

Capabilities

PROJECT



CoreSpotlighSample

TARGETS



CoreSpotlighSample



CoreSpotlighSampl...



CoreSpotlighSampl...



▶ **Target Dependencies (0 it**

▶ **Compile Sources (3 item...**

▼ **Link Binary With Libraries**

Name



CoreSp



Mobile



▶ **Copy Bundle Resources (4**

2. Cree el objeto real de búsqueda de CSS y asocie el identificador único, el identificador de dominio y el conjunto de atributos. Finalmente, indexe el CSSearchableItem usando `[[CSSearchableIndex defaultSearchableIndex] ...]` como se muestra a continuación.

```

4 //
5 // Created by Mayqiyue on 7/10/15.
6 // Copyright © 2015 mayqiyue. All rights reserved.
7 //
8
9 #import "ViewController.h"
10 #import <CoreSpotlight/CoreSpotlight.h>
11 #import <MobileCoreServices/MobileCoreServices.h>
12
13 @interface ViewController ()
14
15 @end
16
17 @implementation ViewController
18
19 - (void)viewDidLoad {
20     [super viewDidLoad];
21     [self setupCoreSpotlightSearch];
22 }
23
24 - (void)setupCoreSpotlightSearch {
25     CSSearchableItemAttributeSet *attributeSet =
26         [CSSearchableItemAttributeSet searchItemAttributeSetWithAttributes:
27             (NSString *)kUTTypeImage];
28     attributeSet.title = @"My First Spotlight Search";
29     attributeSet.contentDescription = @"A sample image";
30     attributeSet.keywords = [NSArray arrayWithObjects:@"keyword1", @"keyword2", nil];
31     UIImage *image = [UIImage imageNamed:@"sampleImage.png"];
32     NSData *imageData = [NSData dataWithBytes:[image CGImage] length:[image CGImage].bytes];
33     attributeSet.thumbnailData = imageData;
34
35     CSSearchableItem *item = [[CSSearchableItem searchItemWithAttributes:attributeSet
36                             domainIdentifier:@"spotlight.sampleImage"]
37                             initWithIdentifier:@"sampleImage"];
38     [[CSSearchableIndex defaultSearchableIndex] addItem:item];
39 }

```


3. OK! Prueba el índice!

Capítulo 49: Cortar un UIImage en un círculo

Examples

Cortar una imagen en un círculo - Objetivo C

importar `#include <math.h>`

El código en `viewDidLoad` o `loadView` debería verse algo como esto

```
- (void)loadView
{
    [super loadView];
    UIImageView *imageView=[[UIImageView alloc]initWithFrame:CGRectMake(0, 50, 320, 320)];
    [self.view addSubview:imageView];
    UIImage *image=[UIImage imageNamed:@"Dubai-Photos-Images-Travel-Tourist-Images-Pictures-800x600.jpg"];
    imageView.image=[self circularScaleAndCropImage:[UIImage imageNamed:@"Dubai-Photos-Images-Travel-Tourist-Images-Pictures-800x600.jpg"] frame:CGRectMake(0, 0, 320, 320)];
}
```

Finalmente, la función que realiza el trabajo pesado `circularScaleAndCropImage` es como se define a continuación

```
- (UIImage*)circularScaleAndCropImage:(UIImage*)image frame:(CGRect)frame {
    // This function returns a newImage, based on image, that has been:
    // - scaled to fit in (CGRect) rect
    // - and cropped within a circle of radius: rectWidth/2

    //Create the bitmap graphics context
    UIGraphicsBeginImageContextWithOptions(CGSizeMake(frame.size.width, frame.size.height),
    NO, 0.0);
    CGContextRef context = UIGraphicsGetCurrentContext();

    //Get the width and heights
    CGFloat imageWidth = image.size.width;
    CGFloat imageHeight = image.size.height;
    CGFloat rectWidth = frame.size.width;
    CGFloat rectHeight = frame.size.height;

    //Calculate the scale factor
    CGFloat scaleFactorX = rectWidth/imageWidth;
    CGFloat scaleFactorY = rectHeight/imageHeight;

    //Calculate the centre of the circle
    CGFloat imageCentreX = rectWidth/2;
    CGFloat imageCentreY = rectHeight/2;

    // Create and CLIP to a CIRCULAR Path
    // (This could be replaced with any closed path if you want a different shaped clip)
    CGFloat radius = rectWidth/2;
    CGContextBeginPath (context);
    CGContextAddArc (context, imageCentreX, imageCentreY, radius, 0, 2*M_PI, 0);
    CGContextClosePath (context);
```

```

CGContextClip (context);

//Set the SCALE factor for the graphics context
//All future draw calls will be scaled by this factor
CGContextScaleCTM (context, scaleFactorX, scaleFactorY);

// Draw the IMAGE
CGRect myRect = CGRectMake(0, 0, imageWidth, imageHeight);
[image drawInRect:myRect];

UIImage *newImage = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();

return newImage;
}

```

Ejemplo de SWIFT 3

```

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view, typically from a nib.
    let imageView = UIImageView(frame: CGRectMake(x: CGFloat(0), y: CGFloat(50), width:
CGFloat(320), height: CGFloat(320)))
    view.addSubview(imageView)
    let image = UIImage(named: "Dubai-Photos-Images-Travel-Tourist-Images-Pictures-
800x600.jpg")
    imageView.image = circularScaleAndCropImage(UIImage(named: "Dubai-Photos-Images-
Travel-Tourist-Images-Pictures-800x600.jpg")!, frame: CGRectMake(x: CGFloat(0), y: CGFloat(0),
width: CGFloat(100), height: CGFloat(100)))
}

```

Finalmente, la función que realiza el trabajo pesado `circularScaleAndCropImage` es como se define a continuación

```

func circularScaleAndCropImage(_ image: UIImage, frame: CGRect) -> UIImage{
    // This function returns a newImage, based on image, that has been:
    // - scaled to fit in (CGRect) rect
    // - and cropped within a circle of radius: rectWidth/2
    //Create the bitmap graphics context
    UIGraphicsBeginImageContextWithOptions(CGSize(width: CGFloat(frame.size.width),
height: CGFloat(frame.size.height)), false, 0.0)
    let context: CGContext? = UIGraphicsGetCurrentContext()
    //Get the width and heights
    let imageWidth: CGFloat = image.size.width
    let imageHeight: CGFloat = image.size.height
    let rectWidth: CGFloat = frame.size.width
    let rectHeight: CGFloat = frame.size.height
    //Calculate the scale factor
    let scaleFactorX: CGFloat = rectWidth / imageWidth
    let scaleFactorY: CGFloat = rectHeight / imageHeight
    //Calculate the centre of the circle
    let imageCentreX: CGFloat = rectWidth / 2
    let imageCentreY: CGFloat = rectHeight / 2
    // Create and CLIP to a CIRCULAR Path
    // (This could be replaced with any closed path if you want a different shaped clip)
    let radius: CGFloat = rectWidth / 2
    context?.beginPath()
    context?.addArc(center: CGPoint(x: imageCentreX, y: imageCentreY), radius: radius,

```



```
startAngle: CGFloat(0), endAngle: CGFloat(2 * Float.pi), clockwise: false)
    context?.closePath()
    context?.clip()
    //Set the SCALE factor for the graphics context
    //All future draw calls will be scaled by this factor
    context?.scaleBy(x: scaleFactorX, y: scaleFactorY)
    // Draw the IMAGE
    let myRect = CGRect(x: CGFloat(0), y: CGFloat(0), width: imageWidth, height:
imageHeight)
    image.draw(in: myRect)
    let newImage: UIImage? = UIGraphicsGetImageFromCurrentImageContext()
    UIGraphicsEndImageContext()
    return newImage!
}
```

Lea Cortar un UIImage en un círculo en línea: <https://riptutorial.com/es/ios/topic/7222/cortar-un-uiimage-en-un-circulo>

Capítulo 50: Creación de PDF en iOS

Examples

Crea PDF

```
UIGraphicsBeginPDFContextToFile(fileName, CGRectZero, nil);

UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 612, 792), nil);

[self drawText];

UIGraphicsEndPDFContext();
```

fileName es el archivo de documento donde va a adjuntar o adjuntar

```
NSString* temporaryFile = @"firstIOS.PDF";
NSArray *arrayPaths =
    NSSearchPathForDirectoriesInDomains(
        NSDocumentDirectory,
        NSUserDomainMask,
        YES);

NSString *path = [arrayPaths objectAtIndex:0];

NSString* fileName = [path stringByAppendingPathComponent:fileName];
```

Donde **drawText** es

```
(void)drawText
{
    NSString* textToDraw = @"Lorem Ipsum is simply dummy text of the printing and typesetting
industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when
an unknown printer took a galley of type and scrambled it to make a type specimen book.";

    CFStringRef stringRef = (__bridge CFStringRef)textToDraw;

    CFAttributedStringRef currentText = CFAttributedStringCreate(NULL, stringRef, NULL);

    CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedString(currentText);

    CGRect frameRect = CGRectMake(0, 0, 300, 100);

    CGMutablePathRef framePath = CGPathCreateMutable();

    CGPathAddRect(framePath, NULL, frameRect);

    CFRange currentRange = CFRangeMake(0, 0);

    CTFrameRef frameRef = CTFramesetterCreateFrame(framesetter, currentRange, framePath,
NULL);
    CGPathRelease(framePath);

    CGContextRef currentContext = UIGraphicsGetCurrentContext();
```

```
CGContextSetTextMatrix(currentContext, CGAffineTransformIdentity);

CGContextTranslateCTM(currentContext, 0, 450);

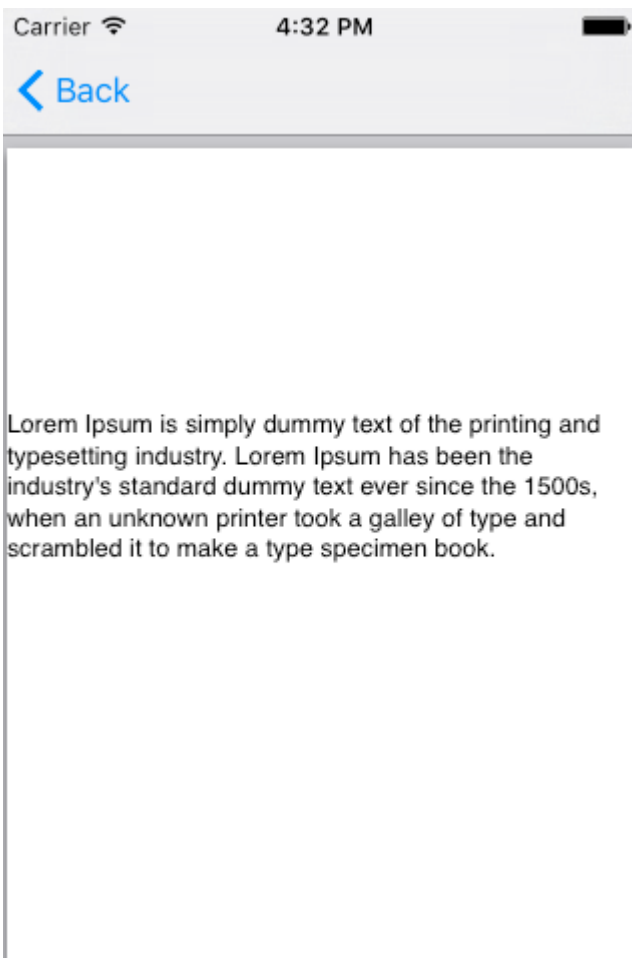
CGContextScaleCTM(currentContext, 2, -2);

CTFrameDraw(frameRef, currentContext);

CFRelease(frameRef);

CFRelease(stringRef);

CFRelease(framesetter);
}
```



Mostrar PDF

```
NSString* fileName = @"firstIOS.PDF";

NSArray *arrayPaths =
NSSearchPathForDirectoriesInDomains(
    NSDocumentDirectory,
    NSUserDomainMask,
    YES);
```

```

NSString *path = [arrayPaths objectAtIndex:0];

NSString* pdfFileName = [path stringByAppendingPathComponent:fileName];

UIWebView* webView = [[UIWebView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];

NSURL *url = [NSURL fileURLWithPath:pdfFileName];

NSURLRequest *request = [NSURLRequest requestWithURL:url];

[webView setScalesPageToFit:YES];

[webView loadRequest:request];

[self.view addSubview:webView];

```

PDF de varias páginas

```

UIGraphicsBeginPDFContextToFile(fileName, CGRectZero, nil);

    UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 600, 792), nil);

    UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 600, 792), nil);

    UIGraphicsBeginPDFPageWithInfo(CGRectMake(0, 0, 600, 792), nil);

    UIGraphicsEndPDFContext();

```

Crear PDF desde cualquier documento de Microsoft cargado en UIWebView

```
#define kPaperSizeA4 CGSizeMake(595.2,841.8)
```

En primer lugar implementar el protocolo `UIPrintPageRenderer`.

```

@interface UIPrintPageRenderer (PDF)

- (NSData*) printToPDF;

@end

@implementation UIPrintPageRenderer (PDF)

- (NSData*) printToPDF
{
    NSMutableData *pdfData = [NSMutableData data];
    UIGraphicsBeginPDFContextToData(pdfData, self.paperRect, nil);
    [self prepareForDrawingPages:NSMakeRange(0, self.numberOfPages)];
    CGRect bounds = UIGraphicsGetPDFContextBounds();
    for (int i = 0; i < self.numberOfPages; i++)
    {
        UIGraphicsBeginPDFPage();
        [self drawPageAtIndex:i inRect:bounds];
    }
    UIGraphicsEndPDFContext();
    return pdfData;
}

```

```
}  
@end
```

Luego, llame al siguiente método después de que el documento haya terminado de cargarse en `UIWebView`

```
-(void)createPDF:(UIWebView *)webView {  
  
    UIPrintPageRenderer *render = [[UIPrintPageRenderer alloc] init];  
    [render addPrintFormatter:webView.viewPrintFormatter startingAtIndex:0];  
  
    float padding = 10.0f;  
    CGRect paperRect = CGRectMake(0, 0, kPaperSizeA4.width, kPaperSizeA4.height);  
    CGRect printableRect = CGRectMake(padding, padding, kPaperSizeA4.width-(padding * 2),  
    kPaperSizeA4.height-(padding * 2));  
  
    [render setValue:[NSValue valueWithCGRect:paperRect] forKey:@"paperRect"];  
    [render setValue:[NSValue valueWithCGRect:printableRect] forKey:@"printableRect"];  
  
    NSData *pdfData = [render printToPDF];  
  
    dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{  
  
        if (pdfData) {  
            [pdfData writeToFile:directoryPath atomically: YES];  
        }  
        else  
        {  
            NSLog(@"PDF couldnot be created");  
        }  
    });  
};
```

Lea Creación de PDF en iOS en línea: <https://riptutorial.com/es/ios/topic/2416/creacion-de-pdf-en-ios>

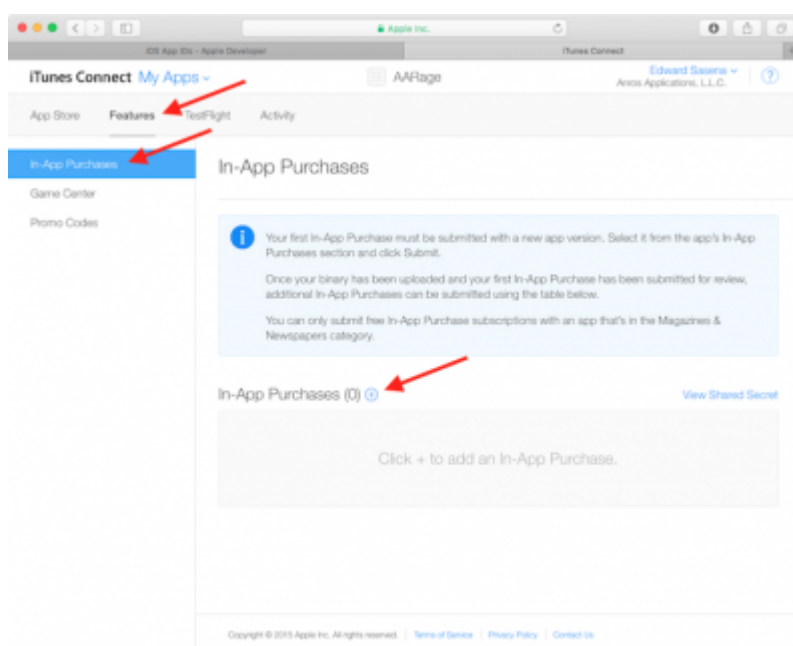
Capítulo 51: Creación de una ID de aplicación

Examples

Creación de productos de compra en la aplicación

- Al ofrecer IAP dentro de una aplicación, primero debe agregar una entrada para cada compra individual dentro de iTunes Connect. Si alguna vez ha incluido una aplicación en venta en la tienda, es un proceso similar e incluye cosas como elegir un nivel de precios para la compra. Cuando el usuario realiza una compra, App Store maneja el complejo proceso de cargar la cuenta de iTunes del usuario. Hay muchos tipos diferentes de IAP que puede agregar:
 - **Consumibles** : Se pueden comprar más de una vez y pueden agotarse. Estas son cosas como vidas extra, moneda del juego, power-ups temporales y similares.
 - **No consumible** : algo que compra una vez y espera tener de forma permanente, como niveles adicionales y contenido desbloqueable.
 - **Suscripción sin renovación** : contenido que está disponible por un período de tiempo fijo.
 - **Suscripción de renovación automática** : una suscripción de repetición, como una suscripción mensual a raywenderlich.com.

Solo puede ofrecer compras en la aplicación para artículos digitales, y no para bienes o servicios físicos. Para obtener más información sobre todo esto, consulte la documentación completa de Apple sobre Creación de productos de compra en la aplicación. Ahora, mientras ve la entrada de su aplicación en iTunes Connect, haga clic en la pestaña Características y luego seleccione Compras dentro de la aplicación. Para agregar un nuevo producto IAP, haga clic en + a la derecha de Compras dentro de la aplicación.



Verás aparecer el siguiente diálogo:

Select the In-App Purchase you want to create.

Consumable

A product that is used once, after which it becomes depleted and must be purchased again.

Example: Fish food for a fishing app.

Non-Consumable

A product that is purchased once and does not expire or decrease with use.

Example: Race track for a game app.

Auto-Renewable Subscription

A product that allows users to purchase dynamic content for a set period. This type of subscription renews automatically unless cancelled by the user.

Example: Monthly subscription for an app offering a streaming service.

Non-Renewing Subscription

A product that allows users to purchase a service with a limited duration. The content of this in-app purchase can be static. This type of subscription does not renew automatically.

Example: Annual subscription to a catalog of archived articles.

[Learn more about In-App Purchases.](#)

Cancel

Create

Cuando un usuario compra un cómic de rabia en su aplicación, deseará que siempre tenga acceso a ella, así que seleccione No consumible y haga clic en Crear. A continuación, complete los detalles para el IAP de la siguiente manera:

- **Nombre de referencia** : un apodo que identifica el IAP dentro de iTunes Connect. Este nombre no aparece en ninguna parte de la aplicación. El título del cómic que desbloquearás con esta compra es "**Girlfriend of Drummer**" , así que ingresa aquí.
- **Identificación del producto** : esta es una cadena única que identifica el IAP. Por lo general, es mejor comenzar con la ID del paquete y luego agregar un nombre único específico a este artículo comprable. Para este tutorial, asegúrate de agregar "GirlfriendOfDrummerRage", ya que esto se usará más adelante dentro de la aplicación para buscar el cómic para desbloquear. Entonces, por ejemplo:
com.theNameYouPickedEarlier.Rage.GirlFriendOfDrummerRage.
- **Compensado para la venta** : Activa o desactiva la venta del IAP. Quieres habilitarlo!
- **Nivel de precio** : El costo del IAP. Elija el nivel 1.

Ahora desplácese hacia abajo hasta la sección de Localizaciones y tenga en cuenta que hay una entrada predeterminada para inglés (EE. UU.). Ingrese "Girlfriend of Drummer" tanto para el Nombre de visualización como para la Descripción. Clic en Guardar. ¡Genial! Has creado tu primer producto IAP.

Localizations ⊕

English (U.S.)

Display Name ?
Girlfriend of Drummer

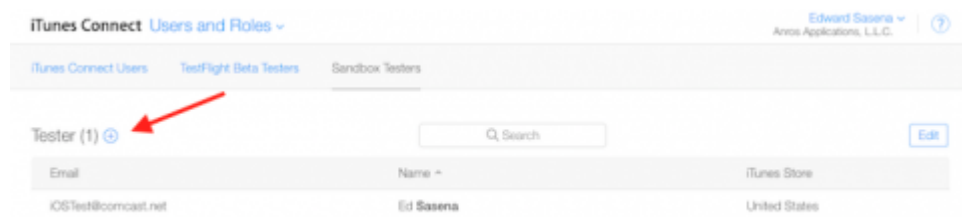
Description ?
Girlfriend of Drummer

234

Se necesita un paso más antes de poder profundizar en algún código. Al probar las compras en la aplicación en una versión de desarrollo de una aplicación, Apple proporciona un entorno de prueba que le permite "comprar" sus productos IAP sin crear transacciones financieras.

Creando un usuario de Sandbox

En iTunes Connect, haga clic en iTunes Connect en la esquina superior izquierda de la ventana para volver al menú principal. Seleccione Usuarios y roles, luego haga clic en la pestaña Probadores de Sandbox. Haga clic en + junto al título "Tester".



Complete la información y haga clic en Guardar cuando haya terminado. Puede inventar el nombre y apellido de su usuario de prueba, pero la dirección de correo electrónico elegida debe ser una dirección de correo electrónico real, ya que Apple enviará una verificación a la dirección. Una vez que reciba ese correo electrónico, asegúrese de hacer clic en el enlace para verificar su dirección. La dirección de correo electrónico que ingrese NO debe estar asociada con una cuenta de ID de Apple. Consejo: si tiene una cuenta de gmail, simplemente puede usar un alias de dirección en lugar de tener que crear una cuenta nueva.

Lea Creación de una ID de aplicación en línea: <https://riptutorial.com/es/ios/topic/10854/creacion-de-una-id-de-aplicacion>

Capítulo 52: Crear un marco personalizado en iOS

Examples

Crear Framework en Swift

Siga estos pasos para crear un marco personalizado en Swift-IOS:

1. Crea un nuevo proyecto. En Xcode
2. Elija iOS / Framework & Library / Cocoa Touch Framework para crear un nuevo framework
3. haga clic en siguiente y establezca el productName
4. haga clic en siguiente y elija el directorio para crear el proyecto allí
5. Agrega código y recursos al proyecto creado

Marco creado con éxito

para agregar el marco creado a otro proyecto, primero debe crear un área de trabajo agregue "proyecto de destino" y "proyecto de marco" al área de trabajo, luego:

1. ir a la pestaña general del proyecto objetivo
2. arrastre el archivo "*.framework" en la carpeta de producto del proyecto de marco a la sección "Binarios incrustados"
3. para usar en cualquier ViewController o clase, simplemente importe el marco en cada archivo

Lea [Crear un marco personalizado en iOS en línea](https://riptutorial.com/es/ios/topic/7331/crear-un-marco-personalizado-en-ios): <https://riptutorial.com/es/ios/topic/7331/crear-un-marco-personalizado-en-ios>

Capítulo 53: Crear un video a partir de imágenes.

Introducción

Creas un video a partir de imágenes usando AVFoundation

Examples

Crear video desde Ullimages

En primer lugar necesitas crear `AVAssetWriter`

```
NSError *error = nil;
NSURL *outputURL = <#NSURL object representing the URL where you want to save the video#>;
AVAssetWriter *assetWriter = [AVAssetWriter assetWriterWithURL:outputURL
fileType:AVFileTypeQuickTimeMovie error:&error];
if (!assetWriter) {
    // handle error
}
```

`AVAssetWriter` necesita al menos una entrada de escritor de activos.

```
NSDictionary *writerInputParams = [NSDictionary dictionaryWithObjectsAndKeys:
    AVVideoCodecH264, AVVideoCodecKey,
    [NSNumber numberWithInt:renderSize.width],
    AVVideoWidthKey,
    [NSNumber numberWithInt:renderSize.height],
    AVVideoHeightKey,
    AVVideoScalingModeResizeAspectFill,
    AVVideoScalingModeKey,
    nil];

AVAssetWriterInput *assetWriterInput = [AVAssetWriterInput
assetWriterInputWithMediaType:AVMediaTypeVideo outputSettings:writerInputParams];
if ([assetWriter canAddInput:assetWriterInput]) {
    [assetWriter addInput:assetWriterInput];
} else {
    // show error message
}
```

Para agregar `CVPixelBufferRef`'s a `AVAssetWriterInput` necesitamos crear

`AVAssetWriterInputPixelBufferAdaptor`

```
NSDictionary *attributes = [NSDictionary dictionaryWithObjectsAndKeys:
    [NSNumber numberWithInt:kCVPixelFormatType_32ARGB],
    (NSString*)kCVPixelBufferPixelFormatTypeKey,
    [NSNumber numberWithBool:YES], (NSString*)
    kCVPixelBufferCGImageCompatibilityKey,
    [NSNumber numberWithBool:YES], (NSString*)
```

```

*)kCVPixelBufferCGBitmapContextCompatibilityKey,
        nil];
AVAssetWriterInputPixelBufferAdaptor *writerAdaptor = [AVAssetWriterInputPixelBufferAdaptor
assetWriterInputPixelBufferAdaptorWithAssetWriterInput:assetWriterInput
sourcePixelBufferAttributes:attributes];

```

Ahora podemos empezar a escribir.

```

[assetWriter startWriting];
[assetWriter startSessionAtSourceTime:kCMTimeZero];
[assetWriterInput requestMediaDataWhenReadyOnQueue:exportingQueue usingBlock:^(
    for (int i = 0; i < images.count; ++i) {
        while (![assetWriterInput isReadyForMoreMediaData]) {
            [NSThread sleepForTimeInterval:0.01];
            // can check for attempts not to create an infinite loop
        }

        UIImage *uIImage = images[i];

        CVPixelBufferRef buffer = NULL;
        CVReturn err = PixelBufferCreateFromImage(uIImage.CGImage, &buffer);
        if (err) {
            // handle error
        }

        // frame duration is duration of single image in seconds
        CMTime presentationTime = CMTimeMakeWithSeconds(i * frameDuration, 1000000);

        [writerAdaptor appendPixelBuffer:buffer withPresentationTime:presentationTime];

        CVPixelBufferRelease(buffer);
    }

[assetWriterInput markAsFinished];
[assetWriter finishWritingWithCompletionHandler:^(
    if (assetWriter.error) {
        // show error message
    } else {
        // outputURL
    }
}];
}];

```

Aquí hay una función para obtener `CVPixelBufferRef` de `CGImageRef`

```

CVReturn PixelBufferCreateFromImage(CGImageRef imageRef, CVPixelBufferRef *outBuffer) {
    CIContext *context = [CIContext context];
    CIImage *ciImage = [CIImage imageWithCGImage:imageRef];

    NSDictionary *attributes = [NSDictionary dictionaryWithObjectsAndKeys:
        [NSNumber numberWithInt:YES], (NSString
*)kCVPixelBufferCGBitmapContextCompatibilityKey,
        [NSNumber numberWithInt:YES], (NSString
*)kCVPixelBufferCGImageCompatibilityKey
        , nil];

    CVReturn err = CVPixelBufferCreate(kCFAllocatorDefault, CGImageGetWidth(imageRef),
CGImageGetHeight(imageRef), kCVPixelFormatType_32ARGB, (__bridge CFDictionaryRef
_Nullable) (attributes), outBuffer);

```

```
if (err) {
    return err;
}

if (outBuffer) {
    [context render:ciImage toCVPixelBuffer:*outBuffer];
}

return kCVReturnSuccess;
}
```

Lea **Crear un video a partir de imágenes.** en línea: <https://riptutorial.com/es/ios/topic/10607/crear-un-video-a-partir-de-imagenes->

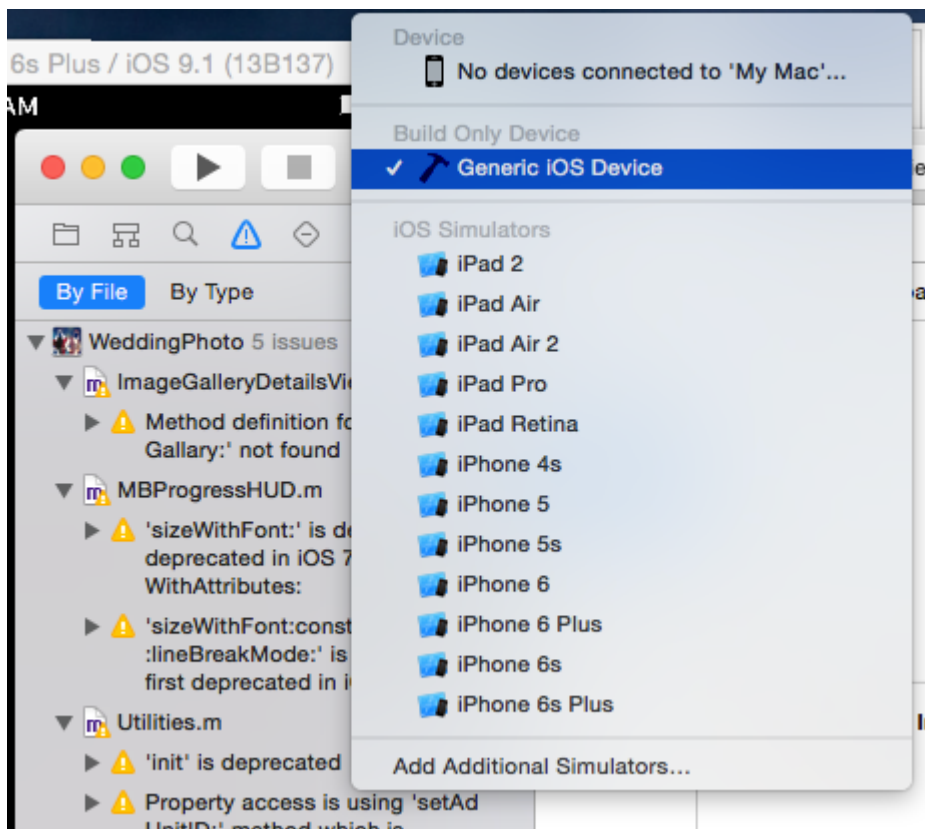
Capítulo 54: Cree un archivo .ipa para cargar en la tienda de aplicaciones con Applicationloader

Examples

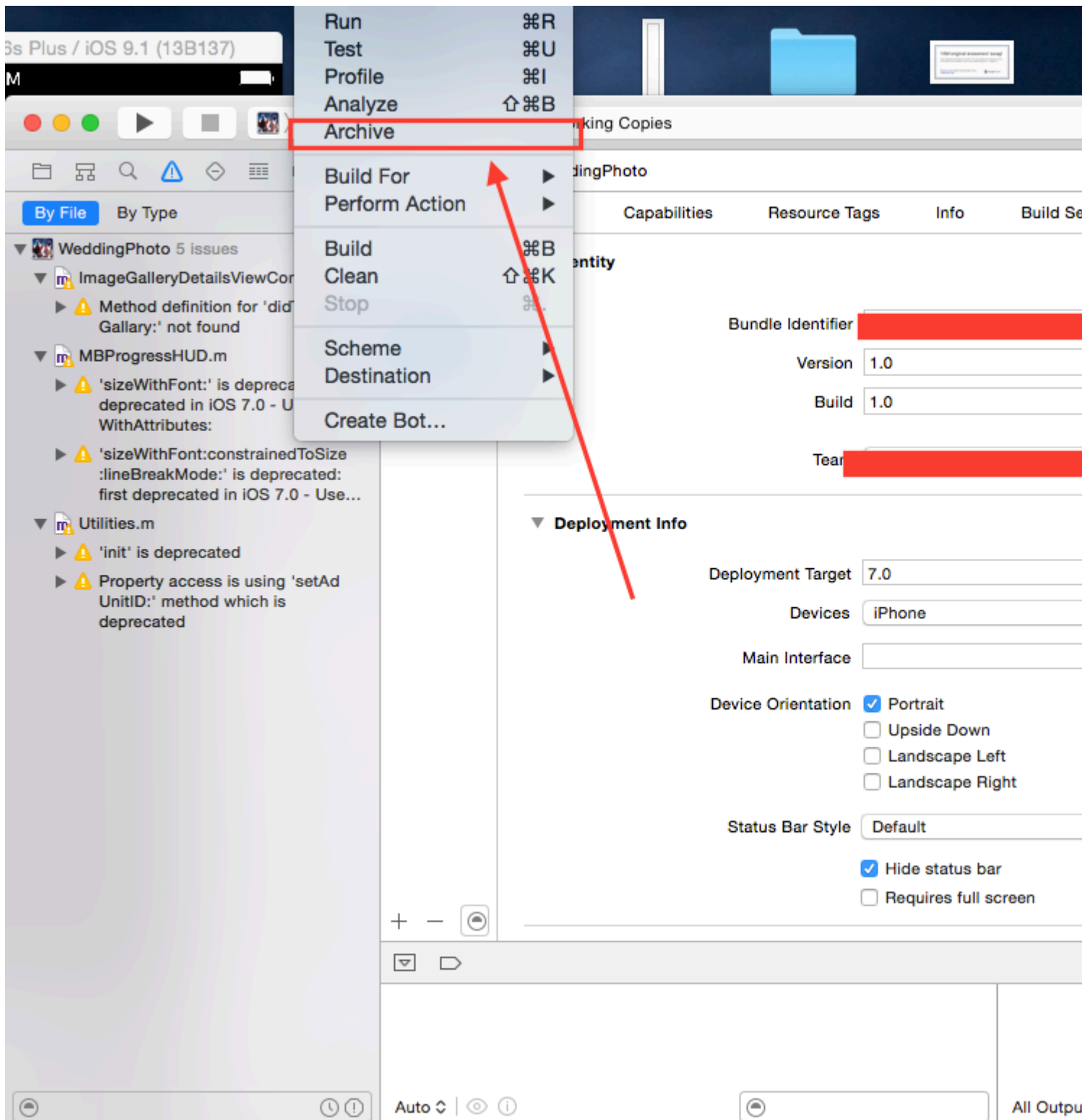
crea el archivo .ipa para cargar la aplicación en appstore con Application Loader

Si desea cargar un archivo .ipa a itunesconnect **sin integrar la cuenta de desarrollador en Xcode** y desea usar el **cargador de aplicaciones** . Entonces puedes **generar .ipa con iTunes** .

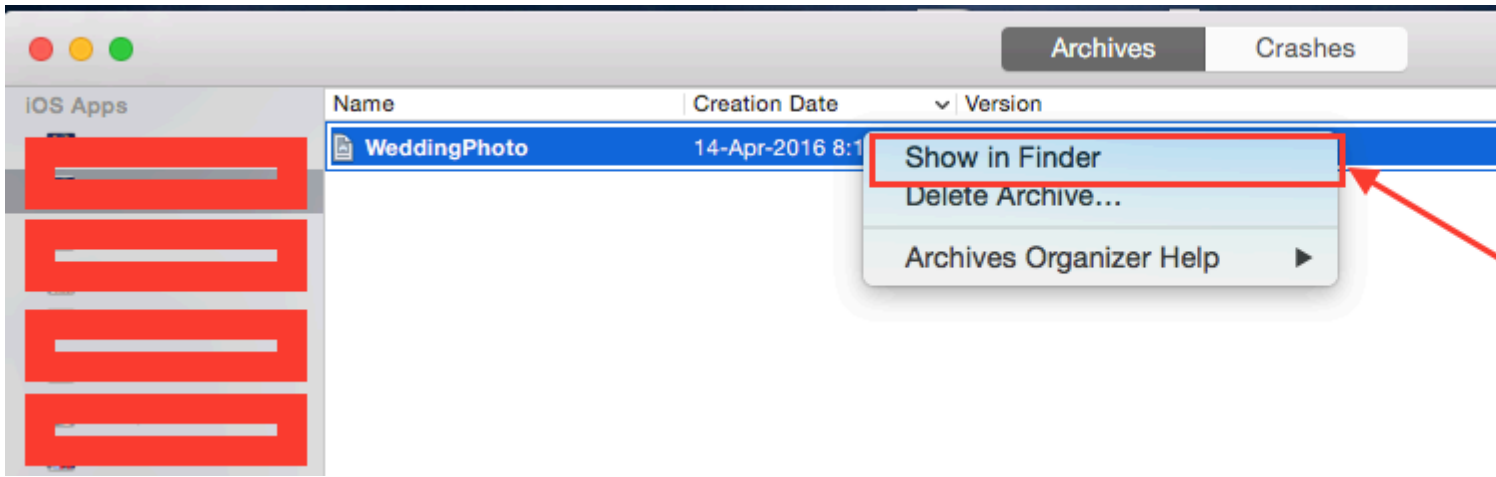
Paso 1: - Selecciona el dispositivo en lugar del simulador.



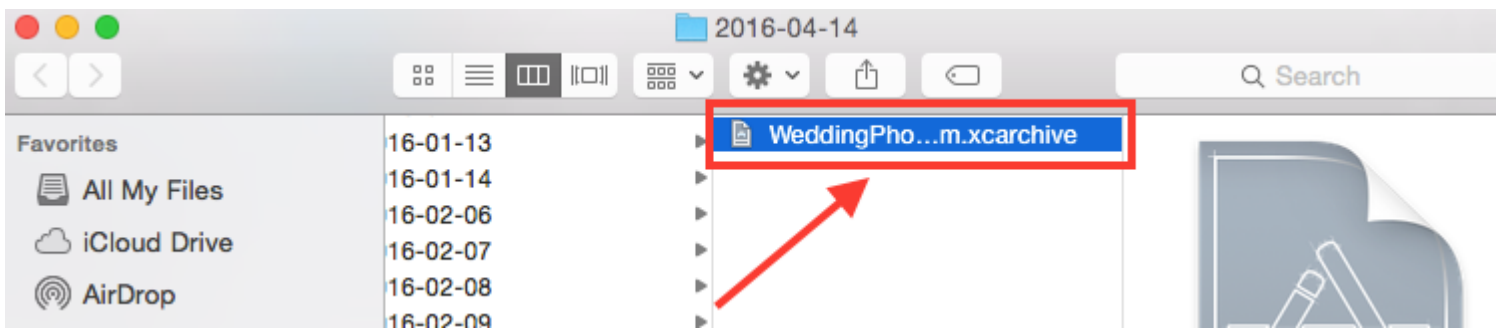
Paso 2: - Ir a Producto -> seleccionar Archivo



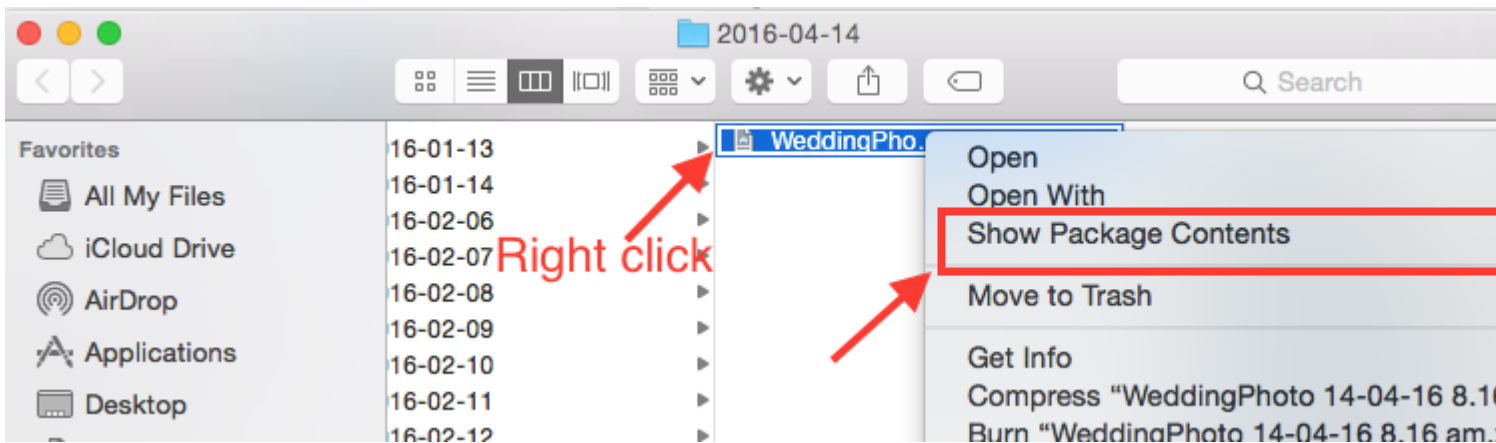
Paso 3: - Después de completar el proceso, haga clic con el botón derecho en su archivo -> y seleccione Mostrar en Finder



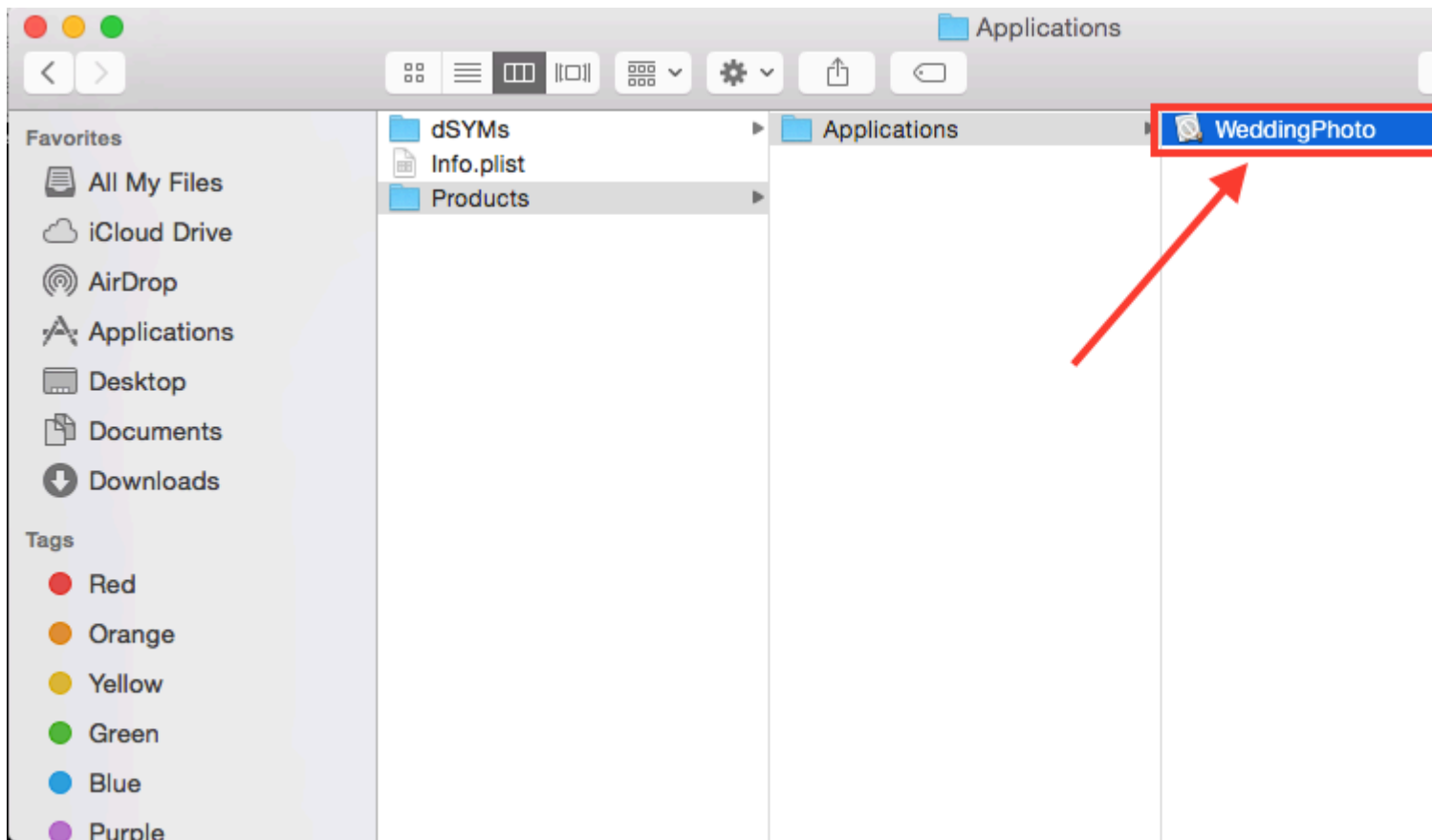
Paso 4: cuando haces clic en mostrar en el buscador redirigirás a la carpeta Archivo, se ve así



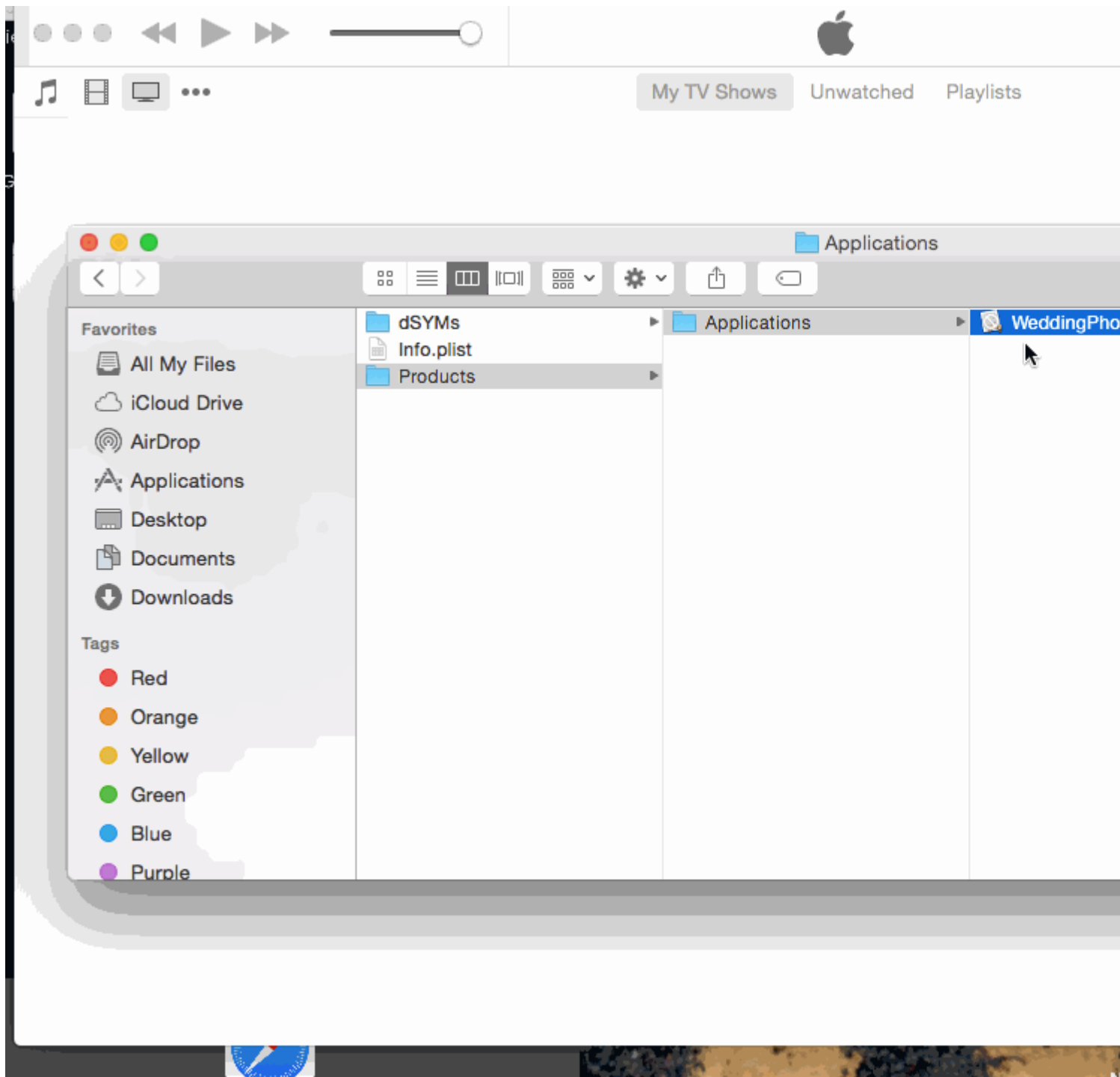
Paso 5: - Haga clic derecho en el archivo .xcarchive -> seleccione Mostrar en la opción del buscador.



Paso 6: - Vaya a Carpeta del producto -> Carpeta de la aplicación -> Encontrará suprojectname.app



Paso 7: - Ahora para convertir .app a .ipa solo arrastra y suelta en itunes. verifique la imagen de abajo,



Paso 8: - Ahora ponga este archivo .ipa en un lugar seguro y utilícelo cuando cargue con el cargador de aplicaciones.

Nota: si desea saber cómo cargar una aplicación con el cargador de aplicaciones, compruebe esto.

[Carga la aplicación con la aplicación Loader](#)

EDITAR: -

ADVERTENCIA: - No haga .ipa cambiando la extensión de .aap a .zip y .zip a .ipa.

He visto en muchas respuestas que, han sugerido comprimir el archivo .app y luego cambiar la

extensión de .zip a .ipa. No está funcionando ahora. Por este método obtendrá error como,

IPA no es válida, no incluye un directorio de carga útil.

Lea [Cree un archivo .ipa para cargar en la tienda de aplicaciones con Applicationloader en línea:](https://riptutorial.com/es/ios/topic/6119/crea-un-archivo--ipa-para-cargar-en-la-tienda-de-aplicaciones-con-applicationloader)

<https://riptutorial.com/es/ios/topic/6119/crea-un-archivo--ipa-para-cargar-en-la-tienda-de-aplicaciones-con-applicationloader>

Capítulo 55: CTCallCenter

Examples

Interceptando llamadas desde tu aplicación incluso desde el fondo

De la documentación de Apple:

Use la clase CTCallCenter para obtener una lista de las llamadas celulares actuales y para responder a los cambios de estado de las llamadas, por ejemplo, de un estado de marcación a un estado conectado. Tales cambios de estado se conocen como eventos de llamada celular.

El propósito de CTCallCenter es brindar al desarrollador la oportunidad de pausar el estado de su aplicación durante una llamada para brindar al usuario la mejor experiencia.

C objetivo:

Primero, definiremos un nuevo miembro de clase dentro de la clase que queremos manejar las intercepciones:

```
@property (atomic, strong) CTCallCenter *callCenter;
```

Dentro de nuestra clase init (constructor) asignaremos nueva memoria a nuestro miembro de la clase:

```
[self setCallCenter:[CTCallCenter new]];
```

Luego, invocaremos nuestro nuevo método que realmente maneja las intercepciones:

```
- (void)registerPhoneCallListener
{
[[self callCenter] setCallEventHandler:^(CTCall * _Nonnull call) {
    NSLog(@"CallEventHandler called - interception in progress");

    if ([call.callState isEqualToString: CTCallStateConnected])
    {
        NSLog(@"Connected");
    }
    else if ([call.callState isEqualToString: CTCallStateDialing])
    {
        NSLog(@"Dialing");
    }
    else if ([call.callState isEqualToString: CTCallStateDisconnected])
    {
        NSLog(@"Disconnected");
    }
    else if ([call.callState isEqualToString: CTCallStateIncoming])
    {
        NSLog(@"Incomming");
    }
}];
```

```
    }  
  }];  
}
```

Eso es todo, si el usuario usará su aplicación y recibirá una llamada telefónica, podría interceptar esta llamada y manejar su aplicación para un estado de guardado.

Vale la pena mencionar que hay 4 estados de llamada que puede interceptar:

```
CTCallStateDialing  
CTCallStateIncoming  
CTCallStateConnected  
CTCallStateDisconnected
```

Rápido:

Defina el miembro de su clase en la clase relevante y defínalo:

```
self.callCenter = CTCallCenter()  
self.callCenter.callEventHandler = { call in  
  // Handle your interception  
  if call.callState == CTCallStateConnected  
  {  
  }  
}
```

¿Qué pasará si su aplicación está en segundo plano y necesita interceptar llamadas mientras la aplicación está en segundo plano?

Por ejemplo, si desarrolla una aplicación **empresarial**, básicamente puede agregar 2 capacidades (VOIP y búsqueda en segundo plano) en la pestaña Capacidades:

Objetivo del proyecto -> Capacidades -> Modos de fondo -> marcar Voz sobre IP y recuperación de fondo

Kit de llamadas - ios 10

```
//Header File  
  
<CallKit/CXCallObserver.h>  
  
CXCallObserver *callObserver = [[CXCallObserver alloc] init];  
  
// If queue is nil, then callbacks will be performed on main queue  
  
[callObserver setDelegate:self queue:nil];  
  
// Don't forget to store reference to callObserver, to prevent it from being released  
  
self.callObserver = callObserver;  
  
// get call status  
- (void)callObserver:(CXCallObserver *)callObserver callChanged:(CXCall *)call {
```

```
if (call.hasConnected) {  
    // perform necessary actions  
}  
}
```

Lea CTCallCenter en línea: <https://riptutorial.com/es/ios/topic/3007/ctcallcenter>

Capítulo 56: CydiaSubstrate tweak

Introducción

Aprende cómo crear ajustes de sustrato de Cydia para iPhones con jailbreak.

Esos ajustes le permitirán modificar el comportamiento del sistema operativo para que actúe de la manera que le gustaría.

Observaciones

Instalando theos

<https://github.com/theos/theos/wiki/Installation>

Examples

Crear nuevo tweak usando Theos

Usa nic para crear un nuevo proyecto

Introduce este comando en tu terminal

```
$THEOS/bin/nic.pl
```

```
NIC 2.0 - New Instance Creator
-----
[1.] iphone/activator_event
[2.] iphone/application_modern
[3.] iphone/cydget
[4.] iphone/flipswitch_switch
[5.] iphone/framework
[6.] iphone/ios7_notification_center_widget
[7.] iphone/library
[8.] iphone/notification_center_widget
[9.] iphone/preference_bundle_modern
[10.] iphone/tool
[11.] iphone/tweak
[12.] iphone/xpc_service
Choose a Template (required):
```

Elegir plantilla [11.] iphone/tweak

Rellene los detalles y obtendrá los siguientes archivos creados:

```
-rw-r--r--@ 1 gkpln3 staff 214B Jun 12 15:09 Makefile
-rw-r--r--@ 1 gkpln3 staff 89B Jun 11 22:58 TorchonFocus.plist
-rw-r--r-- 1 gkpln3 staff 2.7K Jun 12 16:10 Tweak.xml
-rw-r--r-- 1 gkpln3 staff 224B Jun 11 16:17 control
drwxr-xr-x 3 gkpln3 staff 102B Jun 11 16:18 obj
drwxr-xr-x 16 gkpln3 staff 544B Jun 12 16:12 packages
```

Anular método de guardar capturas de pantalla de iOS

abre el archivo `Tweak.xml` usando tu editor de código favorito.

Enganche a un determinado método desde el sistema operativo.

```
%hook SBScreenShotter
- (void) saveScreenshot:(BOOL) screenshot
{
    %orig;
    NSLog(@"saveScreenshot: is called");
}
%end
```

Tenga en cuenta que puede elegir si se debe o no llamar la función original, por ejemplo:

```
%hook SBScreenShotter
- (void) saveScreenshot:(BOOL) screenshot
{
    NSLog(@"saveScreenshot: is called");
}
%end
```

anulará la función sin llamar a la original, por lo que no se guardarán capturas de pantalla.

Lea [CydiaSubstrate tweak en línea](https://riptutorial.com/es/ios/topic/10533/cydia-substrate-tweak): <https://riptutorial.com/es/ios/topic/10533/cydia-substrate-tweak>

Capítulo 57: Datos básicos

Introducción

Core Data es la capa modelo de su aplicación en el sentido más amplio posible. Es el modelo en el patrón Modelo-Vista-Controlador que impregna el SDK de iOS.

Core Data no es la base de datos de su aplicación ni es una API para la persistencia de datos en una base de datos. Core Data es un marco que gestiona un gráfico de objetos. Es tan simple como eso. Los datos básicos pueden persistir en ese gráfico de objetos escribiéndolos en el disco, pero ese no es el objetivo principal del marco.

Examples

Operaciones sobre datos básicos

Para obtener contexto:

```
NSManagedObjectContext *context = ((AppDelegate*)[[UIApplication sharedApplication] delegate]).persistentContainer.viewContext;
```

Para obtener datos:

```
NSFetchRequest<EntityName *> *fetchRequest = [EntityName fetchRequest];
NSError *error ;
NSArray *resultArray= [context executeFetchRequest:fetchRequest error:&error];
```

Para obtener datos con la clasificación:

```
NSFetchRequest<EntityName *> *fetchRequest = [EntityName fetchRequest];
NSSortDescriptor *sortDescriptor = [NSSortDescriptor sortDescriptorWithKey:@"someKey" ascending:YES];
fetchRequest.sortDescriptors = @[sortDescriptor];
NSError *error ;
NSArray *resultArray= [context executeFetchRequest:fetchRequest error:&error];
```

Para añadir datos:

```
NSManagedObject *entityNameObj = [NSEntityDescription insertNewObjectForEntityForName:@"EntityName" inManagedObjectContext:context];
[entityNameObj setValue:@"someValue" forKey:@"someKey"];
```

Para guardar el contexto:

```
(((AppDelegate*)[[UIApplication sharedApplication] delegate]) saveContext);
```

Lea Datos básicos en línea: <https://riptutorial.com/es/ios/topic/9489/datos-basicos>

Capítulo 58: Delegados de multidifusión

Introducción

Patrón para agregar capacidades de multidifusión a los controles existentes de iOS. La adición de multidifusión permite una mayor claridad y la reutilización del código.

Examples

Delegados multicast para cualquier control.

Reenvíe los mensajes de un objeto a otro mediante delegados, transmitiendo estos mensajes a múltiples observadores.

Paso 1: - Crear `NSObject` clase de `RRMulticastDelegate`

Paso 2: - Siguiendo el código implementado en el archivo `RRMulticastDelegate.h`

```
#import <Foundation/Foundation.h>

@interface RRMulticastDelegate : NSObject

{
    //Handle multiple observers of delegate
    NSMutableArray* _delegates;
}

// Delegate method implementation to the list of observers
- (void)addDelegate:(id)delegate;
- (void)removeDelegate:(id)delegate;

// Get multiple delegates
-(NSArray *)delegatesObjects;

@end
```

Paso 3: - Seguir el código implementado en el archivo `RRMulticastDelegate.m`

```
#import "RRMulticastDelegate.h"

@implementation RRMulticastDelegate

- (id)init
{
    if (self = [super init])
    {
        _delegates = [NSMutableArray array];
    }
    return self;
}

-(NSArray *)delegatesObjects
```

```

{
    return _delegates;
}

- (void)removeDelegate:(id)delegate
{
    if ([_delegates containsObject:delegate])
        [_delegates removeObject:delegate];
}

- (void)addDelegate:(id)delegate
{
    if (![_delegates containsObject:delegate])
        [_delegates addObject:delegate];
}

- (BOOL)respondsToSelector:(SEL)aSelector
{
    if ([super respondsToSelector:aSelector])
        return YES;

    // if any of the delegates respond to this selector, return YES
    for(id delegate in _delegates)
    {
        if (!delegate)
            continue;

        if ([delegate respondsToSelector:aSelector])
        {
            return YES;
        }
    }
    return NO;
}

- (NSMethodSignature *)methodSignatureForSelector:(SEL)aSelector
{
    // can this class create the signature?
    NSMethodSignature* signature = [super methodSignatureForSelector:aSelector];

    // if not, try our delegates
    if (!signature)
    {
        for(id delegate in _delegates)
        {
            if (!delegate)
                continue;

            if ([delegate respondsToSelector:aSelector])
            {
                return [delegate methodSignatureForSelector:aSelector];
            }
        }
    }
    return signature;
}

- (void)forwardInvocation:(NSInvocation *)anInvocation
{
    // forward the invocation to every delegate
    for(id delegate in _delegates)

```

```

    {
        if (!delegate)
            continue;

        if ([delegate respondsToSelector:[anInvocation selector]])
        {
            [anInvocation invokeWithTarget:delegate];
        }
    }
}
@end

```

Paso 4: - Crear `NSObject` categoría de clase de `RRProperty`

Paso 5: - Siguiendo el código implementado en el archivo `NSObject+RRProperty.h`

```

#import <Foundation/Foundation.h>

#import "RRMulticastDelegate.h"

@interface NSObject (RRProperty)<UITextFieldDelegate,UITableViewDataSource>

-(void)setObject:(id)block forKey:(NSString *)key;
-(id)objectForKey:(NSString *)key;

#pragma mark - Multicast Delegate

- (RRMulticastDelegate *)multicastDelegate;
- (RRMulticastDelegate *)multicastDatasource;

-(void)addDelegate:(id)delegate;
-(void)addDataSource:(id)datasource;

@end

```

Paso 6: - Siguiendo el código implementado en el archivo `NSObject+RRProperty.m`

```

#import "NSObject+RRProperty.h"

#import <objc/message.h>
#import <objc/runtime.h>

#pragma GCC diagnostic ignored "-Wprotocol"

static NSString *const MULTICASTDELEGATE = @"MULTICASTDELEGATE";
static NSString *const MULTICASTDATASOURCE = @"MULTICASTDATASOURCE";

@implementation NSObject (RRProperty)

-(void)setObject:(id)block forKey:(NSString *)key
{
    objc_setAssociatedObject(self, (__bridge const void *) (key), block,
OBJC_ASSOCIATION_RETAIN);
}

-(id)objectForKey:(NSString *)key
{

```

```

    return objc_getAssociatedObject(self, (__bridge const void *) (key));
}

#pragma mark - Multicast Delegate

- (RRMulticastDelegate *)multicastDelegate
{
    id multicastDelegate = [self objectForKey: MULTICASTDELEGATE];
    if (multicastDelegate == nil) {
        multicastDelegate = [[RRMulticastDelegate alloc] init];

        [self setObject: multicastDelegate forKey: MULTICASTDELEGATE];
    }

    return multicastDelegate;
}

- (RRMulticastDelegate *)multicastDatasource
{
    id multicastDatasource = [self objectForKey: MULTICASTDATASOURCE];
    if (multicastDatasource == nil) {
        multicastDatasource = [[RRMulticastDelegate alloc] init];

        [self setObject: multicastDatasource forKey: MULTICASTDATASOURCE];
    }

    return multicastDatasource;
}

- (void) addDelegate: (id) delegate
{
    [self.multicastDelegate addDelegate: delegate];

    UITextField *text = (UITextField *) self;
    text.delegate = self.multicastDelegate;
}

- (void) addDataSource: (id) datasource
{
    [self.multicastDatasource addDelegate: datasource];
    UITableView *text = (UITableView *) self;
    text.dataSource = self.multicastDatasource;
}

@end

```

Finalmente, puede usar delegado de multidifusión para cualquier control ...

Por ej ...

Importe su clase viewcontroller en el archivo `NSObject+RRProperty.h` para acceder a sus métodos para **configurar un delegado / fuente de datos de multidifusión** .

```

UITextView *txtView = [[UITextView alloc] initWithFrame: txtframe];
[txtView addDelegate: self];

UITableView *tblView = [[UITableView alloc] initWithFrame: tblframe];
[tblView addDelegate: self];
[tblView addDataSource: self];

```

Lea Delegados de multidifusión en línea: <https://riptutorial.com/es/ios/topic/10081/delegados-de-multidifusion>

Capítulo 59: Depuración se bloquea

Examples

Encontrar información sobre un accidente

Cuando su aplicación falla, Xcode ingresará al depurador y le mostrará más información sobre la falla:

test > My Mac

test PID 12093

- CPU 0%
- Memory 1.3 MB
- Energy Impact Zero
- Disk Zero KB/s
- Network Zero KB/s

Thread 1 Queue: com....hread (serial)

- 0 __pthread_kill
- 10 +[NSArray arrayWithObject:]
- 11 main**
- 12 start
- 13 start

```

1 //
2 //  main.m
3 //  test
4 //
5 //  Created
6 //  Copyright
7 //
8
9 #import <Fou
10
11 int main(int
12 {
13     @autorel
14         id o
15     NSLo
16 }
17     return 0
18 }
19
20

```

, obtendrá una representación textual del seguimiento de pila que puede copiar y pegar:

```
(lldb) bt
* thread #1: tid = 0x3aaec5, 0x00007fff91055f06 libsystem_kernel.dylib`__pthread_kill + 10,
queue = 'com.apple.main-thread', stop reason = signal SIGABRT
  frame #0: 0x00007fff91055f06 libsystem_kernel.dylib`__pthread_kill + 10
  frame #1: 0x000000010008142d libsystem_pthread.dylib`pthread_kill + 90
  frame #2: 0x00007fff96dc76e7 libsystem_c.dylib`abort + 129
  frame #3: 0x00007fff8973bf81 libc++abi.dylib`abort_message + 257
  frame #4: 0x00007fff89761a47 libc++abi.dylib`default_terminate_handler() + 267
  frame #5: 0x00007fff94f636ae libobjc.A.dylib`_objc_terminate() + 103
  frame #6: 0x00007fff8975f19e libc++abi.dylib`std::__terminate(void (*)()) + 8
  frame #7: 0x00007fff8975ec12 libc++abi.dylib`__cxa_throw + 121
  frame #8: 0x00007fff94f6108c libobjc.A.dylib`objc_exception_throw + 318
  frame #9: 0x00007fff8d067372 CoreFoundation`-[__NSPlaceholderArray initWithObjects:count:]
+ 290
  frame #10: 0x00007fff8d0eaalf CoreFoundation`+[NSArray arrayWithObject:] + 47
* frame #11: 0x0000000100001b54 test`main(argc=1, argv=0x00007fff5fbff808) + 68 at main.m:15
  frame #12: 0x00007fff8bea05ad libdyld.dylib`start + 1
  frame #13: 0x00007fff8bea05ad libdyld.dylib`start + 1
```

La depuración de SIGABRT y EXC_BAD_INSTRUCTION se bloquea

Un SIGABRT o un EXC_BAD_INSTRUCTION por lo general significa que la aplicación se bloqueó intencionalmente porque falló alguna comprobación. Estos deben registrar un mensaje en la consola del depurador con más información; Consulte allí para obtener más información.

Muchos SIGABRT son causados por excepciones Objective-C no detectadas. Hay *muchas* razones por las que se pueden lanzar excepciones, y *siempre* registrarán mucha información útil en la consola.

- `NSInvalidArgumentException`, lo que significa que la aplicación pasó un argumento no válido a un método
- `NSRangeException`, lo que significa que la aplicación intentó acceder a un índice fuera de los límites de un objeto como una `NSArray` o una `NSString`
- `NSInternalInconsistencyException` significa que un objeto descubrió que estaba en un estado inesperado.
- `NSUnknownKeyException` generalmente significa que tiene una conexión incorrecta en un XIB. Prueba algunas de las respuestas a [esta pregunta](#).

Depurando EXC_BAD_ACCESS

`EXC_BAD_ACCESS` significa que el proceso intentó acceder a la memoria de forma no válida, como eliminar la referencia a un puntero `NULL` o escribir en la memoria de solo lectura. Este es el tipo de bloqueo más difícil de depurar, ya que generalmente no tiene un mensaje de error, y algunos bloqueos pueden ser *muy* difíciles de reproducir y / o pueden ocurrir en un código completamente ajeno al problema. Este error es muy raro en Swift, pero si ocurre, a menudo puede obtener fallos más fáciles de depurar al reducir las optimizaciones del compilador.

La `EXC_BAD_ACCESS` errores de `EXC_BAD_ACCESS` se producen al intentar `EXC_BAD_ACCESS` referencia de un puntero `NULL`. Si este es el caso, la dirección que aparece en la flecha roja generalmente será

un número hexadecimal que es más bajo que una dirección de memoria normal, a menudo `0x0` . Establezca puntos de interrupción en el depurador o agregue sentencias `printf / NSLog` ocasionales para descubrir por qué ese puntero es `NULL` .

Un `EXC_BAD_ACCESS` que se produce de forma menos confiable o que no tiene ningún sentido podría ser el resultado de un problema de administración de memoria. Los problemas comunes que pueden causar esto son:

- Usando la memoria que ha sido desasignada
- Intentar escribir más allá del final de una matriz C u otro tipo de búfer
- Usando un puntero que no ha sido inicializado

En la sección de Diagnósticos del Editor de Esquemas, Xcode incluye algunas herramientas útiles para ayudar a depurar problemas de memoria:



test > My Mac



test > My Mac

Build
1 target

Run
Debug

Test
Debug

Profile
Release

Analyze
Debug

Archive
Release

No Debug Session

<https://riptutorial.com/es/ios/topic/4745/depuracion-se-bloquea>

Capítulo 60: Detección de rostro utilizando CoreImage / OpenCV

Examples

Detección de rostro y rasgos

C objetivo

Importe lo siguiente a su ViewController

```
#import <CoreImage/CoreImage.h>
#import <CoreImage/CoreImage.h>
#import <QuartzCore/QuartzCore.h>
```

Llamar a la función

```
[self faceDetector];
```

Definición de la función:

```
-(void)faceDetector
{
    // Load the picture for face detection
    UIImageView* image = [[UIImageView alloc] initWithImage:[UIImage
imageNamed:@"download.jpeg"]];

    // Draw the face detection image
    [self.view addSubview:image];

    // Execute the method used to markFaces in background
    [self performSelectorInBackground:@selector(markFaces:) withObject:image];

    // flip image on y-axis to match coordinate system used by core image
    [image setTransform:CGAffineTransformMakeScale(1, -1)];

    // flip the entire window to make everything right side up
    [self.view setTransform:CGAffineTransformMakeScale(1, -1)];
}
```

Marcar función de cara

```
//Adds face squares and color masks to eyes and mouth
-(void)markFaces:(UIImageView *)facePicture
{
    // draw a CI image with the previously loaded face detection picture
    CIImage* image = [CIImage imageWithCGImage:facePicture.image.CGImage];
```

```

// create a face detector - since speed is not an issue we'll use a high accuracy
// detector
CIDetector* detector = [CIDetector detectorOfType:CIDetectorTypeFace
                        context:nil options:[NSDictionary
dictionaryWithObject:CIDetectorAccuracyHigh forKey:CIDetectorAccuracy]];

// create an array containing all the detected faces from the detector
NSArray* features = [detector featuresInImage:image];
NSLog(@"Number of faces %d",[features count]);

// we'll iterate through every detected face. CIFaceFeature provides us
// with the width for the entire face, and the coordinates of each eye
// and the mouth if detected. Also provided are BOOL's for the eye's and
// mouth so we can check if they already exist.
// for (features in image)
// {
for(CIFaceFeature* faceFeature in features)
{
    // get the width of the face
    CGFloat faceWidth = faceFeature.bounds.size.width;

    // create a UIView using the bounds of the face
    UIView* faceView = [[UIView alloc] initWithFrame:faceFeature.bounds];

    // add a border around the newly created UIView
    faceView.layer.borderWidth = 1;
    faceView.layer.borderColor = [[UIColor redColor] CGColor];

    // add the new view to create a box around the face
    [self.view addSubview:faceView];

    if(faceFeature.hasLeftEyePosition)
    {
        // create a UIView with a size based on the width of the face
        UIView* leftEyeView = [[UIView alloc]
initWithFrame:CGRectMake(faceFeature.leftEyePosition.x-faceWidth*0.15,
faceFeature.leftEyePosition.y-faceWidth*0.15, faceWidth*0.3, faceWidth*0.3)];
        // change the background color of the eye view
        [leftEyeView setBackgroundColor:[UIColor blueColor]
colorWithAlphaComponent:0.3]];
        // set the position of the leftEyeView based on the face
        [leftEyeView setCenter:faceFeature.leftEyePosition];
        // round the corners
        leftEyeView.layer.cornerRadius = faceWidth*0.15;
        // add the view to the window
        [self.view addSubview:leftEyeView];
    }

    if(faceFeature.hasRightEyePosition)
    {
        // create a UIView with a size based on the width of the face
        UIView* leftEye = [[UIView alloc]
initWithFrame:CGRectMake(faceFeature.rightEyePosition.x-faceWidth*0.15,
faceFeature.rightEyePosition.y-faceWidth*0.15, faceWidth*0.3, faceWidth*0.3)];
        // change the background color of the eye view
        [leftEye setBackgroundColor:[UIColor blueColor] colorWithAlphaComponent:0.3]];
        // set the position of the rightEyeView based on the face
        [leftEye setCenter:faceFeature.rightEyePosition];
        // round the corners
        leftEye.layer.cornerRadius = faceWidth*0.15;
        // add the new view to the window
    }
}

```

```

        [self.view addSubview:leftEye];
    }

    if(faceFeature.hasMouthPosition)
    {
        // create a UIView with a size based on the width of the face
        UIView* mouth = [[UIView alloc]
initWithFrame:CGRectMake(faceFeature.mouthPosition.x-faceWidth*0.2,
faceFeature.mouthPosition.y-faceWidth*0.2, faceWidth*0.4, faceWidth*0.4)];
        // change the background color for the mouth to green
        [mouth setBackgroundColor:[UIColor greenColor] colorWithAlphaComponent:0.3];
        // set the position of the mouthView based on the face
        [mouth setCenter:faceFeature.mouthPosition];
        // round the corners
        mouth.layer.cornerRadius = faceWidth*0.2;
        // add the new view to the window
        [self.view addSubview:mouth];
    }
}

// }
}

```

El simulador de pantalla para la función



Lea Detección de rostro utilizando CoreImage / OpenCV en línea:

<https://riptutorial.com/es/ios/topic/7298/deteccion-de-rostro-utilizando-coreimage---opencv>

Capítulo 61: Diseño automático

Introducción

El diseño automático calcula dinámicamente el tamaño y la posición de todas las vistas en su jerarquía de vistas, según las restricciones puestas en esas vistas. [Fuente](#)

Sintaxis

- `NSLayoutConstraint` (item: Any, atributo: `NSLayoutConstraintAttribute`, relatedBy: `NSLayoutConstraintRelation`, toItem: Any, attribute: `NSLayoutConstraintAttribute`, multiplicador: `CGFloat`, constante: `CGFloat`) // Crear un constraint

Examples

Establecer restricciones programáticamente

Ejemplo de código repetitivo

```
override func viewDidLoad() {
    super.viewDidLoad()

    let myView = UIView()
    myView.backgroundColor = UIColor.blueColor()
    myView.translatesAutoresizingMaskIntoConstraints = false
    view.addSubview(myView)

    // Add constraints code here
    // ...
}
```

En los ejemplos a continuación, el Estilo de Anclaje es el método preferido sobre el Estilo `NSLayoutConstraint`, sin embargo, solo está disponible desde iOS 9, por lo que si está soportando iOS 8, entonces aún debe usar el Estilo `NSLayoutConstraint`.

Fijación

Estilo de ancla

```
let margins = view.layoutMarginsGuide
myView.leadingAnchor.constraintEqualToAnchor(margins.leadingAnchor, constant: 20).active = true
```

- Además de `leadingAnchor`, también hay `trailingAnchor`, `topAnchor` y `bottomAnchor`.

NSLayoutConstraint Style

```
NSLayoutConstraint(item: myView, attribute: NSLayoutConstraintAttribute.Leading, relatedBy:
NSLayoutConstraintRelation.Equal, toItem: view, attribute: NSLayoutConstraintAttribute.LeadingMargin, multiplier:
1.0, constant: 20.0).active = true
```

- Además de `.Leading` también hay `.Trailing`, `.Top` y `.Bottom`.
- Además de `.LeadingMargin` también hay `.TrailingMargin`, `.TopMargin` y `.BottomMargin`.

Estilo de lenguaje de formato visual

```
NSLayoutConstraint.constraintsWithVisualFormat("H:|-20-[myViewKey]", options: [], metrics:
nil, views: ["myViewKey": myView])
```

Anchura y altura

Estilo de ancla

```
myView.widthAnchor.constraintEqualToAnchor(nil, constant: 200).active = true
myView.heightAnchor.constraintEqualToAnchor(nil, constant: 100).active = true
```

NSLayoutConstraint Style

```
NSLayoutConstraint(item: myView, attribute: NSLayoutConstraintAttribute.Width, relatedBy:
NSLayoutConstraintRelation.Equal, toItem: nil, attribute: NSLayoutConstraintAttribute.NotAnAttribute, multiplier:
1, constant: 200).active = true
NSLayoutConstraint(item: myView, attribute: NSLayoutConstraintAttribute.Height, relatedBy:
NSLayoutConstraintRelation.Equal, toItem: nil, attribute: NSLayoutConstraintAttribute.NotAnAttribute, multiplier:
1, constant: 100).active = true
```

Estilo de lenguaje de formato visual

```
NSLayoutConstraint.constraintsWithVisualFormat("H:[myViewKey(200)]", options: [], metrics:
nil, views: ["myViewKey": myView])
NSLayoutConstraint.constraintsWithVisualFormat("V:[myViewKey(100)]", options: [], metrics:
nil, views: ["myViewKey": myView])
```

Centro en contenedor

Estilo de ancla

```
myView.centerXAnchor.constraintEqualToAnchor(view.centerXAnchor).active = true
myView.centerYAnchor.constraintEqualToAnchor(view.centerYAnchor).active = true
```

NSLayoutConstraint Style

```
NSLayoutConstraint(item: myView, attribute: NSLayoutConstraintAttribute.CenterX, relatedBy:
NSLayoutConstraintRelation.Equal, toItem: view, attribute: NSLayoutConstraintAttribute.CenterX, multiplier: 1,
constant: 0).active = true
NSLayoutConstraint(item: myView, attribute: NSLayoutConstraintAttribute.CenterY, relatedBy:
```



```
NSLayoutConstraint.Equal, toItem: view, attribute: NSLayoutConstraint.CenterY, multiplier: 1, constant: 0).active = true
```

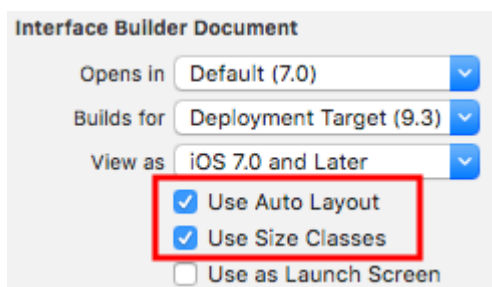
Estilo de lenguaje de formato visual

```
NSLayoutConstraint.constraintsWithVisualFormat("V:[viewKey]-(<=0)-[myViewKey]", options: NSLayoutConstraint.FormatOptions.AlignAllCenterX, metrics: nil, views: ["myViewKey": myView, "viewKey": view])\nNSLayoutConstraint.constraintsWithVisualFormat("H:[viewKey]-(<=0)-[myViewKey]", options: NSLayoutConstraint.FormatOptions.AlignAllCenterY, metrics: nil, views: ["myViewKey": myView, "viewKey": view])
```

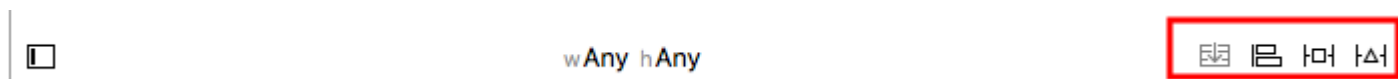
Cómo usar Auto Layout

El diseño automático se utiliza para organizar las vistas para que se vean bien en cualquier dispositivo y orientación. Las restricciones son las reglas que indican cómo debe establecerse todo. Incluyen bordes de fijación, centrado y configuración de tamaños, entre otras cosas.

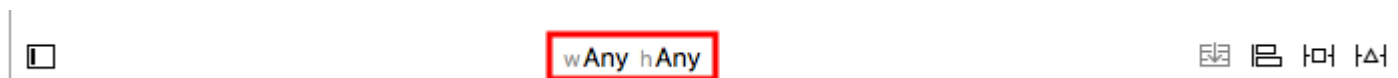
El diseño automático está habilitado de forma predeterminada, pero puede verificarlo dos veces. Si hace clic en *Main.storyboard* en el Project Navigator y luego muestra el inspector de archivos. Asegúrese de que las clases de diseño automático y tamaño estén marcadas:



Las restricciones de diseño automático se pueden establecer en el generador de interfaz o en el código. En el Creador de interfaces, encontrará las herramientas de diseño automático en la parte inferior derecha. Al hacer clic en ellos se revelarán diferentes opciones para establecer las restricciones en una vista.

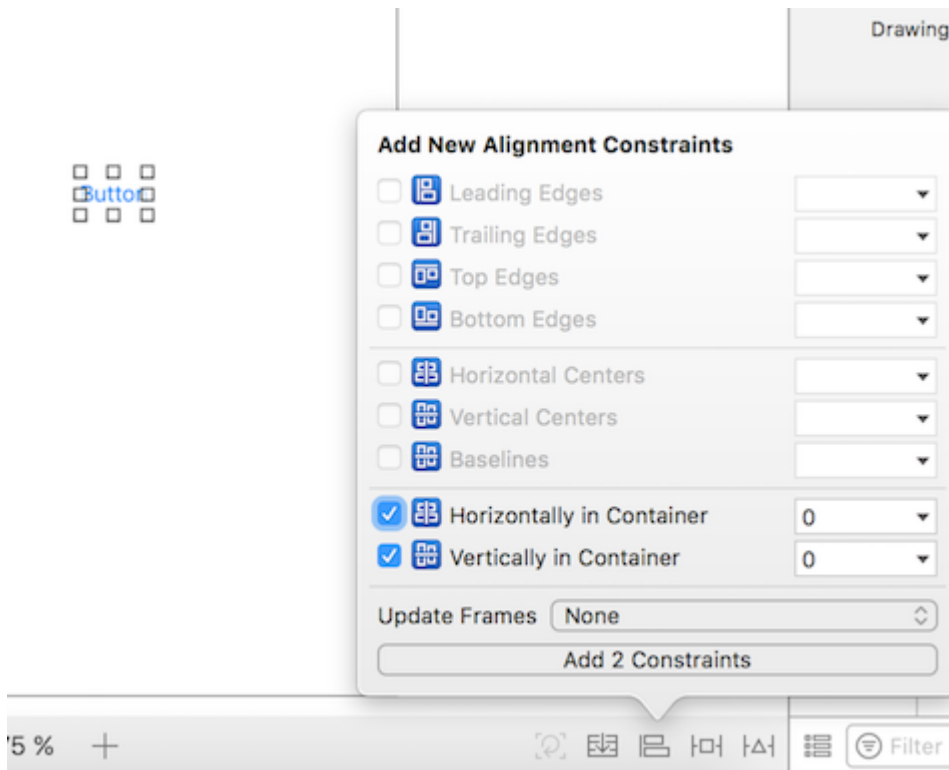


Si desea tener diferentes restricciones para diferentes tamaños de dispositivos u orientaciones, puede configurarlas en todas las opciones de cualquier clase de tamaño que se encuentren en la parte inferior central.

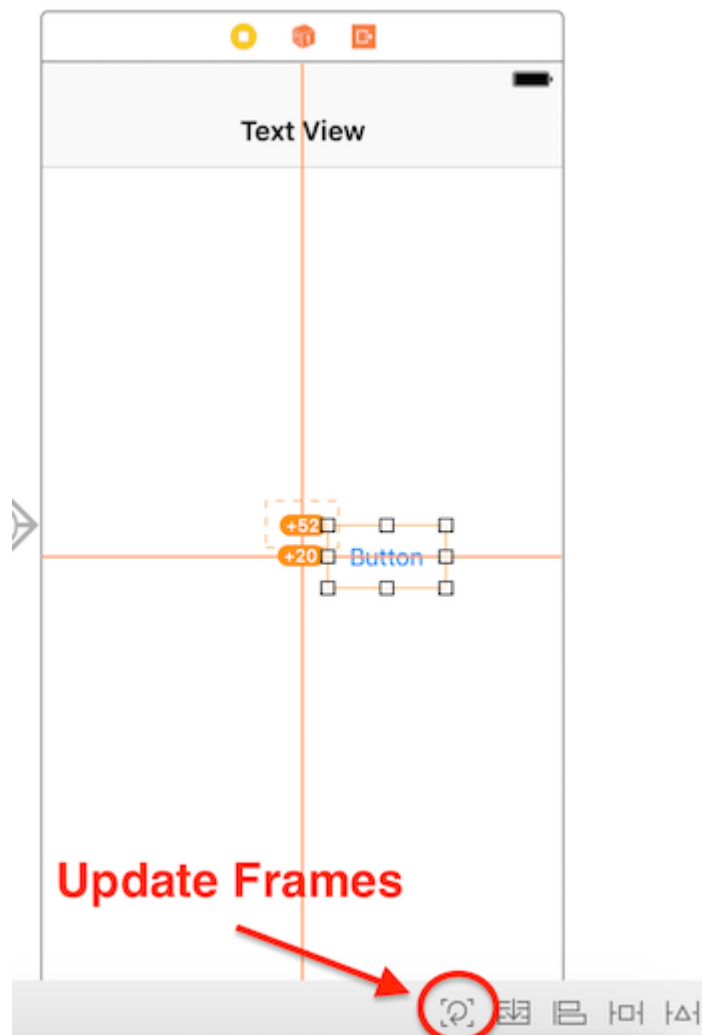


Restricciones del centro

Seleccione su botón (o la vista que desee centrar) en el **guión gráfico** . Luego haga clic en el botón de alinear en la parte inferior derecha. Seleccione `Horizontally in Container` y `Vertically in Container` . Haga clic en "Añadir 2 restricciones".



Si ya no estaba perfectamente centrado, es posible que tengas que hacer una cosa más. Haga clic en el botón "Actualizar marcos" que se encuentra dos a la izquierda del botón "Incrustar en la

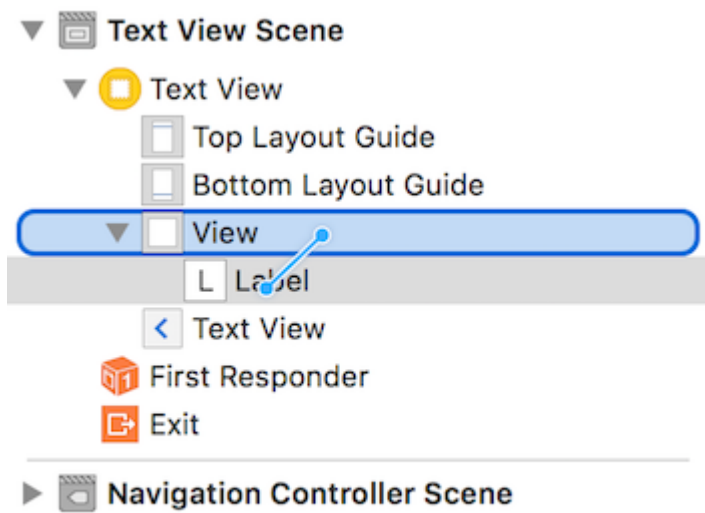


pila" en la barra inferior.

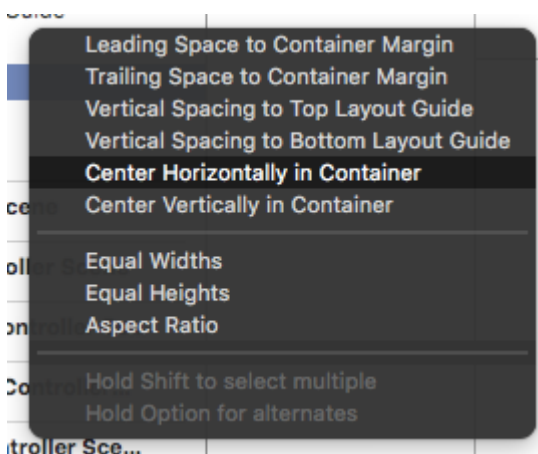
También puede "actualizar los cuadros según sea necesario" presionando juntos $\square + \square + =$ (Comando + Opción e igual) después de seleccionar la vista, esto puede ahorrar algo de tiempo.

Ahora, cuando ejecute su aplicación, debería estar centrada, independientemente del tamaño del dispositivo que esté utilizando.

Otra forma de centrar las vistas utilizando el Generador de Interfaz es mediante la tecla de control-clic-arrastrar. Digamos que quieres centrar una `UILabel` en una vista. Abra el `Document Outline` del `Document Outline` en su guión gráfico haciendo clic en el botón de la barra lateral en la parte inferior izquierda. Haga clic y arrastre desde la etiqueta a la vista mientras mantiene presionado `ctrl` (control), y debería aparecer una línea azul:



Al publicarse, aparecerá un menú de opciones de restricción:



Seleccione "Centrar horizontalmente en contenedor" y "Centrar verticalmente en contenedor". Actualizar marcos según sea necesario, y ¡voilà! Una etiqueta centrada.

Alternativamente, puede agregar las restricciones programáticamente. Cree las restricciones y agréguelas a los elementos y vistas deseados de la interfaz de usuario, como se describe en el siguiente ejemplo, donde creamos un botón y lo alineamos en el centro, horizontal y verticalmente a su supervisión:

C objetivo

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    UIButton *yourButton = [[UIButton alloc] initWithFrame:CGRectMake(0, 0, 100, 18)];
    [yourButton setTitle:@"Button" forState:UIControlStateNormal];

    [self.view addConstraint:[NSLayoutConstraint constraintWithItem:yourButton
attribute:NSLayoutAttributeCenterY relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeCenterY multiplier:1 constant:0]]; //Align vertically center to
superView

    [self.view addConstraint:[NSLayoutConstraint constraintWithItem:yourButton
attribute:NSLayoutAttributeCenterX relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeCenterX multiplier:1 constant:0]]; //Align horizontally center to
superView

    [self.view addSubview:yourButton]; //Add button to superView
}
```

Rápido

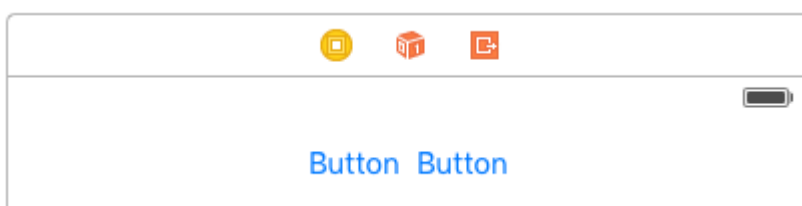
```
override func viewDidLoad()
{
    super.viewDidLoad()
    let yourButton: UIButton = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 18))
    yourButton.setTitle("Button", forState: .Normal)

    let centerVertically = NSLayoutConstraint(item: yourButton,
attribute: .CenterX,
relatedBy: .Equal,
toItem: view,
attribute: .CenterX,
multiplier: 1.0,
constant: 0.0)

    let centerHorizontally = NSLayoutConstraint(item: yourButton,
attribute: .CenterY,
relatedBy: .Equal,
toItem: view,
attribute: .CenterY,
multiplier: 1.0,
constant: 0.0)

    NSLayoutConstraint.activateConstraints([centerVertically, centerHorizontally])
}
```

Vistas espaciales uniformemente



Es común que dos vistas estén una al lado de la otra, centradas en su supervisión. La respuesta

común dada en Stack Overflow es incrustar estas dos vistas en una vista `UIView` y centrar la vista `UIView`. Esto no es necesario ni recomendado. De los documentos de [UILayoutGuide](#) :

Hay una serie de costos asociados con la adición de vistas ficticias a su jerarquía de vistas. Primero, está el costo de crear y mantener la vista en sí. En segundo lugar, la vista ficticia es un miembro completo de la jerarquía de vistas, lo que significa que agrega sobrecarga a cada tarea que realiza la jerarquía. Lo peor de todo es que la vista ficticia invisible puede interceptar mensajes destinados a otras vistas, lo que causa problemas que son muy difíciles de encontrar.

Puede usar `UILayoutGuide` para hacer esto, en lugar de agregar los botones a una vista `UIView` innecesaria. Un `UILayoutGuide` es esencialmente un espacio rectangular que puede interactuar con el diseño automático. Usted coloca un `UILayoutGuide` en los lados izquierdo y derecho de los botones y establece que sus anchos sean iguales. Esto centrará los botones. Aquí está cómo hacerlo en código:

Estilo de lenguaje de formato visual

```
view.addSubview(button1)
view.addSubview(button2)

let leftSpace = UILayoutGuide()
view.addLayoutGuide(leftSpace)

let rightSpace = UILayoutGuide()
view.addLayoutGuide(rightSpace)

let views = [
    "leftSpace" : leftSpace,
    "button1" : button1,
    "button2" : button2,
    "rightSpace" : rightSpace
]

// Lay the buttons and layout guides out horizontally in a line.
// Put the layout guides on each end.
NSLayoutConstraint.activateConstraints(NSLayoutConstraint.constraintsWithVisualFormat("H:|[leftSpace][button2][rightSpace]|", options: [], metrics: nil, views: views))

// Now set the layout guides widths equal, so that the space on the
// left and the right of the buttons will be equal
leftSpace.widthAnchor.constraintEqualToAnchor(rightSpace.widthAnchor).active = true
```

Estilo de ancla

```
let leadingSpace = UILayoutGuide()
let trailingSpace = UILayoutGuide()
view.addLayoutGuide(leadingSpace)
view.addLayoutGuide(trailingSpace)

leadingSpace.widthAnchor.constraintEqualToAnchor(trailingSpace.widthAnchor).active = true

leadingSpace.leadingAnchor.constraintEqualToAnchor(view.leadingAnchor).active = true
leadingSpace.trailingAnchor.constraintEqualToAnchor(button1.leadingAnchor).active = true
```

```
trailingSpace.leadingAnchor.constraintEqualToAnchor(button2.trailingAnchor).active = true
trailingSpace.trailingAnchor.constraintEqualToAnchor(view.trailingAnchor).active = true
```

También deberá agregar restricciones verticales a esto, ¡pero esto centrará los botones en la vista sin agregar ninguna vista "ficticia"! Esto evitará que el sistema pierda tiempo de CPU al mostrar esas vistas "ficticias". Este ejemplo utiliza botones, pero puede intercambiar botones para cualquier vista en la que quiera poner restricciones.

Si admite iOS 8 o una versión anterior, la forma más sencilla de crear este diseño es agregar vistas ficticias ocultas. Con iOS 9 puede reemplazar las vistas ficticias con guías de diseño.

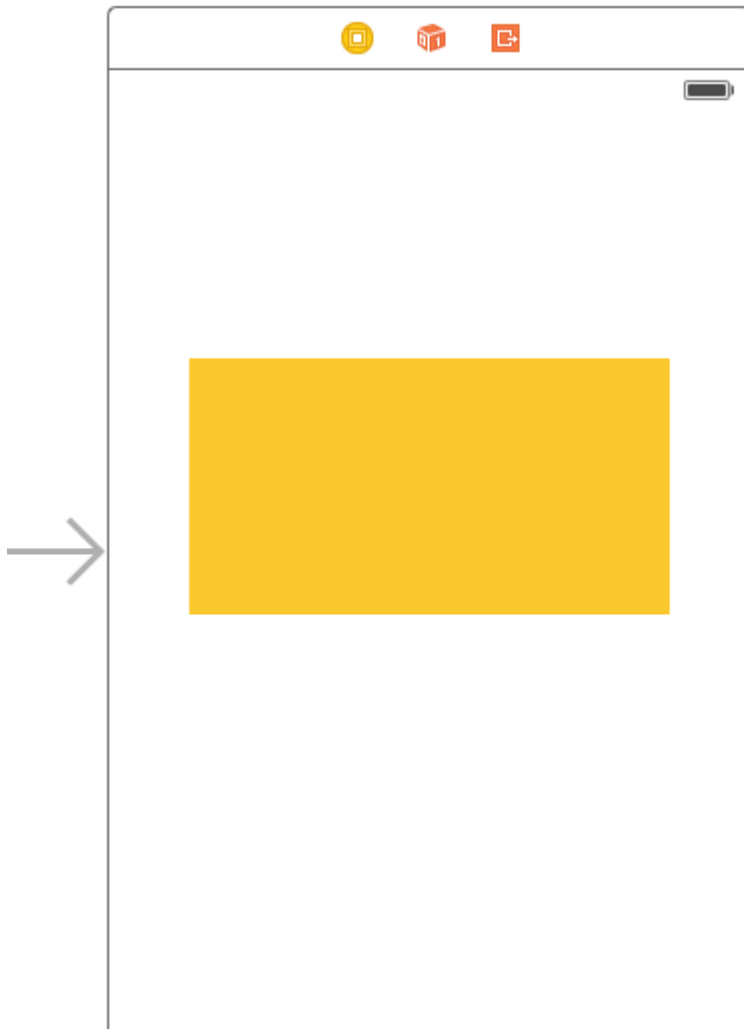
Nota: Interface Builder aún no admite guías de diseño (Xcode 7.2.1). Por lo tanto, si desea usarlos, debe crear sus restricciones en el código. [Fuente](#)

UILabel tamaño intrínseco

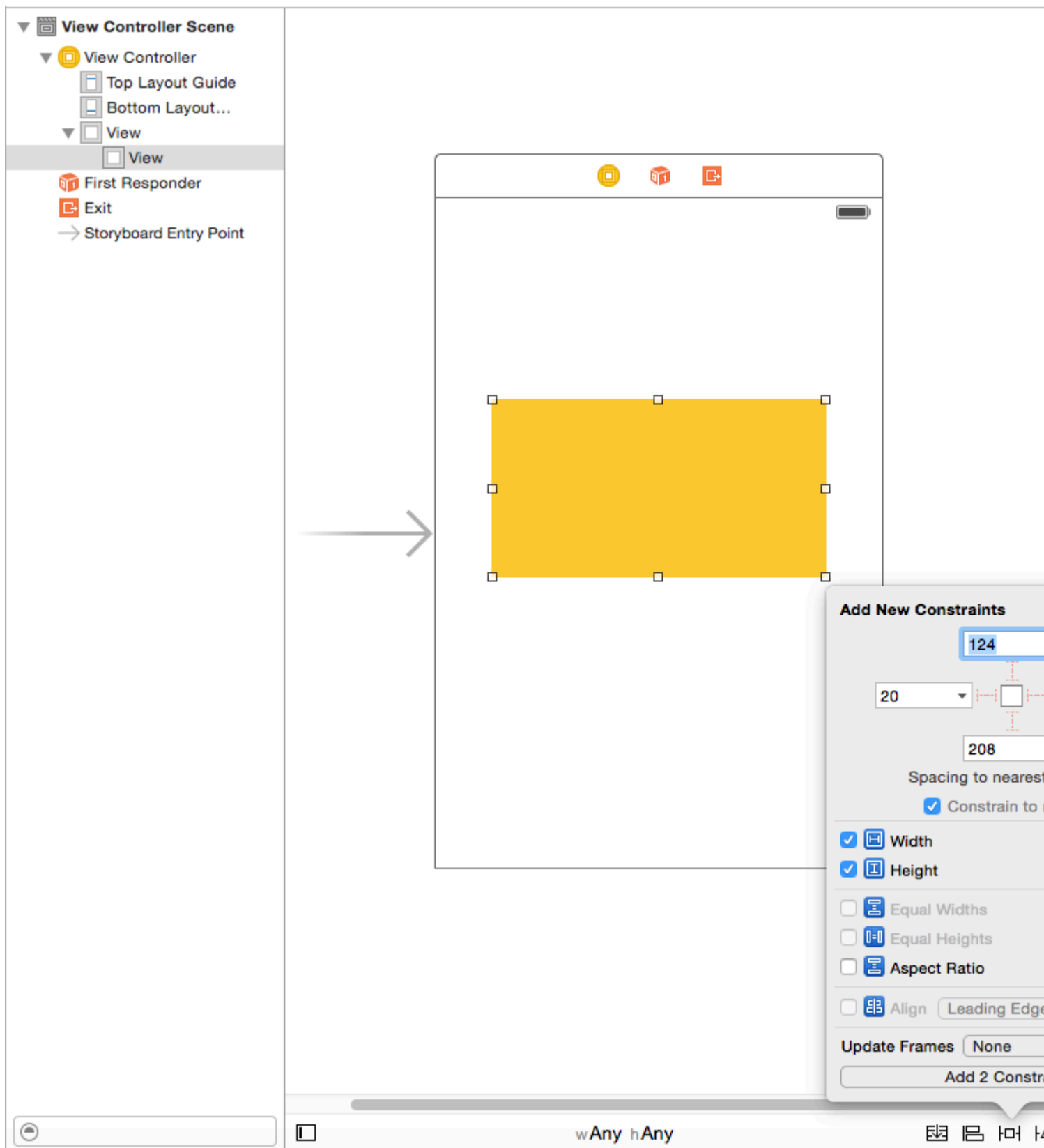
Tenemos que crear una vista que tendrá un prefijo de imagen a un texto. el texto puede ser de longitud variable. Tenemos que lograr un resultado en el que, en Imagen +, el texto esté siempre en el centro de la vista principal.



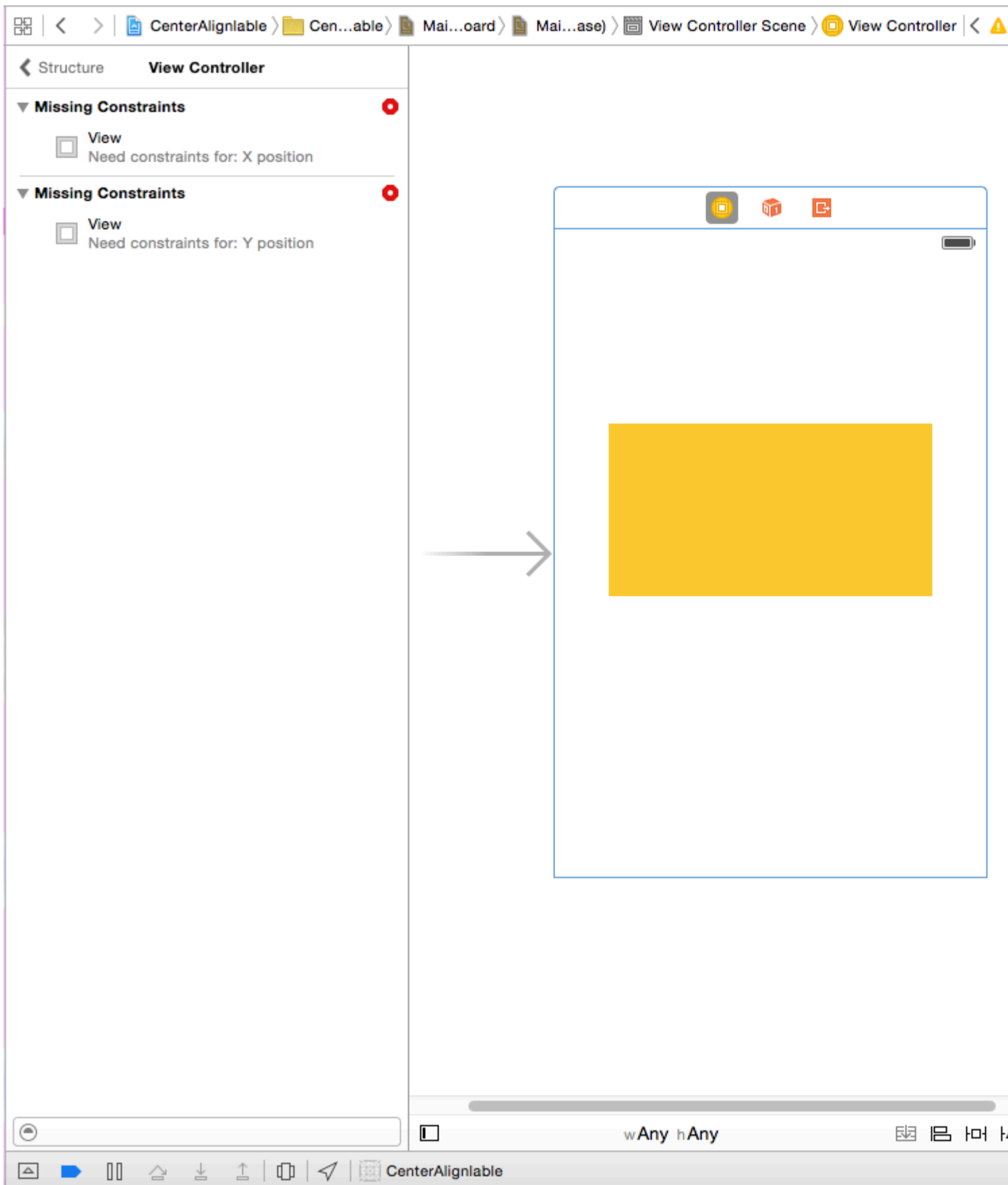
Paso 1: primero cree un proyecto de vista única y asígnele el nombre que prefiera y abra la primera vista del guión gráfico. Arrastre una vista con un tamaño razonable y configure el color de fondo en amarillo. He cambiado el tamaño de mi controlador de vista a 3.5 ". La resultante La vista debería verse algo como esto.



Paso 2: Ahora agregaremos restricciones a la vista amarilla. Para comenzar, agregaremos restricciones de ancho y altura (¿Espera un minuto? ¿No dijimos que la vista tendrá un ancho dinámico? Ok, volveremos a ella más adelante) Agregar el Las siguientes restricciones, según la imagen a continuación, no molestan con el valor de ancho, cualquier valor funcionará bien para el ancho, pero manténgalo lo suficientemente grande para que podamos agregar las salidas automáticas correctamente.



Después de agregar estas dos restricciones, verá que XCode le está dando errores, como se muestra en la imagen de abajo, para verlos y comprenderlos.

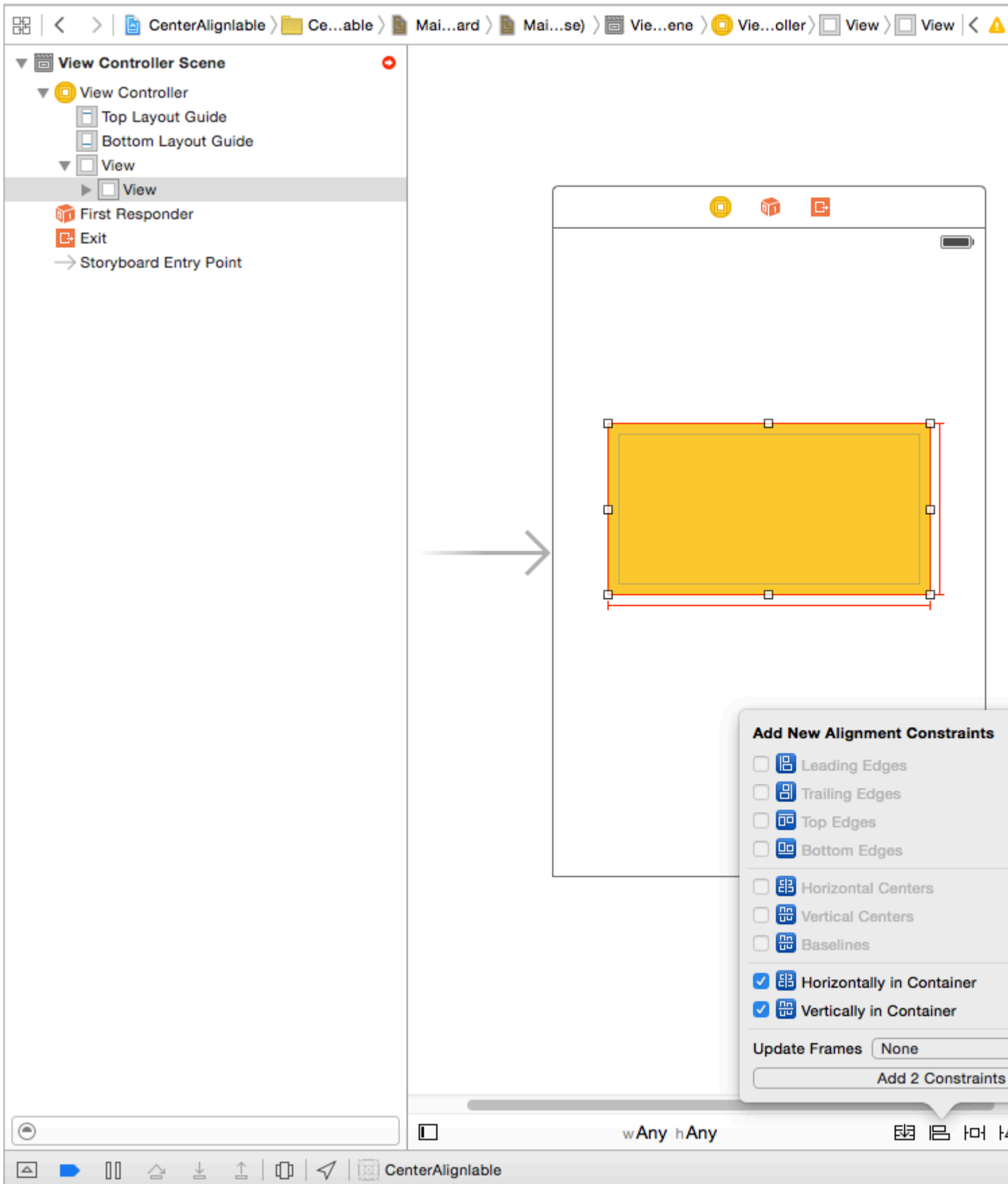


Tenemos dos errores (rojo significa error) Como se explicó anteriormente, revisemos la parte de ambigüedad

Restricciones que faltan: Necesidad de restricciones para: Posición X: - Como se mencionó anteriormente, le hemos dado a la vista un ancho y una altura, por lo que se define su

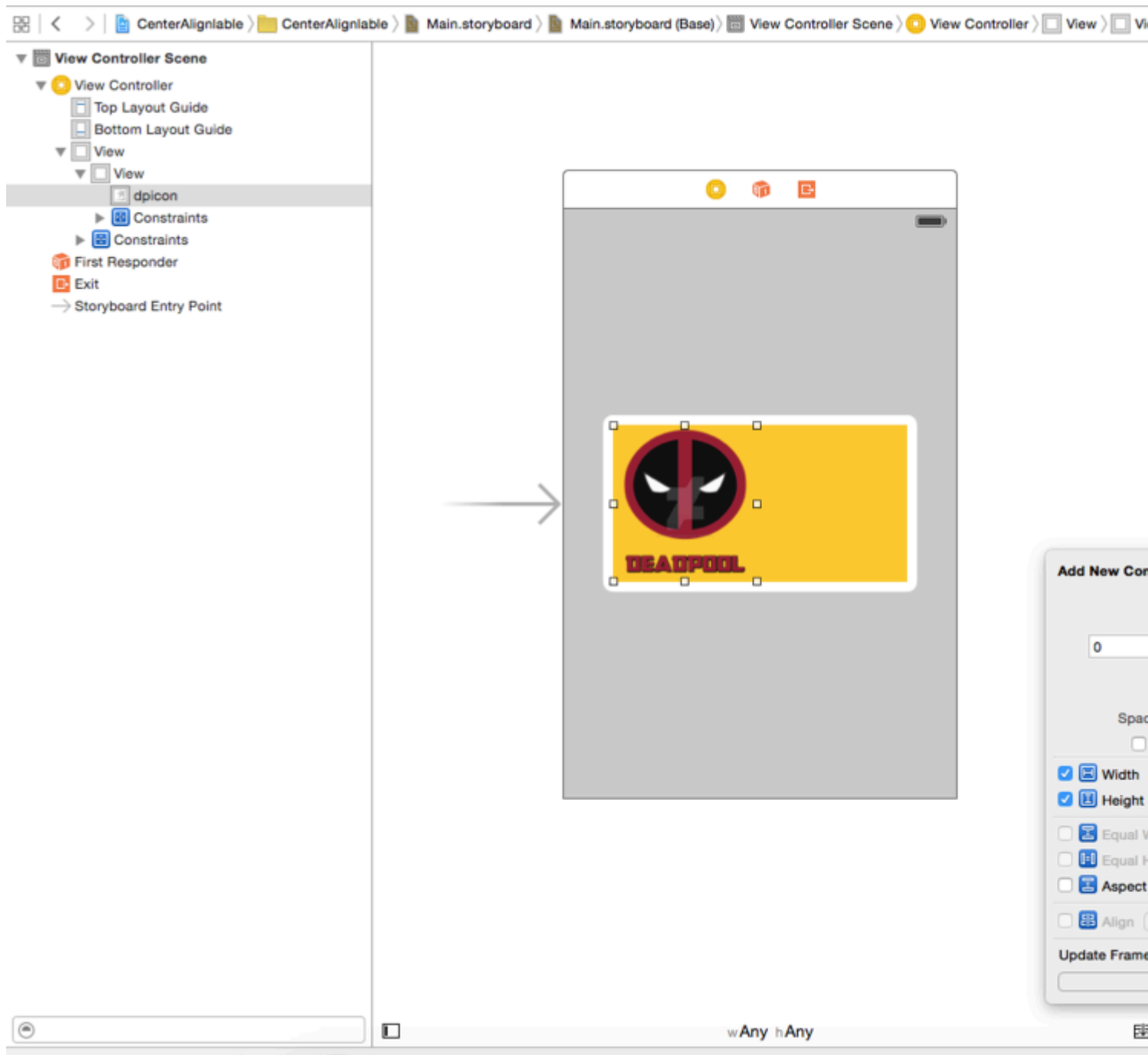
"RAZONES", pero no hemos dado su origen, por lo que no se define su "MARCO". Autolayout no puede determinar cuál será la posición X de nuestra vista amarilla

Restricciones que faltan: Necesidad de restricciones para: Posición Y: - Como se mencionó anteriormente, le hemos dado a la vista un ancho y una altura, por lo que se define su "RAZONES", pero no hemos dado su origen, por lo que no se define su "MARCO". Autolayout no puede determinar cuál será la posición Y de nuestra vista amarilla. Para resolver esto, tenemos que darle a Autolayout algo para resolver X e Y. Ya que no podemos establecer fotogramas, lo haremos de forma automática. Agregue las siguientes restricciones según la La imagen que figura a continuación la explicaré más adelante.



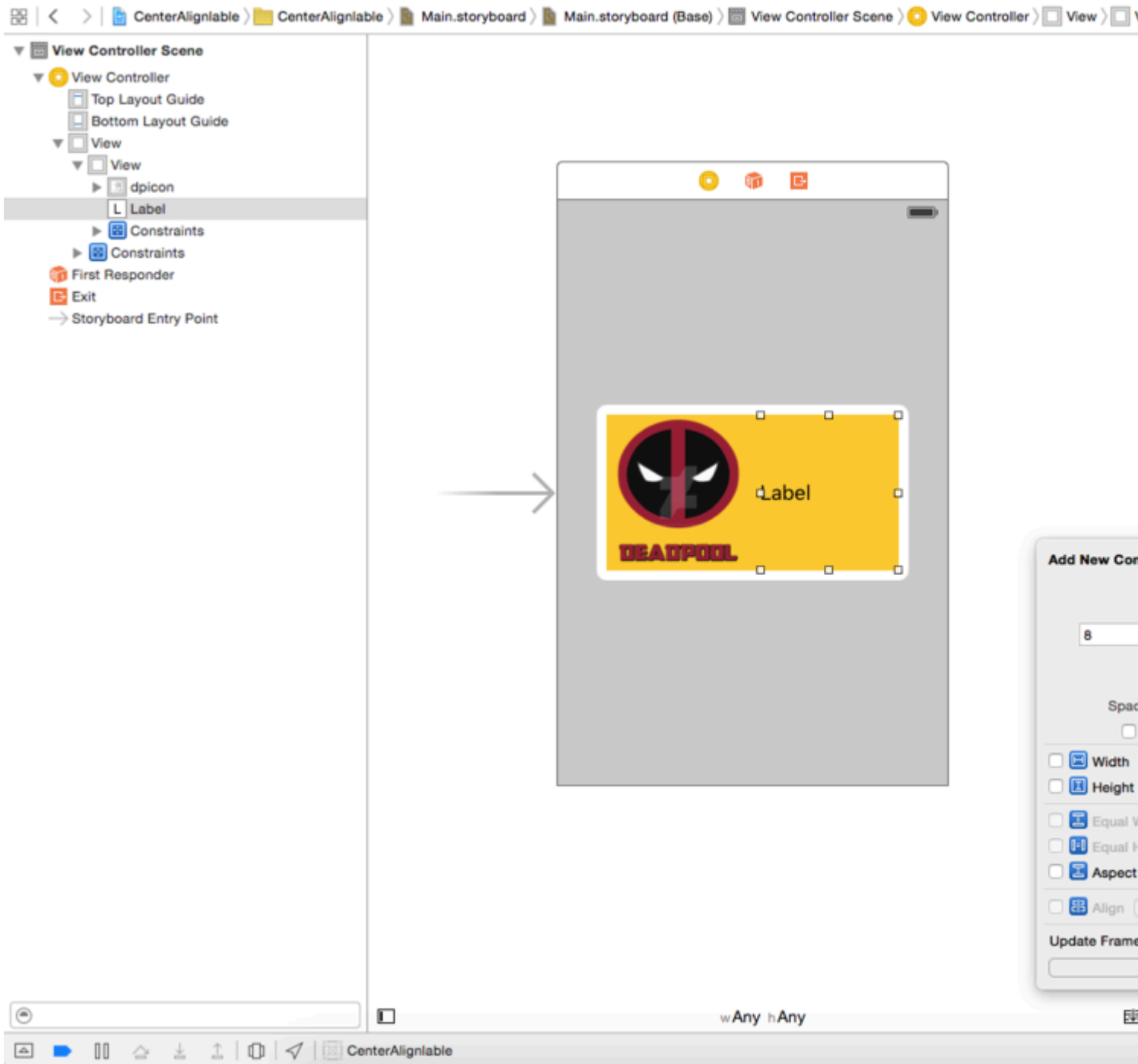
Lo que hemos hecho es: hemos agregado un "Centro vertical" y un "Centro horizontal". Estas restricciones indican a la disposición automática que nuestra vista amarilla siempre estará en el centro horizontalmente: por lo tanto, X en el mismo determinado está con restricción vertical y se determina Y. podría tener que ajustar el marco).

Paso 3: Por ahora nuestra vista base amarilla está lista. Agregaremos la imagen de prefijo como subvista de nuestra vista amarilla con las siguientes restricciones. Puede elegir cualquier imagen de su elección.



Como tenemos una dimensión fija para nuestra imagen de prefijo, tendremos una altura de ancho fija para esta vista de imagen. Agregue las restricciones y continúe con el siguiente paso.

Paso 4: agregue una UILabel como la vista secundaria de nuestra vista amarilla y agregue las siguientes restricciones

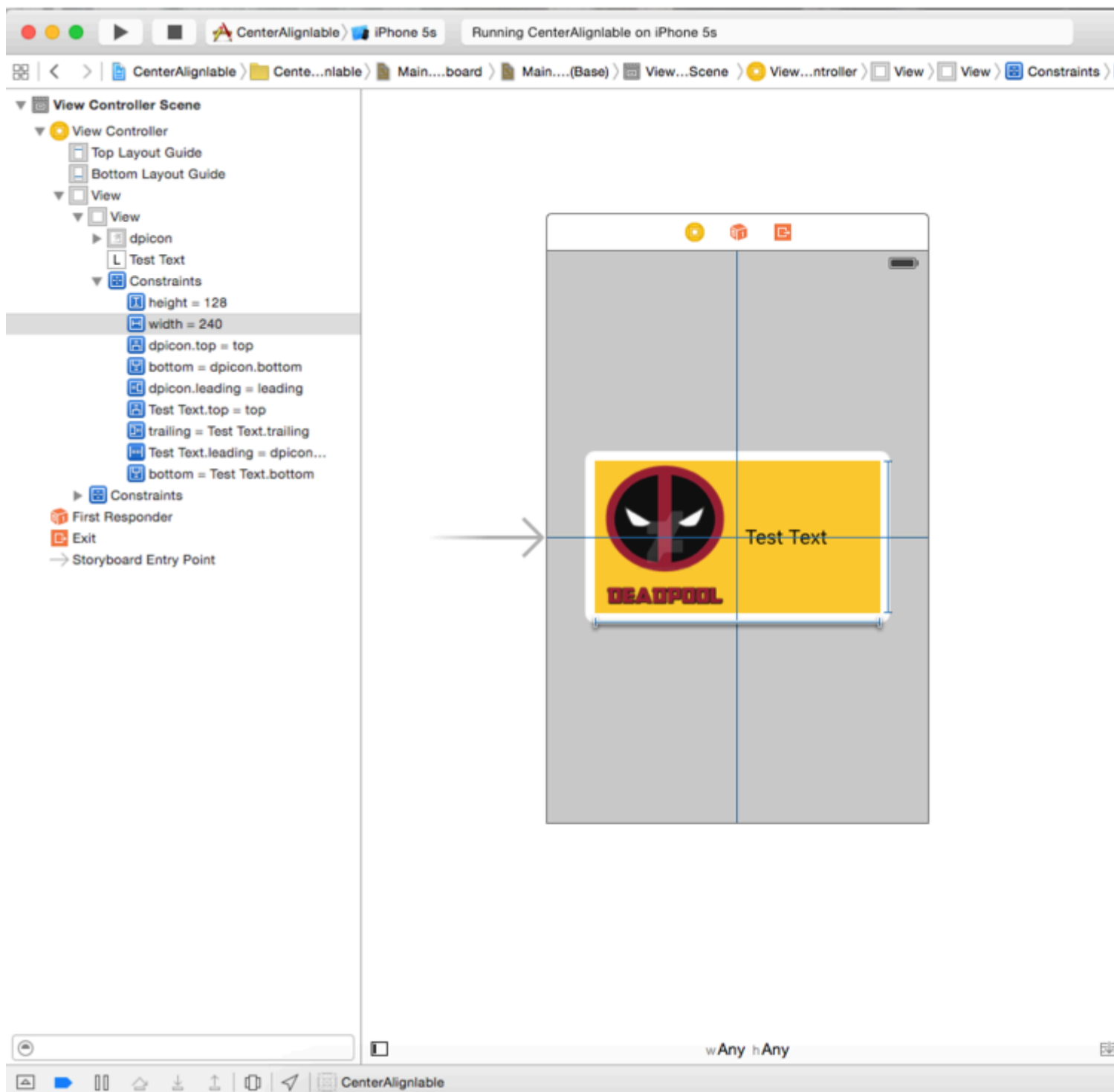


Como puede ver, solo le he dado restricciones relativas a nuestra UILabel. Es 8 puntos desde la imagen del prefijo y 0,0,0 arriba y abajo desde la vista amarilla. Ya que queremos que el ancho sea dinámico, no daremos restricciones de ancho o altura. .

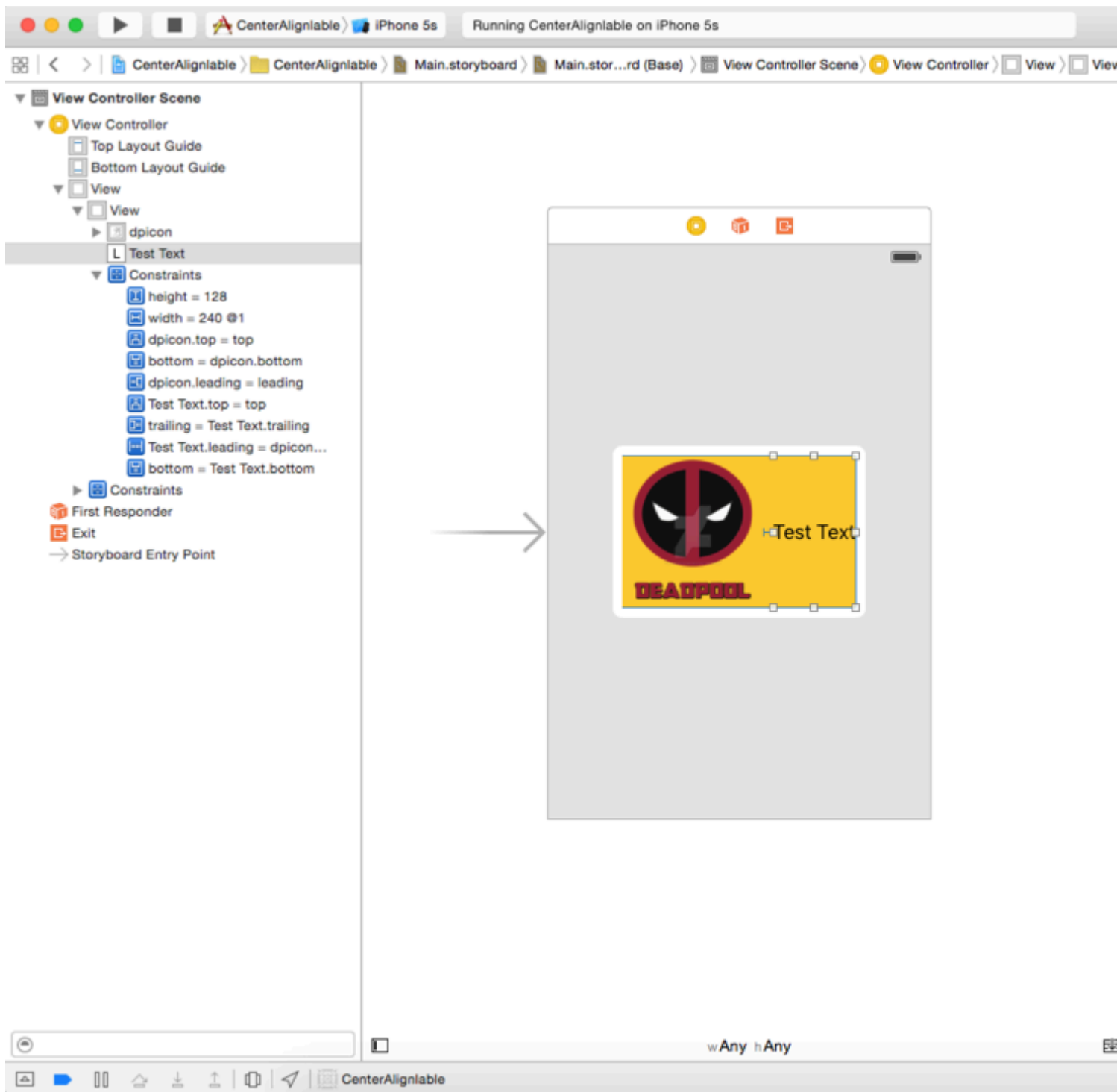
P: ¿Por qué no estamos recibiendo ningún error ahora, no hemos dado ningún ancho y altura?
Respuesta: - Recibimos un error o una advertencia solo cuando el diseño automático no es capaz de resolver cualquier cosa que sea necesaria para representar una vista en la pantalla. Sea de ancho o de origen. Ya que nuestra etiqueta es relativa a la vista amarilla y la imagen del prefijo y sus marcos Está bien definido. Autolayout es capaz de calcular el marco de nuestra etiqueta.

Paso 5: Ahora si recordamos, nos daremos cuenta de que hemos dado una vista fija a nuestra vista amarilla, pero queremos que sea dinámica y dependa del texto de nuestra etiqueta. Por lo tanto, modificaremos nuestra restricción de ancho de la vista amarilla. Ancho de la vista amarilla

es necesario para resolver la ambigüedad, pero queremos que se anule en el tiempo de ejecución en función del contenido de UILabel. Así que seleccionaremos nuestra vista amarilla e iremos al inspector de tamaño y reduciremos la prioridad de la restricción de ancho a 1 para que quede descartada. Sigue la imagen de abajo.

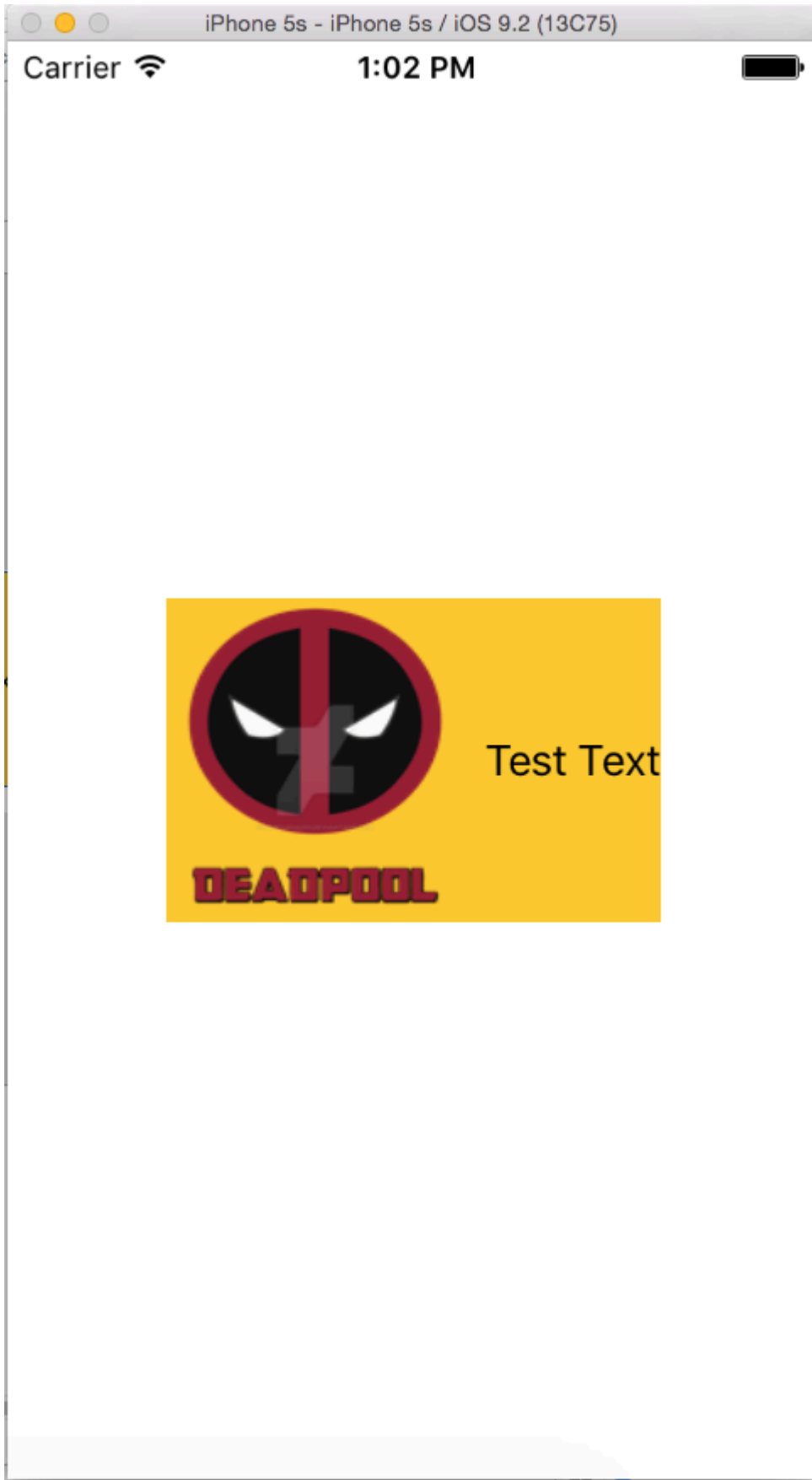


Paso 6: Queremos que UILabel se expanda de acuerdo con el texto y amplíe nuestra vista amarilla. Por lo tanto, hemos reducido la prioridad del ancho de la vista amarilla. Ahora aumentaremos la prioridad de la resistencia de compresión de texto de nuestra UILabel. Queremos que nuestra vista se reduzca Bueno, así aumentaremos la prioridad de contenido de UILabel. Siga la imagen a continuación.



Como puede ver, hemos aumentado la prioridad de compatibilidad de contenido a 500 y la prioridad de resistencia a la compresión a 751, lo que superará con éxito la prioridad 1 de la restricción de ancho.

Ahora construye y ejecuta verás algo como lo siguiente.



Cómo animar con diseño automático

Sin diseño automático, la animación se realiza cambiando el marco de una vista a lo largo del tiempo. Con Auto Layout, las restricciones dictan el marco de la vista, por lo que tiene que animar

las restricciones en su lugar. Esta indirección hace que la animación sea más difícil de visualizar.

Estas son las formas de animar con Auto Layout:

1. **Cambie la constante de la restricción** después de la creación mediante llamadas periódicas (`CADisplayLink` , `dispatch_source_t` , `dispatch_after` , `NSTimer`). Luego llame a `layoutIfNeeded` para actualizar la restricción. Ejemplo:

C objetivo:

```
self.someConstraint.constant = 10.0;
[UIView animateWithDuration:0.25 animations:^(
    [self.view layoutIfNeeded];
)];
```

Rápido:

```
self.someConstraint.constant = 10.0
UIView.animate(withDuration: 0.25, animations: self.view.layoutIfNeeded)
```

2. **Cambie las restricciones** y llame a `[view layoutIfNeeded]` dentro de un bloque de animación. Esto se interpola entre las dos posiciones haciendo caso omiso de las restricciones durante la animación.

```
[UIView animateWithDuration:0.5 animations:^(
    [view layoutIfNeeded];
)]
```

3. **Cambiar la prioridad de las restricciones** . Esto requiere menos CPU que agregar y eliminar restricciones.
4. **Eliminar todas las restricciones y utilizar máscaras de tamaño automático** . Para más adelante, debe configurar `view.translatesAutoresizingMaskIntoConstraints = YES` .
5. **Use restricciones que no interfieran con la animación deseada** .
6. **Utilice una vista de contenedor** . Posiciona la supervisión con restricciones. Luego, agregue una subvista con restricciones que no contrarresten la animación, por ejemplo: un centro en relación con la supervisión. Esto descarga parte de las restricciones a la vista supervisada, por lo que no combaten la animación en la subvista.
7. **Animar capas en lugar de vistas** . Las transformaciones de capa no activan el diseño automático.

```
CABasicAnimation* ba = [CABasicAnimation animationWithKeyPath:@"transform"];
ba.autoreverses = YES;
ba.duration = 0.3;
ba.toValue = [NSValue valueWithCATransform3D:CATransform3DMakeScale(1.1, 1.1, 1)];
[v.layer addAnimation:ba forKey:nil];
```

8. **Reemplazar `layoutSubviews`** . Llame a `[super layoutSubviews]` y afine las restricciones.

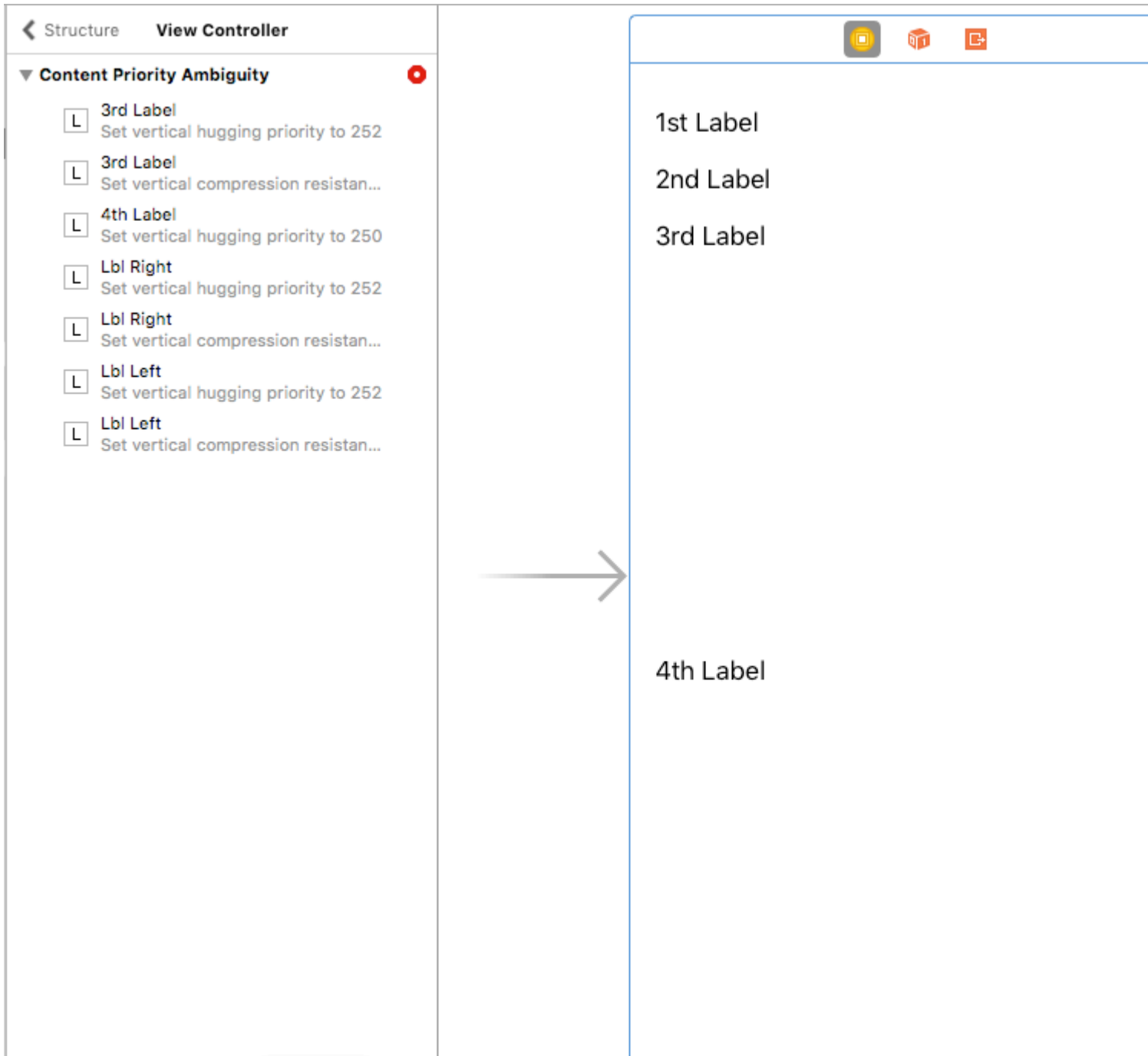
9. **Cambie el marco en `viewDidLayoutSubviews`** . El diseño automático se aplica en `layoutSubviews` , así que una vez hecho, cámbielo en `viewDidLayoutSubviews` .

10. **Optar por salir de `Auto Layout`** y establecer vistas manualmente. Puede hacer esto anulando `layoutSubviews / layout` sin llamar a la implementación de la súper clase.

Sugerencia rápida: si el elemento primario de la vista animada no se está interpolando (es decir, la animación salta del estado inicial al final), llame a `layoutIfNeeded()` en la vista más profunda que sea el elemento primario de la vista animada (en otras palabras, , que no se ve afectado por la animación). No sé exactamente por qué funciona esto.

Resolver el conflicto de prioridad de `UILabel`

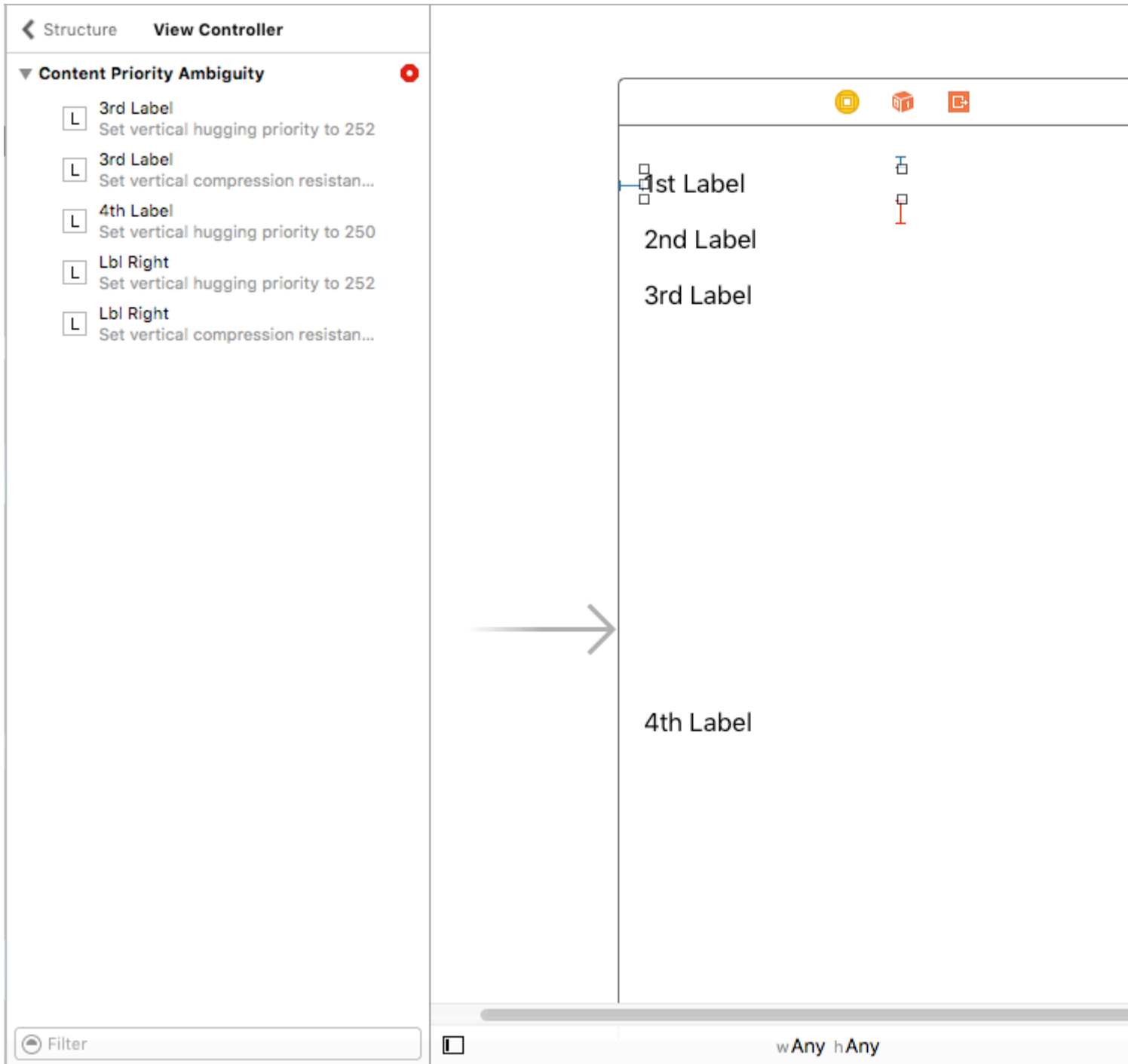
Problema : cuando usa muchas etiquetas dentro de una vista, quizás reciba una **advertencia** :

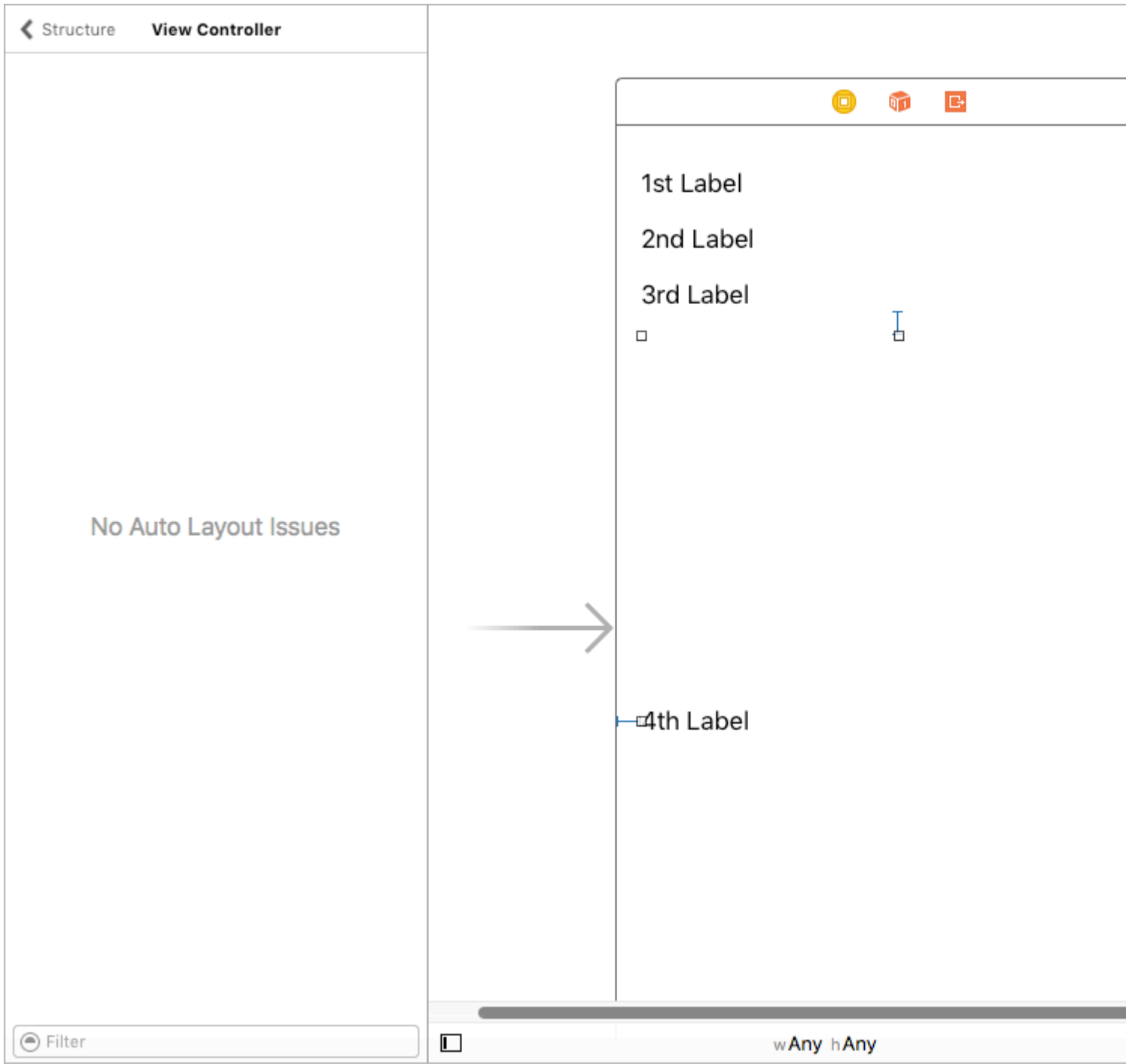


¿Cómo podemos arreglar esta **advertencia** ?

Solución : Calculamos y establecemos las prioridades en orden. Las prioridades deben ser diferentes de las etiquetas. Significa que lo que es importante tendrá mayor prioridad. Por ejemplo, en mi caso, establezco las prioridades verticales para que mis etiquetas se vean así:

Establecí la prioridad más alta para la primera etiqueta y la más baja para la cuarta etiqueta.





En un ViewController, creo que es difícil ver el efecto de esas prioridades. Sin embargo, está muy claro con la altura de celda estimada de UITableViewCell +.

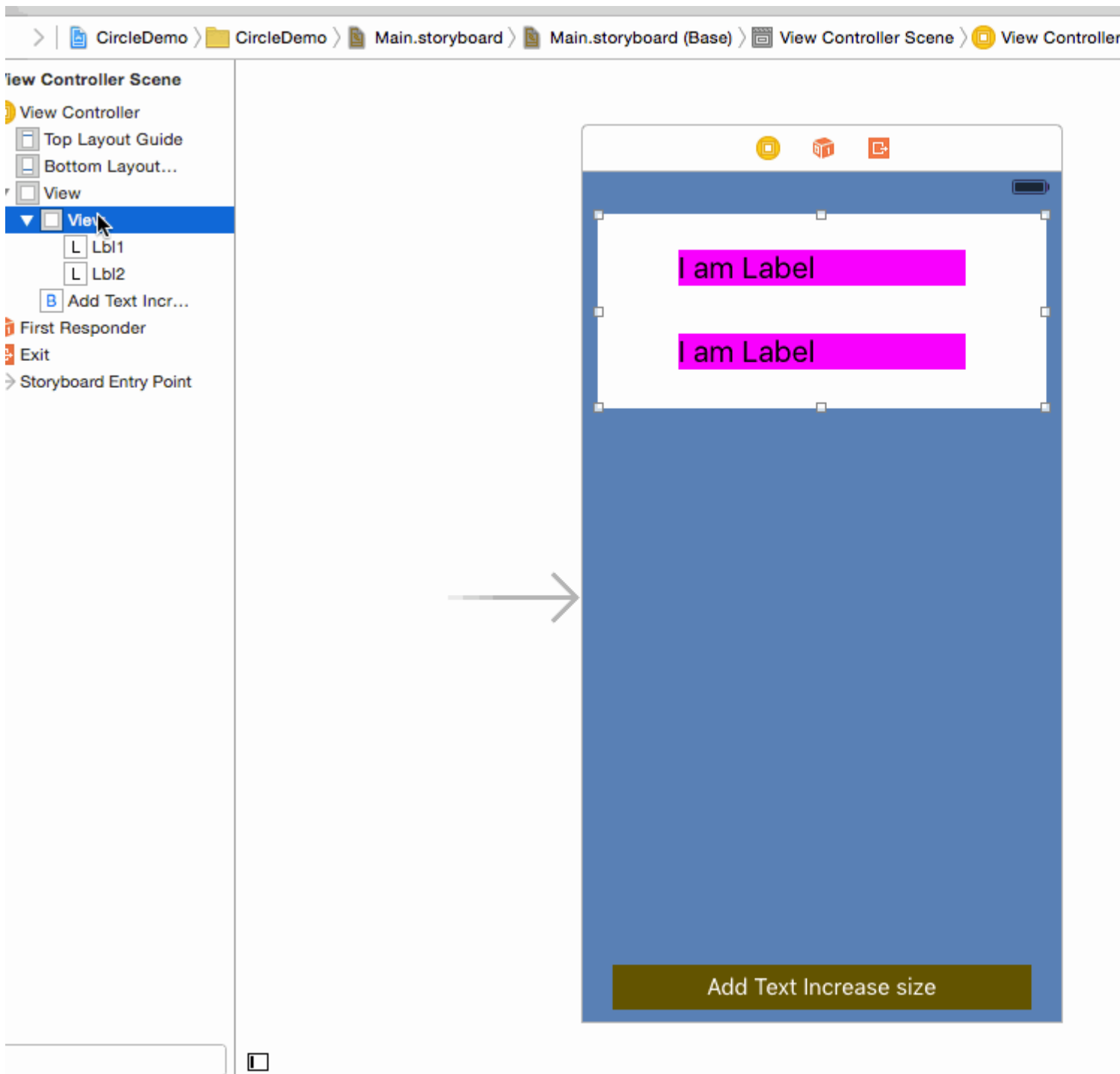
Espero que esto ayude.

Tamaño de UILabel y Parentview según el texto en UILabel

Guía paso por paso :-

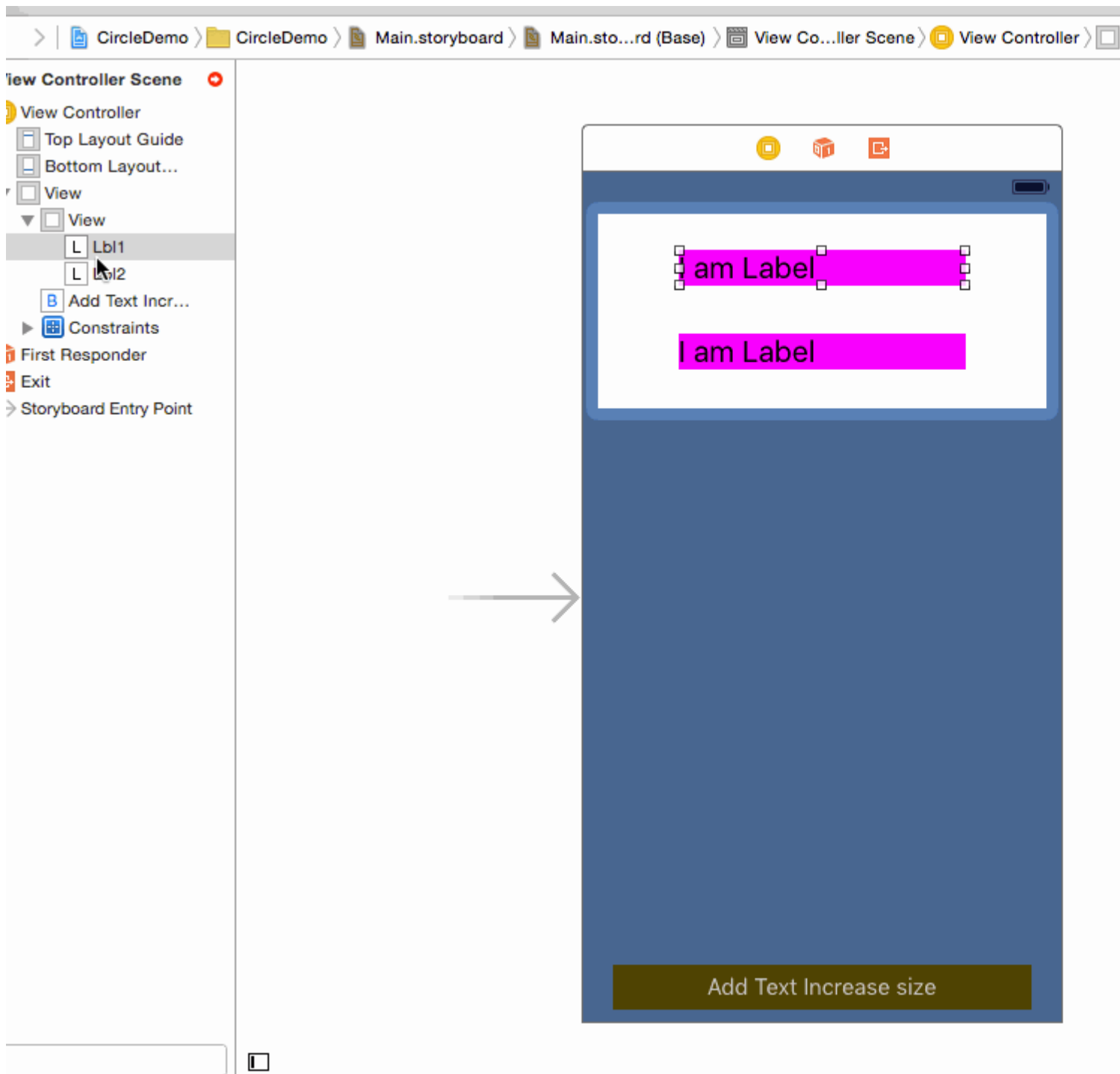
Paso 1: - Establecer restricción a UIView

1. Líder. 2) Arriba. 3) que se arrastra. (De mainview)



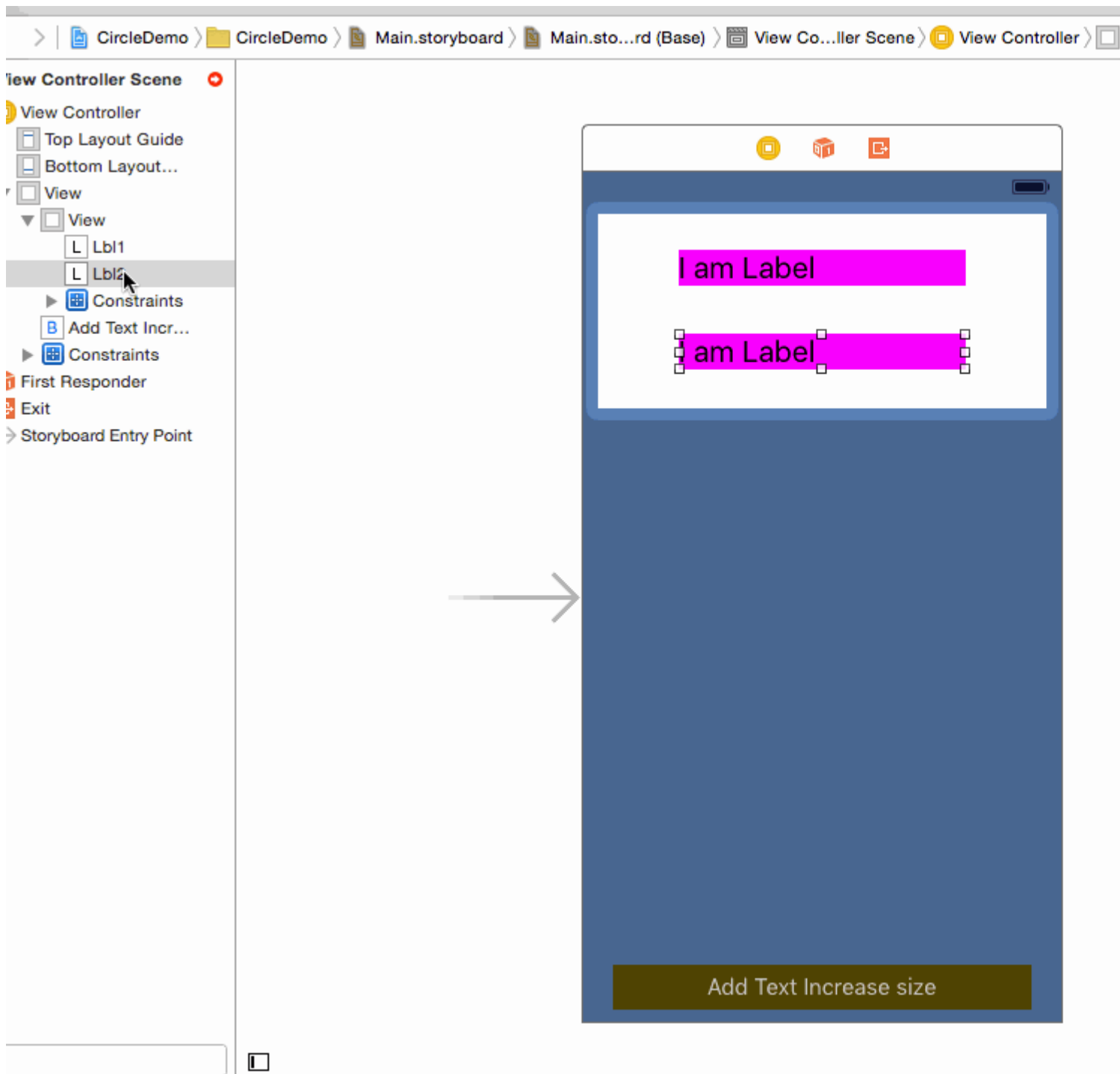
Paso 2: - Establecer restricción a la etiqueta 1

1. Liderando 2) Top 3) Arrastrándose (desde su supervisión)

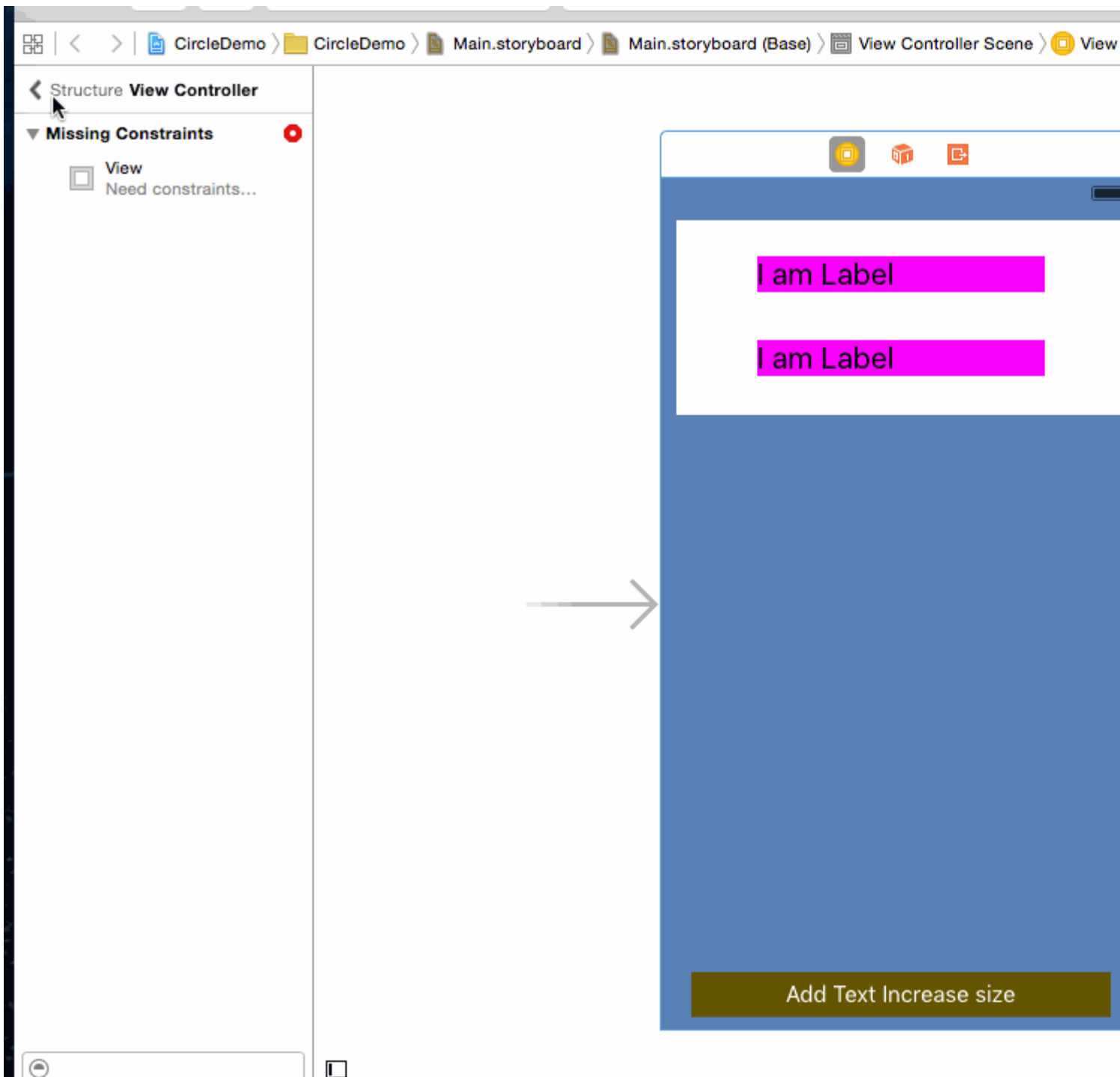


Paso 3: - Establecer restricción en Etiqueta 2

1. Liderando 2) Top 3) arrastrando (desde su supervisión)

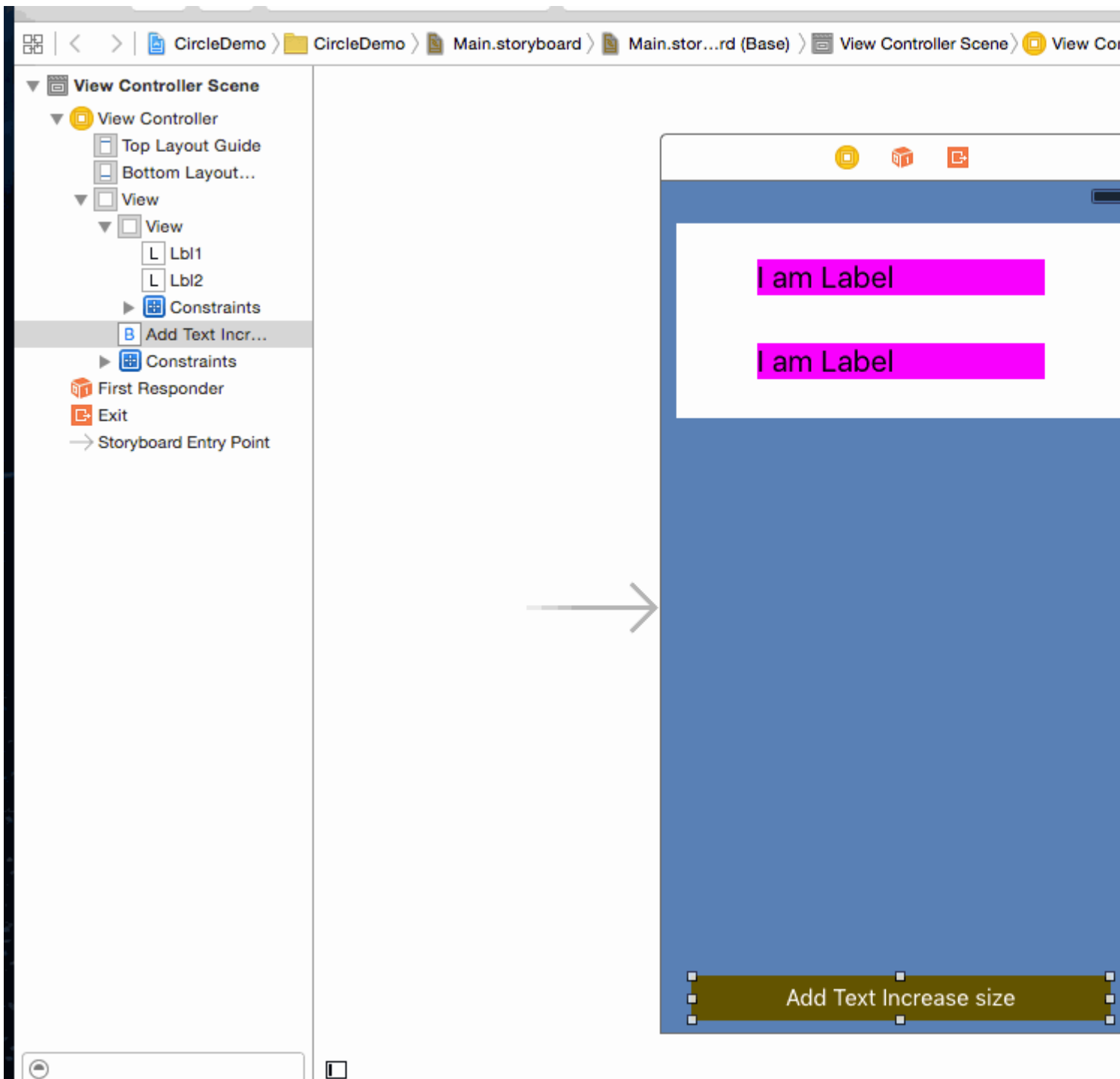


Paso 4: - Lo más complicado le da un fondo a UILabel desde UIView.



Paso 5: - (Opcional) Establecer restricción a UIButton

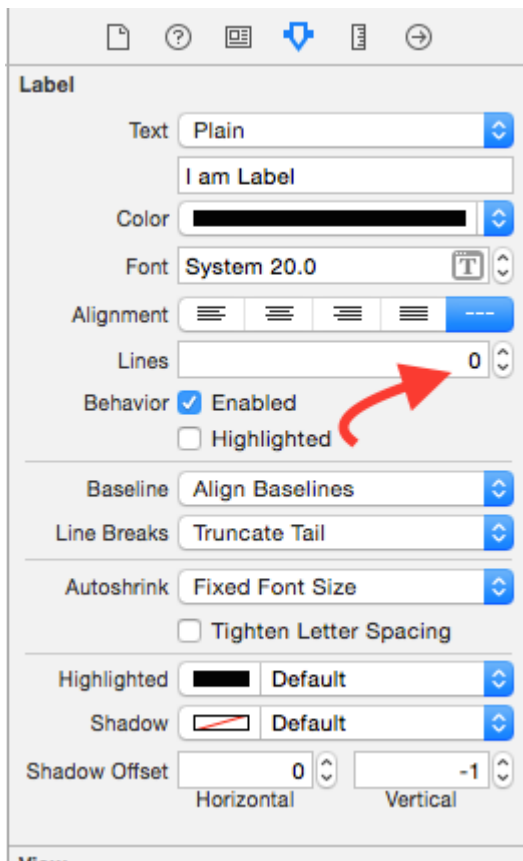
1. Líder
- 2) Parte inferior
- 3) Arrastre
- 4) Altura fija (desde la vista principal)



Salida: -



Nota: - Asegúrese de haber establecido el Número de líneas = 0 en la propiedad Etiqueta.



Espero que esta información sea suficiente para comprender Autoresize UIView según la altura de UILabel y Autoresize UILabel según texto.

Conceptos básicos del lenguaje de formato visual: restricciones en el código!

HVFL es un lenguaje diseñado para restringir los elementos de la interfaz de usuario de una manera simple y rápida. En general, VFL tiene una ventaja sobre la personalización de UI tradicional en el Interface Builder porque es mucho más legible, accesible y compacto.

Aquí hay un ejemplo de VFL, en el que tres UIViews están restringidas de izquierda a derecha, llenando `superView.width`, con `aGradeView`

```
"H:|[bgView][aGradeView(40)][bGradeView(40)]|"
```

Hay dos ejes en los que podemos restringir los objetos de la interfaz de usuario, horizontal y verticalmente.

Cada línea de VFL siempre comienza con `H:` o `V:`. Si ninguno de los dos está presente, la opción predeterminada es `H:`.

Continuando, tenemos una tubería. `|` Este símbolo, o la tubería, se refiere a la supervisión. Si observa más detenidamente el fragmento de código VFL que se encuentra arriba, observará dos de estas tuberías.

Esto significa los dos extremos horizontales de la supervisión, los límites externos y exteriores.

A continuación verá algunos corchetes, dentro del primer conjunto de corchetes, tenemos `bgView`.

Cuando tenemos corchetes, se refiere a un elemento de la interfaz de usuario, ahora puede preguntarse cómo establecemos un vínculo entre el nombre y el elemento de la interfaz de usuario real, ¿quizás una salida?

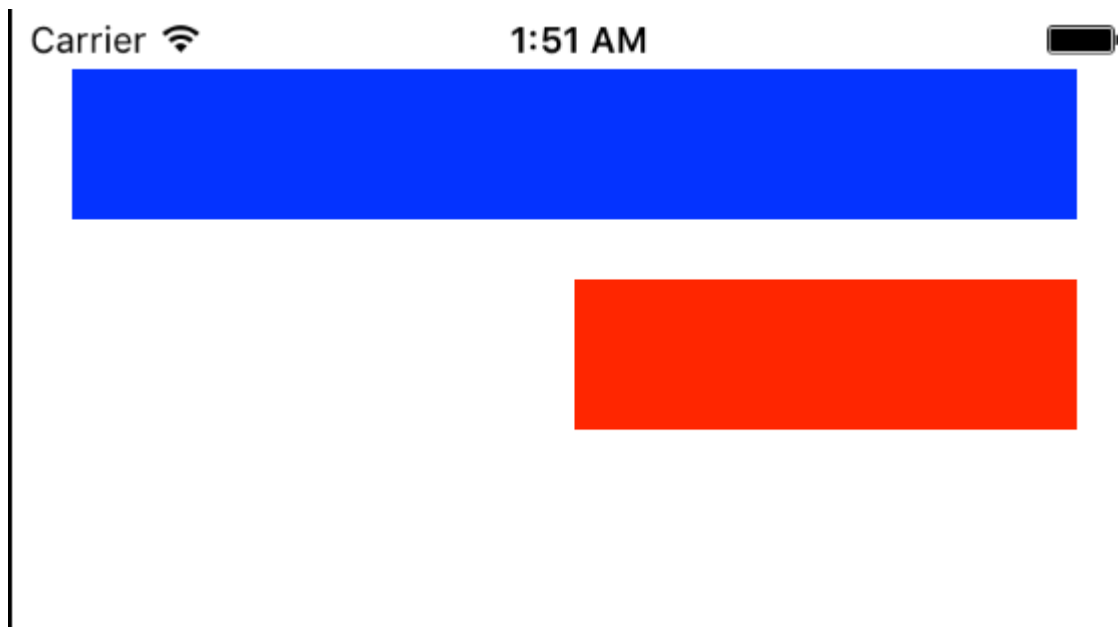
Cubriré eso al final del post.

Si observa el segundo par de corchetes `[aGradeView(50)]`, también tenemos algunos paréntesis encapsulados dentro, cuando está presente, define el ancho / alto según los ejes, que en este caso es 50 píxeles de ancho.

Los primeros corchetes `[bgView]` no tenían un ancho definido explícitamente, lo que significa que abarcaría todo lo posible.

Bien, eso es todo por lo básico, más sobre las cosas avanzadas en otro ejemplo.

por ejemplo:



```
// 1. create views
UIView *blueView = [[UIView alloc] init];
blueView.backgroundColor = [UIColor blueColor];
[self.view addSubview:blueView];

UIView *redView = [[UIView alloc] init];
redView.backgroundColor = [UIColor redColor];
[self.view addSubview:redView];

// 2. forbid Autoresizing
blueView.translatesAutoresizingMaskIntoConstraints = NO;
redView.translatesAutoresizingMaskIntoConstraints = NO;

// 3. make constraints
// horizontal
NSArray *blueH = [NSLayoutConstraint constraintsWithVisualFormat:@"H:|-20-[blueView]-20-|"
options:NSLayoutFormatAlignAllLeft metrics:nil views:@{@"blueView" : blueView}];
[self.view addConstraints:blueH];

// vertical
```

```

    NSArray *blueVandRedV = [NSLayoutConstraint constraintsWithVisualFormat:@"V:|-20-[blueView(50)]-20-[redView(==blueView)]" options:NSLayoutFormatAlignAllTrailing metrics:nil
    views:@{@"blueView" : blueView, @"redView" : redView}];
    [self.view addConstraints:blueVandRedV];

    NSLayoutConstraint *redW = [NSLayoutConstraint constraintWithItem:redView
    attribute:NSLayoutAttributeWidth relatedBy:NSLayoutRelationEqual toItem:blueView
    attribute:NSLayoutAttributeWidth multiplier:0.5 constant:0];
    [self.view addConstraint:redW];

```

Uso mixto de diseño automático con diseño no automático

En ocasiones, es posible que desee realizar algunas acciones adicionales para los cálculos de **Disposición automática** realizados por el propio `UIKit`.

Ejemplo: cuando tiene una `UIView` que tiene un `maskLayer`, es posible que necesite actualizar `maskLayer` tan pronto como **Auto Layout** cambie el `frame` `UIView`

```

// CustomView.m
- (void)layoutSubviews {
    [super layoutSubviews];
    // now you can assume Auto Layout did its job
    // you can use view's frame in your calculations
    CALayer maskLayer = self.maskLayer;
    maskLayer.bounds = self.bounds;
    ...
}

```

o si desea realizar alguna acción adicional al **diseño automático** en `ViewController`

```

- (void)viewDidLayoutSubviews {
    [super viewDidLayoutSubviews];
    // now you can assume all your subviews are positioned/resized correctly
    self.customView.frame = self.containerView.frame;
}

```

Diseño proporcional

Restricción creada como

```

NSLayoutConstraint(item: myView, attribute: NSLayoutConstraintAttribute.Leading, relatedBy:
NSLayoutConstraintRelation.Equal, toItem: view, attribute: NSLayoutConstraintAttribute.LeadingMargin, multiplier:
1.0, constant: 20.0)

```

o, desde el punto de vista matemático:

```

view.attribute * multiplier + constant (1)

```

Puede usar el multiplicador para crear un diseño proporcional para un factor de tamaño diferente.

Ejemplo:

La vista turquesa (V1) es un cuadrado con ancho de supervisión proporcional de ancho con relación 1: 1.1

Gary square (V2) es una subvista de V1. Espacio inferior establecido por constante = 60, Espacio al final establecido por multiplicador = 1.125 y constante = 0

Espacio al final establecido proporcionalmente, espacio inferior establecido como una constante.





- C objetivo

```
// 1. create views
UIView *blueView = [[UIView alloc] init];
blueView.backgroundColor = [UIColor blueColor];
[self.view addSubview:blueView];

UIView *redView = [[UIView alloc] init];
redView.backgroundColor = [UIColor redColor];
[self.view addSubview:redView];

// 2. forbid Autoresizing
blueView.translatesAutoresizingMaskIntoConstraints = NO;
redView.translatesAutoresizingMaskIntoConstraints = NO;

// 3. make constraints
// 3.1 blueView
NSLayoutConstraint *blueLeft = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeLeft relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeLeft multiplier:1 constant:20];
[self.view addConstraint:blueLeft];

NSLayoutConstraint *blueTop = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeTop relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeTop multiplier:1 constant:20];
[self.view addConstraint:blueTop];

NSLayoutConstraint *blueRight = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeRight relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeRight multiplier:1 constant:-20];
[self.view addConstraint:blueRight];

NSLayoutConstraint *blueHeight = [NSLayoutConstraint constraintWithItem:blueView
attribute:NSLayoutAttributeHeight relatedBy:NSLayoutRelationEqual toItem:nil
attribute:NSLayoutAttributeNotAnAttribute multiplier:1 constant:50];
[self.view addConstraint:blueHeight];

// 3.2 redView
NSLayoutConstraint *redTop = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeTop relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeBottom multiplier:1 constant:20];
[self.view addConstraint:redTop];
```

```
NSLayoutConstraint *redRight = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeRight relatedBy:NSLayoutRelationEqual toItem:self.view
attribute:NSLayoutAttributeRight multiplier:1 constant:-20];
[self.view addConstraint:redRight];

NSLayoutConstraint *redHeight = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeHeight relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeHeight multiplier:1 constant:0];
[self.view addConstraint:redHeight];

NSLayoutConstraint *redWidth = [NSLayoutConstraint constraintWithItem:redView
attribute:NSLayoutAttributeWidth relatedBy:NSLayoutRelationEqual toItem:blueView
attribute:NSLayoutAttributeWidth multiplier:0.5 constant:0];
[self.view addConstraint:redWidth];
```

Lea Diseño automático en línea: <https://riptutorial.com/es/ios/topic/792/disenio-automatico>

Capítulo 62: Enlace profundo en iOS

Observaciones

Útil [documentación de Apple](#) con ejemplos y aclaraciones.

Examples

Abrir una aplicación basada en su esquema de URL

Para abrir una aplicación con el esquema de URL definido `todolist://`:

C objetivo

```
NSURL *myURL = [NSURL URLWithString:@"todolist://there/is/something/to/do"];
[[UIApplication sharedApplication] openURL:myURL];
```

Rápido

```
let urlString = "todolist://there/is/something/to/do"
if let url = NSURL(string: urlString) {
    UIApplication.shared().openURL(url)
}
```

HTML

```
<a href="todolist://there/is/something/to/do">New SMS Message</a>
```

Nota: es útil verificar si se puede abrir el enlace para mostrar un mensaje apropiado para el usuario. Esto se puede hacer usando el método `canOpenURL:`

Agregando un esquema de URL a tu propia aplicación

Digamos que está trabajando en una aplicación llamada `MyTasks`, y desea permitir que las URL entrantes creen una nueva tarea con un título y un cuerpo. La URL que estás diseñando puede verse algo como esto:

```
mytasks://create?title=hello&body=world
```

(Por supuesto, los parámetros de `text` y `body` se utilizan para completar nuestra tarea que estamos creando.)

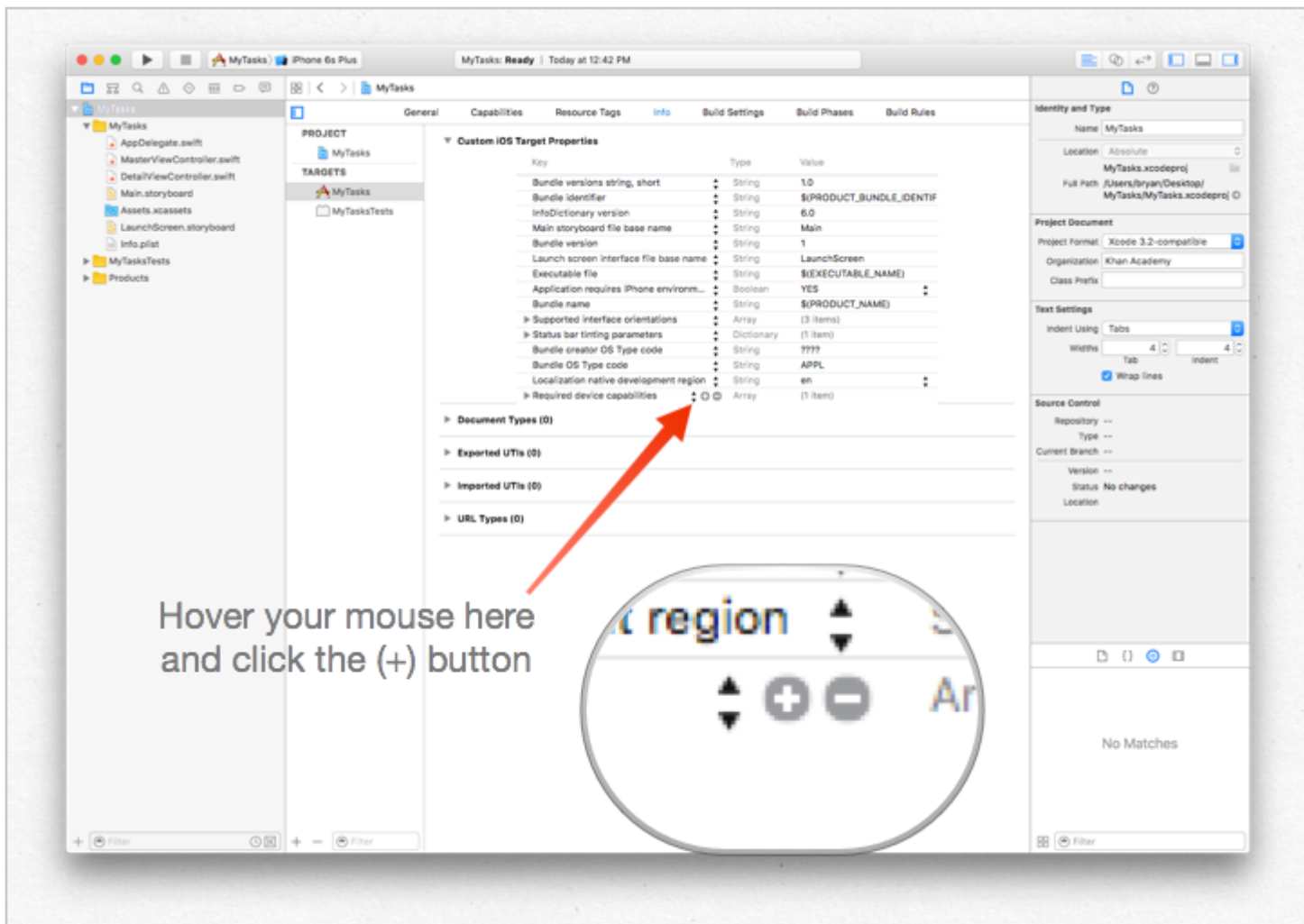
Aquí están los grandes pasos para agregar este esquema de URL a su proyecto:

1. Registre un esquema de URL en el archivo `Info.plist` su aplicación, para que el sistema sepa cuándo enrutar una URL a su aplicación.

2. Agregue una función a su `UIApplicationDelegate` que acepte y maneje las URL entrantes.
3. Realice cualquier tarea que deba ocurrir cuando se abra esa URL.

Paso uno: Registrar un esquema de URL en Info.plist:

Primero, debemos agregar una entrada de "Tipos de URL" a nuestro archivo Info.plist. Haga clic en el botón (+) aquí:



... luego ingrese un identificador único para su aplicación, así como el esquema de URL que desea usar. ¡Se específico! No quieres que el esquema de URL entre en conflicto con la implementación de otra aplicación. ¡Es mejor ser demasiado largo aquí que demasiado corto!

▼ URL types	▲	Array	(5 items)
▼ Item 0	▼	Dictionary	(2 items)
URL identifier	▲	String	com.mycompany
▼ URL Schemes	▲	Array	(1 item)
Item 0	⊕ ⊖	String	mytasks
▼ Item 1		Dictionary	(1 item)

Paso dos: Manejar la URL en

UIApplicationDelegate

Necesitamos implementar `application:openURL:options:` en nuestro `UIApplicationDelegate`. ¡Inspeccionaremos la URL entrante y veremos si hay alguna acción que podamos tomar!

Una implementación sería esta:

```
func application(app: UIApplication, openURL url: NSURL, options: [String : AnyObject]) -> Bool {
    if url.scheme == "mytasks" && url.host == "create" {
        let title = // get the title out of the URL's query using a method of your choice
        let body = // get the title out of the URL's query using a method of your choice
        self.rootViewController.createTaskWithTitle(title, body: body)
        return true
    }

    return false
}
```

Paso tres: realizar una tarea en función de la URL.

Cuando un usuario abre su aplicación a través de una URL, probablemente espera que suceda *algo*. Tal vez es navegar por una parte del contenido, tal vez es crear un nuevo elemento. En este ejemplo, vamos a crear una nueva tarea en la aplicación.

En el código anterior, podemos ver una llamada a

`self.rootViewController.createTaskWithTitle(:body:)` - por lo tanto, asumiendo que su `AppDelegate` tiene un puntero a su controlador de vista raíz que implementa la función correctamente, ¡está todo listo!

Configuración de Deeplink para su aplicación

La configuración de enlaces profundos para su aplicación es fácil. Solo necesita una url pequeña con la que desea abrir su aplicación.

Sigue los pasos para configurar la vinculación profunda para tu aplicación.

1. Permite crear un proyecto y llamarlo DeepLinkPOC.
2. Ahora selecciona el objetivo de tu proyecto.
3. Después de seleccionar el objetivo, seleccione la pestaña 'información'.
4. Desplácese hasta la parte inferior hasta que vea una opción de **Tipos de URL**
5. Haga clic en la opción '+

6. Verá que los **esquemas de URL** agregan una cadena mediante la cual desea abrir su aplicación. Permite agregar " **DeepLinking** " en los esquemas de URL.

Entonces, para abrir su aplicación, puede iniciarla escribiendo "**DeepLinking: //**" en su safari. Tu cadena de enlace profundo tiene el siguiente formato.

```
[scheme]://[host]/[path] --> DeepLinking://path/Page1
```

donde, Esquema: Host "DeepLinking": ruta "path": "Page1"

Nota : incluso si no agrega el host y la ruta, la aplicación se iniciará, así que no se preocupe. Pero puede agregar el host y la ruta para redirigir adicionalmente a una página en particular después del inicio de la aplicación.

7. Ahora agregue el siguiente método a su appdelegate.

Rápido:

```
func application(application: UIApplication, openURL url: NSURL, sourceApplication: String?, annotation: AnyObject) -> Bool
```

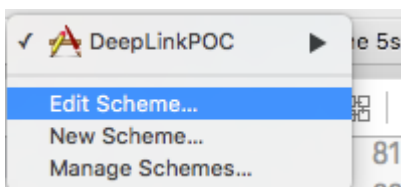
C objetivo:

```
-(BOOL)application:(UIApplication *)application  
    openURL:(NSURL *)url  
    sourceApplication:(NSString *)sourceApplication  
    annotation:(id)annotation
```

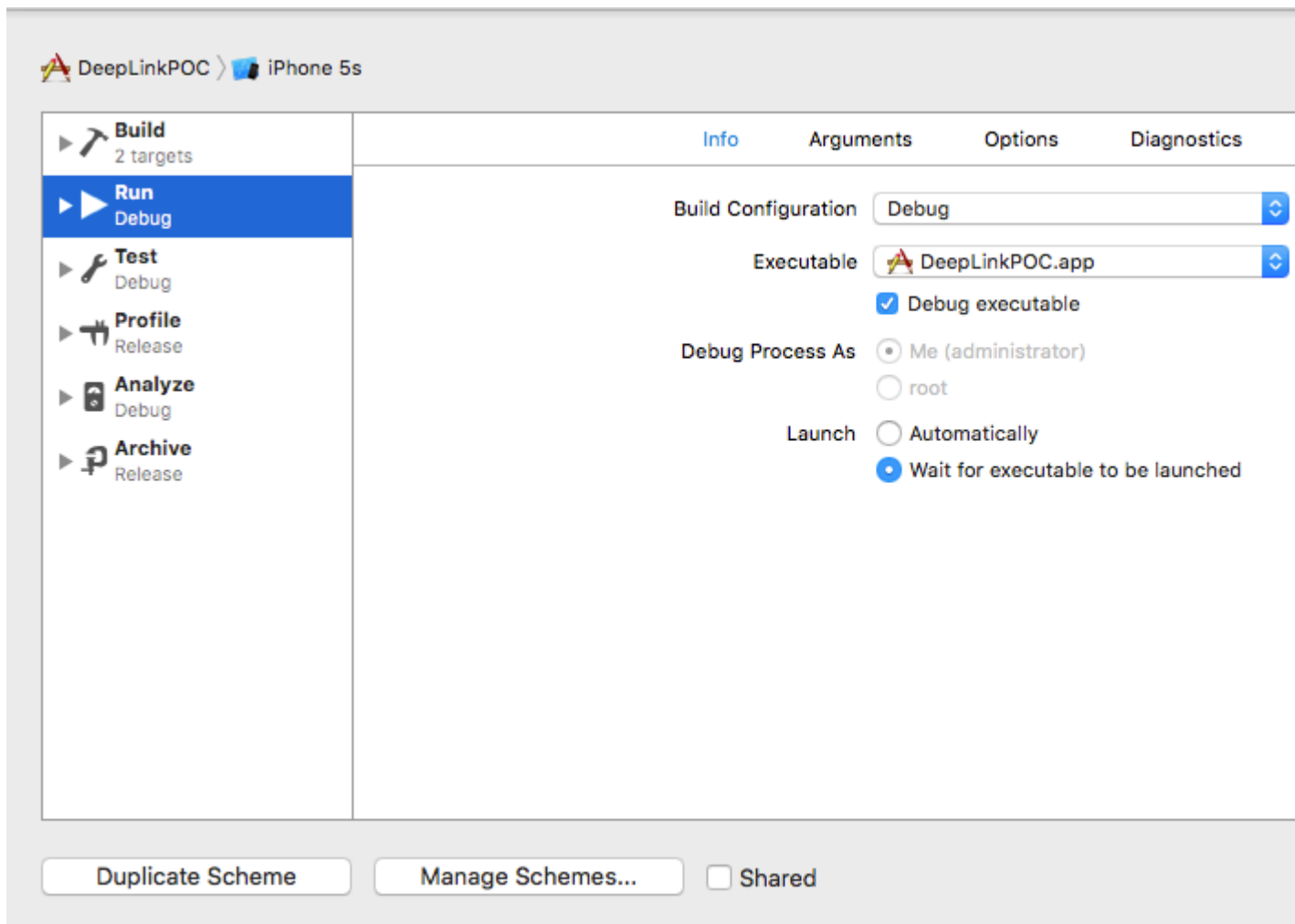
Se llama al método anterior siempre que su aplicación se inicie utilizando una cadena de enlace profundo que estableció para su aplicación.

8. Ahora es el momento de instalar su aplicación, pero espere a que salte directamente al botón de ejecución. Hagamos un pequeño cambio en el método de inicio de la aplicación del esquema.

- Seleccione y edite su esquema como



- Cambia su tipo de lanzamiento y cierra.



9. Ahora haga clic en el botón Ejecutar (si lo desea, puede agregar un punto de interrupción a sus métodos `didFinishLaunchingWithOptions` y `openURL` para observar los valores)
10. Verá un mensaje "Esperando que se inicie DeepLinkPOC (o el nombre de su aplicación)".
11. Abra safari y escriba " **DeepLinking: //** " en la barra de búsqueda. Esto le mostrará "abra esta página en DeepLinkPOC" haga clic en abrir para iniciar su aplicación.

Espero que tengas que saber cómo configurar enlaces profundos para tu aplicación :)

Lea Enlace profundo en iOS en línea: <https://riptutorial.com/es/ios/topic/5173/enlace-profundo-en-ios>

Capítulo 63: Enlaces universales

Observaciones

1. Cuando admite enlaces universales, los usuarios de iOS 9 pueden tocar un enlace a su sitio web y redirigirse sin problemas a su aplicación instalada sin pasar por Safari. Si su aplicación no está instalada, al tocar un enlace a su sitio web se abre su sitio web en Safari.
2. En general, cualquier enlace compatible que se `UIWebView` clic en Safari o en instancias de `UIWebView / WKWebView` debería abrir la aplicación.
3. Para iOS 9.2 y menos, esto solo funcionará en un dispositivo. iOS 9.3 también soporta el simulador.
4. iOS recuerda la elección del usuario al abrir Universal Links. Si tocan la ruta de navegación superior derecha para abrir el enlace en Safari, todos los demás clics los llevarán a Safari, no a la aplicación. Pueden volver a abrir la aplicación por defecto seleccionando Abrir en el banner de la aplicación en el sitio web.

Examples

Servidor de configuración

Necesitas tener un servidor corriendo en línea. Para asociar de forma segura su aplicación iOS con un servidor, Apple requiere que ponga a disposición un archivo de configuración, denominado `apple-app-site-association`. Este es un archivo `JSON` que describe el dominio y las rutas admitidas.

El archivo `apple-app-site-association` debe estar accesible a través de `HTTPS`, sin redirecciones, en `https:// {domain} / apple-app-site-association`.

El archivo se ve así:

```
{
  "applinks": {
    "apps": [ ],
    "details": [
      {
        "appID": "{app_prefix}.{app_identifier}",
        "paths": [ "/path/to/content", "/path/to/other/*", "NOT /path/to/exclude" ]
      },
      {
        "appID": "TeamID.BundleID2",
        "paths": [ "*" ]
      }
    ]
  }
}
```

NOTA : no agregue `.json` al nombre de archivo `apple-app-site-association`.

Las claves son las siguientes:

apps

: debe tener una matriz vacía como su valor y debe estar presente. Así es como Apple lo quiere.
`details` : es una variedad de diccionarios, uno para cada aplicación iOS compatible con el sitio web. Cada diccionario contiene información sobre la aplicación, el equipo y las identificaciones del paquete.

Hay 3 formas de definir caminos:

`Static` : toda la ruta admitida está codificada para identificar un enlace específico, por ejemplo, / static / términos

`Wildcards` : se puede usar A * para hacer coincidir las rutas dinámicas, por ejemplo, / books / * puede coincidir con la ruta a la página de cualquier autor. ? Dentro de los componentes del camino específico, por ejemplo, libros / 1? se puede utilizar para emparejar cualquier libro cuyo ID comience con 1.

`Exclusions` : al anular una ruta con NO se excluye esa ruta para que no coincida.

El orden en que se mencionan las rutas en la matriz es importante. Los índices anteriores tienen mayor prioridad. Una vez que una ruta coincide, la evaluación se detiene y otras rutas se ignoran. Cada camino distingue entre mayúsculas y minúsculas.

Código del sitio web

El código del sitio web se puede encontrar en la rama de gh-pages en <https://github.com/vineetchoudhary/iOS-Universal-Links/tree/gh-pages>

Apoyo a múltiples dominios

Cada dominio admitido en la aplicación debe tener disponible su propio archivo de asociación de sitio de aplicación de Apple. Si el contenido servido por cada dominio es diferente, entonces el contenido del archivo también cambiará para admitir las rutas respectivas. De lo contrario, se puede utilizar el mismo archivo, pero debe ser accesible en todos los dominios compatibles.

Firma del archivo App-Site-Association

Nota : puede omitir esta parte si su servidor utiliza `HTTPS` para servir contenido y pasar a la guía de configuración de la aplicación.

Si su aplicación apunta a iOS 9 y su servidor utiliza `HTTPS` para servir contenido, no necesita firmar el archivo. Si no (por ejemplo, cuando se admite Handoff en iOS 8), debe firmarse con un certificado `SSL` de una autoridad de certificados reconocida.

Nota : este no es el certificado proporcionado por Apple para enviar su aplicación a la App Store. Debe ser proporcionado por un tercero, y se recomienda utilizar el mismo certificado que usa para su servidor `HTTPS` (aunque no es obligatorio).

Para firmar el archivo, primero cree y guarde una versión simple de `.txt`. A continuación, en el terminal, ejecute el siguiente comando:

```
cat <unsigned_file>.txt | openssl smime -sign -inkey example.com.key -signer example.com.pem -
```

```
certfile intermediate.pem -noattr -nodetach -outform DER > apple-app-site-association
```

Esto dará salida al archivo firmado en el directorio actual. Los archivos `example.com.key` , `example.com.pem` , y `intermediate.pem` son los archivos que su Autoridad de Certificación pondría a su disposición.

Nota : Si el archivo no está firmado, debe tener un `Content-Type` de `application/json` . De lo contrario, debería ser `application/pkcs7-mime` .

Valide su servidor con la herramienta de validación de búsqueda de aplicaciones de Apple

Pruebe su página web para iOS 9 API de búsqueda. Ingrese una URL y Applebot rastreará su página web y le mostrará cómo puede optimizar para obtener los mejores resultados

<https://search.developer.apple.com/appsearch-validation-tool/>

Instalar la aplicación iOS (habilitando los enlaces universales)

La configuración en el lado de la aplicación requiere dos cosas:

1. Configuración de los derechos de la aplicación y habilitación de los enlaces universales activando la capacidad de Dominios asociados en el proyecto.
2. Manejo de enlaces entrantes en su `AppDelegate` .

1. Configuración de los derechos de la aplicación y habilitación de enlaces universales.

El primer paso para configurar los derechos de su aplicación es habilitarla para su ID de aplicación. Haz esto en el Centro de Miembros Desarrolladores de Apple. Haga clic en Certificados, Identificadores y Perfiles y luego en Identificadores. Seleccione su ID de aplicación (créela primero si es necesario), haga clic en Editar y habilite el derecho de Dominios asociados.

ID: com.Universal-Links		
Application Services:		
Service	Development	Distribution
App Group	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Associated Domains	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
Data Protection	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Game Center	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
HealthKit	<input type="radio"/> Disabled	<input type="radio"/> Disabled
HomeKit	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Wireless Accessory Configuration	<input type="radio"/> Disabled	<input type="radio"/> Disabled
iCloud	<input type="radio"/> Disabled	<input type="radio"/> Disabled
In-App Purchase	<input checked="" type="radio"/> Enabled	<input checked="" type="radio"/> Enabled
Inter-App Audio	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Apple Pay	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Wallet	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Push Notifications	<input type="radio"/> Disabled	<input type="radio"/> Disabled
Personal VPN	<input type="radio"/> Disabled	<input type="radio"/> Disabled

A continuación, obtenga el prefijo y el sufijo de la ID de la aplicación haciendo clic en la ID de la aplicación correspondiente.

El prefijo y el sufijo de la ID de la aplicación deben coincidir con el del archivo de asociación-sitio-asociación-apple.

A continuación en `Xcode`, seleccione el destino de su aplicación, haga clic en Capacidades y active Dominios asociados para Activar. Agregue una entrada para cada dominio que admita su aplicación, con el prefijo de los **enlaces de la aplicación**:

Por ejemplo, **aplinks: YourCustomDomainName.com**

Que se ve así para la aplicación de muestra:

General
Capabilities
Resource Tags
Info

PROJECT

- Universal Links

TARGETS

- Universal Links

- ▶ **Apple Pay**

- ▶ **In-App Purchase**

- ▶ **Personal VPN**

- ▶ **Maps**

- ▶ **Keychain Sharing**

- ▶ **Background Modes**

- ▶ **Inter-App Audio**

- ▼ **Associated Domains**

Domains:

+ -

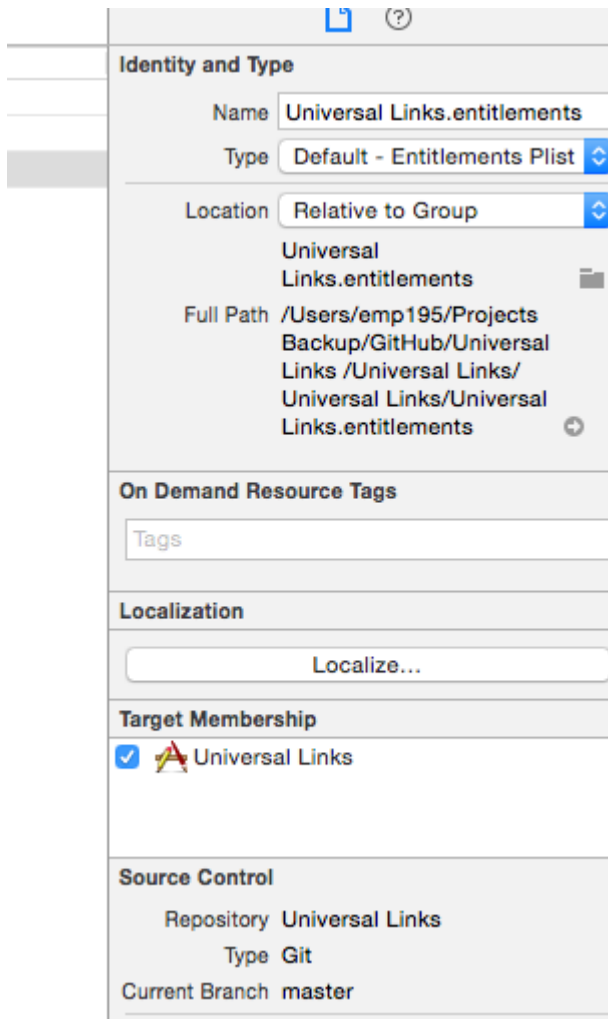
Steps: Add the "Associated Domain

Add the "Associated Domain

- ▶ **App Groups**

- ▶ **Data Protection**

Nota : Asegúrese de haber seleccionado el mismo equipo e ingresado la misma ID de paquete que la ID de aplicación registrada en el Centro de miembros. También asegúrese de que Xcode incluya el archivo de derechos al seleccionar el archivo y, en el inspector de archivos, asegúrese de que su destino esté marcado.



2. Manejo de enlaces entrantes en su AppDelegate

Todos los redireccionamientos desde Safari a la aplicación para enlaces universales se realizan a través del método a continuación en la clase AppDelegate de la aplicación. Analiza esta URL para determinar la acción correcta en la aplicación.

```
[UIApplicationDelegate application: continueUserActivity: restorationHandler:]
```

C objetivo

```
-(BOOL)application:(UIApplication *)application continueUserActivity:(NSUserActivity *)userActivity restorationHandler:(void (^)(NSArray * _Nullable))restorationHandler{  
    ///Checking whether the activity was from a web page redirect to the app.  
    if ([userActivity.activityType isEqualToString: NSUserActivityTypeBrowsingWeb]) {  
        ///Getting the URL from the UserActivity Object.  
        NSURL *url = userActivity.webpageURL;  
        UIStoryboard *storyBoard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];  
        UINavigationController *navigationController = (UINavigationController *)_window.rootViewController;  
        if ([url.pathComponents containsObject:@"home"]) {  
            [navigationController pushViewController:[storyBoard instantiateViewControllerWithIdentifier:@"HomeScreenId"] animated:YES];  
        }else if ([url.pathComponents containsObject:@"about"]){  
            [navigationController pushViewController:[storyBoard
```

```
instantiateViewControllerWithIdentifier:@"AboutScreenId"] animated:YES];  
    }  
}  
return YES;  
}
```

Swift:

```
func application(application: UIApplication, continueUserActivity userActivity:  
NSUserActivity, restorationHandler: ([AnyObject]? -> Void) -> Bool {  
    if userActivity.activityType == NSUserActivityTypeBrowsingWeb {  
        let url = userActivity.webpageURL!  
        //handle url  
    }  
    return true  
}
```

Código de aplicación de iOS

El código de la aplicación se puede encontrar la rama principal [aquí](#) .

Lea Enlaces universales en línea: <https://riptutorial.com/es/ios/topic/2362/enlaces-universales>

Capítulo 64: Entrega por paracaídas

Examples

Entrega por paracaídas

C objetivo

Airdrop se puede utilizar desde `UIActivityViewController`. La clase `UIActivityViewController` es un controlador de vista estándar que proporciona varios servicios estándar, como copiar elementos al portapapeles, compartir contenido a sitios de redes sociales, enviar elementos a través de Mensajes, AirDrop y algunas aplicaciones de terceros.

En este caso estaríamos enviando una imagen a través de `UIActivityViewController`

```
UIImage *hatImage = [UIImage imageNamed:@"logo.png"];
if (hatImage)//checks if the image file is not nil
{
//Initialise a UIActivityViewController
UIActivityViewController *controller = [[UIActivityViewController alloc]
initWithActivityItems:[hatImage] applicationActivities:nil];
//Excludes following options from the UIActivityViewController menu
NSArray *excludeActivities = @[UIActivityTypePostToWeibo,UIActivityTypePrint,
UIActivityTypeMail,UIActivityTypeMessage,UIActivityTypePostToTwitter,UIActivityTypePostToFacebook,
UIActivityTypeCopyToPasteboard,UIActivityTypeAssignToContact,
UIActivityTypeSaveToCameraRoll,UIActivityTypeAddToReadingList,
UIActivityTypePostToFlickr,UIActivityTypePostToVimeo,
UIActivityTypePostToTencentWeibo];
controller.excludedActivityTypes = excludeActivities;
[self presentViewController:controller animated:YES completion:nil];
}
```

Rápido

```
if ((newImage) != nil)
{
let activityVC = UIActivityViewController(activityItems: [newImage],
applicationActivities: nil)
activityVC.excludedActivityTypes =[UIActivityTypeAddToReadingList]
self.presentViewController(activityVC, animated: true, completion: nil)
}
```

Lea Entrega por paracaídas en línea: <https://riptutorial.com/es/ios/topic/7360/entrega-por-paracaidas>

Capítulo 65: Escáner de códigos QR

Introducción

Los códigos QR (respuesta rápida) son códigos de barras bidimensionales que se utilizan ampliamente en etiquetas ópticas legibles por máquina. iOS proporciona una forma de leer los códigos QR utilizando el marco `AVFoundation` desde iOS 7 en adelante. Este marco proporciona un conjunto de API para configurar / abrir la cámara y leer códigos QR de la alimentación de la cámara.

Examples

UIViewController busca QR y muestra la entrada de video

```
import AVFoundation
class QRScannerViewController: UIViewController,
    AVCaptureMetadataOutputObjectsDelegate {

    func viewDidLoad() {
        self.initCaptureSession()
    }

    private func initCaptureSession() {
        let captureDevice = AVCaptureDevice
            .defaultDevice(withMediaType: AVMediaTypeVideo)
        do {
            let input = try AVCaptureDeviceInput(device: captureDevice)
            let captureMetadataOutput = AVCaptureMetadataOutput()
            self.captureSession?.addOutput(captureMetadataOutput)
            captureMetadataOutput.setMetadataObjectsDelegate(self,
                queue: DispatchQueue.main)
            captureMetadataOutput
                .metadataObjectTypes = [AVMetadataObjectTypeQRCode]

            self.videoPreviewLayer =
                AVCaptureVideoPreviewLayer(session: self.captureSession)
            self.videoPreviewLayer?
                .videoGravity = AVLayerVideoGravityResizeAspectFill
            self.videoPreviewLayer?.frame =
                self.view.layer.bounds

            self._viewController?.view.layer
                .addSublayer(videoPreviewLayer!)
            self.captureSession?.startRunning()
        } catch {
            //TODO: handle input open error
        }
    }

    private func dismissCaptureSession() {
        if let running = self.captureSession?.isRunning, running {
            self.captureSession?.stopRunning()
        }
        self.captureSession = nil
    }
}
```

```

        self.videoPreviewLayer?.removeFromSuperLayer()
        self.videoPreviewLayer = nil
    }

    func captureOutput(_ captureOutput: AVCaptureOutput,
                      didOutputMetadataObjects metadataObjects: [Any]!,
                      from connection: AVCaptureConnection) {
        guard metadataObjects != nil && metadataObjects.count != 0 else {
            //Nothing captured
            return
        }

        if let metadataObj =
            metadataObjects[0] as? AVMetadataMachineReadableCodeObject {
            guard metadataObj.type == AVMetadataObjectTypeQRCode else {
                return
            }

            let barCodeObject = videoPreviewLayer?
                .transformedMetadataObject(for:
                    metadataObj as AVMetadataMachineReadableCodeObject)
                as! AVMetadataMachineReadableCodeObject

            if let qrValue = metadataObj.stringValue {
                self.handleQRRead(value: qrValue)
            }
        }
    }

    private handleQRRead(value: String) {
        //TODO: Handle the read qr
    }
    private captureSession: AVCaptureSession?
    private videoPreviewLayer: AVCaptureVideo
}

```

handleQRRead : se handleQRRead en un escaneo exitoso initCaptureSession - inicialice el escaneo para QR y la entrada de la cámara dismissCaptureSession - oculte la entrada de la cámara y detenga el escaneo

Escanear el código QR con el marco de AVFoudation

Antes de iOS 7, cuando desee escanear un código QR, es posible que tengamos que confiar en marcos de terceros o bibliotecas como [zBar](#) o [zXing](#) . Pero Apple introdujo AVCaptureMetaDataOutput de iOS 7 para leer códigos de barras.

Para leer el código QR usando AVFoundation necesitamos configurar / crear AVCaptureSession y usar captureOutput:didOutputMetadataObjects:fromConnection: delegate method.

Paso 1

Importe el marco de AVFoundation y confirme al protocolo AVCaptureMetadataOutputObjectsDelegate

```
import AVFoundation
```

```
class ViewController: UIViewController, AVCaptureMetadataOutputObjectsDelegate
```

Paso 2

La lectura de códigos QR está totalmente basada en la captura de video. Entonces, para capturar video continuo, cree una `AVCaptureSession` y configure la entrada y salida del dispositivo. Agregue el siguiente código en el controlador `viewDidLoad` controlador de vista

```
// Create an instance of the AVCaptureDevice and provide the video as the media type parameter.
let captureDevice = AVCaptureDevice.defaultDevice(withMediaType: AVMediaTypeVideo)

do {
    // Create an instance of the AVCaptureDeviceInput class using the device object and initialise capture session
    let input = try AVCaptureDeviceInput(device: captureDevice)
    captureSession = AVCaptureSession()
    captureSession?.addInput(input)

    // Create a instance of AVCaptureMetadataOutput object and set it as the output device the capture session.
    let captureMetadataOutput = AVCaptureMetadataOutput()
    captureSession?.addOutput(captureMetadataOutput)
    // Set delegate with a default dispatch queue
    captureMetadataOutput.setMetadataObjectsDelegate(self, queue: DispatchQueue.main)
    //set meta data object type as QR code, here we can add more then one type as well
    captureMetadataOutput.metadataObjectTypes = [AVMetadataObjectTypeQRCode]

    // Initialize the video preview layer and add it as a sublayer to the viewcontroller view's layer.
    videoPreviewLayer = AVCaptureVideoPreviewLayer(session: captureSession)
    videoPreviewLayer?.videoGravity = AVLayerVideoGravityResizeAspectFill
    videoPreviewLayer?.frame = view.layer.bounds
    view.layer.addSublayer(videoPreviewLayer!)

    // Start capture session.
    captureSession?.startRunning()
} catch {
    // If any error occurs, let the user know. For the example purpose just print out the error
    print(error)
    return
}
```

Paso 3

Implementar el método de delegado `AVCaptureMetadataOutputObjectsDelegate` para leer el código QR

```
func captureOutput(_ captureOutput: AVCaptureOutput!, didOutputMetadataObjects metadataObjects: [Any]!, from connection: AVCaptureConnection!) {
```

```

// Check if the metadataObjects array contains at least one object. If not no QR code is
in our video capture
if metadataObjects == nil || metadataObjects.count == 0 {
    // NO QR code is being detected.
    return
}

// Get the metadata object and cast it to `AVMetadataMachineReadableCodeObject`
let metadataObj = metadataObjects[0] as! AVMetadataMachineReadableCodeObject

if metadataObj.type == AVMetadataObjectTypeQRCode {
    // If the found metadata is equal to the QR code metadata then get the string value
from meta data
    let barCodeObject = videoPreviewLayer?.transformedMetadataObject(for: metadataObj)

    if metadataObj.stringValue != nil {
        // metadataObj.stringValue is our QR code
    }
}
}
}

```

aquí, el objeto de metadatos también puede proporcionarle los límites del código QR leído en la fuente de la cámara. Para obtener los límites, simplemente pase el objeto de metadatos al método `transformedMetadataObject` `videoPreviewLayer` como se muestra a continuación.

```

let barCodeObject = videoPreviewLayer?.transformedMetadataObject(for: metadataObj)
qrCodeFrameView?.frame = barCodeObject!.bounds

```

Lea Escáner de códigos QR en línea: <https://riptutorial.com/es/ios/topic/7963/escaner-de-codigos-qr>

Capítulo 66: Establecer fondo de vista

Examples

Establecer fondo de vista

C objetivo:

```
view.backgroundColor = [UIColor redColor];
```

Rápido:

```
view.backgroundColor! = UIColor.redColor()
```

Swift 3

```
view.backgroundColor = UIColor.redColor
```

Rellene la imagen de fondo de una vista

C objetivo

```
UIGraphicsBeginImageContext(self.view.frame.size);
[[UIImage imageNamed:@"image.png"] drawInRect:self.view.bounds];
UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
UIGraphicsEndImageContext();
self.view.backgroundColor = [UIColor colorWithPatternImage:image];
```

Establecer vista de fondo con imagen

```
self.view.backgroundColor = [UIColor colorWithPatternImage:[UIImage
imageNamed:@"Background.png"]];
```

Creación de una vista de fondo degradado

Para crear un fondo con un degradado puede usar la clase [CAGradientLayer](#) :

Swift 3.1:

```
func createGradient() {
    let caLayer = CAGradientLayer()
    caLayer.colors = [UIColor.white, UIColor.green, UIColor.blue]
    caLayer.locations = [0, 0.5, 1]
    caLayer.bounds = self.bounds
    self.layer.addSublayer(caLayer)
}
```

Esto se puede llamar en viewDidLoad () así:

```
override func viewDidLoad() {  
    super.viewDidLoad()  
    createGradient()  
}
```

Las variables de ubicación y límites de CAGradientLayer pueden tomar varios valores para crear una capa de degradado con la cantidad de colores que desee. De la documentación:

De manera predeterminada, los colores se distribuyen uniformemente en la capa, pero opcionalmente puede especificar ubicaciones para controlar las posiciones de color a través del degradado.

Lea **Establecer fondo de vista en línea**: <https://riptutorial.com/es/ios/topic/6854/establecer-fondo-de-vista>

Capítulo 67: EventKit

Examples

Solicitando Permiso

Su aplicación no puede acceder a sus recordatorios y su calendario sin permiso. En su lugar, debe mostrar una alerta al usuario, solicitándole que otorgue acceso a eventos para la aplicación.

Para comenzar, importe el framework `EventKit` :

Rápido

```
import EventKit
```

C objetivo

```
#import <EventKit/EventKit.h>
```

Haciendo un `EKEventStore`

Entonces, hacemos un objeto `EKEventStore` . Este es el objeto desde el que podemos acceder a los datos de calendario y recordatorios:

Rápido

```
let eventStore = EKEventStore()
```

C objetivo

```
EKEventStore *eventStore = [[EKEventStore alloc] init];
```

Nota

Hacer un objeto `EKEventStore` cada vez que necesitamos acceder al calendario no es eficiente. Intenta hacerlo una vez y utilízalo en cualquier parte de tu código.

Comprobando disponibilidad

La disponibilidad tiene tres estados diferentes: Autorizado, Denegado y No determinado. No determinado significa que la aplicación debe otorgar acceso.

Para verificar la disponibilidad, usamos el método `EKEventStore authorizationStatusForEntityType()` del objeto `EKEventStore` :

Rápido

```
switch EKEventStore.authorizationStatusForEntityType(EKEntityTypeEvent) {
    case .Authorized: //...
    case .Denied: //...
    case .NotDetermined: //...
    default: break
}
```

C objetivo

```
switch ([EKEventStore authorizationStatusForEntityType:EKEntityTypeEvent]){
    case EKAuthorizationStatus.Authorized:
        //...
        break;
    case EKAuthorizationStatus.Denied:
        //...
        break;
    case EKAuthorizationStatus.NotDetermined:
        //...
        break;
    default:
        break;
}
```

Solicitando Permiso

Ponga el siguiente código en el caso `NotDetermined` :

Rápido

```
eventStore.requestAccessToEntityType(EKEntityTypeEvent, completion: { [weak self]
    (userGrantedAccess, _) -> Void in
    if userGrantedAccess{
        //access calendar
    }
})
```

Accediendo a diferentes tipos de calendarios.

Accediendo a la gama de calendarios.

Para acceder a la matriz de `EKCalendar`s, usamos el método `calendarsForEntityType` :

Rápido

```
let calendarsArray = eventStore.calendarsForEntityType(EKEntityType.Event) as! [EKCalendar]
```

Iterando a través de calendarios

Solo usa un simple bucle `for` :

Rápido

```
for calendar in calendarsArray{  
    //...  
}
```

Accediendo al título y color del calendario.

Rápido

```
let calendarColor = UIColor(CGColor: calendar.CGColor)  
let calendarTitle = calendar.title
```

C objetivo

```
UIColor *calendarColor = [UIColor initWithCGColor: calendar.CGColor];  
NSString *calendarTitle = calendar.title;
```

Añadiendo un evento

Creando el objeto evento

Rápido

```
var event = EKEvent(eventStore: eventStore)
```

C objetivo

```
EKEvent *event = [EKEvent initWithEventStore:eventStore];
```

Configuración de calendario relacionado, título y fechas

Rápido

```
event.calendar = calendar
event.title = "Event Title"
event.startDate = startDate //assuming startDate is a valid NSDate object
event.endDate = endDate //assuming endDate is a valid NSDate object
```

Añadiendo evento al calendario

Rápido

```
try {
    do eventStore.saveEvent(event, span: EKSpan.ThisEvent)
} catch let error as NSError {
    //error
}
```

C objetivo

```
NSError *error;
BOOL *result = [eventStore saveEvent:event span:EKSpanThisEvent error:&error];
if (result == NO){
    //error
}
```

Lea EventKit en línea: <https://riptutorial.com/es/ios/topic/5854/eventkit>

Capítulo 68: Extensión para notificaciones push enriquecidas - iOS 10.

Introducción

iOS 10 nos dio `UserNotifications.framework`, la nueva API para notificaciones locales / remotas. Ofrece la visualización de archivos adjuntos multimedia o la respuesta a mensajes directamente desde la notificación.

El contenido de la notificación consiste en: título, subtítulo, cuerpo y adjunto. El archivo adjunto puede contener imágenes / gifs / videos de hasta 50 mb.

Examples

Extensión de contenido de notificación

¿Por qué lo necesitamos?

La extensión de contenido nos ayuda a crear una interfaz de usuario personalizada al momento de la notificación.

Utilice este marco para definir una extensión que recibe los datos de notificación y proporciona la representación visual correspondiente. Su extensión también puede responder a acciones personalizadas asociadas con esas notificaciones.

Implementación

1. En la ventana de xCode `Navigator` vaya a la sección de `Targets`. Presione `Add New Target`.
2. Seleccione la plantilla de `Notification Content Extension`:

Choose a template for your new target:

iOS watchOS tvOS macOS Cross-Platform

Call Directory Extension

Content Blocker Extension

iMessage

Intents Extension

Notification Service Extension

Photo Editing Extension

Spotlight Index

Sticker Pack

Cancel

UNNotificationExtensionCategory :

▼ NSExtension

▼ NSExtensionAttributes

UNNotificationExtensionDefaultContentHidden

UNNotificationExtensionCategory

UNNotificationExtensionInitialContentSizeRatio

NSExtensionMainStoryboard

NSExtensionPointIdentifier

Atributos de NSExtension :

UNNotificationExtensionCategory (Requerido)

El valor de esta clave es una cadena o una matriz de cadenas. Cada cadena contiene el identificador de una categoría declarada por la aplicación que utiliza la clase de Categoría UNNotification.

UNNotificationExtensionInitialContentSizeRatio (Requerido)

Número que representa el tamaño inicial de la vista de su controlador de vista expresado como una relación de su altura a su ancho.

UNNotificationExtensionDefaultContentHidden (opcional)

Cuando se establece en SÍ, el sistema muestra solo su controlador de vista personalizado en la interfaz de notificación. Cuando se establece en NO, el sistema muestra el contenido de notificación predeterminado además del contenido del controlador de vista.

UNNotificationExtensionOverridesDefaultTitle (opcional)

El valor de esta clave es un booleano. Cuando se establece en verdadero, el sistema usa la propiedad del título de su controlador de vista como el título de la notificación. Cuando se establece en falso, el sistema establece el título de la notificación al nombre de su aplicación. Si no especifica esta clave, el valor predeterminado se establece en falso.

4. Crear vista personalizada en el archivo `NotificationViewController.swift`
5. Agregue una nueva `category key` y establezca su valor según lo que escribimos en la Lista de información (paso 3):

Empujar:

```
{  
  aps: {
```

```
alert: { ... },
category: 'io.swifting.notification-category'
}
}
```

Local:

```
let mutableNotificationContent = UNMutableNotificationContent()
mutableNotificationContent.category = "io.swifting.notification-category"
mutableNotificationContent.title = "Swifting.io Notifications"
mutableNotificationContent.subtitle = "Swifting.io presents"
mutableNotificationContent.body = "Custom notifications"
```

También puedes ver la referencia oficial de la API:

https://developer.apple.com/reference/usernotificationsui/unnotificationcontentextension?utm_source=swift

Lea Extensión para notificaciones push enriquecidas - iOS 10. en línea:

<https://riptutorial.com/es/ios/topic/9501/extension-para-notificaciones-push-enriquecidas---ios-10->

Capítulo 69: FacebookSDK

Examples

Integración de FacebookSDK

Paso 1: Instala el SDK

Puede instalar el SDK [manualmente](#) o a través de `CocoaPods` . La última opción es muy recomendable.

Ponga estas líneas en `Podfile` :

```
target 'MyApp' do
  use_frameworks!

  pod 'FBSDKCoreKit'
  pod 'FBSDKLoginKit'
  pod 'FBSDKShareKit'
end
```

Ejecute `pod install` en el terminal y abra `.xcworkspace` lugar de `.xcodeproj` después.

`FBSDKLoginKit` y `FBSDKShareKit` son opcionales. Puede o no puede necesitarlos.

Paso 2: Crea una aplicación en Facebook

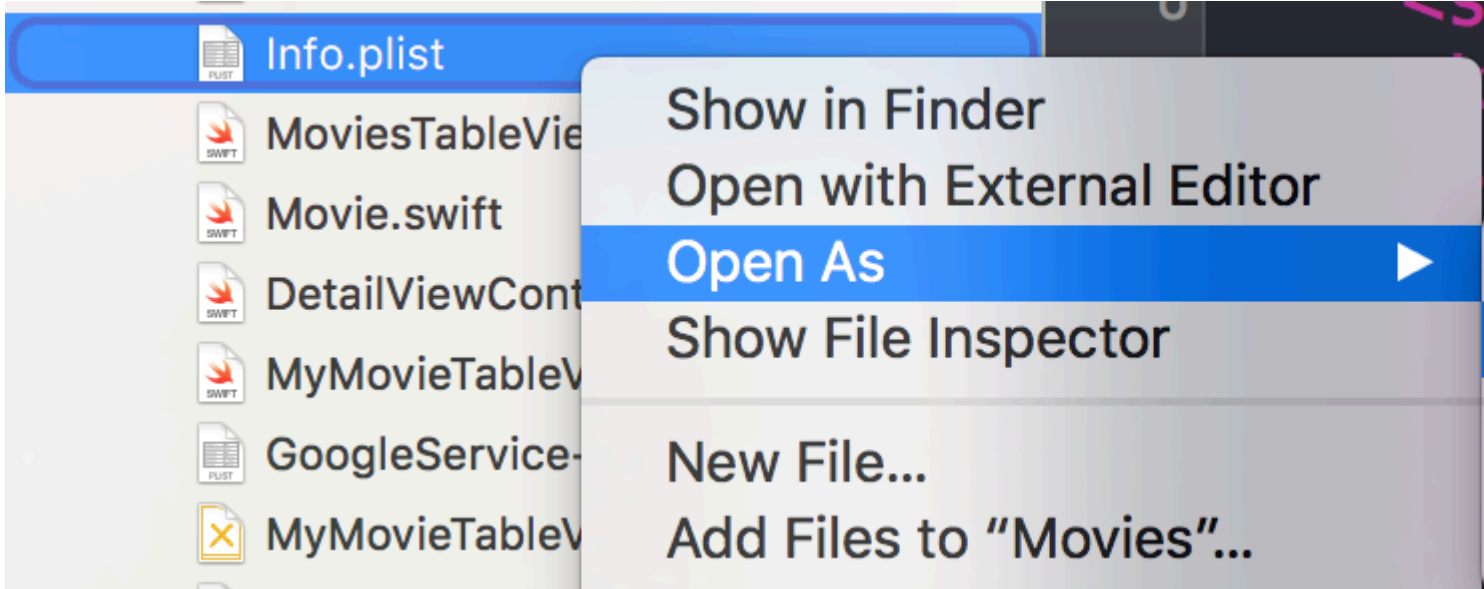
Vaya a [Inicio rápido: Facebook para desarrolladores](#) para crear una aplicación.

Facebook te pedirá que descargues el SDK después de crear la aplicación. Puede omitir esta parte si ya instaló el SDK a través de `CocoaPods`.

Paso 3: Editar `.plist`

a. Para que su aplicación pueda "comunicarse" con Facebook, debe poner algunas configuraciones en su archivo `.plist` . Facebook te dará el fragmento personalizado en la página de inicio rápido.

segundo. Edite su archivo `.plist` como código fuente.



do. Pega tu fragmento personalizado en el código fuente. **¡Ten cuidado!** El fragmento debe ser exactamente el hijo de la etiqueta `<dict>` . Su código fuente debe ser algo como:

```
<plist version="1.0">
<dict>
  // ...
  //some default settings
  // ...
  <key>CFBundleURLTypes</key>
  <array>
    <dict>
      <key>CFBundleURLSchemes</key>
      <array>
        <string>fb{FBAppId}</string>
      </array>
    </dict>
  </array>
  <key>FacebookAppID</key>
  <string>{FBAppId}</string>
  <key>FacebookDisplayName</key>
  <string>{FBAppName}</string>
  <key>LSApplicationQueriesSchemes</key>
  <array>
    <string>fbapi</string>
    <string>fb-messenger-api</string>
    <string>fbauth2</string>
    <string>fbshareextension</string>
  </array>
  <key>NSAppTransportSecurity</key>
  <dict>
    <key>NSExceptionDomains</key>
    <dict>
      <key>facebook.com</key>
      <dict>
        <key>NSIncludesSubdomains</key>
        <true/>
        <key>NSEnvironmentRequiresForwardSecrecy</key>
        <false/>
      </dict>
      <key>fbcdn.net</key>
    </dict>
  </dict>
</plist>
```



```

        <key>NSIncludesSubdomains</key>
        <true/>
        <key>NSExceptionRequiresForwardSecrecy</key>
        <false/>
    </dict>
    <key>akamaihd.net</key>
    <dict>
        <key>NSIncludesSubdomains</key>
        <true/>
        <key>NSExceptionRequiresForwardSecrecy</key>
        <false/>
    </dict>
</dict>
</plist>

```

Si pega el fragmento en un lugar incorrecto, se encontrará con problemas.

Paso 4: informe a Facebook su identificador de paquete en la página de inicio rápido.

=> [Cómo obtener el identificador de paquete](#)

Paso 5: Edita tu `AppDelegate.swift`

a.

```
import FBSDKCoreKit
```

segundo.

```

func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[NSObject: AnyObject]?) -> Bool {
    FBSDKApplicationDelegate.sharedInstance().application(application,
didFinishLaunchingWithOptions: launchOptions)
    return true
}

func application(application: UIApplication, openURL url: NSURL, sourceApplication: String?,
annotation: AnyObject) -> Bool {
    return FBSDKApplicationDelegate.sharedInstance().application(application, openURL: url,
sourceApplication: sourceApplication, annotation: annotation)
}

```

Creando su propio botón "Iniciar sesión con Facebook" personalizado

A veces, queremos diseñar nuestra propia interfaz de usuario para el botón "Iniciar sesión con Facebook" en lugar del botón original que viene con FacebookSDK.

1. En tu guión gráfico, arrastra tu UIButton y configúralo como quieras que sea.
2. Ctrl + arrastre su botón a su controlador de vista como IBAction.
3. **Dentro** del método IBAction, tendrá que simular un toque en el botón real de Facebook de la siguiente manera:

Rápido:

```

let loginButton = FBSDKLoginButton()
loginButton.delegate = self
// Your Custom Permissions Array
loginButton.readPermissions =
[
    "public_profile",
    "email",
    "user_about_me",
    "user_photos"
]
// Hiding the button
loginButton.hidden = true
self.view.addSubview(loginButton)
// Simulating a tap for the actual Facebook SDK button
loginButton.sendActionsForControlEvents(UIControlEvents.TouchUpInside)

```

C objetivo:

```

FBSDKLoginButton *FBButton = [FBSDKLoginButton new];

// Your Custom Permissions Array
FBButton.readPermissions = @[@"public_profile",
    @"email",
    @"user_about_me",
    @"user_photos"
];

FBButton.loginBehavior = FBSDKLoginBehaviorNative;
[FBButton setDelegate:self];
[FBButton setHidden:true];
[loginButton addSubview:FBButton];

[FBButton sendActionsForControlEvents:UIControlEventTouchUpInside];

```

Estas hecho

Obteniendo los datos de usuario de facebook

Después de que el usuario haya `FBButton.readPermissions` en Facebook desde su aplicación, ahora es el momento de buscar los datos que solicitó en `FBButton.readPermissions`.

Rápido:

```

enum FacebookParametesField : String
{
    case FIELDS_KEY = "fields"
    case FIELDS_VALUE = "id, email, picture, first_name, last_name"
}

if FBSDKAccessToken.currentAccessToken() != nil
{
    // Getting user facebook data
    FBSDKGraphRequest(graphPath: "me",
        parameters: [FacebookParametesField.FIELDS_KEY.rawValue :
    FacebookParametesField.FIELDS_VALUE.rawValue])
    .startWithCompletionHandler({ (graphConnection : FBSDKGraphRequestConnection!, result :
    AnyObject!, error : NSError!) -> Void in

```

```

if error == nil
{
    print("Facebook Graph phaze")

    let email = result["email"]
    let facebookToken = FBSDKAccessToken.currentAccessToken().tokenString
    let userFacebookId = result["id"]
    let firstName = result["first_name"]
    let lastName = result["last_name"]

    if let result = result as? Dictionary<String, AnyObject>
    {
        if let picture = result["picture"] as? Dictionary<String,AnyObject>
        {
            if let data = picture["data"] as? Dictionary <String,AnyObject>
            {
                if let url = data["url"] as? String
                {
                    // Profile picture URL
                    let profilePictureURL = url
                }
            }
        }
    }
}
})
}

```

Lea FacebookSDK en línea: <https://riptutorial.com/es/ios/topic/2972/facebooksdk>

Capítulo 70: FileHandle

Introducción

Leer el archivo en trozos del directorio de documentos

Examples

Lea el archivo del directorio de documentos en trozos

Obtengo la ruta del archivo del directorio de documentos y leo ese archivo en trozos de 1024 y lo `NSData (NSMutableData)` al objeto `NSMutableData` o puede escribir directamente en el socket.

```
// MARK: - Get file data as chunks Methode.
func getFileDataInChunks() {

    let documentDirectoryPath = NSSearchPathForDirectoriesInDomains(.documentDirectory,
        .userDomainMask, true)[0] as NSString
    let filePath = documentDirectoryPath.appendingPathComponent("video.mp4")

    //Check file exists at path or not.
    if FileManager.default.fileExists(atPath: filePath) {

        let chunkSize = 1024 // divide data into 1 kb

        //Create NSMutableData object to save read data.
        let ReadData = NSMutableData()

        do {

            //open file for reading.
            outputFileHandle = try FileHandle(forReadingFrom: URL(fileURLWithPath: filePath))

            // get the first chunk
            var datas = outputFileHandle?.readData(ofLength: chunkSize)

            //check next chunk is empty or not.
            while !(datas?.isEmpty)! {

                //here I write chunk data to ReadData or you can directly write to socket.
                ReadData.append(datas!)

                // get the next chunk
                datas = outputFileHandle?.readData(ofLength: chunkSize)

                print("Running: \(ReadData.length) ")
            }

            //close outputFileHandle after reading data complete.
            outputFileHandle?.closeFile()

            print("File reading complete")
        }
    }
}
```

```
    }catch let error as NSError {  
        print("Error : \(error.localizedDescription)")  
    }  
}  
}
```

Después de completar la lectura del archivo, obtendrá los datos del archivo en la variable `ReadData`
Aquí `outputFileHandle` es un objeto de `FileHandle`

```
var outputFileHandle:FileHandle?
```

Lea `FileHandle` en línea: <https://riptutorial.com/es/ios/topic/10665/filehandle>

Capítulo 71: Filtros de CoreImage

Examples

Ejemplo de filtro de imagen central

C objetivo

Simplemente inicie sesión para ver cómo usar un filtro en particular.

```
NSArray *properties = [CIFilter filterNamesInCategory:kCICategoryBuiltIn];

for (NSString *filterName in properties)
{
    CIFilter *fltr = [CIFilter filterWithName:filterName];
    NSLog(@"%@", [fltr attributes]);
}
```

En el caso de CISepiaTone el registro del sistema es el siguiente

```
CIAttributeFilterDisplayName = "Sepia Tone";
CIAttributeFilterName = CISepiaTone;
    CIAttributeReferenceDocumentation = "http://developer.apple.com/cgi-
bin/apple_ref.cgi?apple_ref=//apple_ref/doc/filter/ci/CISepiaTone";
    inputImage =
    {
        CIAttributeClass = CIImage;
        CIAttributeDescription = "The image to use as an input image. For filters that also
use a background image, this is the foreground image.";
        CIAttributeDisplayName = Image;
        CIAttributeType = CIAttributeTypeImage;
    };
    inputIntensity =
    {
        CIAttributeClass = NSNumber;
        CIAttributeDefault = 1;
        CIAttributeDescription = "The intensity of the sepia effect. A value of 1.0 creates a
monochrome sepia image. A value of 0.0 has no effect on the image.";
        CIAttributeDisplayName = Intensity;
        CIAttributeIdentity = 0;
        CIAttributeMin = 0;
        CIAttributeSliderMax = 1;
        CIAttributeSliderMin = 0;
        CIAttributeType = CIAttributeTypeScalar;
    };
}
```

Usando el registro del sistema anterior, configuramos el filtro de la siguiente manera:

```
CIImage *beginImage = [CIImage imageWithCGImage:[myImageView.image CGImage]];
CIContext *context = [CIContext contextWithOptions:nil];
//select Filter Name and Intensity
CIFilter *filter = [CIFilter filterWithName:@"CISepiaTone" keysAndValues:
kCIInputImageKey, beginImage, @"inputIntensity", [NSNumber numberWithInt:0.8], nil];
CIImage *outputImage = [filter outputImage];
```

```

CGImageRef cgimg = [context createCGImage:outputImage fromRect:[outputImage extent]];
UIImage *newImg = [UIImage imageWithCGImage:cgimg];

[myImageView1 setImage:newImg];

CGImageRelease(cgimg);

```

Imagen generada por el código anterior.



Una forma alternativa de configurar un filtro.

```

UIImageView *imageView1=[[UIImageView alloc] initWithFrame:CGRectMake(0, 0,
self.view.frame.size.width, self.view.frame.size.height/2)];
UIImageView *imageView2=[[UIImageView alloc] initWithFrame:CGRectMake(0,
self.view.frame.size.height/2, self.view.frame.size.width, self.view.frame.size.height/2)];
imageView1.image=[UIImage imageNamed:@"image.png"];

CIImage *beginImage = [CIImage imageWithCGImage:[imageView1.image CGImage]];
CIText *context = [CIText contextWithOptions:nil];
//select Filter Name and Intensity

CIFilter *filter = [CIFilter filterWithName:@"CIColorPosterize"];

```

```

[filter setValue:beginImage forKey:kCIInputImageKey];
[filter setValue:[NSNumber numberWithFloat:8.0] forKey:@"inputLevels"];
CIImage *outputImage = [filter outputImage];

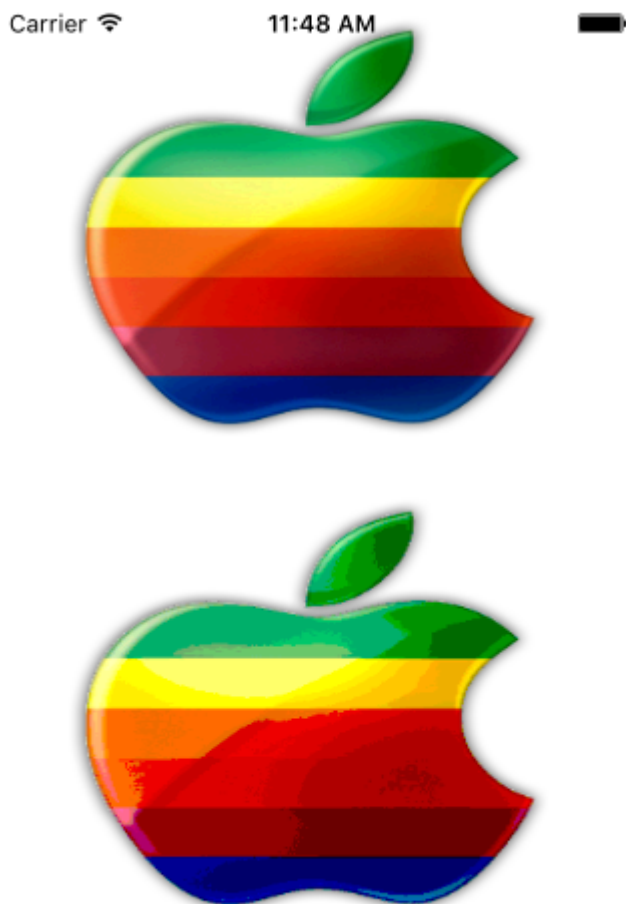
CGImageRef cgimg = [context createCGImage:outputImage fromRect:[outputImage extent]];
UIImage *newImg = [UIImage imageWithCGImage:cgimg];

[imageView2 setImage:newImg];

CGImageRelease(cgimg);
[self.view addSubview:imageView1];
[self.view addSubview:imageView2];

```

Imagen generada a partir de este código.



Todos los filtros disponibles son los siguientes

```

/* CIAccordionFoldTransition,
   CIAdditionCompositing,
   CIAffineClamp,
   CIAffineTile,
   CIAffineTransform,
   CIColorAverage,
   CIColorHistogram,
   CIColorMaximum,
   CIColorMaximumAlpha,
   CIColorMinimum,
   CIColorMinimumAlpha,
   CIAztecCodeGenerator,

```


CIBarsSwipeTransition,
CIBlendWithAlphaMask,
CIBlendWithMask,
CIBloom,
CIBoxBlur,
CIBumpDistortion,
CIBumpDistortionLinear,
CICheckerboardGenerator,
CICircleSplashDistortion,
CICircularScreen,
CICircularWrap,
CICMYKHalftone,
CICode128BarcodeGenerator,
CIColorBlendMode,
CIColorBurnBlendMode,
CIColorClamp,
CIColorControls,
CIColorCrossPolynomial,
CIColorCube,
CIColorCubeWithColorSpace,
CIColorDodgeBlendMode,
CIColorInvert,
CIColorMap,
CIColorMatrix,
CIColorMonochrome,
CIColorPolynomial,
CIColorPosterize,
CIColumnAverage,
CIComicEffect,
CIConstantColorGenerator,
CIConvolution3X3,
CIConvolution5X5,
CIConvolution7X7,
CIConvolution9Horizontal,
CIConvolution9Vertical,
CICopyMachineTransition,
CICrop,
CICrystallize,
CIDarkenBlendMode,
CIDepthOfField,
CIDifferenceBlendMode,
CIDiscBlur,
CIDisintegrateWithMaskTransition,
CIDisplacementDistortion,
CIDissolveTransition,
CIDivideBlendMode,
CIDotScreen,
CIDroste,
CIEdges,
CIEdgeWork,
CIEightfoldReflectedTile,
CIExclusionBlendMode,
CIExposureAdjust,
CIFalseColor,
CIFlashTransition,
CIFourfoldReflectedTile,
CIFourfoldRotatedTile,
CIFourfoldTranslatedTile,
CIGammaAdjust,
CIGaussianBlur,
CIGaussianGradient,

CIGlassDistortion,
CIGlassLozenge,
CI.GlideReflectedTile,
CI.Gloom,
CI.HardLightBlendMode,
CI.HatchedScreen,
CI.HeightFieldFromMask,
CI.HexagonalPixellate,
CI.HighlightShadowAdjust,
CI.HistogramDisplayFilter,
CI.HoleDistortion,
CI.HueAdjust,
CI.HueBlendMode,
CI.Kaleidoscope,
CI.LanczosScaleTransform,
CI.LenticularHaloGenerator,
CI.LightenBlendMode,
CI.LightTunnel,
CI.LinearBurnBlendMode,
CI.LinearDodgeBlendMode,
CI.LinearGradient,
CI.LinearToSRGBToneCurve,
CI.LineOverlay,
CI.LineScreen,
CI.LuminosityBlendMode,
CI.MaskedVariableBlur,
CI.MaskToAlpha,
CI.MaximumComponent,
CI.MaximumCompositing,
CI.MedianFilter,
CI.MinimumComponent,
CI.MinimumCompositing,
CI.ModTransition,
CI.MotionBlur,
CI.MultiplyBlendMode,
CI.MultiplyCompositing,
CI.NoiseReduction,
CI.OpTile,
CI.OverlayBlendMode,
CI.PageCurlTransition,
CI.PageCurlWithShadowTransition,
CI.ParallelogramTile,
CI.PDF417BarcodeGenerator,
CI.PerspectiveCorrection,
CI.PerspectiveTile,
CI.PerspectiveTransform,
CI.PerspectiveTransformWithExtent,
CI.PhotoEffectChrome,
CI.PhotoEffectFade,
CI.PhotoEffectInstant,
CI.PhotoEffectMono,
CI.PhotoEffectNoir,
CI.PhotoEffectProcess,
CI.PhotoEffectTonal,
CI.PhotoEffectTransfer,
CI.PinchDistortion,
CI.PinLightBlendMode,
CI.Pixellate,
CI.Pointillize,
CI.QRCodeGenerator,
CI.RadialGradient,

```
CIRandomGenerator,  
CIRippleTransition,  
CIRowAverage,  
CISaturationBlendMode,  
CIScreenBlendMode,  
CISepiaTone,  
CIShadedMaterial,  
CISharpenLuminance,  
CISixfoldReflectedTile,  
CISixfoldRotatedTile,  
CISmoothLinearGradient,  
CISoftLightBlendMode,  
CISourceAtopCompositing,  
CISourceInCompositing,  
CISourceOutCompositing,  
CISourceOverCompositing,  
CISpotColor,  
CISpotLight,  
CISRGBToneCurveToLinear,  
CIStarShineGenerator,  
CIStraightenFilter,  
CISretchCrop,  
CIStripesGenerator,  
CISubtractBlendMode,  
CISunbeamsGenerator,  
CISwipeTransition,  
CITemperatureAndTint,  
CIToneCurve,  
CITorusLensDistortion,  
CITriangleKaleidoscope,  
CITriangleTile,  
CITwelvefoldReflectedTile,  
CITwirlDistortion,  
CIUnsharpMask,  
CIVibrance,  
CIVignette,  
CIVignetteEffect,  
CIVortexDistortion,  
CIWhitePointAdjust,  
CIZoomBlur*/
```

Lea Filtros de CoreImage en línea: <https://riptutorial.com/es/ios/topic/7278/filtros-de-coreimage>

Capítulo 72: Firma de código

Examples

Perfiles de aprovisionamiento

Para crear un archivo IPA en XCode, debe firmar su solicitud con un certificado y un perfil de aprovisionamiento. Estos se pueden crear en

<https://developer.apple.com/account/ios/profile/create>

Tipos de perfil de aprovisionamiento

Los perfiles de aprovisionamiento se dividen en dos tipos, desarrollo y distribución:

Desarrollo

- Desarrollo de aplicaciones iOS / desarrollo de aplicaciones tvOS: se utiliza en el desarrollo para instalar su aplicación en un dispositivo de prueba.

Distribución

- App Store / tvOS App Store: se utiliza para firmar su solicitud de carga en la tienda de aplicaciones.
- En casa: se utiliza para la distribución empresarial de su aplicación a dispositivos dentro de su empresa.
- Ad Hoc / tvOS Ad Hoc: se utiliza para distribuir su aplicación a un número limitado de dispositivos específicos (por ejemplo, deberá conocer los UDID de los dispositivos en los que desea instalar su aplicación).

Lea Firma de código en línea: <https://riptutorial.com/es/ios/topic/6055/firma-de-codigo>

Capítulo 73: Fuentes personalizadas

Examples

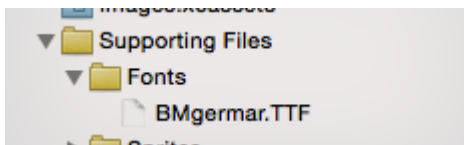
Incrustar fuentes personalizadas

Soporte de fuente personalizado

Las aplicaciones que desean usar fuentes personalizadas ahora pueden incluir esas fuentes en su paquete de aplicaciones y registrarlas con el sistema al incluir la clave `UIAppFonts` en su archivo `Info.plist`. El valor de esta clave es una serie de cadenas que identifican los archivos de fuentes en el paquete de la aplicación. Cuando el sistema ve la clave, carga las fuentes especificadas y las pone a disposición de la aplicación.

Una vez que las fuentes se hayan configurado en la `Info.plist`, puede usar sus fuentes personalizadas como cualquier otra fuente en IB o programáticamente.

1. Arrastre y suelte su fuente en la carpeta de archivos de soporte de Xcode. No olvides marcar tu aplicación en la sección "Agregar a objetivos". A partir de este momento, puede utilizar esta fuente en IB y elegirla de la paleta de fuentes.



2. Para que esta fuente esté disponible en el dispositivo, abra `Info.plist` y agregue las `Fonts provided by application` key (`UIAppFonts`). Agregue el nombre de la fuente como valor a la clave del elemento 0. Nota: El nombre de la fuente puede variar de su nombre de archivo de fuente.

▼ Fonts provided by application	Array	(1 item)
Item 0	String	BMgermar.TTF

3. Obtenga el nombre de fuente agregado personalizado usando el siguiente fragmento de código

[Swift 3]

```
for family in UIFont.familyNames {
    print("\(family)")

    for name in UIFont.fontNames(forFamilyName: family) {
        print("    \(name)")
    }
}
```

[Objetivo - C]

```
for (NSString *familyName in [UIFont familyNames]){
    NSLog(@"Family name: %@", familyName);
    for (NSString *fontName in [UIFont fontNamesForFamilyName:familyName]) {
        NSLog(@"--Font name: %@", fontName);
    }
}
```

Fuentes personalizadas con guión gráfico

Las fuentes personalizadas para los componentes de la interfaz de usuario desde el guión gráfico se pueden lograr fácilmente con los [atributos de tiempo de ejecución definidos por el usuario](#) en el guión gráfico y las [categorías](#) .

Las ventajas son como,

- No es necesario definir salidas para el elemento ui
- No hay necesidad de configurar la fuente para elementos programáticamente.

Pasos a seguir

1. **Archivo de fuente:** agregue el archivo de fuente (.ttf) al paquete de la aplicación y agregue la entrada para la fuente en Info.plist bajo **Fuente provista por la aplicación** como en esta [documentación](#) de fuentes personalizadas.
2. **Definir categorías:** agregue un archivo como **UIKit + IBExtensions** y agregue las categorías para los elementos de la interfaz de usuario como UILabel, UIButton, etc. para los que desea establecer una fuente personalizada. Todas las categorías tendrán una propiedad personalizada, por ejemplo, **FontName** . Esto se usará más adelante en el guión gráfico para configurar la fuente personalizada (como en el paso 4).

UIKit + IBExtensions.h

```
#import <UIKit/UIKit.h>

//Category extension for UILabel
@interface UILabel (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end

// Category extension for UITextField
@interface UITextField (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end

// Category extension for UIButton
@interface UIButton (IBExtensions)

@property (nonatomic, copy) NSString *fontName;
@end
```

3. **Getters and Setters:** defina getters y setters para la propiedad `fontName` en cada categoría agregada.

UIKit + IBExtensions.m

```
#import "UIKit+IBExtensions.h"

@implementation UILabel (IBExtensions)

- (NSString *)fontName {
    return self.font.fontName;
}

- (void)setFontName:(NSString *)fontName {
    self.font = [UIFont fontWithName:fontName size:self.font.pointSize];
}

@end

@implementation UITextField (IBExtensions)

- (NSString *)fontName {
    return self.font.fontName;
}

- (void)setFontName:(NSString *)fontName {
    self.font = [UIFont fontWithName:fontName size:self.font.pointSize];
}

@end

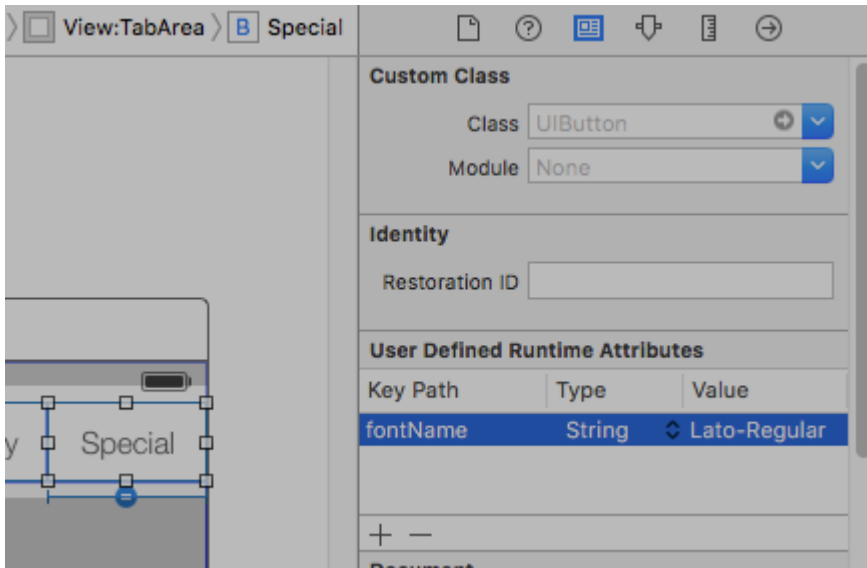
@implementation UIButton (IBExtensions)

- (NSString *)fontName {
    return self.titleLabel.font.fontName;
}

- (void)setFontName:(NSString *)fontName{
    self.titleLabel.font = [UIFont fontWithName:fontName size:self.titleLabel.font.pointSize];
}

@end
```

4. **Configuración de la fuente en el guión gráfico:** agregue una entrada en Atributos de tiempo de ejecución definidos por el usuario con **fontName** como `keyPath` y el nombre de su **fuente personalizada** como valor con el tipo como `String` como se muestra.



Esto establecerá su fuente personalizada mientras se ejecuta la aplicación.

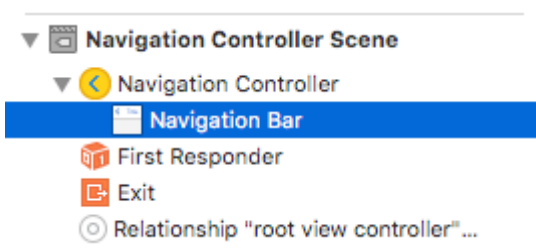
Notas:

- Lato-Regular es la fuente personalizada que he usado.
- El mismo nombre en el archivo **.ttf** agregado en el paquete debe usarse sin extensión en el guión gráfico.
- El tamaño de fuente será el mismo que se define en el inspector de atributos del elemento UI.

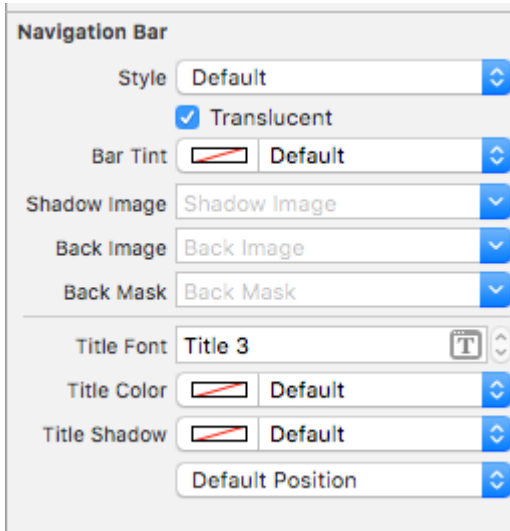
Aplicar fuentes personalizadas a los controles dentro de un Storyboard

El siguiente ejemplo muestra cómo aplicar fuentes personalizadas a una barra de navegación e incluye correcciones para algunos comportamientos extraños que se encuentran en Xcode. También se pueden aplicar las fuentes personalizadas a **cualquier otro UIControls** como **UILabels**, **UIButtons** y más utilizando el inspector de atributos después de agregar la fuente personalizada al proyecto. Tenga en cuenta los enlaces externos a las muestras de trabajo y videos cerca de la parte inferior.

1. Seleccione su barra de navegación dentro de su controlador de navegación



2. Cambiar la fuente del título en el inspector de atributos

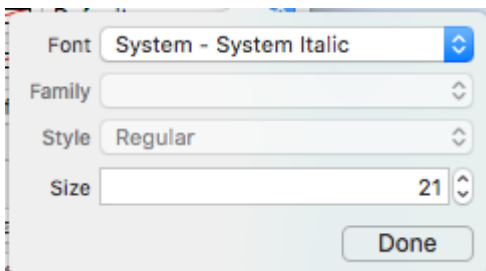


(Es probable que tengas que alternar el Tinte de barra para la Barra de navegación antes de que Xcode seleccione la nueva fuente)

Notas (advertencias)

Verificado que esto funciona en Xcode 7.1.1+. (**Vea las muestras abajo**)

1. Es necesario que alterne el tinte de la barra de navegación antes de que la fuente surta efecto (parece un error en Xcode; puede volver a la configuración predeterminada y la fuente se mantendrá)
2. Si elige una fuente del sistema ~ Asegúrese de asegurarse de que el tamaño no sea 0.0 (de lo contrario, se ignorará la nueva fuente)



3. Parece que esto funciona sin problemas cuando solo una barra de navegación está en la jerarquía de vistas. Parece que las barras de navegación secundarias en la misma pila se ignoran. (Tenga en cuenta que si muestra la barra de navegación del controlador de navegación maestro, se ignorarán todas las demás configuraciones personalizadas de la barra de navegación).

Gotchas (deux)

Algunos de estos se repiten, lo que significa que es muy probable que sean dignos de mención.

1. A veces el storyboard xml se corrompe. Esto requiere que revise la estructura en el Guión gráfico como modo de Código fuente (haga clic con el botón derecho en el archivo del guión gráfico> Abrir como ...)

2. En algunos casos, la etiqueta de NavigationItem asociada con el atributo de tiempo de ejecución definido por el usuario se estableció como un elemento secundario XML de la etiqueta de vista en lugar de la etiqueta de controlador de vista. Si es así, quítelo de entre las etiquetas para que funcione correctamente.
3. Alterne el Tinte de la barra de navegación para asegurarse de que se utiliza la fuente personalizada.
4. Verifique el parámetro de tamaño de la fuente a menos que use un estilo de fuente dinámico
5. La jerarquía de vistas anulará la configuración. Parece que una fuente por pila es posible.

Resultado

Muestras

- [Video que muestra múltiples fuentes en un proyecto avanzado](#)
- [Descarga de fuente simple](#)
- [Descarga avanzada del proyecto ~ Muestra múltiples fuentes NavBar y soluciones de fuentes personalizadas](#)
- [Video que muestra múltiples fuentes y fuentes personalizadas](#)

Manejo de fuentes personalizadas

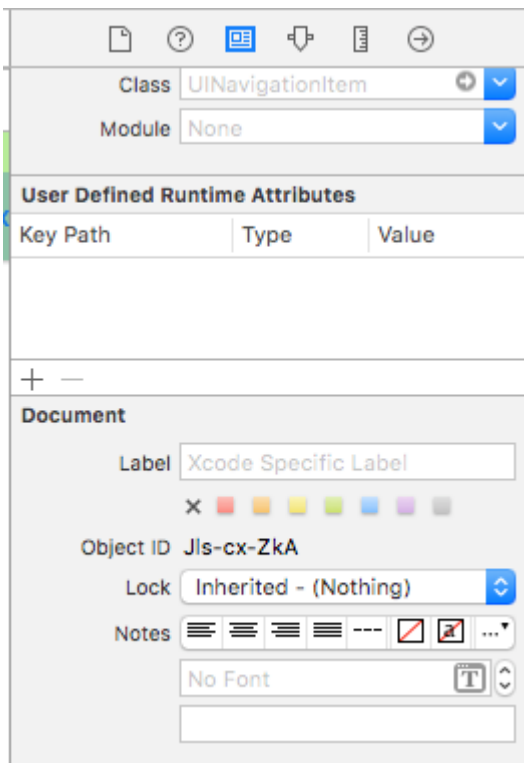
Nota ~ Puede encontrar una [buena lista de verificación](#) en el sitio web de Code With Chris y puede ver el proyecto de descarga de muestra.

Si tiene su propia fuente y desea usarla en su guión gráfico, entonces hay un conjunto decente de respuestas en la siguiente [pregunta SO](#) . Una respuesta identifica estos pasos.

1. Obtenga su archivo de fuente personalizado (.ttf, .ttc)
2. Importe los archivos de fuentes a su proyecto Xcode
3. En app-info.plist, agregue una clave llamada Fuentes provista por la aplicación. Es un tipo de matriz, agregue todos los nombres de los archivos de fuentes a la matriz, tenga en cuenta: incluida la extensión del archivo.
4. En el guión gráfico, en la barra de navegación, vaya al inspector de atributos, haga clic en el botón derecho del icono del área de selección de fuente. En el panel emergente, elija Fuente a personalizar y elija la Familia de su nombre de fuente incrustado.

Solución de fuente personalizada

Así que, naturalmente, parece que Xcode puede manejar fuentes personalizadas en UINavigationController, pero esa característica simplemente no se actualiza correctamente (la fuente seleccionada se ignora).



Para solucionar esto:

Una forma es arreglar usando el guión gráfico y agregando una línea de código: primero agregue una UIView (UIButton, UILabel, o alguna otra subclase UIView) al controlador de vista (no el elemento de navegación ... Xcode actualmente no permite que lo haga ese). Después de agregar el control, puede modificar la fuente en el guión gráfico y agregar una referencia como salida a su View Controller. Simplemente asigne esa vista al UINavigationController.titleView. También puede establecer el nombre del texto en código si es necesario. Error informado (23600285).

```
@IBOutlet var customFontTitleView: UIButton!  
  
//Sometime later...  
self.navigationItem.titleView = customFontTitleView
```

Nota: este ejemplo se deriva de una respuesta que publiqué en SO ([aquí](#)).

Lea Fuentes personalizadas en línea: <https://riptutorial.com/es/ios/topic/1504/fuentes-personalizadas>

Capítulo 74: GCD (Grand Central Dispatch)

Introducción

Grand Central Dispatch (GCD) es la respuesta de Apple al multihilo. Es un marco liviano para realizar tareas de forma síncrona o asíncrona en colas y maneja los hilos de la CPU para usted entre bastidores.

Tema relacionado: [Concurrencia](#)

Examples

Crear una cola de envío

Puedes crear tu propia cola usando `dispatch_queue_create`

C objetivo

```
dispatch_queue_t queue = dispatch_queue_create("com.example.myqueue", DISPATCH_QUEUE_SERIAL);
```

Rápido

```
// Before Swift 3
let queue = dispatch_queue_create("com.example.myqueue", DISPATCH_QUEUE_SERIAL)
// Swift 3
let queue = DispatchQueue(label: "com.example.myqueue") //default is serial queue, unless
.concurrent is specified as an attribute otherwise
```

Conseguir la cola principal

La cola principal es la cola de envío en la que tienen lugar todas las actualizaciones de la interfaz de usuario y se coloca el código que implica los cambios de la interfaz de usuario.

`NSURLSession` llegar a la cola principal para actualizar la interfaz de usuario al finalizar un proceso asíncrono como `NSURLSession`

Hay dos tipos de llamadas de cola principales `synchronous` y `asynchronous`. Cuando invocas algo de forma `synchronously`, significa que el hilo que inició esa operación esperará a que la tarea termine antes de continuar. `Asynchronous` significa que no esperará.

Código Objetivo-C

Llamada de cola principal `synchronous`

```
dispatch_queue_t queue = dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
```

Llamada `Asynchronous` la cola principal

```
dispatch_async(dispatch_get_main_queue(), ^{
    // do work here to Usually to update the User Interface
});
```

SWIFT 3

Llamada `Asynchronous` la cola principal

```
DispatchQueue.main.async {
}
}
```

Llamada de cola principal `Synchronous`

```
DispatchQueue.main.sync {
}
}
```

Grupo de despacho

`DispatchGroup` permite la sincronización agregada del trabajo. Puede usarlos para enviar varios elementos de trabajo diferentes y realizar un seguimiento cuando se completan, incluso aunque se ejecuten en diferentes colas. Este comportamiento puede ser útil cuando no se puede avanzar hasta completar todas las tareas especificadas.

Un escenario en el que esto podría ser útil es si tiene varias llamadas de servicio web que todas necesitan terminar antes de continuar. Por ejemplo, necesita descargar varios conjuntos de datos que deben ser procesados por alguna función. Debe esperar a que todos los servicios web se completan antes de llamar a la función para procesar todos los datos recibidos.

Swift 3

```
func doLongTasksAndWait () {
    print("starting long running tasks")
    let group = DispatchGroup() //create a group for a bunch of tasks we are about to
do
    for i in 0...3 { //launch a bunch of tasks (eg a bunch of webservice
calls that all need to be finished before proceeding to the next ViewController)
        group.enter() //let the group know that something is being added
        DispatchQueue.global().async { //run tasks on a background thread
            sleep(arc4random() % 4) //do some long task eg webservice or database lookup
(here we are just sleeping for a random amount of time for demonstration purposes)
            print("long task \(i) done!")
            group.leave() //let group know that the task is finished
        }
    }
    group.wait() //will block whatever thread we are on here until all
the above tasks have finished (so maybe dont use this function on your main thread)
    print("all tasks done!")
}
```

```
}
```

Alternativamente, si no desea esperar a que finalicen los grupos, sino que desea ejecutar una función una vez que todas las tareas hayan finalizado, use la función de `notify` en lugar de `group.wait()`

```
group.notify(queue: DispatchQueue.main) { //the queue: parameter is which queue this block
will run on, if you need to do UI updates, use the main queue
    print("all tasks done!")           //this will execute when all tasks have left the
group
}
```

Ejemplo de salida:

```
starting long running tasks
long task 0 done!
long task 3 done!
long task 1 done!
long task 2 done!
all tasks done!
```

Para obtener más información, consulte los [documentos de Apple](#) o el [tema](#) relacionado

Despido semáforo

`DispatchSemaphore` proporciona una implementación eficiente de un semáforo de conteo tradicional, que se puede usar para controlar el acceso a un recurso en múltiples contextos de ejecución.

Un escenario para cuándo usar un semáforo podría ser si está leyendo o escribiendo un archivo, si varias tareas intentan leer y escribir desde un archivo al mismo tiempo, podría aumentar su rendimiento para hacer que cada tarea espere su turno para que Para no sobrecargar el controlador de E / S.

Swift 3

```
func do2TasksAtATime () {
    print("starting long running tasks (2 at a time)")
    let sem = DispatchSemaphore(value: 2)           //this semaphore only allows 2 tasks to
run at the same time (the resource count)
    for i in 0...7 {                               //launch a bunch of tasks
        DispatchQueue.global().async {            //run tasks on a background thread
            sem.wait()                            //wait here if no resources available
            sleep(2)                              //do some long task eg file access (here
we are just sleeping for a 2 seconds for demonstration purposes)
            print("long task \(i) done! \(Date())")
            sem.signal()                          //let the semaphore know this resource is
now available
        }
    }
}
```

Ejemplo de salida: (observe las marcas de tiempo)

```
starting long running tasks (2 at a time)
long task 0 done! 2017-02-16 07:11:53 +0000
long task 1 done! 2017-02-16 07:11:53 +0000
long task 2 done! 2017-02-16 07:11:55 +0000
long task 3 done! 2017-02-16 07:11:55 +0000
long task 5 done! 2017-02-16 07:11:57 +0000
long task 4 done! 2017-02-16 07:11:57 +0000
long task 6 done! 2017-02-16 07:11:59 +0000
long task 7 done! 2017-02-16 07:11:59 +0000
```

Para más información, consulte los [documentos de Apple](#).

Serial vs Colas de despacho concurrentes

Swift 3

Cola de serie

```
func serialQueues () {
    let serialQueue = DispatchQueue(label: "com.example.serial") //default queue type is a
    serial queue
    let start = Date ()
    for i in 0...3 {
        serialQueue.async {
            thread, using our serial queue
            sleep(2)
            webservice or database lookup
            let timeTaken = Date().timeIntervalSince(start)
            print("serial long task \(i) done! total time taken: \(timeTaken)")
        }
    }
}
```

Ejemplo de salida:

```
serial long task 0 done! total time taken: 2.07241100072861
serial long task 1 done! total time taken: 4.16347700357437
serial long task 2 done! total time taken: 6.23209798336029
serial long task 3 done! total time taken: 8.30682599544525
```

Cola concurrente

```
func concurrentQueues () {
    let concurrentQueue = DispatchQueue(label: "com.example.concurrent", attributes:
    .concurrent) //explicitly specify the queue to be a concurrent queue
    let start = Date ()
    for i in 0...3 {
        concurrentQueue.async {
            sleep(2)
            let timeTaken = Date().timeIntervalSince(start)
            print("concurrent long task \(i) done! total time taken: \(timeTaken)")
        }
    }
}
```

Ejemplo de salida:

```
concurrent long task 3 done! total time taken: 2.07092100381851
concurrent long task 0 done! total time taken: 2.07087397575378
concurrent long task 2 done! total time taken: 2.07086700201035
concurrent long task 1 done! total time taken: 2.07089096307755
```

Discusión

Como podemos ver en los ejemplos anteriores, una cola en serie completará cada tarea en el orden en que se envían a la cola. Cada tarea esperará a que la tarea anterior termine antes de ejecutarse. En cuanto a la cola concurrente, cada tarea no espera a las otras en la cola y se ejecuta tan pronto como sea posible; La ventaja es que todas las tareas en la cola se ejecutarán al mismo tiempo en subprocesos separados, haciendo que una cola concurrente tome menos tiempo que una cola en serie.

Si el orden de ejecución de las tareas no es importante, siempre use una cola concurrente para la mejor eficiencia.

Lea GCD (Grand Central Dispatch) en línea: <https://riptutorial.com/es/ios/topic/4626/gcd--grand-central-dispatch->

Capítulo 75: Gráfico (Coreplot)

Examples

Haciendo gráficos con CorePlot

Core Plot proporciona un podspec, de modo que puede usar cocoapods como su administrador de bibliotecas, lo que debería hacer que la instalación y actualización sean mucho más sencillas.

Instala cocoapods en tu sistema.

En el directorio del proyecto, agregue un archivo de texto a su proyecto llamado Podfile escribiendo `pod init` en el directorio de su proyecto

En el archivo Pod, agregue la línea `pod 'CorePlot', '~> 1.6'`

En la terminal, `cd` a su directorio de proyecto y ejecute `pod install`

Cocoapods generará un archivo `xcworkspace`, que debe usar para iniciar su proyecto (el archivo `.xcodeproj` no incluirá las bibliotecas de pod)

Abra el `.xcworkspace` generado por CocoaPods

En el archivo `ViewController.h`

```
#import <CorePlot/ios/CorePlot.h>
//#import "CorePlot-CocoaTouch.h" or the above import statement
@interface ViewController : UIViewController<CPTPlotDataSource>
```

En el archivo `ViewController.m`

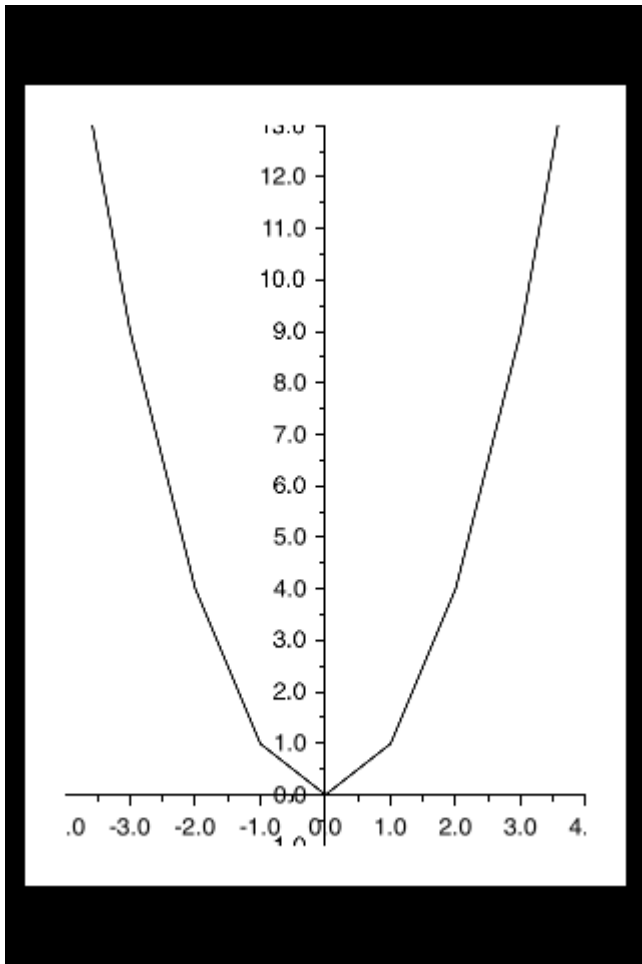
```
-(void)loadView
{
    [super loadView];
    // We need a hostview, you can create one in IB (and create an outlet) or just do this:
    CPTGraphHostingView* hostView = [[CPTGraphHostingView alloc] initWithFrame:CGRectMake(10,
    40, 300, 400)];
    hostView.backgroundColor=[UIColor whiteColor];
    self.view.backgroundColor=[UIColor blackColor];
    [self.view addSubview: hostView];
    // Create a CPTGraph object and add to hostView
    CPTGraph* graph = [[CPTXYGraph alloc] initWithFrame:CGRectMake(10, 40, 300, 400)];
    hostView.hostedGraph = graph;
    // Get the (default) plotSpace from the graph so we can set its x/y ranges
    CPTXYPlotSpace *plotSpace = (CPTXYPlotSpace *) graph.defaultPlotSpace;
    // Note that these CPTPlotRange are defined by START and LENGTH (not START and END) !!
    [plotSpace setYRange: [CPTPlotRange plotRangeWithLocation:CPTDecimalFromFloat( 0 )
    length:CPTDecimalFromFloat( 20 )]];
    [plotSpace setXRange: [CPTPlotRange plotRangeWithLocation:CPTDecimalFromFloat( -4 )
    length:CPTDecimalFromFloat( 8 )]];
    // Create the plot (we do not define actual x/y values yet, these will be supplied by the
    datasource...)
```

```

    CPTScatterPlot* plot = [[CPTScatterPlot alloc] initWithFrame:CGRectZero];
    // Let's keep it simple and let this class act as datasource (therefore we implemtn
    <CPTPlotDataSource>)
    plot.dataSource = self;
    // Finally, add the created plot to the default plot space of the CPTGraph object we
    created before
    [graph addPlot:plot toPlotSpace:graph.defaultPlotSpace];
}
// This method is here because this class also functions as datasource for our graph
// Therefore this class implements the CPTPlotDataSource protocol
-(NSUInteger)numberOfRecordsForPlot:(CPTPlot *)plotnumberOfRecords
{
    return 9; // Our sample graph contains 9 'points'
}
// This method is here because this class also functions as datasource for our graph
// Therefore this class implements the CPTPlotDataSource protocol
-(NSNumber *)numberForPlot:(CPTPlot *)plot field:(NSUInteger)fieldEnum
recordIndex:(NSUInteger)index
{
    // We need to provide an X or Y (this method will be called for each) value for every
    index
    int x = index - 4;
    // This method is actually called twice per point in the plot, one for the X and one for
    the Y value
    if(fieldEnum == CPTScatterPlotFieldX)
    {
        // Return x value, which will, depending on index, be between -4 to 4
        return [NSNumber numberWithInt: x];
    } else
    {
        // Return y value, for this example we'll be plotting y = x * x
        return [NSNumber numberWithInt: x * x];
    }
}
}

```

La salida generada es la siguiente:



Lea Gráfico (Coreplot) en línea: <https://riptutorial.com/es/ios/topic/7302/grafico--coreplot->

Capítulo 76: Grupo de despacho

Introducción

Temas relacionados:

[Grand Central Dispatch](#)

[Concurrencia](#)

Examples

Introducción

Supongamos que tiene múltiples hilos en ejecución. Cada hilo está haciendo una tarea. Desea que se le notifique ya sea en el subproceso principal O en otro subproceso, cuando se hayan completado todos los subprocesos de tareas.

La solución más sencilla para este problema es un `DispatchGroup`.

Al usar un Grupo de `DispatchGroup`, para cada solicitud, `enter` al grupo y para cada solicitud completada, `leave` el grupo.

Cuando ya no haya solicitudes en el grupo, se le `notify` (notificará).

Uso:

```
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        let dispatchGroup = DispatchGroup() //Create a group for the tasks.
        let session: URLSession = URLSession.shared

        dispatchGroup.enter() //Enter the group for the first task.

        let firstTask = session.dataTask(with: URLRequest(url: URL(string:
"https://stackoverflow.com")!)) { (data, response, error) in

            //Process Response..

            dispatchGroup.leave() //Leave the group for the first task.
        }

        dispatchGroup.enter() //Enter the group for the second task.
```

```

        let secondTask = session.dataTask(with: URLRequest(url: URL(string:
"https://google.ca"!)) { (data, response, error) in

            //Process Response..

            dispatchGroup.leave() //Leave the group for the second task.
        }

        //Get notified on the main thread/queue.. when ALL of the tasks above has been
completed.
        dispatchGroup.notify(queue: DispatchQueue.main) {

            print("Every task is complete")

        }

        //Start the tasks.
        firstTask.resume()
        secondTask.resume()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}

```

Con lo anterior, no tiene que `wait` infinitamente hasta que se completen todas las tareas. Puede mostrar un cargador ANTES de que se hayan iniciado todas las tareas y descartar el cargador DESPUÉS de que se completen todas las tareas. De esta manera, su hilo principal no se bloquea y su código permanece limpio.

Ahora suponga que también desea que se `ordered` las tareas o agregue sus respuestas a una matriz secuencialmente. Podrías hacer lo siguiente:

```

import UIKit

//Locking mechanism..
func synchronized(_ lock: AnyObject, closure: () -> Void) {
    objc_sync_enter(lock)
    closure()
    objc_sync_exit(lock)
}

class ViewController: UIViewController {

    let lock = NSObject() //Object to lock on.
    var responseArray = Array<Data?>() //Array of responses.

    override func viewDidLoad() {
        super.viewDidLoad()

        let dispatchGroup = DispatchGroup()
        let session: URLSession = URLSession.shared

        dispatchGroup.enter() //Enter the group for the first task.
    }
}

```

```

    let firstTask = session.dataTask(with: URLRequest(url: URL(string:
"https://stackoverflow.com"!)) { (data, response, error) in

        //Process Response..

        synchronized(self.lock, closure: { () -> Void in
            self.responseArray[0] = data ?? nil
        })

        dispatchGroup.leave() //Leave the group for the first task.
    }

    dispatchGroup.enter() //Enter the group for the second task.

    let secondTask = session.dataTask(with: URLRequest(url: URL(string:
"https://google.ca"!)) { (data, response, error) in

        //Process Response..

        synchronized(self.lock, closure: { () -> Void in
            self.responseArray[1] = data ?? nil
        })

        dispatchGroup.leave() //Leave the group for the second task.
    }

    //Get notified on the main thread.. when ALL of the requests above has been completed.
    dispatchGroup.notify(queue: DispatchQueue.main) {

        print("Every task is complete..")

        for i in 0..

```

Cada entrada debe tener una salida en un `DispatchGroup` . Si olvidas `leave` después de `entering` , te estás preparando. NUNCA se le notificará cuando se completen las tareas.

La cantidad de `enter` debe ser igual a la cantidad de `leave` .

Lea Grupo de despacho en línea: <https://riptutorial.com/es/ios/topic/4624/grupo-de-despacho>

Capítulo 77: Guía para elegir los mejores patrones de arquitectura iOS

Introducción

Desmitificando MVC, MVP, MVVM y VIPER o cualquier otro patrón de diseño para elegir el mejor enfoque para construir una aplicación

Examples

Patrón MVC

```
import UIKit

struct Person { // Model
    let firstName: String
    let lastName: String
}

class GreetingViewController : UIViewController { // View + Controller
    var person: Person!
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self, action: "didTapButton:", forControlEvents:
.TouchUpInside)
    }

    func didTapButton(button: UIButton) {
        let greeting = "Hello" + " " + self.person.firstName + " " + self.person.lastName
        self.greetingLabel.text = greeting
    }

    // layout code goes here
}

// Assembling of MVC
let model = Person(firstName: "David", lastName: "Blaine")
let view = GreetingViewController()
view.person = model
```

Patrones de MVP

```
import UIKit

struct Person { // Model
    let firstName: String
    let lastName: String
}
```



```

protocol GreetingView: class {
    func setGreeting(greeting: String)
}

protocol GreetingViewPresenter {
    init(view: GreetingView, person: Person)
    func showGreeting()
}

class GreetingPresenter : GreetingViewPresenter {
    unowned let view: GreetingView
    let person: Person
    required init(view: GreetingView, person: Person) {
        self.view = view
        self.person = person
    }
    func showGreeting() {
        let greeting = "Hello" + " " + self.person.firstName + " " + self.person.lastName
        self.view.setGreeting(greeting)
    }
}

class GreetingViewController : UIViewController, GreetingView {
    var presenter: GreetingViewPresenter!
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self, action: "didTapButton:", forControlEvents:
.TouchUpInside)
    }

    func didTapButton(button: UIButton) {
        self.presenter.showGreeting()
    }

    func setGreeting(greeting: String) {
        self.greetingLabel.text = greeting
    }

    // layout code goes here
}
// Assembling of MVP
let model = Person(firstName: "David", lastName: "Blaine")
let view = GreetingViewController()
let presenter = GreetingPresenter(view: view, person: model)
view.presenter = presenter

```

Patrón MVVM

```

import UIKit

struct Person { // Model
    let firstName: String
    let lastName: String
}

protocol GreetingViewModelProtocol: class {

```

```

    var greeting: String? { get }
    var greetingDidChange: ((GreetingViewModelProtocol) -> ())? { get set } // function to
call when greeting did change
    init(person: Person)
    func showGreeting()
}

class GreetingViewModel : GreetingViewModelProtocol {
    let person: Person
    var greeting: String? {
        didSet {
            self.greetingDidChange?(self)
        }
    }
    var greetingDidChange: ((GreetingViewModelProtocol) -> ())?
    required init(person: Person) {
        self.person = person
    }
    func showGreeting() {
        self.greeting = "Hello" + " " + self.person.firstName + " " + self.person.lastName
    }
}

class GreetingViewController : UIViewController {
    var viewModel: GreetingViewModelProtocol! {
        didSet {
            self.viewModel.greetingDidChange = { [unowned self] viewModel in
                self.greetingLabel.text = viewModel.greeting
            }
        }
    }
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self.viewModel, action: "showGreeting",
forControlEvents: .TouchUpInside)
    }
    // layout code goes here
}
// Assembling of MVVM
let model = Person(firstName: "David", lastName: "Blaine")
let viewModel = GreetingViewModel(person: model)
let view = GreetingViewController()
view.viewModel = viewModel

```

Patrón VIPER

```

import UIKit

struct Person { // Entity (usually more complex e.g. NSManagedObject)
    let firstName: String
    let lastName: String
}

struct GreetingData { // Transport data structure (not Entity)
    let greeting: String
    let subject: String
}

```

```

}

protocol GreetingProvider {
    func provideGreetingData()
}

protocol GreetingOutput: class {
    func receiveGreetingData(greetingData: GreetingData)
}

class GreetingInteractor : GreetingProvider {
    weak var output: GreetingOutput!

    func provideGreetingData() {
        let person = Person(firstName: "David", lastName: "Blaine") // usually comes from data
        access layer
        let subject = person.firstName + " " + person.lastName
        let greeting = GreetingData(greeting: "Hello", subject: subject)
        self.output.receiveGreetingData(greeting)
    }
}

protocol GreetingViewEventHandler {
    func didTapShowGreetingButton()
}

protocol GreetingView: class {
    func setGreeting(greeting: String)
}

class GreetingPresenter : GreetingOutput, GreetingViewEventHandler {
    weak var view: GreetingView!
    var greetingProvider: GreetingProvider!

    func didTapShowGreetingButton() {
        self.greetingProvider.provideGreetingData()
    }

    func receiveGreetingData(greetingData: GreetingData) {
        let greeting = greetingData.greeting + " " + greetingData.subject
        self.view.setGreeting(greeting)
    }
}

class GreetingViewController : UIViewController, GreetingView {
    var eventHandler: GreetingViewEventHandler!
    let showGreetingButton = UIButton()
    let greetingLabel = UILabel()

    override func viewDidLoad() {
        super.viewDidLoad()
        self.showGreetingButton.addTarget(self, action: "didTapButton:", forControlEvents:
.TouchUpInside)
    }

    func didTapButton(button: UIButton) {
        self.eventHandler.didTapShowGreetingButton()
    }

    func setGreeting(greeting: String) {
        self.greetingLabel.text = greeting
    }
}

```

```
    }  
  
    // layout code goes here  
}  
// Assembling of VIPER module, without Router  
let view = GreetingViewController()  
let presenter = GreetingPresenter()  
let interactor = GreetingInteractor()  
view.eventHandler = presenter  
presenter.view = view  
presenter.greetingProvider = interactor  
interactor.output = presenter
```

Lea Guía para elegir los mejores patrones de arquitectura iOS en línea:

<https://riptutorial.com/es/ios/topic/10029/guia-para-elegir-los-mejores-patrones-de-arquitectura-ios>

Capítulo 78: Guión gráfico

Introducción

Normalmente, los controladores de vista en un guión gráfico se crean instancias y se crean automáticamente en respuesta a las acciones definidas dentro del mismo guión gráfico. Sin embargo, puede usar un objeto de guión gráfico para crear una instancia del controlador de vista inicial en un archivo de guión gráfico o crear una instancia de otros controladores de vista que desee presentar mediante programación. A continuación encontrará ejemplos de ambos casos de uso.

Examples

Inicializar

```
//Swift
let storyboard = UIStoryboard(name: "Main", bundle: NSBundle.mainBundle())

//Objective-c
UINavigationController *storyboard = [UINavigationController storyboardWithName:@"Main" bundle:[NSBundle mainBundle]];
```

Fetch Initial ViewController

```
//Swift
let mainScreen = storyboard.instantiateInitialViewController()

//Objective-c
UIViewController *initialScreen = [storyboard instantiateInitialViewController];
```

Fetch ViewController

```
//Swift
let viewController = storyboard.instantiateViewControllerWithIdentifier("identifier")

//Objective-c
UIViewController *viewController = [storyboard
instantiateViewControllerWithIdentifier:@"identifier"];
```

Lea Guión gráfico en línea: <https://riptutorial.com/es/ios/topic/3514/guion-grafico>

Capítulo 79: Hacer selectivas esquinas UIView redondeadas.

Examples

Código de objetivo C para hacer la esquina seleccionada de un UIView redondeado

Primero **importe** `#import <QuartzCore/QuartzCore.h>` en su clase `ViewController`. Así es como configuro mi vista en código.

```
UIView *view1=[[UIView alloc]init];
view1.backgroundColor=[UIColor colorWithRed:255/255.0 green:193/255.0 blue:72/255.0
alpha:1.0];
CGRect view1Frame = view1.frame;
view1Frame.size.width = SCREEN_WIDTH*0.97;
view1Frame.size.height = SCREEN_HEIGHT*0.2158;
view1Frame.origin.x = 0;
view1Frame.origin.y = 0.1422*SCREEN_HEIGHT-10;
view1.frame = view1Frame;
[self setMaskTo:view1 byRoundingCorners:UIRectCornerBottomRight|UIRectCornerTopRight];
[self.view addSubview:view1];
```

Aquí está la función que hace el trabajo pesado y redondea los bordes seleccionados, que es el borde inferior derecho y el borde superior derecho en nuestro caso

```
- (void)setMaskTo:(UIView*)view byRoundingCorners:(UIRectCorner)corners
{
    UIBezierPath *rounded = [UIBezierPath bezierPathWithRoundedRect:view.bounds
                                                                    byRoundingCorners:corners
                                                                    cornerRadii:CGSizeMake(20.0, 20.0)];

    CAShapeLayer *shape = [[CAShapeLayer alloc] init];
    [shape setPath:rounded.CGPath];
    view.layer.mask = shape;
}
```

Lea [Hacer selectivas esquinas UIView redondeadas](https://riptutorial.com/es/ios/topic/7224/hacer-selectivas-esquinas-uiview-redondeadas). en línea:

[https://riptutorial.com/es/ios/topic/7224/hacer-selectivas-esquinas-uiview-redondeadas-](https://riptutorial.com/es/ios/topic/7224/hacer-selectivas-esquinas-uiview-redondeadas)

Capítulo 80: iBeacon

Parámetros

Parámetros	Detalles
gerente	Referencia de CLLocationManager
región	CLRegion podría ser una región circular (geofence o región de baliza)
balizas	Array of CLBeacon contiene todas las balizas a distancia

Observaciones

Las balizas son objetos IOT. Nos centramos en aquellos que cumplen con el protocolo iBeacon un estándar de Apple. Cada baliza es un dispositivo de una sola vía que transmite 3 cosas.

1. UUID
2. Mayor
3. Menor

Podemos escanear iBeacons configurando nuestro objeto de administrador de CLLocation para escanear en busca de balizas para UUID en particular. Todas las balizas con el UUID dado serán escaneadas.

El administrador de CLLocation también proporciona llamadas al entrar y salir de la región de baliza.

Examples

Operación básica iBeacon

1. Configurar balizas de monitoreo

```
func initiateRegion(ref:BeaconHandler) {
    let uuid: NSUUID = NSUUID(UUIDString: "<UUID>")
    let beacon = CLBeaconRegion(proximityUUID: uuid, identifier: "")
    locationManager?.requestAlwaysAuthorization() //cllocation manager obj.
    beacon?.notifyOnEntry = true
    beacon?.notifyOnExit = true
    beacon?.notifyEntryStateOnDisplay = true
    locationManager?.startMonitoringForRegion(beacon!)
    locationManager?.delegate = self;
    // Check if beacon monitoring is available for this device
    if (!CLLocationManager.isMonitoringAvailableForClass(CLBeaconRegion)) {
        print("error")
    }
}
```

```
locationManager!.startRangingBeaconsInRegion(self.beacon!)
}
```

2. Gerente de localización entrar y salir de la región

```
func locationManager(manager: CLLocationManager, didEnterRegion region: CLRegion) {
    if(region.isKindOfClass(CLBeaconRegion)) {
        locationManager!.startRangingBeaconsInRegion(self.beacon!)
    }
}

func locationManager(manager: CLLocationManager, didExitRegion region: CLRegion) {
    if(region.isKindOfClass(CLBeaconRegion)) {
        locationManager!.stopRangingBeaconsInRegion(self.beacon!)
    }
}
```

3. Centro de localización gama baliza

```
func locationManager(manager: CLLocationManager, didRangeBeacons beacons: [CLBeacon], inRegion
region: CLBeaconRegion) {
    print(beacons.first.major)
}
```

Escaneando balizas específicas

```
beacon = CLBeaconRegion(proximityUUID: <#NSUUID#>, major: <#CLBeaconMajorValue#>, identifier:
<#String#>) // listening to all beacons with given UUID and major value
beacon = CLBeaconRegion(proximityUUID: <#NSUUID#>, major: <##CLBeaconMajorValue#>, minor:
<##CLBeaconMinorValue#>, identifier: <##String#>) // listening to all beacons with given UUID
and major and minor value
```

Alineando iBeacons

Primero, hay que solicitar la autorización de los servicios de localización.

```
let locationManager = CLLocationManager()
locationManager.delegate = self
locationManager.requestWhenInUseAuthorization()
// OR locationManager.requestAlwaysAuthorization()
```

Entonces puede obtener toda la información de iBeacons dentro de `didRangeBeacons`

```
func locationManager(manager: CLLocationManager, didRangeBeacons beacons: [CLBeacon], inRegion
region: CLBeaconRegion) {
    for beacon in beacons {
        print(beacon.major)
        print(beacon.minor)
    }
}
```

Lea iBeacon en línea: <https://riptutorial.com/es/ios/topic/1958/ibeacon>

Capítulo 81: IBOutlet

Observaciones

IBOutlet no es una palabra reservada ni una variable o clase, es azúcar sintáctica para Interface Builder. Una vez que el código fuente de Objective-C es preprocesado, se resuelve en nada.

En Swift se resuelve como nula.

Se declara en <UIKit/UINibDeclarations.h> como

```
#ifndef IBOutlet
#define IBOutlet
#endif
```

Examples

Usando un IBOutlet en un elemento UI

En general, los IBOutlet se utilizan para conectar un objeto de interfaz de usuario a otro objeto, en este caso un UIViewController. La conexión sirve para permitir que el objeto se vea afectado por mi código o eventos programáticamente. Esto se puede hacer simplemente usando el asistente de un guión gráfico y haciendo clic con el botón de control del elemento en la sección de propiedad .h del controlador de vista, pero también se puede hacer mediante la conexión manual y manual del código IBOutlet a la pestaña "conexiones" del objeto. La barra de servicios a la derecha. Aquí hay un ejemplo de objetivo-c de un UIViewController con una salida de etiqueta:

```
//ViewController.h
#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

//This is the declaration of the outlet
@property (nonatomic, weak) IBOutlet UILabel *myLabel;

@end

//ViewController.m
#import "ViewController.h"

@implementation ViewController

@synthesize myLabel;

-(void) viewDidLoad {

    [super viewDidLoad];
    //Editing the properties of the outlet
    myLabel.text = @"TextHere";
}
```

```
}  
  
@end
```

Y rápido:

```
import UIKit  
class ViewController: UIViewController {  
    //This is the declaration of the outlet  
    @IBOutlet weak var myLabel: UILabel!  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        //Editing the properties of the outlet  
        myLabel.text = "TextHere"  
    }  
}
```

La conexión entre el objeto del guión gráfico y el objeto programado se puede verificar como conectado si se rellena el punto a la izquierda de la declaración de la salida en el .h. Un círculo vacío implicaba una conexión incompleta.

Lea IBOutlets en línea: <https://riptutorial.com/es/ios/topic/4713/iboutlets>

Capítulo 82: Imágenes de caché en línea

Examples

AlamofireImage

Imágenes de caché en línea utilizando AlamofireImage . Funciona en la parte superior de Alamofire en Swift. Instala AlamofireImage utilizando cocoapods

```
pod 'AlamofireImage', '~> 3.1'
```

Preparar:

1. Importar AlamofireImage y Alamofire
2. Configure la memoria caché de la imagen: `let imageCache = AutoPurgingImageCache(memoryCapacity: 111_111_111, preferredMemoryUsageAfterPurge: 90_000_000)`
3. Hacer una solicitud y agregar la imagen a la caché:

```
Alamofire.request(self.nameUrl[i]).responseImage { response in
    if response.result.value != nil {
        let image = UIImage(data: response.data!, scale: 1.0)!
        imageCache.add(image, withIdentifier: self.nameUrl[i])
    }
}
```

4. Recuperar imágenes de caché:

```
if let image = imageCache.image(withIdentifier: self.nameUrl[self.a])
{
    self.localImageView.image = image
}
```

Para más información siga [este enlace](#).

Lea Imágenes de caché en línea en línea: <https://riptutorial.com/es/ios/topic/9450/imagenes-de-cache-en-linea>

Capítulo 83: Instantánea de UIView

Examples

Obtención de la instantánea

```
- (UIImage *)getSnapshot
{
    UIScreen *screen = [UIScreen mainScreen];
    CGRect bounds = [self.view bounds];
    UIGraphicsBeginImageContextWithOptions(bounds.size, false, screen.scale);
    CGContextRef context = UIGraphicsGetCurrentContext();
    CGContextSetInterpolationQuality(context, kCGInterpolationHigh);
    [self.view drawViewHierarchyInRect:bounds afterScreenUpdates:YES];
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return image;
}
```

Rápido

```
var screenshot: UIImage
{
    UIGraphicsBeginImageContext(self.bounds.size);
    let context = UIGraphicsGetCurrentContext();
    self.layer.render(in: context)
    let screenShot = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return screenShot
}
```

Instantánea con subvista con otro marcado y texto.

- Soporta retrato y paisaje tanto tipo de imagen
- El dibujo y otras subvistas se pueden combinar en mi caso. Estoy agregando una etiqueta al dibujo.

```
{
    CGSize fullSize = getImageForEdit.size;
    CGSize sizeInView = AVMakeRectWithAspectRatioInsideRect(imgViewFake.image.size,
imgViewFake.bounds).size;
    CGFloat orgScale = orgScale = fullSize.width/sizeInView.width;
    CGSize newSize = CGSizeMake(orgScale * img.image.size.width, orgScale *
img.image.size.height);
    if(newSize.width <= fullSize.width && newSize.height <= fullSize.height){
        newSize = fullSize;
    }
    CGRect offsetRect;
    if (getImageForEdit.size.height > getImageForEdit.size.width){
        CGFloat scale = newSize.height/fullSize.height;
        CGFloat offset = (newSize.width - fullSize.width*scale)/2;
        offsetRect = CGRectMake(offset, 0, newSize.width-offset*2, newSize.height);
    }
}
```

```
    }
    else{
        CGFloat scale = newSize.width/fullSize.width;
        CGFloat offset = (newSize.height - fullSize.height*scale)/2;
        offsetRect = CGRectMake(0, offset, newSize.width, newSize.height-offset*2);
    }
    UIGraphicsBeginImageContextWithOptions(newSize, NO, getImageForEdit.scale);
    [getImageForEdit drawAtPoint:offsetRect.origin];
    // [img.image drawInRect:CGRectMake(0,0,newSize.width,newSize.height)];
    CGFloat oldScale = img.contentScaleFactor;
    img.contentScaleFactor = getImageForEdit.scale;
    [img drawViewHierarchyInRect:CGRectMake(0, 0, newSize.width, newSize.height)
    afterScreenUpdates:YES];
    img.contentScaleFactor = oldScale;
    UIImage *combImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    imageData = UIImageJPEGRepresentation(combImage, 1);
}
```

Lea Instantánea de UIView en línea: <https://riptutorial.com/es/ios/topic/4622/instantanea-de-UIView>

Capítulo 84: Integración SqlCipher

Introducción

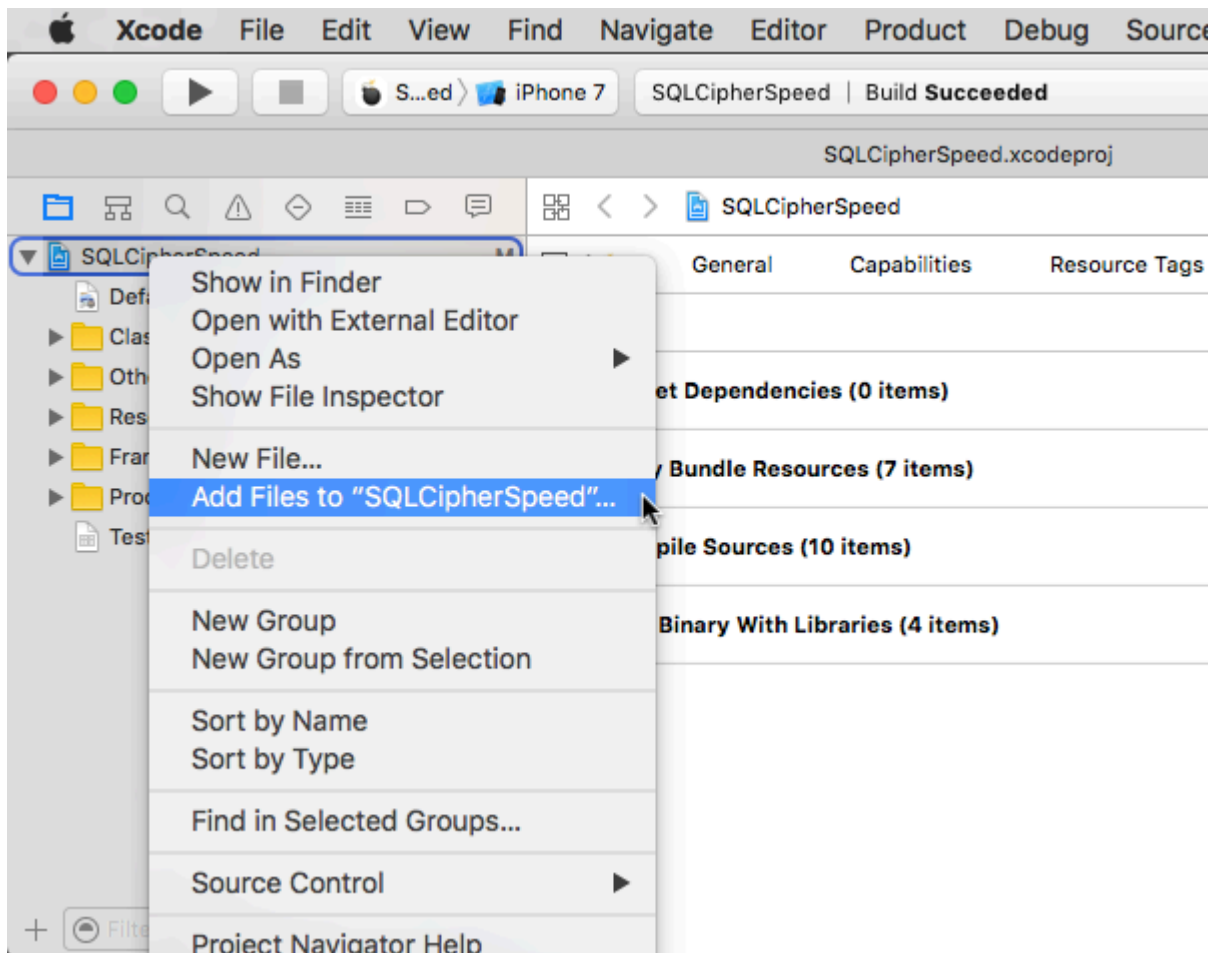
SQLite ya es una API popular para el almacenamiento de datos persistentes en aplicaciones iOS, por lo que la ventaja para el desarrollo es obvia. Como programador, trabajas con una API estable y bien documentada que tiene muchos envoltorios buenos disponibles en Objective-C, como FMDB y datos encriptados. Todos los problemas de seguridad están claramente desconectados del código de la aplicación y son administrados por el marco subyacente.

Observaciones

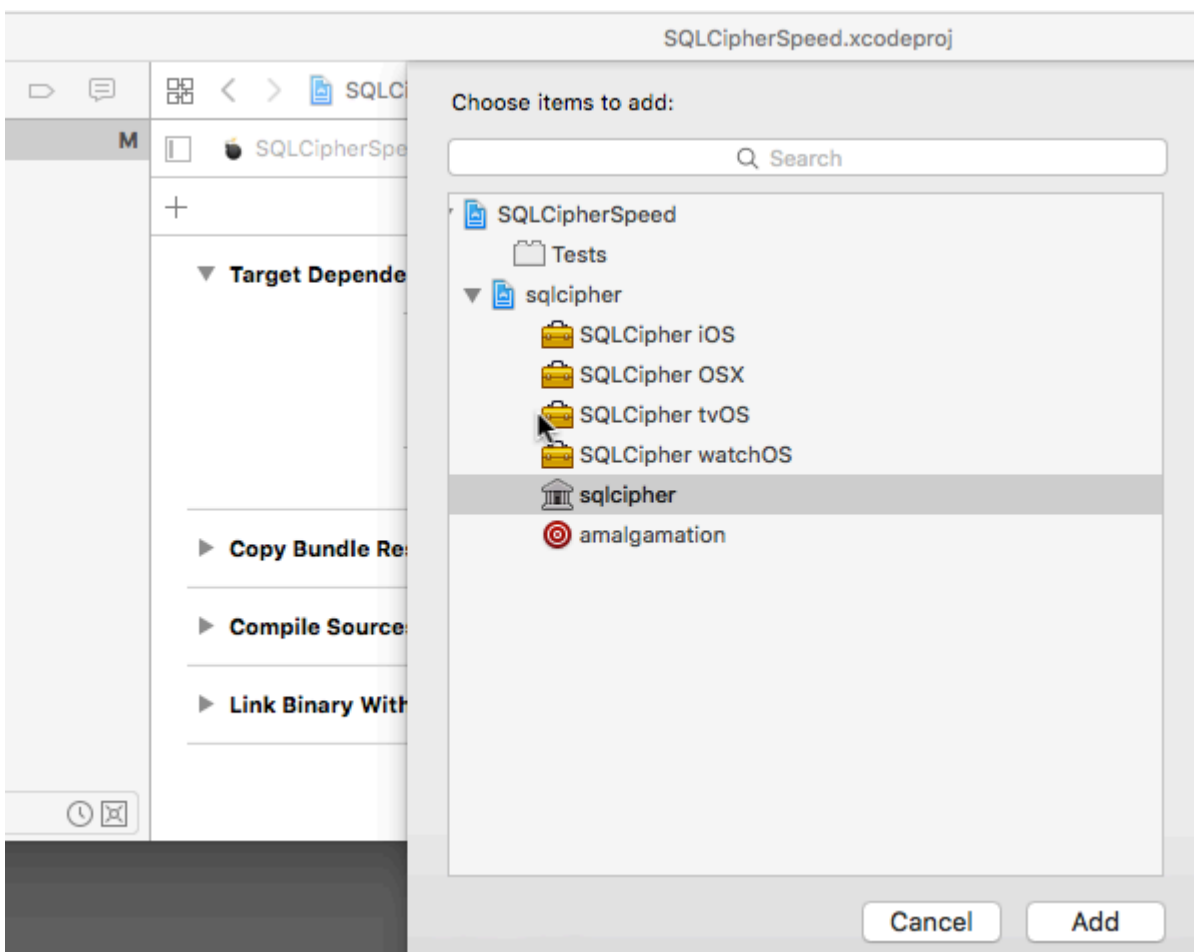
1. Abra la Terminal, cambie al directorio raíz de su proyecto y verifique el código del proyecto SQLCipher usando Git:

```
$ git clone https://github.com/sqlcipher/sqlcipher.git
```

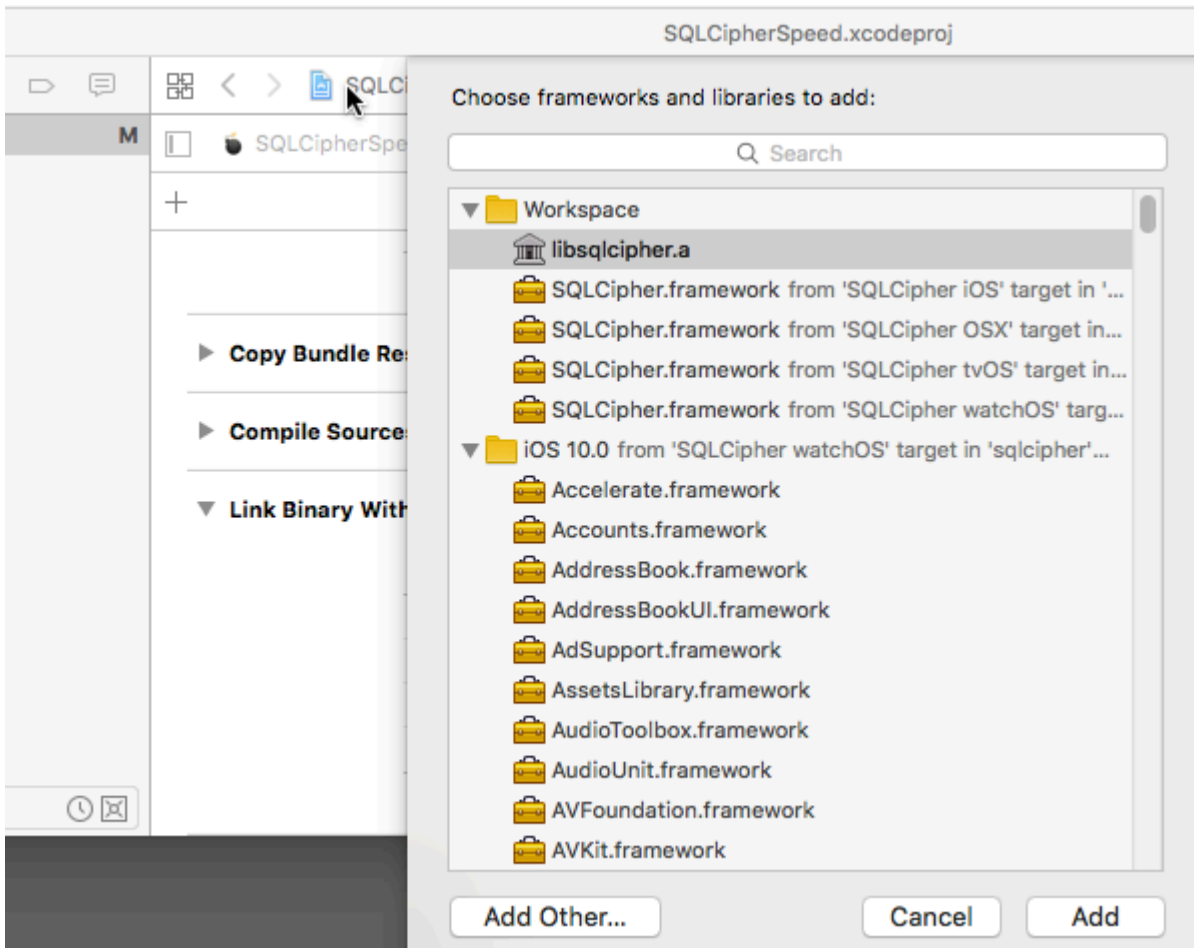
2. Haga clic derecho en el proyecto y elija "Agregar archivos a" Mi aplicación "" (la etiqueta variará según el nombre de su aplicación). Ya que clonamos SQLCipher directamente en la misma carpeta que su aplicación iOS, debería ver una carpeta sqlcipher en su carpeta de proyecto raíz. Abra esta carpeta y seleccione **sqlcipher.xcodeproj**



3. Seleccione el panel Crear configuración. En el campo de búsqueda, escriba "Rutas de búsqueda de encabezado". Haga doble clic en el campo debajo de la columna de destino y agregue la siguiente ruta: **\$ (PROJECT_DIR) / sqlcipher / src**
4. Comience a escribir "Otros indicadores de vinculador" en el campo de búsqueda hasta que aparezca la configuración, haga doble clic para editarlo y agregue el siguiente valor: **\$ (BUILT_PRODUCTS_DIR) /libsqcipher.a**
5. Comience a escribir "Otros indicadores C" en el campo de búsqueda hasta que aparezca la configuración, haga doble clic para editarlo y, en la **ventana** emergente, agregue el siguiente valor: **-DSQLITE_HAS_CODEC**
6. Expanda las Dependencias de destino y haga clic en el botón + al final de la lista. En el navegador que se abre, seleccione el **destino de** la biblioteca estática **sqlcipher** :



7. Expanda Enlace binario con bibliotecas, haga clic en el botón + al final de la lista y seleccione la biblioteca **libsqcipher.a** .



8. Finalmente, también en Enlace con bibliotecas, agregue **Security.framework** .

Examples

Integración de código:

Integración para abrir la base de datos mediante contraseña.

```

- (void) checkAndOpenDB{
    sqlite3 *db;
    NSString *strPassword = @"password";

    if (sqlite3_open_v2([[databaseURL path] UTF8String], &db, SQLITE_OPEN_READWRITE |
    SQLITE_OPEN_CREATE, NULL) == SQLITE_OK) {
        const char* key = [strPassword UTF8String];
        sqlite3_key(db, key, (int)strlen(key));
        if (sqlite3_exec(db1, (const char*) "SELECT count(*) FROM sqlite_master;", NULL,
        NULL, NULL) == SQLITE_OK) {
            NSLog(@"Password is correct, or a new database has been initialized");
        } else {
            NSLog(@"Incorrect password!");
        }
        sqlite3_close(db);
    }
}

- (NSURL *)databaseURL
{

```



```
NSArray *URLs = [[NSFileManager defaultManager] URLsForDirectory:NSDocumentDirectory
inDomains:NSUserDomainMask];
NSURL *directoryURL = [URLs firstObject];
NSURL *databaseURL = [directoryURL URLByAppendingPathComponent:@"database.sqlite"];
return databaseURL;
}
```

Lea Integración SqlCipher en línea: <https://riptutorial.com/es/ios/topic/9969/integracion-sqlcipher>

Capítulo 85: Interoperabilidad rápida y objetiva-C

Examples

Usando clases de Objective-C en Swift

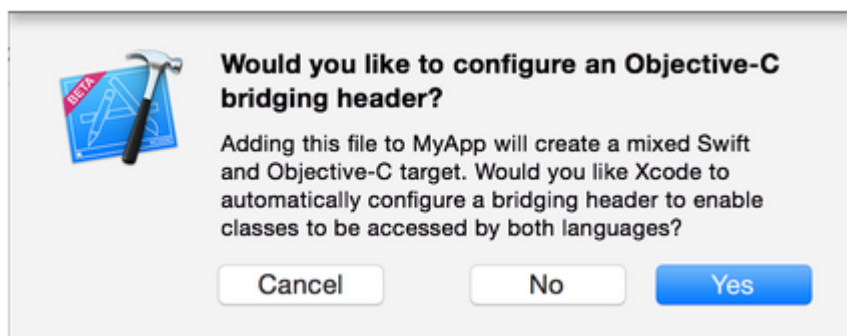
Si tiene una clase existente que le gustaría usar, realice el **Paso 2** y luego salte al **Paso 5**. (Para algunos casos, tuve que agregar un `#import <Foundation/Foundation.h` explícito a un archivo ObjC más antiguo)

Paso 1: Agregar implementación de Objective-C - .m

Agregue un archivo `.m` a su clase y `CustomObject.m` nombre `CustomObject.m`

Paso 2: Añadir encabezado puente

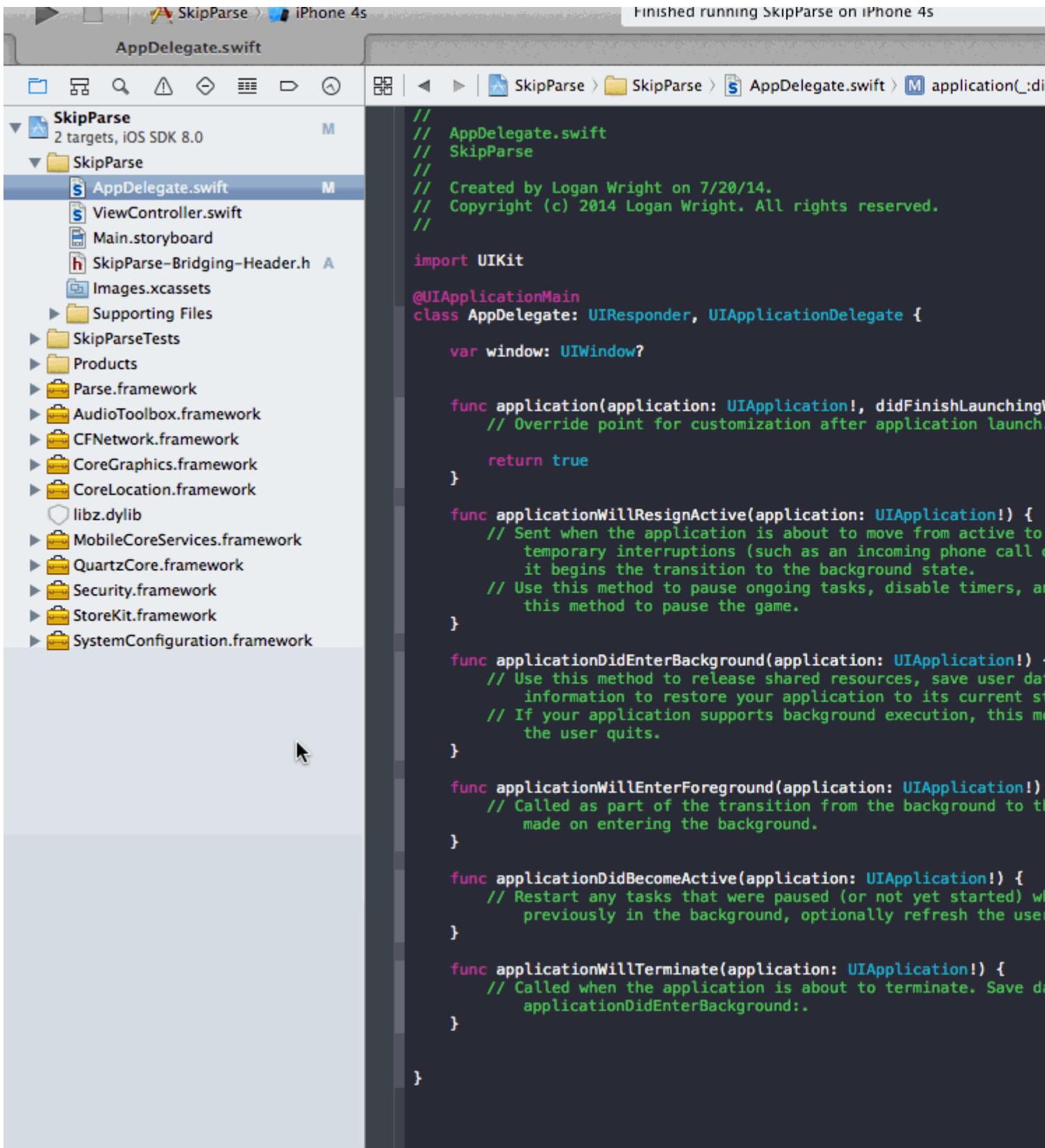
Cuando agregue su archivo `.m`, es probable que reciba un mensaje que se parece a esto:



Haga clic en **SÍ** !

Si no vio el indicador o borró accidentalmente el encabezado de puente, agregue un nuevo archivo `.h` a su proyecto y `<#YourProjectName#>-Bridging-Header.h` nombre `<#YourProjectName#>-Bridging-Header.h`

En algunas situaciones, especialmente cuando se trabaja con marcos ObjC, no agrega una clase Objective-C explícitamente y Xcode no puede encontrar el enlazador. En este caso, cree su archivo `.h` el nombre mencionado anteriormente, luego asegúrese de vincular su ruta en la configuración del proyecto de su objetivo de esta manera:



Nota

Es una buena práctica vincular su proyecto usando la macro `$(SRCROOT)` para que, si mueve su proyecto, o trabaje en él con otros usando un repositorio remoto, aún funcione. `$(SRCROOT)` se puede considerar como el directorio que contiene su archivo `.xcodeproj`. Podría verse así:

```
$(SRCROOT)/Folder/Folder/<#YourProjectName#>-Bridging-Header.h
```

Paso 3: Añadir encabezado Objective-C - .h

Agregue otro archivo `.h` y `CustomObject.h` nombre `CustomObject.h`

Paso 4: Construye tu clase de Objective-C

En `CustomObject.h`

```
#import <Foundation/Foundation.h>

@interface CustomObject : NSObject

@property (strong, nonatomic) id someProperty;

- (void) someMethod;

@end
```

En `CustomObject.m`

```
#import "CustomObject.h"

@implementation CustomObject

- (void) someMethod {
    NSLog(@"SomeMethod Ran");
}

@end
```

Paso 5: Agregar clase a Bridging-Header

En `YourProject-Bridging-Header.h`:

```
#import "CustomObject.h"
```

Paso 6: Usa tu objeto

En `SomeSwiftFile.swift`:

```
var instanceOfCustomObject: CustomObject = CustomObject()
instanceOfCustomObject.someProperty = "Hello World"
println(instanceOfCustomObject.someProperty)
instanceOfCustomObject.someMethod()
```

No es necesario importar explícitamente, para eso sirve el encabezado de puente.

Usando clases rápidas en Objective-C

Paso 1: Crear nueva clase Swift

Agregue un archivo `.swift` a su proyecto y `MySwiftObject.swift` nombre `MySwiftObject.swift`

En `MySwiftObject.swift` :

```
import Foundation

class MySwiftObject : NSObject {

    var someProperty: AnyObject = "Some Initializer Val"

    init() {}

    func someFunction(someArg:AnyObject) -> String {
        var returnVal = "You sent me \(someArg)"
        return returnVal
    }

}
```

Paso 2: Importar archivos Swift a la clase ObjC

En `SomeRandomClass.m` :

```
#import "<#YourProjectName>-Swift.h"
```

El archivo: `<#YourProjectName>-Swift.h` ya debe crearse automáticamente en su proyecto, incluso si no puede verlo.

Paso 3: Usa tu clase

```
MySwiftObject * myOb = [MySwiftObject new];
NSLog(@"MyOb.someProperty: %@", myOb.someProperty);
myOb.someProperty = @"Hello World";
NSLog(@"MyOb.someProperty: %@", myOb.someProperty);
NSString * retString = [myOb someFunction:@"Arg"];
NSLog(@"RetString: %@", retString);
```

Nota:

1. CodeCompletion no se estaba comportando con tanta precisión como me gustaría. En mi sistema, ejecutar una compilación rápida con "cmd + r" pareció ayudar a Swift a encontrar parte del código Objc y viceversa.
2. Si agrega el archivo `.swift` a un proyecto anterior y obtiene un error: `dyld: Library not loaded: @rpath/libswift_stdlib_core.dylib` , intente [reiniciar](#) completamente [Xcode](#).
3. Aunque originalmente era posible usar clases Swift puras en Objective-C usando el prefijo `@objc` , después de Swift 2.0, esto ya no es posible. Ver historial de edición para la explicación original. Si esta funcionalidad se vuelve a habilitar en futuras versiones de Swift, la respuesta se actualizará en consecuencia.

Lea Interoperabilidad rápida y objetiva-C en línea:

<https://riptutorial.com/es/ios/topic/1497/interoperabilidad-rapida-y-objetiva-c>

Capítulo 86: iOS - Implementación de XMPP con el framework Robbie Hanson

Examples

Ejemplo de iOS XMPP Robbie Hanson con Openfire

SRXMPPDemo

Descargue el ejemplo y todas las clases aquí - <https://github.com/SahebRoy92/SRXMPPDemo>

Una demostración de XMPP en Objective C, con varias características simples y complejas implementadas en él. Todas las características de XMPP se realizan mediante funciones xmpp "en banda". Pocas características de este proyecto son:

SRXMPP : una envoltura de clase Singleton que casi tiene todas las funciones necesarias para la aplicación de chat uno a uno.

- chat individual
- Implementación de datos básicos del chat (mensaje de texto), con lo que se guardan mensajes anteriores, mensajes fuera de línea.
- implementación de vCard (información de perfil del usuario, propia y de otros también) a partir de XML y Core Data proporcionados por el propio marco de Robbie Hanson.
- Disponibilidad de estado de amigos (en línea / fuera de línea / escribiendo)

Pasos a seguir

Si desea utilizar este proyecto como referencia, puede hacer lo siguiente:

1. Instale Openfire en un servidor activo: alquile un servidor, instale openfire.

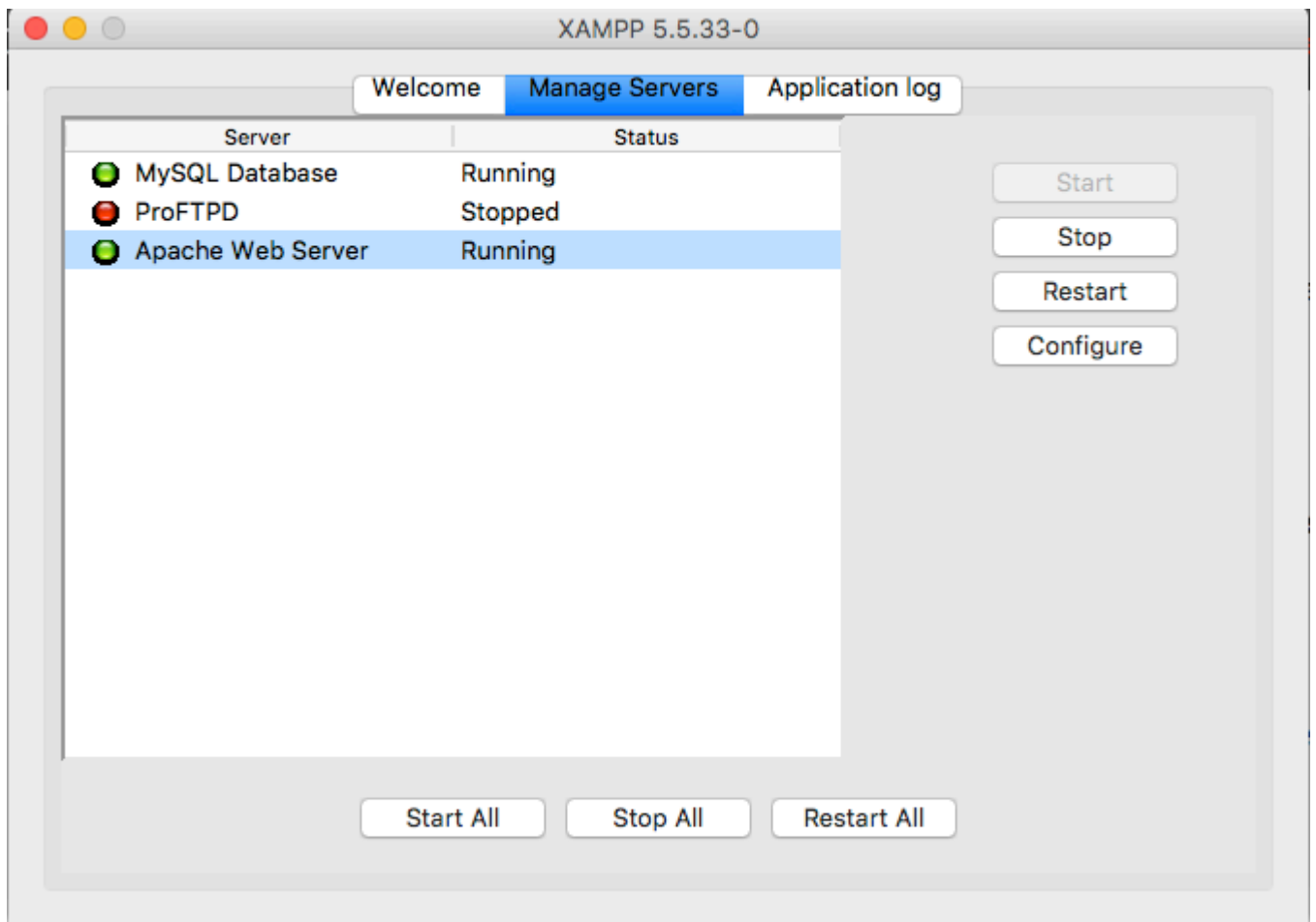
2. Desea probarlo sin problemas en su propia computadora : debe descargar, instalar y configurar 3 cosas para comenzar

a. Java -

- Descarga e instala Java para Mac.

segundo. XAMPP -

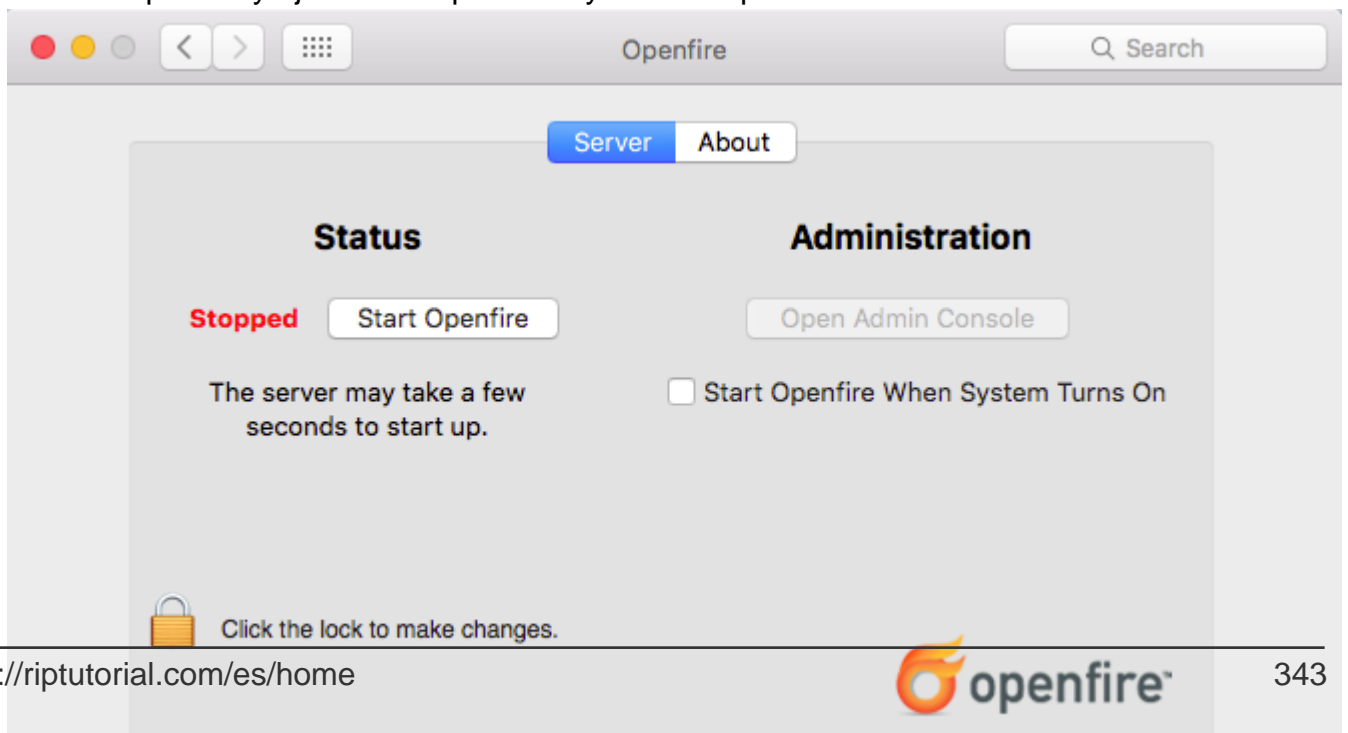
- Instalar XAMPP es relativamente fácil.
- Después de la instalación, simplemente inicie XAMPP e inicie **Database (SQL)** y **Apache Server** .



- Luego abra el navegador y pegue esta URL [<http://localhost/phpmyadmin/>]
- . Crea una nueva base de datos desde el panel lateral izquierdo.
- **Dale un nombre al DB pero recuerda este nombre, supongamos que lo llamamos ChatDB**

do. Fuego abierto

- Instalar Openfire y ejecutar la aplicación y "Iniciar Openfire"



- Abra el navegador y pegue esta URL - [<http://localhost:9090/setup/index.jsp>] (<http://localhost:9090/setup/index.jsp>)
- Hacer la configuración normal
 - Seleccionar idioma>
 - Configuración del servidor, deje como está, solo continúe>
 - Configuración de la base de datos, deje como está como "Conexión estándar a la base de datos seleccionada">
 - Configuración de la base de datos - Conexión estándar ". Ahora recuerde que el nombre de la base de datos que estableció fue **ChatDB** .
 - Seleccione Presets del controlador de base de datos como * " *MySQL* ". Deje JDBC Driver Class como es. Ahora en la URL de la base de datos puede ver, los corchetes mencionan el nombre de host y el nombre de la base de datos. Simplemente cambie el Nombre de host a "**localhost**" , y el nombre de la base de datos a "**ChatDB**" , o cualquier otro nombre de DB que haya establecido anteriormente, mientras configura XAMPP. Deje el nombre de usuario y la contraseña en blanco. Complete detalles como la imagen aquí

Database Settings - Standard Connection

Specify a JDBC driver and connection properties to connect to your database. If you need more information about this pro

Note: Database scripts for most popular databases are included in the server distribution at [Openfire_HOME]/resc

The screenshot shows the 'Database Settings - Standard Connection' configuration page. It contains the following fields and values:

- Database Driver Presets:
- JDBC Driver Class:
- Database URL:
- Username:
- Password:
- Minimum Connections:
- Maximum Connections:
- Connection Timeout: Days

- Luego complete la configuración dando un nombre de usuario y contraseña y reconfirmándolo. Eso es todo lo que has hecho Configuración de Openfire.

Ahora viene la parte cuando tienes que cambiar un pequeño detalle en el código.

Importante Necesitamos ir a la clase - **SRXMPP.m** , localizar el SRXMPP_Hostname externo de **NSString** (en la parte superior) y sobrescribir el valor de este a la

- IP del servidor donde está instalado OpenFire, **O**
- si lo ha instalado localmente, sobrescriba el valor a - "**localhost**" .

Eso es todo, estás listo para usar este proyecto de ejemplo y comenzar a codificar y convertirlo en un mejor proyecto por tu cuenta.

Este paquete de inicio lo ayudará a comprender mejor la estructura de XMPP, así como a familiarizarse con los protocolos XMPP.

Puede encontrar otros protocolos XMPP aquí en este sitio: [<https://xmpp.org/rfcs/rfc3920.html>]
◆(<https://xmpp.org/rfcs/rfc3920.html>)

Aún queda desarrollo y partes donde espero incluirlas más adelante.

1. Grupo de chat
2. Soporte de envío de imágenes.

En resumen, este proyecto de ejemplo, junto con el singleton, tiene casi todas las características que se necesitan para que tenga una aplicación de chat One-to-One.

Lea iOS - Implementación de XMPP con el framework Robbie Hanson en línea:
<https://riptutorial.com/es/ios/topic/1475/ios---implementacion-de-xmpp-con-el-framework-robbie-hanson>

Capítulo 87: iOS TTS

Introducción

Encuentre cómo producir voz sintetizada a partir de texto en un dispositivo iOS

Examples

Texto a voz

C objetivo

```
AVSpeechSynthesizer *synthesizer = [[AVSpeechSynthesizer alloc] init];
AVSpeechUtterance *utterance = [AVSpeechUtterance speechUtteranceWithString:@"Some text"];
[utterance setRate:0.2f];
[synthesizer speakUtterance:utterance];
```

Rápido

```
let synthesizer = AVSpeechSynthesizer()
let utterance = AVSpeechUtterance(string: "Some text")
utterance.rate = 0.2
```

También puedes cambiar la voz de esta manera:

```
utterance.voice = AVSpeechSynthesisVoice(language: "fr-FR")
```

Y luego speak

- En Swift 2: `synthesizer.speakUtterance(utterance)`
- En Swift 3: `synthesizer.speak(utterance)`

No te olvides de importar AVFoundation

Métodos útiles

Puede detener o pausar todo el habla usando estos dos métodos:

```
- (BOOL)pauseSpeakingAtBoundary:(AVSpeechBoundary)boundary;
- (BOOL)stopSpeakingAtBoundary:(AVSpeechBoundary)boundary;
```

`AVSpeechBoundary` indica si la voz debe `AVSpeechBoundaryImmediate` o detenerse inmediatamente (`AVSpeechBoundaryImmediate`) o debe `AVSpeechBoundaryImmediate` o detenerse después de la palabra

hablada actualmente (`AVSpeechBoundaryWord`).

Lea iOS TTS en línea: <https://riptutorial.com/es/ios/topic/8909/ios-tts>

Capítulo 88: Kit de juego

Examples

Generando numeros al azar

Aunque `GameplayKit` (que se presenta con iOS 9 SDK) trata de implementar la lógica del juego, también se puede usar para generar números aleatorios, lo cual es muy útil en aplicaciones y juegos.

Además del `GKRandomSource.sharedRandom` que se usa en los siguientes capítulos, hay tres tipos adicionales de `GKRandomSource` 's out of the box.

- **GKARC4RandomSource** que utiliza el algoritmo ARC4
- **GKLinearCongruentialRandomSource** ¿Qué es un rápido pero no tan al azar `GKRandomSource`
- **GKMersenneTwisterRandomSource** que implementa un algoritmo MersenneTwister. Es más lento pero más aleatorio.

En el siguiente capítulo solo usamos el método `nextInt()` de un `GKRandomSource`. Además de esto hay el `nextBool() -> Bool` y el `nextUniform() -> Float`

Generacion

Primero, importa `GameplayKit`:

Rápido

```
import GameplayKit
```

C objetivo

```
#import <GameplayKit/GameplayKit.h>
```

Luego, para generar un número aleatorio, usa este código:

Rápido

```
let randomNumber = GKRandomSource.sharedRandom().nextInt()
```

C objetivo

```
int randomNumber = [[GKRandomSource sharedRandom] nextInt];
```

Nota

La función `nextInt()`, cuando se usa sin parámetros, devolverá un número aleatorio entre -2,147,483,648 y 2,147,483,647, incluidos ellos mismos, por lo que no estamos seguros de que siempre sea un número positivo o distinto de cero.

Generando un número de 0 a n

Para lograr esto, debe `nextIntWithUpperBound(n)` al método `nextIntWithUpperBound()` :

Rápido

```
let randomNumber = GKRandomSource.sharedRandom().nextInt(upperBound: 10)
```

C objetivo

```
int randomNumber = [[GKRandomSource sharedRandom] nextIntWithUpperBound: 10];
```

Este código nos dará un número entre 0 y 10, incluidos ellos mismos.

Generando un número de ma n

Para hacer esto, creas un objeto `GKRandomDistribution` con un `GKRandomSource` y pasas los límites. Se puede `GKRandomDistribution` una `GKRandomDistribution` para cambiar el comportamiento de distribución, como `GKGaussianDistribution` o `GKShuffledDistribution` .

Después de eso, el objeto se puede usar como cualquier `GKRandomSource` regular, ya que también implementa el protocolo `GKRandom` .

Rápido

```
let randomizer = GKRandomDistribution(randomSource: GKRandomSource(), lowestValue: 0, highestValue: 6)
let randomNumberInBounds = randomizer.nextInt()
```

Objective-C *obsoleto*

```
int randomNumber = [[GKRandomSource sharedRandom] nextIntWithUpperBound: n - m] + m;
```

Por ejemplo, para generar un número aleatorio entre 3 y 10, utiliza este código:

Rápido

```
let randomNumber = GKRandomSource.sharedRandom().nextInt(upperBound: 7) + 3
```

Objective-C *obsoleto*

```
int randomNumber = [[GKRandomSource sharedRandom] nextIntWithUpperBound: 7] + 3;
```

GKEntity y GKComponent

Una entidad representa un objeto de un juego como una figura de jugador o una figura enemiga. Como este objeto no hace mucho sin brazos y piernas, podemos agregarle los componentes.

Para crear este sistema, Apple tiene las clases `GKEntity` y `GKComponent`.

Asumamos que tenemos la siguiente clasificación para los siguientes capítulos:

```
class Player: GKEntity{}
class PlayerSpriteComponent: GKComponent {}
```

GKEntity

Una entidad es una colección de componentes y ofrece varias funciones para agregar, eliminar e interactuar con los componentes de la misma.

Si bien podríamos usar `GKEntity`, es común Subclase para un tipo específico de entidad de juego.

Es importante que solo sea posible agregar un componente de una clase una vez. En caso de que agregue un segundo componente de la misma clase, se anulará el primer componente existente dentro de `GKEntity`

```
let otherComponent = PlayerSpriteComponent()
var player = Player()
player.addComponent(PlayerSpriteComponent())
player.addComponent(otherComponent)
print(player.components.count) //will print 1
print(player.components[0] === otherComponent) // will print true
```

Puedes preguntar por qué. La razón de esto es los métodos llamados `component(for: T.Type)` que devuelve el componente de un tipo específico de la entidad.

```
let component = player.component(ofType: PlayerSpriteComponent.self)
```

Además de los componentes-métodos, tiene un método de `update` que se utiliza para delegar la

hora delta o la hora actual de la lógica del juego a sus componentes.

```
var player = Player()
player.addComponent(PlayerSpriteComponent())
player.update(deltaTime: 1.0) // will call the update method of the PlayerSpriteComponent
added to it
```

GKComponent

Un componente representa algo de una entidad, por ejemplo, el componente visual o el componente lógico.

Si se llama a un método de actualización de una entidad, se lo delegará a todos sus componentes. La anulación de este método se utiliza para manipular una entidad.

```
class PlayerSpriteComponent: GKComponent {
    override func update(deltaTime seconds: TimeInterval) {
        //move the sprite depending on the update time
    }
}
```

Además de esto, es posible anular el método `didAddToEntity` y `willRemoveFromEntity` para informar a otros componentes sobre su eliminación o adición.

Para manipular otro componente dentro de un componente, es posible obtener la `GKEntity` a la que se agrega el componente.

```
override func update(deltaTime seconds: TimeInterval) {
    let controller = self.entity?.component(ofType: PlayerControlComponent.self)
    //call methods on the controller
}
```

Aunque esto es posible, **no** es un patrón común, ya que los alambres de los dos componentes juntos.

GKComponentSystem

Si bien acabamos de hablar sobre el uso del mecanismo de actualización de delegados de `GKEntity` para actualizar los `GKComponents` hay una forma diferente de actualizar `GKComponents` que se llama `GKComponentSystem`.

Se utiliza en caso de que sea necesario que todos los componentes de un tipo específico se actualicen de una sola vez.

Se `GKComponentSystem` un `GKComponentSystem` para un tipo específico de componente.

```
let system = GKComponentSystem(componentClass: PlayerSpriteComponent.self)
```


Para agregar un componente puedes usar el método add:

```
system.addComponent (PlayerSpriteComponent ())
```

Pero una forma más común es pasar la entidad creada con sus componentes al `GKComponentSystem` y encontrará un componente coincidente dentro de la entidad.

```
system.addComponent (foundIn: player)
```

Para actualizar todos los componentes de un tipo específico, llame a la actualización:

```
system.update (deltaTime: delta)
```

En caso de que desee utilizar `GKComponentSystem` lugar de un mecanismo de actualización basado en entidades, debe tener un `GKComponentSystem` para cada componente y llamar a la actualización en todos los sistemas.

Lea Kit de juego en línea: <https://riptutorial.com/es/ios/topic/4966/kit-de-juego>

Capítulo 89: Kit de salud

Examples

Kit de salud

C objetivo

Primero ve a `Target->Capabilities` y habilita `HealthKit`. Esto configuraría la entrada `info.plist`.

Cree una nueva `CocoaClass` de `CocoaClass` del tipo `NSObject`. El nombre de archivo que di es `GSHealthKitManager` y el archivo de encabezado es el que se muestra a continuación

GSHealthKitManager.h

```
#import <Foundation/Foundation.h>
#import <HealthKit/HealthKit.h>
@interface GSHealthKitManager : NSObject

+ (GSHealthKitManager *)sharedManager;

- (void)requestAuthorization;

- (NSDate *)readBirthDate;
- (void)writeWeightSample:(double)weight;
- (NSString *)readGender;

@end
```

GSHealthKitManager.m

```
#import "GSHealthKitManager.h"
#import <HealthKit/HealthKit.h>

@interface GSHealthKitManager ()

@property (nonatomic, retain) HKHealthStore *healthStore;

@end

@implementation GSHealthKitManager

+ (GSHealthKitManager *)sharedManager {
    static dispatch_once_t pred = 0;
    static GSHealthKitManager *instance = nil;
    dispatch_once(&pred, ^{
        instance = [[GSHealthKitManager alloc] init];
        instance.healthStore = [[HKHealthStore alloc] init];
    });
    return instance;
}
```

```

- (void)requestAuthorization {

    if ([HKHealthStore isHealthDataAvailable] == NO) {
        // If our device doesn't support HealthKit -> return.
        return;
    }

    NSArray *readTypes = @[
        [HKObjectType
        characteristicTypeForIdentifier:HKCharacteristicTypeIdentifierDateOfBirth],
        [HKObjectType
        characteristicTypeForIdentifier:HKCharacteristicTypeIdentifierBiologicalSex]];

    [self.healthStore requestAuthorizationToShareTypes:nil readTypes:[NSSet
    setWithArray:readTypes] completion:nil];
}

- (NSDate *)readBirthDate {
    NSError *error;
    NSDate *dateOfBirth = [self.healthStore dateOfBirthWithError:&error]; // Convenience
    method of HKHealthStore to get date of birth directly.

    if (!dateOfBirth) {
        NSLog(@"Either an error occurred fetching the user's age information or none has been
        stored yet. In your app, try to handle this gracefully.");
    }

    return dateOfBirth;
}

- (NSString *)readGender
{
    NSError *error;
    HKBiologicalSexObject *gen=[self.healthStore biologicalSexWithError:&error];
    if (gen.biologicalSex==HKBiologicalSexMale)
    {
        return(@"Male");
    }
    else if (gen.biologicalSex==HKBiologicalSexFemale)
    {
        return(@"Female");
    }
    else if (gen.biologicalSex==HKBiologicalSexOther)
    {
        return(@"Other");
    }
    else{
        return(@"Not Set");
    }
}

@end

```

Llamando desde ViewController

```

- (IBAction)pressed:(id)sender {

    [[GSHealthKitManager sharedManager] requestAuthorization];
    NSDate *birthDate = [[GSHealthKitManager sharedManager] readBirthDate];
}

```

```
NSLog(@"birthdate %@", birthDate);
NSLog(@"gender 2131321 %@", [[GSHealthKitManager sharedManager] readGender]);

}
```

Salida de registro

```
2016-10-13 14:41:39.568 random[778:26371] birthdate 1992-11-29 18:30:00 +0000
2016-10-13 14:41:39.570 random[778:26371] gender 2131321 Male
```

Lea Kit de salud en línea: <https://riptutorial.com/es/ios/topic/7412/kit-de-salud>

Capítulo 90: Llaverero

Sintaxis

- `kSecClassGenericPassword` // Una clave de valor que representa una contraseña que no es de Internet
- `kSecClassInternetPassword` // Una clave de valor que representa una contraseña de Internet
- `kSecClassCertificate` // Una clave de valor que representa un certificado
- `kSecClassCertificate` // Una clave de valor que representa una clave
- `kSecClassIdentity` // Una clave de valor que representa una identidad, que es un certificado más una clave

Observaciones

iOS almacena información privada como contraseñas, claves de cifrado, certificados e identidades en un área de almacenamiento seguro llamada Llaverero. Esta área de almacenamiento es administrada completamente por un co-procesador llamado Secure Enclave, que está integrado dentro del procesador de la aplicación. Debido a que el llaverero se encuentra en un espacio aislado en iOS, los elementos del llaverero solo pueden ser recuperados por la aplicación que los colocó allí en primer lugar.

En algunos casos, debe activar el uso compartido de llaves en Xcode para evitar errores.

Para interactuar con el llaverero, usamos un marco de CA llamado Servicios de Llaverero. Para obtener más información, consulte [la Guía de programación de servicios de llaverero de Apple](#).

Debido a que Keychain Services está por debajo del nivel `Foundation`, está restringido al uso de los tipos `CoreFoundation`. Como resultado, la mayoría de los objetos se representan internamente como `CFDictionary`s con `CFString`s como sus claves y una variedad de tipos de `CoreFoundation` como sus valores.

Mientras que Keychain Services se incluye como parte del marco de `Security`, la importación de `Foundation` suele ser una buena opción, ya que incluye algunas funciones de ayuda en el backend.

Además, si no desea tratar directamente con Keychain Services, Apple ofrece el proyecto de muestra Swift de [Keychain genérico](#) que proporciona tipos Swift que usan los servicios de Keychain entre bastidores.

Examples

Agregando una contraseña al llaverero

Cada elemento de llaverero se representa con mayor frecuencia como un `CFDictionary`. Sin embargo, puede simplemente usar `NSDictionary` en Objective-C y aprovechar el puenteo, o en

Swift puede usar `Dictionary` y convertir explícitamente a `CFDictionary` .

Podrías construir una contraseña con el siguiente diccionario:

Rápido

```
var dict = [String : AnyObject]()
```

Primero, necesita un par de clave / valor que le permita al llavero saber que esta es una contraseña. Tenga en cuenta que dado que nuestra clave `dict` es una `String` , debemos convertir cualquier `CFString` a una `String` explícitamente en Swift 3. `CFString` no se puede usar como la clave de un Diccionario Swift porque no es Hashable.

Rápido

```
dict[kSecClass as String] = kSecClassGenericPassword
```

A continuación, nuestra contraseña puede tener una serie de atributos para describirla y ayudarnos a encontrarla más adelante. [Aquí hay una lista de atributos para contraseñas genéricas](#) .

Rápido

```
// The password will only be accessible when the device is unlocked
dict[kSecAttrAccessible as String] = kSecAttrAccessibleWhenUnlocked
// Label may help you find it later
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
// Service name
dict[kSecAttrService as String] = "MyService" as CFString
```

Por último, necesitamos nuestros datos privados reales. Asegúrese de no mantener esto en la memoria por mucho tiempo. Esto debe ser `CFData` .

Rápido

```
dict[kSecValueData as String] = "my_password!".data(using: .utf8) as! CFData
```

Finalmente, la función de agregar Servicios de llavero quiere saber cómo debe devolver el elemento de llavero recién construido. Dado que no debe conservar los datos mucho tiempo en la memoria, aquí le explicamos cómo solo podría devolver los atributos:

Rápido

```
dict[kSecReturnAttributes as String] = kCFBooleanTrue
```

Ahora hemos construido nuestro artículo. Vamos a añadirlo:

Rápido

```
var result: AnyObject?
let status = withUnsafeMutablePointer(to: &result) {
    SecItemAdd(dict as CFDictionary, UnsafeMutablePointer($0))
}
let newAttributes = result as! Dictionary<String, AnyObject>
```

Esto coloca los nuevos atributos dictados dentro del `result . SecItemAdd` el diccionario que construimos, así como un puntero hacia donde nos gustaría obtener nuestro resultado. La función luego devuelve un `OSStatus` indica éxito o un código de error. Los códigos de resultados se describen [aquí](#) .

Encontrar una contraseña en el llavero

Para construir una consulta, necesitamos representarla como un `CFDictionary` . También puede usar `NSDictionary` en Objective-C o `Dictionary` en Swift y convertir a `CFDictionary` .

Necesitamos una clave de clase:

Rápido

```
var dict = [String : AnyObject]()
dict[kSecClass as String] = kSecClassGenericPassword
```

A continuación, podemos especificar atributos para limitar nuestra búsqueda:

Rápido

```
// Label
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
// Service name
dict[kSecAttrService as String] = "MyService" as CFString
```

También podemos especificar claves modificadoras de búsqueda especiales descritas [aquí](#) .

Finalmente, necesitamos decir cómo nos gustaría que nos devolvieran nuestros datos. A continuación, solicitaremos que solo la contraseña privada sea devuelta como un objeto `CFData` :

Rápido

```
dict[kSecReturnData as String] = kCFBooleanTrue
```

Ahora, busquemos:

Rápido

```
var queryResult: AnyObject?
let status = withUnsafeMutablePointer(to: &queryResult) {
    SecItemCopyMatching(dict as CFDictionary, UnsafeMutablePointer($0))
}
// Don't keep this in memory for long!!
let password = String(data: queryResult as! Data, encoding: .utf8)!
```

Aquí, `SecItemCopyMatching` un diccionario de consulta y un puntero hacia donde desea que vaya el resultado. Devuelve un `OSStatus` con un código de resultado. [Aquí](#) están las posibilidades.

Actualizando una contraseña en el llavero

Como de costumbre, primero necesitamos un `CFDictionary` para representar el elemento que queremos actualizar. Esto debe contener todos los valores antiguos para el artículo, incluidos los datos privados antiguos. Luego toma un `CFDictionary` de cualquier atributo o los datos en sí que le gustaría cambiar.

Así que primero, vamos a construir una clave de clase y una lista de atributos. Estos atributos pueden limitar nuestra búsqueda, pero debe incluir los atributos y los valores antiguos si los va a cambiar.

Rápido

```
var dict = [String : AnyObject]()
dict[kSecClass as String] = kSecClassGenericPassword
// Label
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
```

Ahora debemos agregar los datos antiguos:

Rápido


```
dict[kSecValueData as String] = "my_password!!".data(using: .utf8) as! CFData
```

Ahora vamos a crear los mismos atributos pero una contraseña diferente:

Rápido

```
var newDict = [String : AnyObject]()
newDict[kSecClass as String] = kSecClassGenericPassword
// Label
newDict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
newDict[kSecAttrAccount as String] = "My Name" as CFString
// New password
newDict[kSecValueData as String] = "new_password!!".data(using: .utf8) as! CFData
```

Ahora, solo lo pasamos a Keychain Services:

Rápido

```
let status = SecItemUpdate(dict as CFDictionary, newDict as CFDictionary)
```

`SecItemUpdate` devuelve un código de estado. Los resultados se describen [aquí](#) .

Eliminar una contraseña del llavero

Solo necesitamos una cosa para eliminar un elemento del llavero: un `CFDictionary` con atributos que describen los elementos que se eliminarán. Cualquier elemento que coincida con el diccionario de consulta se eliminará de forma permanente, por lo que si solo tiene la intención de eliminar un solo elemento, asegúrese de ser específico con su consulta. Como siempre, podemos usar un `NSDictionary` en Objective-C o en Swift podemos usar un `Dictionary` y luego convertirlo en `CFDictionary` .

Un diccionario de consultas, en este contexto, incluye exclusivamente una clave de clase para describir qué es el elemento y los atributos para describir información sobre el elemento. No se permite la inclusión de restricciones de búsqueda como `kSecMatchCaseInsensitive` .

Rápido

```
var dict = [String : AnyObject]()
dict[kSecClass as String] = kSecClassGenericPassword
// Label
dict[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
// Username
dict[kSecAttrAccount as String] = "My Name" as CFString
```

Y ahora podemos simplemente eliminarlo:

Rápido

```
let status = SecItemDelete(dict as CFDictionary)
```

`SecItemDelete` devuelve un `OSStatus` . Los códigos de resultados se describen [aquí](#) .

Llavero Añadir, actualizar, eliminar y buscar operaciones utilizando un archivo.

Llavero.h

```
#import <Foundation/Foundation.h>
typedef void (^KeychainOperationBlock)(BOOL successfulOperation, NSData *data, OSStatus status);

@interface Keychain : NSObject

-(id) initWithService:(NSString *) service_ withGroup:(NSString*)group_;

-(void)insertKey:(NSString *)key withData:(NSData *)data
withCompletion:(KeychainOperationBlock)completionBlock;
-(void)updateKey:(NSString*)key withData:(NSData*) data
withCompletion:(KeychainOperationBlock)completionBlock;
-(void)removeDataForKey:(NSString*)key
withCompletionBlock:(KeychainOperationBlock)completionBlock;
-(void)findDataForKey:(NSString*)key
withCompletionBlock:(KeychainOperationBlock)completionBlock;

@end
```

Llavero.m

```
#import "Keychain.h"
#import <Security/Security.h>

@implementation Keychain

{
    NSString * keychainService;
    NSString * keychainGroup;
}

-(id) initWithService:(NSString *)service withGroup:(NSString*)group
{
    self =[super init];
    if(self) {
        keychainService = [NSString stringWithString:service];
        if(group) {
            keychainGroup = [NSString stringWithString:group];
        }
    }
}
```

```

    return self;
}

-(void)insertKey:(NSString *)key
    withData:(NSData *)data
    withCompletion:(KeychainOperationBlock)completionBlock
{
    NSMutableDictionary * dict =[self prepareDict:key];
    [dict setObject:data forKey:(__bridge id)kSecValueData];
    [dict setObject:keychainService forKey:(id)kSecAttrService];

    OSStatus status = SecItemAdd((__bridge CFDictionaryRef)dict, NULL);
    if(errSecSuccess != status) {
        DLog(@"Unable add item with key =%@ error:%d",key, (int)status);
        if (completionBlock) {
            completionBlock(errSecSuccess == status, nil, status);
        }
    }
    if (status == errSecDuplicateItem) {
        [self updateKey:key withData:data withCompletion:^(BOOL successfulOperation, NSData
*updateData, OSStatus updateStatus) {
            if (completionBlock) {
                completionBlock(successfulOperation, updateData, updateStatus);
            }
            DLog(@"Found duplication item -- updating key with data");
        }]];
    }
}

-(void)findDataForKey:(NSString *)key
    withCompletionBlock:(KeychainOperationBlock)completionBlock
{
    NSMutableDictionary *dict = [self prepareDict:key];
    [dict setObject:(__bridge id)kSecMatchLimitOne forKey:(__bridge id)kSecMatchLimit];
    [dict setObject:keychainService forKey:(id)kSecAttrService];
    [dict setObject:(id)kCFBooleanTrue forKey:(__bridge id)kSecReturnData];
    CFTypeRef result = NULL;
    OSStatus status = SecItemCopyMatching((__bridge CFDictionaryRef)dict,&result);

    if( status != errSecSuccess) {
        DLog(@"Unable to fetch item for key %@ with error:%d",key, (int)status);
        if (completionBlock) {
            completionBlock(errSecSuccess == status, nil, status);
        }
    } else {
        if (completionBlock) {
            completionBlock(errSecSuccess == status, (__bridge NSData *)result, status);
        }
    }
}

-(void)updateKey:(NSString *)key
    withData:(NSData *)data
    withCompletion:(KeychainOperationBlock)completionBlock
{
    NSMutableDictionary * dictKey =[self prepareDict:key];

    NSMutableDictionary * dictUpdate =[NSMutableDictionary alloc] init];
    [dictUpdate setObject:data forKey:(__bridge id)kSecValueData];
    [dictUpdate setObject:keychainService forKey:(id)kSecAttrService];
    OSStatus status = SecItemUpdate((__bridge CFDictionaryRef)dictKey, (__bridge

```

```

CFDictionaryRef)dictUpdate);
    if( status != errSecSuccess) {
        DLog(@"Unable to remove item for key %@ with error:%d",key,(int)status);
    }
    if (completionBlock) {
        completionBlock(errSecSuccess == status, nil, status);
    }
}

-(void)removeDataForKey:(NSString *)key
withCompletionBlock:(KeychainOperationBlock)completionBlock {
    NSMutableDictionary *dict = [self prepareDict:key];
    OSStatus status = SecItemDelete((__bridge CFDictionaryRef)dict);
    if( status != errSecSuccess) {
        DLog(@"Unable to remove item for key %@ with error:%d",key,(int)status);
    }
    if (completionBlock) {
        completionBlock(errSecSuccess == status, nil, status);
    }
}

#pragma mark Internal methods

-(NSMutableDictionary*) prepareDict:(NSString *) key {

    NSMutableDictionary *dict = [[NSMutableDictionary alloc] init];
    [dict setObject:(__bridge id)kSecClassGenericPassword forKey:(__bridge id)kSecClass];

    NSData *encodedKey = [key dataUsingEncoding:NSUTF8StringEncoding];
    [dict setObject:encodedKey forKey:(__bridge id)kSecAttrGeneric];
    [dict setObject:encodedKey forKey:(__bridge id)kSecAttrAccount];
    [dict setObject:keychainService forKey:(__bridge id)kSecAttrService];
    [dict setObject:(__bridge id)kSecAttrAccessibleAlwaysThisDeviceOnly forKey:(__bridge
id)kSecAttrAccessible];

    //This is for sharing data across apps
    if(keychainGroup != nil) {
        [dict setObject:keychainGroup forKey:(__bridge id)kSecAttrAccessGroup];
    }

    return dict;
}

@end

```

Llavero de control de acceso (TouchID con contraseña de retorno)

Keychain permite guardar elementos con el atributo SecAccessControl especial que permitirá obtener elementos de Keychain solo después de que el usuario sea autenticado con Touch ID (o código de acceso si se permite dicha reserva). Solo se notifica a la aplicación si la autenticación fue exitosa o no, la IU completa es administrada por iOS.

Primero, se debe crear el objeto SecAccessControl:

Rápido

```
let error: Unmanaged<CFError>?

guard let accessControl = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly, .userPresence, &error) else {
    fatalError("Something went wrong")
}
```

A continuación, agréguelo al diccionario con la clave `kSecAttrAccessControl` (que se excluye mutuamente con la clave `kSecAttrAccessible` que ha estado usando en otros ejemplos):

Rápido

```
var dictionary = [String : Any]()

dictionary[kSecClass as String] = kSecClassGenericPassword
dictionary[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
dictionary[kSecAttrAccount as String] = "My Name" as CFString
dictionary[kSecValueData as String] = "new_password!".data(using: .utf8) as! CFData
dictionary[kSecAttrAccessControl as String] = accessControl
```

Y guárdalo como lo has hecho antes:

Rápido

```
let lastResultCode = SecItemAdd(query as CFDictionary, nil)
```

Para acceder a los datos almacenados, solo consulte Keychain para obtener una clave. Keychain Services presentará un cuadro de diálogo de autenticación al usuario y devolverá los datos, o nada, dependiendo de si se proporcionó una huella dactilar adecuada o si se hizo coincidir el código de acceso.

Opcionalmente, se puede especificar una cadena de solicitud:

Rápido

```
var query = [String: Any]()

query[kSecClass as String] = kSecClassGenericPassword
query[kSecReturnData as String] = kCFBooleanTrue
query[kSecAttrAccount as String] = "My Name" as CFString
query[kSecAttrLabel as String] = "com.me.myapp.myaccountpassword" as CFString
query[kSecUseOperationPrompt as String] = "Please put your fingers on that button" as CFString

var queryResult: AnyObject?
let status = withUnsafeMutablePointer(to: &queryResult) {
    SecItemCopyMatching(query as CFDictionary, UnsafeMutablePointer($0))
}
```

Preste atención a que el `status` será `err` si el usuario rechazó, canceló o falló la autorización.

Rápido

```
if status == noErr {
    let password = String(data: queryResult as! Data, encoding: .utf8)!
    print("Password: \(password)")
} else {
    print("Authorization not passed")
}
```

Lea Llaveró en línea: <https://riptutorial.com/es/ios/topic/6839/llaveró>

Capítulo 91: Localización

Introducción

La **localización** es una función provista por iOS que traduce su aplicación a múltiples idiomas. Para la **localización**, es necesaria la **internacionalización**. La **internacionalización** es el proceso de hacer que la aplicación iOS pueda adaptar diferentes culturas, idiomas y regiones.

Examples

Localización en iOS

Cree un archivo individualizable de `Localizable.strings` para cada idioma. El lado derecho sería diferente para cada idioma. Piense en ello como un par clave-valor:

```
"str" = "str-language";
```

Acceso str en Objective-C:

```
//Try to provide description on the localized string to be able to create a proper  
documentation if needed  
NSString *str = NSLocalizedString(@"string", @"description of the string");
```

Acceso str en Swift:

```
let str = NSLocalizedString("string", comment: "language");
```

Lea **Localización en línea**: <https://riptutorial.com/es/ios/topic/1579/localizacion>

Capítulo 92: Manejando el teclado

Examples

Desplazando un UIScrollView / UITableView al visualizar el teclado

Hay pocos enfoques disponibles allí:

1. Puede suscribirse para recibir notificaciones de eventos de apariencia de teclado y cambiar la compensación manualmente:

```
//Swift 2.0+
override func viewDidLoad() {
    super.viewDidLoad()

    NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(YourVCClassName.keyboardWillShow(_:)), name: UIKeyboardWillShowNotification, object:
nil)
    NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(YourVCClassName.keyboardWillHide(_:)), name: UIKeyboardWillHideNotification, object:
nil)
}

func keyboardWillShow(notification: NSNotification) {
    if let userInfo = notification.userInfo {
        if let keyboardHeight =
userInfo[UIKeyboardFrameEndUserInfoKey]?.CGRectValue.size.height {
            tableView.contentInset = UIEdgeInsetsMake(0, 0, keyboardHeight, 0)
        }
    }
}

func keyboardWillHide(notification: NSNotification) {
    tableView.contentInset = UIEdgeInsetsMake(0, 0, 0, 0)
}

//Objective-C
- (void)viewDidLoad {

    [super viewDidLoad];

    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillShow:) name:UIKeyboardWillShowNotification object:nil];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillHide:) name:UIKeyboardWillHideNotification object:nil];
}

- (void)keyboardWillShow:(NSNotification *)notification {

    NSDictionary *userInfo = [notification userInfo];

    if (userInfo) {

        CGRect keyboardEndFrame;
```



```

        [[userInfo objectForKey:UIKeyboardFrameEndUserInfoKey] getValue:&keyboardEndFrame];
        tableView.contentInset = UIEdgeInsetsMake(0, 0, keyboardEndFrame.size.height, 0);
    }
}

- (void)keyboardWillHide:(NSNotification *)notification {
    tableView.contentInset = UIEdgeInsetsMake(0, 0, 0, 0);
}

```

2. O use soluciones ya hechas como `TPKeyboardAvoidingTableView` o `TPKeyboardAvoidingScrollView` <https://github.com/michaeltyson/TPKeyboardAvoiding>

Descartar un teclado con toque en la vista

Si desea ocultar un teclado tocando fuera de él, es posible utilizar este truco pirateado (solo funciona con Objective-C):

```

- (void)viewDidLoad {
    [super viewDidLoad];

    // dismiss keyboard when tap outside a text field
    UITapGestureRecognizer *tapGestureRecognizer = [[UITapGestureRecognizer alloc]
initWithTarget:self.view action:@selector(endEditing:)];
    [tapGestureRecognizer setCancelsTouchesInView:NO];
    [self.view addGestureRecognizer:tapGestureRecognizer];
}

```

Para Swift habrá un poco más de código:

```

override func viewDidLoad() {
    super.viewDidLoad()

    // dismiss keyboard when tap outside a text field
    let tapGestureRecognizer: UITapGestureRecognizer = UITapGestureRecognizer(target: self,
action: #selector(YourVCName.dismissKeyboard))
    view.addGestureRecognizer(tapGestureRecognizer)
}

//Calls this function when the tap is recognized.
func dismissKeyboard() {
    //Causes the view (or one of its embedded text fields) to resign the first responder
status.
    view.endEditing(true)
}

```

Otro ejemplo de Swift 3 / iOS 10

```

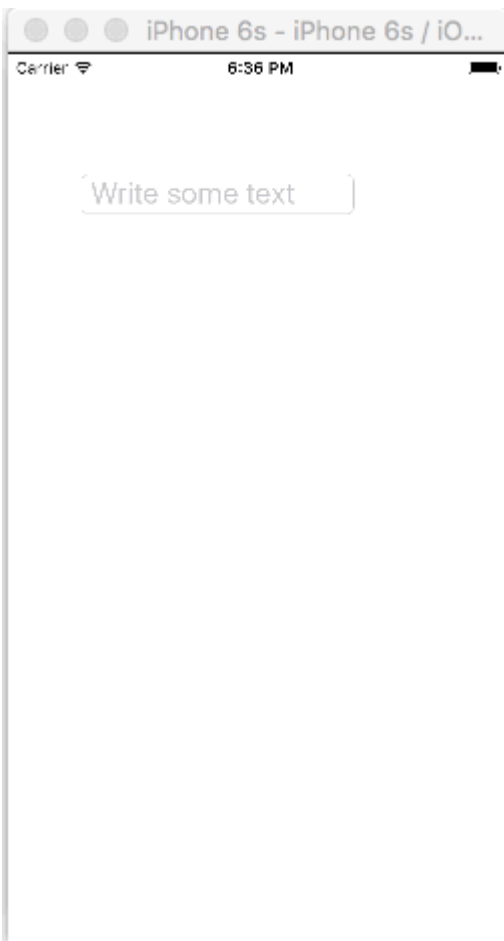
class vc: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
    }
}

```

```
        txtSomeField.delegate = self
    }
}

extension vc: UITextFieldDelegate {
    //Hide the keyboard for any text field when the UI is touched outside of the keyboard.
    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)
    {
        self.view.endEditing(true) //Hide the keyboard
    }
}
```

Crear un teclado personalizado en la aplicación



Este es un teclado básico en la aplicación. El mismo método podría usarse para hacer casi cualquier distribución de teclado. Aquí están las principales cosas que deben hacerse:

- Cree la distribución del teclado en un archivo `.xib`, cuyo propietario es una clase Swift o Objective-C que es una subclase `UIView`.
- Dígame al `UITextField` que use el teclado personalizado.
- Utilice un delegado para comunicarse entre el teclado y el controlador de vista principal.

Crear el archivo de diseño de teclado `.xib`

- En Xcode vaya a **Archivo > Nuevo > Archivo ... > iOS > Interfaz de usuario > Ver** para crear el archivo .xib.
- Llamé al mío teclado.xib
- Añade los botones que necesites.
- Use restricciones de diseño automático para que, independientemente del tamaño del teclado, los botones cambien de tamaño según corresponda.
- Establezca el propietario del archivo (no la vista raíz) para que sea la clase de `Keyboard`. Esta es una fuente común de error. Vas a crear esta clase en el siguiente paso. Ver la nota al final.

Cree el archivo de teclado de subclase .swift UIView

- En Xcode vaya a **Archivo > Nuevo > Archivo ... > iOS > Fuente > Cocoa Touch Class** para crear la **clase** Swift o Objective-C. Elija `UIView` como una superclase para la clase recién creada
- Llamé a la mía `Keyboard.swift` (clase de `Keyboard` en Objective-C)
- Agregue el siguiente código para Swift:

```
import UIKit

// The view controller will adopt this protocol (delegate)
// and thus must contain the keyWasTapped method
protocol KeyboardDelegate: class {
    func keyWasTapped(character: String)
}

class Keyboard: UIView {

    // This variable will be set as the view controller so that
    // the keyboard can send messages to the view controller.
    weak var delegate: KeyboardDelegate?

    // MARK:- keyboard initialization

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        initializeSubviews()
    }

    override init(frame: CGRect) {
        super.init(frame: frame)
        initializeSubviews()
    }

    func initializeSubviews() {
        let xibName = "Keyboard" // xib extension not included
        let view = NSBundle.mainBundle().loadNibNamed(xibName, owner: self,
options: nil)[0] as! UIView
        self.addSubview(view)
    }
}
```

```

        view.frame = self.bounds
    }

    // MARK:- Button actions from .xib file

    @IBAction func keyTapped(sender: UIButton) {
        // When a button is tapped, send that information to the
        // delegate (ie, the view controller)
        self.delegate?.keyWasTapped(sender.titleLabel!.text!) // could alternatively
        send a tag value
    }
}

```

- Agregue el siguiente código para Objective-C:

Archivo keyboard.h

```

#import <UIKit/UIKit.h>

// The view controller will adopt this protocol (delegate)
// and thus must contain the keyWasTapped method
@protocol KeyboardDelegate<NSObject>
- (void)keyWasTapped:(NSString *)character;
@end

@interface Keyboard : UIView
@property (nonatomic, weak) id<KeyboardDelegate> delegate;
@end

```

Archivo keyboard.m

```

#import "Keyboard.h"

@implementation Keyboard

- (id)initWithCoder:(NSCoder *)aDecoder {
    self = [super initWithCoder:aDecoder];
    [self initializeSubviews];
    return self;
}

- (id)initWithFrame:(CGRect)frame {
    self = [super initWithFrame:frame];
    [self initializeSubviews];
    return self;
}

- (void)initializeSubviews {
    NSString *xibName = @"Keyboard"; // xib extension not included
    UIView *view = [[[NSBundle mainBundle] loadNibNamed:xibName owner:self
options:nil] firstObject];
    [self addSubview:view];
    view.frame = self.bounds;
}

// MARK:- Button actions from .xib file

```

```

-(IBAction)keyTapped:(UIButton *)sender {
    // When a button is tapped, send that information to the
    // delegate (ie, the view controller)
    [self.delegate keyWasTapped:sender.titleLabel.text]; // could alternatively send a
    tag value
}

@end

```

- Controle las acciones de arrastre desde los botones hasta la devolución de llamada en el archivo `@IBAction` método `@IBAction` en el propietario de Swift o Objective-C para conectarlos todos.
- Tenga en cuenta que el protocolo y el código delegado. Vea [esta respuesta](#) para una explicación simple sobre cómo trabajan los delegados.

Configurar el controlador de vista

- Agregue un `UITextField` a su guión gráfico principal y conéctelo a su controlador de vista con un `IBOutlet`. Llámalo `textField`.
- Use el siguiente código para View Controller en Swift:

```

import UIKit

class ViewController: UIViewController, KeyboardDelegate {

    @IBOutlet weak var textField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()

        // initialize custom keyboard
        let keyboardView = Keyboard(frame: CGRect(x: 0, y: 0, width: 0, height: 300))
        keyboardView.delegate = self // the view controller will be notified by the
        keyboard whenever a key is tapped

        // replace system keyboard with custom keyboard
        textField.inputView = keyboardView
    }

    // required method for keyboard delegate protocol
    func keyWasTapped(character: String) {
        textField.insertText(character)
    }
}

```

- Usa el siguiente código para Objective-C:

archivo .h

```

#import <UIKit/UIKit.h>

@interface ViewController : UIViewController

```

```
@end
```

archivo .m

```
#import "ViewController.h"
#import "Keyboard.h"

@interface ViewController ()<KeyboardDelegate>

@property (nonatomic, weak) IBOutlet UITextField *textField;

@end

@implementation ViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.

    // initialize custom keyboard
    Keyboard *keyboardView = [[Keyboard alloc] initWithFrame:CGRectMake(0, 0, 0, 300)];
    keyboardView.delegate = self; // the view controller will be notified by the keyboard
    whenever a key is tapped

    // replace system keyboard with custom keyboard
    self.textField.inputView = keyboardView;
}

- (void)keyWasTapped:(NSString *)character {
    [self.textField insertText:character];
}

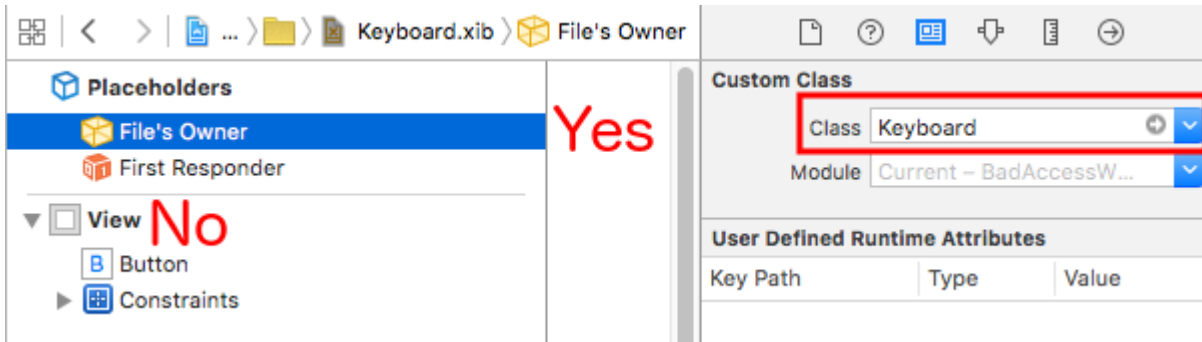
@end
```

- Tenga en cuenta que el controlador de vista adopta el protocolo `KeyboardDelegate` que definimos anteriormente.

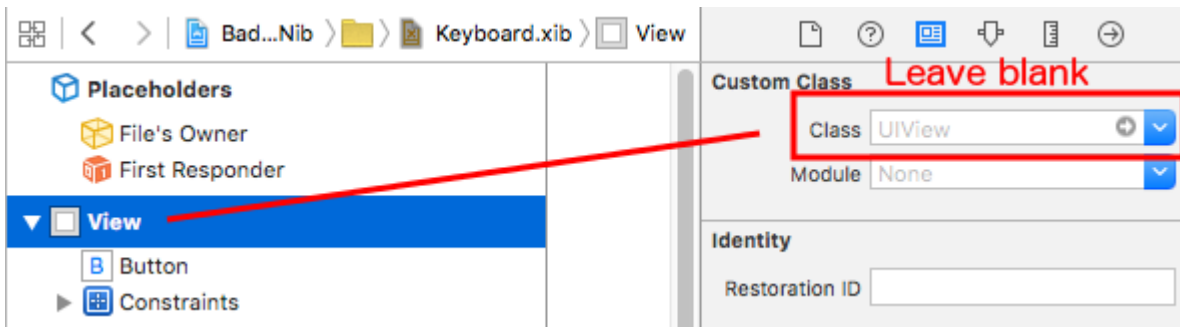
Error común

Si `EXC_BAD_ACCESS` un error `EXC_BAD_ACCESS`, probablemente se deba a que configura la clase personalizada de la vista como `Keyboard` lugar de hacerlo para el propietario del archivo de plumilla.

Seleccione `Keyboard.nib` y luego elija Propietario del archivo.



Asegúrese de que la clase personalizada para la vista raíz esté en blanco.



Notas

Este ejemplo proviene originalmente de [esta respuesta de desbordamiento de pila](#).

Administrar el teclado usando un delegado de Singleton +

Cuando empecé a administrar el teclado, usaba notificaciones separadas en cada ViewController.

Método de notificación (mediante NSNotification):

```
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
        NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(ViewController.keyboardNotification(_:)), name:
UIKeyboardWillChangeFrameNotification, object: nil)
    }

    func keyboardNotification(notification: NSNotification) {
        guard let userInfo = notification.userInfo else { return }

        let endFrame = (userInfo[UIKeyboardFrameEndUserInfoKey] as? NSValue)?.CGRectValue()
        let duration: NSTimeInterval = (userInfo[UIKeyboardAnimationDurationUserInfoKey] as?
NSNumber)?.doubleValue ?? 0
        let animationCurveRawNSN = userInfo[UIKeyboardAnimationCurveUserInfoKey] as? NSNumber
        let animationCurveRaw = animationCurveRawNSN?.unsignedLongValue ??
UIViewAnimationOptions.CurveEaseOut.rawValue
        let animationCurve: UIViewAnimationOptions = UIViewAnimationOptions(rawValue:
animationCurveRaw)

        if endFrame?.origin.y >= UIScreen.mainScreen().bounds.size.height {
            lowerViewBottomConstraint.constant = 0
        }
    }
}
```

```

    } else {
        lowerViewBottomConstraint.constant = endFrame?.size.height ?? 0.0
    }
    view.animateConstraintWithDuration(duration, delay: NSTimeInterval(0), options:
animationCurve, completion: nil)
    }
}

```

Mi problema fue que me encontré escribiendo este código una y otra vez para cada ViewController. Después de experimentar un poco, descubrí que usar un patrón Singleton + Delegate me permitió reutilizar un montón de código y organizar toda la administración del teclado en un solo lugar.

Singleton + Método de Delegado:

```

protocol KeyboardManagerDelegate: class {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions)
}

class KeyboardManager {

    weak var delegate: KeyboardManagerDelegate?

    class var sharedInstance: KeyboardManager {
        struct Singleton {
            static let instance = KeyboardManager()
        }
        return Singleton.instance
    }

    init() {
        NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(KeyboardManager.keyboardWillChangeFrameNotification(_:)), name:
UIKeyboardWillChangeFrameNotification, object: nil)
    }

    @objc func keyboardWillChangeFrameNotification(notification: NSNotification) {
        guard let userInfo = notification.userInfo else { return }

        let endFrame = (userInfo[UIKeyboardFrameEndUserInfoKey] as? NSValue)?.CGRectValue()
        let duration: NSTimeInterval = (userInfo[UIKeyboardAnimationDurationUserInfoKey] as?
NSNumber)?.doubleValue ?? 0
        let animationCurveRawNSN = userInfo[UIKeyboardAnimationCurveUserInfoKey] as? NSNumber
        let animationCurveRaw = animationCurveRawNSN?.unsignedLongValue ??
UIViewAnimationOptions.CurveEaseOut.rawValue
        let animationCurve: UIViewAnimationOptions = UIViewAnimationOptions(rawValue:
animationCurveRaw)

        delegate?.keyboardWillChangeFrame(endFrame, duration: duration, animationCurve:
animationCurve)
    }
}

```

Ahora, cuando quiero administrar el teclado desde un ViewController, todo lo que tengo que hacer es configurar el delegado en ese ViewController e implementar cualquier método de delegado.


```

class ViewController: UIViewController {
    override func viewWillAppear(animated: Bool) {
        super.viewWillAppear(animated)
        KeyboardManager.sharedInstance.delegate = self
    }
}

// MARK: - Keyboard Manager

extension ViewController: KeyboardManagerDelegate {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions) {
        if endFrame?.origin.y >= UIScreen.mainScreen().bounds.size.height {
            lowerViewBottomConstraint.constant = 0
        } else {
            lowerViewBottomConstraint.constant = (endFrame?.size.height ?? 0.0)
        }
        view.animateConstraintWithDuration(duration, delay: NSTimeInterval(0), options:
animationCurve, completion: nil)
    }
}

```

Este método es muy personalizable también! Digamos que queremos agregar funcionalidad para `UIKeyboardWillHideNotification` . Esto es tan fácil como agregar un método a nuestro `KeyboardManagerDelegate` .

`KeyboardManagerDelegate` **with** `UIKeyboardWillHideNotification` :

```

protocol KeyboardManagerDelegate: class {
    func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions)
    func keyboardWillHide(notificationUserInfo: [NSObject: AnyObject])
}

class KeyboardManager {
    init() {
        NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(KeyboardManager.keyboardWillChangeFrameNotification(_:)), name:
UIKeyboardWillChangeFrameNotification, object: nil)
        NotificationCenter.defaultCenter().addObserver(self, selector:
#selector(KeyboardManager.keyboardWillHide(_:)), name: UIKeyboardWillHideNotification, object:
nil)
    }

    func keyboardWillHide(notification: NSNotification) {
        guard let userInfo = notification.userInfo else { return }
        delegate?.keyboardWillHide(userInfo)
    }
}

```

Digamos que solo queremos implementar `func keyboardWillHide(notificationUserInfo: [NSObject: AnyObject])` en un `ViewController`. También podemos hacer este método opcional.

```

typealias KeyboardManagerDelegate = protocol<KeyboardManagerModel,
KeyboardManagerConfigureable>

protocol KeyboardManagerModel: class {

```

```

func keyboardWillChangeFrame(endFrame: CGRect?, duration: NSTimeInterval, animationCurve:
UIViewAnimationOptions)
}

@objc protocol KeyboardManagerConfigurable {
    optional func keyboardWillHide(userInfo: [NSObject: AnyObject])
}

```

* Tenga en cuenta que este patrón ayuda a evitar el uso excesivo de `@objc`. ¡Vea <http://www.jessesquires.com/avoiding-objc-in-swift/> para más detalles!

En resumen, he encontrado que usar un Delegado de Singleton + para administrar el teclado es más eficiente y fácil de usar que usar Notificaciones

Mover la vista hacia arriba o hacia abajo cuando el teclado está presente

Nota: esto solo funciona para el teclado incorporado provisto por iOS

RÁPIDO:

Para que la vista de un **UIViewController** aumente el origen del marco cuando se presenta y lo disminuya cuando esté oculto, agregue las siguientes funciones a su clase:

```

func keyboardWillShow(notification: NSNotification) {

    if let keyboardSize = (notification.userInfo?[UIKeyboardFrameBeginUserInfoKey] as?
NSValue)?.cgRectValue {
        if self.view.frame.origin.y == 0{
            self.view.frame.origin.y -= keyboardSize.height
        }
    }
}

func keyboardWillHide(notification: NSNotification) {
    if let keyboardSize = (notification.userInfo?[UIKeyboardFrameBeginUserInfoKey] as?
NSValue)?.cgRectValue {
        if self.view.frame.origin.y != 0{
            self.view.frame.origin.y += keyboardSize.height
        }
    }
}

```

Y en el método `viewDidLoad()` de su clase, agregue los siguientes observadores:

```

NotificationCenter.default.addObserver(self, selector: #selector(Login.keyboardWillShow),
name: NSNotification.Name.UIKeyboardWillShow, object: nil)
NotificationCenter.default.addObserver(self, selector: #selector(Login.keyboardWillHide),
name: NSNotification.Name.UIKeyboardWillHide, object: nil)

```

Y esto funcionará para cualquier tamaño de pantalla, usando la propiedad de altura del teclado.

C OBJETIVO:

Para hacer lo mismo en Objective-C, este código se puede usar:

```
- (void) viewWillAppear:(BOOL) animated {
    [super viewWillAppear:animated];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillShow:) name:UIKeyboardWillShowNotification object:nil];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(keyboardWillHide:) name:UIKeyboardWillHideNotification object:nil];
}

- (void) viewWillDisappear:(BOOL) animated {
    [super viewWillDisappear:animated];
    [[NSNotificationCenter defaultCenter] removeObserver:self
name:UIKeyboardWillShowNotification object:nil];
    [[NSNotificationCenter defaultCenter] removeObserver:self
name:UIKeyboardWillHideNotification object:nil];
}

- (void) keyboardWillShow:(NSNotification *) notification
{
    CGSize keyboardSize = [[[notification userInfo]
objectForKey:UIKeyboardFrameBeginUserInfoKey] CGRectValue].size;

    [UIView animateWithDuration:0.3 animations:^(
        CGRect f = self.view.frame;
        f.origin.y = -keyboardSize.height;
        self.view.frame = f;
    )];
}

- (void) keyboardWillHide:(NSNotification *) notification
{
    [UIView animateWithDuration:0.3 animations:^(
        CGRect f = self.view.frame;
        f.origin.y = 0.0f;
        self.view.frame = f;
    )];
}
```

Lea **Manejando el teclado en línea**: <https://riptutorial.com/es/ios/topic/436/manejando-el-teclado>

Capítulo 93: Manejar múltiples entornos usando macro

Examples

Manejar múltiples entornos usando múltiples objetivos y macro

Por ejemplo, tenemos dos entornos: CI: puesta en escena y queremos agregar algunas personalizaciones para cada entorno. Aquí intentaré personalizar la URL del servidor, nombre de la aplicación.

Primero, creamos dos objetivos para 2 entornos duplicando el objetivo principal:

- MultipleEnvironments M
 - MultipleEnvironments
 - AppDelegate.h
 - AppDelegate.m
 - ViewController.h
 - ViewController.m
 - Main.storyboard
 - Assets.xcassets
 - LaunchScreen.storyboard
 - Info.plist
 - Supporting Files
 - Products
 - MultipleEnviron...s copy-Info.plist A
 - CI copy-Info.plist A

- PROJECT
 - MultipleEnvironme...
- TARGETS
 - MultipleEnvironme...
 - CI
 - Staging

General

▼ Identity

▼ Deployment Info

▼ App Icons and Launch Images

▼ Embedded Binaries

▼ Linked Frameworks and Libraries

- Si ejecutamos / archivamos utilizando el destino de CI, SERVER_URL es <http://ci.api.example.com/>
- Si ejecutamos / archivamos usando STAGING target, SERVER_URL es <http://stg.api.example.com/>

Si desea personalizar más, por ejemplo: Cambiar el nombre de la aplicación para cada objetivo:



Xcode

File

Edit

View

Find

Navigate



CI >



iPhone 6s



M



MultipleEnvironments

M



MultipleEnvironments



AppDelegate.h



AppDelegate.m



ViewController.h



ViewController.m



Main.storyboard



Assets.xcassets



LaunchScreen.storyboard



Info.plist



Supporting Files



AppConfigurations.h

A



Products



MultipleEnviron...s copy-Info.plist

A



CI copy-Info.plist

A



PROJECT



MultipleEnv

TARGETS



MultipleEnv



CI



Staging



MultipleEn...



App CI



App STG

<https://riptutorial.com/es/ios/topic/6849/manejar-multiples-entornos-usando-macro>

Capítulo 94: Manejo de esquemas de URL

Sintaxis

1. // **El método `canOpenURL`** verifica si hay alguna aplicación que pueda manejar el esquema de URL indicado.

2. // Swift

```
UIApplication.sharedApplication (). CanOpenURL (_ aUrl: NSURL)
```

3. // C objetivo

```
[[UIApplication sharedApplication] canOpenURL: (NSURL *) aUrl];
```

4. // **El método `openURL`** intenta abrir un recurso ubicado por URL. **SÍ** / verdadero si se abrió de otro modo **NO** / falso.

5. // Swift

```
UIApplication.sharedApplication (). OpenURL (_ aUrl: NSURL)
```

6. // C objetivo

```
[[UIApplication sharedApplication] openURL: (NSURL *) aUrl];
```

Parámetros

Parámetro	Sentido
todo	una instancia de <code>NSURL</code> que almacena una cadena de esquema incorporada o personalizada

Observaciones

En iOS9 y superior, su aplicación debe enumerar los esquemas de URL que desee consultar. Esto se hace agregando `LSApplicationQueriesSchemes` a `Info.plist`

iOS tiene soporte incorporado para los esquemas de `tel`, `http` / `https`, `sms`, `mailto`, `facetime`. También admite URL basadas en `http` para aplicaciones de `Youtube`, `Maps` e `iTunes`.

Ejemplos de esquemas de URL incorporados:

tel : `tel://123456890` **O** `tel:123456890`

http : `http://www.google.com`

facetime : `facetime://azimov@demo.com`

mailto : `mailto://azimov@demo.com`

sms : `sms://123456890` **O** `sms:123456890`

Youtube : `https://www.youtube.com/watch?v=-eCaif2QKfA`

Mapas :

- **Usando la dirección:** `http://maps.apple.com/?address=1,Infinite+Loop,Cupertino,California`
- **Uso de coordenadas:** `http://maps.apple.com/?ll=46.683155557,6.683155557`

iTunes : `https://itunes.apple.com/us/artist/andy-newman/id200900`

Nota : No todos los caracteres especiales son compatibles con el esquema de `tel` (por ejemplo, * o #). Esto se hace debido a preocupaciones de seguridad para evitar que los usuarios redireccionen llamadas no autorizadas, por lo que en este caso, la aplicación del `Phone` no se abrirá.

Examples

Usando el esquema de URL incorporado para abrir la aplicación Mail

Rápido:

```
if let url = URL(string: "mailto://azimov@demo.com") {
    if UIApplication.shared.canOpenURL(url) {
        UIApplication.shared.openURL(url)
    } else {
        print("Cannot open URL")
    }
}
```

C objetivo:

```
NSURL *url = [NSURL URLWithString:@"mailto://azimov@demo.com"];
if ([[UIApplication sharedApplication] canOpenURL:url]) {
    [[UIApplication sharedApplication] openURL:url];
} else {
    NSLog(@"Cannot open URL");
}
```

Esquemas de URL de Apple

Estos son esquemas de URL admitidos por aplicaciones nativas en iOS, OS X y watchOS 2 y

posteriores.

Abrir enlace en Safari:

C objetivo

```
NSString *stringURL = @"http://stackoverflow.com/";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Rápido:

```
let stringURL = "http://stackoverflow.com/"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

Comenzando una conversación telefónica

C objetivo

```
NSString *stringURL = @"tel:1-408-555-5555";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Rápido:

```
let stringURL = "tel:1-408-555-5555"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

HTML

```
<a href="tel:1-408-555-5555">1-408-555-5555</a>
```

Comenzando una conversación de FaceTime

C objetivo

```
NSString *stringURL = @"facetime:14085551234";
NSURL *url = [NSURL URLWithString:stringURL];
[[UIApplication sharedApplication] openURL:url];
```

Rápido:

```
let stringURL = "facetime:14085551234"
if let url = URL(string: stringURL) {
    UIApplication.shared.openURL(url)
}
```

HTML

```
<a href="facetime:14085551234">Connect using FaceTime</a>  
<a href="facetime:user@example.com">Connect using FaceTime</a>
```

Apertura de la aplicación Messages para componer un sms al destinatario:

C objetivo

```
NSString *stringURL = @"sms:1-408-555-1212";  
NSURL *url = [NSURL URLWithString:stringURL];  
[[UIApplication sharedApplication] openURL:url];
```

Rápido:

```
let stringURL = "sms:1-408-555-1212"  
if let url = URL(string: stringURL) {  
    UIApplication.shared.openURL(url)  
}
```

HTML

```
<a href="sms:">Launch Messages App</a>  
<a href="sms:1-408-555-1212">New SMS Message</a>
```

Apertura de la aplicación Mail para redactar un correo electrónico al destinatario:

C objetivo

```
NSString *stringURL = @"mailto:foo@example.com";  
NSURL *url = [NSURL URLWithString:stringURL];  
[[UIApplication sharedApplication] openURL:url];
```

Rápido:

```
let stringURL = "mailto:foo@example.com"  
if let url = URL(string: stringURL) {  
    UIApplication.shared.openURL(url)  
}
```

HTML

```
<a href="mailto:frank@wwdcdemo.example.com">John Frank</a>
```

También puede incluir un campo de asunto, un mensaje y varios destinatarios en los campos Para, Cc y Cco. (En iOS, se ignora el atributo from). El siguiente ejemplo muestra una URL mailto que incluye varios atributos diferentes:

```
mailto:foo@example.com?cc=bar@example.com&subject=Greetings%20from%20Cupertino!&body=Wish%20you%20were
```

Nota: El diálogo de `MFMailComposeViewController` correo electrónico también se puede presentar dentro de la aplicación usando `MFMailComposeViewController` .

Lea Manejo de esquemas de URL en línea: <https://riptutorial.com/es/ios/topic/3646/manejo-de-esquemas-de-url>

Capítulo 95: Marco de contactos

Observaciones

Enlaces útiles

- [Documentación de Apple](#)
- [Preguntas y respuestas relacionadas con el desbordamiento de pila](#)
- [WWDC15 Sesión de vídeo](#)

Examples

Autorizar el acceso de contacto

Importando el framework

Rápido

```
import Contacts
```

C objetivo

```
#import <Contacts/Contacts.h>
```

Comprobando la accesibilidad

Rápido

```
switch CNContactStore.authorizationStatusForEntityType (CNEntityType.Contacts) {  
case .Authorized: //access contacts  
case .Denied, .NotDetermined: //request permission  
default: break  
}
```

C objetivo

```
switch ([CNContactStore authorizationStatusForEntityType:CNEntityType.Contacts]) {
```

```
case CNAuthorizationStatus.Authorized:
    //access contacts
    break;
case CNAuthorizationStatus.Denied:
    //request permission
    break;
case CNAuthorizationStatus.NotDetermined:
    //request permission
    break;
}
```

Solicitando Permiso

Rápido

```
var contactStore = CKContactStore()
contactStore.requestAccessForEntityType(CKEntityType.Contacts, completionHandler: { (ok, _) ->
Void in
    if access{
        //access contacts
    }
}
```

Acceso a los contactos

Aplicando un filtro

Para acceder a los contactos, debemos aplicar un filtro de tipo `NSPredicate` a nuestra variable `NSPredicate` que definimos en el ejemplo Autorizar el acceso de contacto. Por ejemplo, aquí queremos ordenar los contactos con nombre que coincida con el nuestro:

Rápido

```
let predicate = CNContact.predicateForContactsMatchingName("Some Name")
```

C objetivo

```
NSPredicate *predicate = [CNContact predicateForContactsMatchingName:@"Some Name"];
```

Especificando claves para buscar

Aquí, queremos obtener el nombre, el apellido y la imagen del perfil del contacto:

Rápido

```
let keys = [CNContactGivenNameKey, CNContactFamilyNameKey, CNContactImageDataKey]
```

Trayendo contactos

Rápido

```
do {
    let contacts = try contactStore.unifiedContactsMatchingPredicate(predicate, keysToFetch:
keys)
} catch let error as NSError {
    //...
}
```

Accediendo a los datos de contacto

Rápido

```
print(contacts[0].givenName)
print(contacts[1].familyName)
let image = contacts[2].imageData
```

Agregar un contacto

Rápido

```
import Contacts

// Creating a mutable object to add to the contact
let contact = CNMutableContact()

contact.imageData = NSData() // The profile picture as a NSData object

contact.givenName = "John"
contact.familyName = "Appleseed"

let homeEmail = CNLabeledValue(label:CNLabelHome, value:"john@example.com")
let workEmail = CNLabeledValue(label:CNLabelWork, value:"j.appleseed@icloud.com")
contact.emailAddresses = [homeEmail, workEmail]

contact.phoneNumbers = [CNLabeledValue(
    label:CNLabelPhoneNumberiPhone,
    value:CNPhoneNumber(stringValue:"(408) 555-0126"))]

let homeAddress = CNMutablePostalAddress()
```

```
homeAddress.street = "1 Infinite Loop"
homeAddress.city = "Cupertino"
homeAddress.state = "CA"
homeAddress.postalCode = "95014"
contact.postalAddresses = [CNLabeledValue(label:CNLabelHome, value:homeAddress)]

let birthday = NSDateComponents()
birthday.day = 1
birthday.month = 4
birthday.year = 1988 // You can omit the year value for a yearless birthday
contact.birthday = birthday

// Saving the newly created contact
let store = CNContactStore()
let saveRequest = CNSaveRequest()
saveRequest.addContact(contact, toContainerWithIdentifier:nil)
try! store.executeSaveRequest(saveRequest)
```

Lea Marco de contactos en línea: <https://riptutorial.com/es/ios/topic/5872/marco-de-contactos>

Capítulo 96: Mensajería FCM en Swift

Observaciones

FCM: <https://firebase.google.com/docs/cloud-messaging/ios/client>

Examples

Inicializar FCM en Swift

sigua el siguiente paso para agregar FCM en su proyecto swift

1- Si todavía no tienes un proyecto Xcode, crea uno ahora. Crea un Podfile si no tienes uno:

```
$ cd tu directorio de proyectos
$ pod init
```

2- Agrega los pods que quieres instalar. Puedes incluir un Pod en tu Podfile así:

```
pod 'Firebase / Core'
pod 'Firebase / Messaging'
```

3- Instale los pods y abra el archivo .xcworkspace para ver el proyecto en Xcode.

```
$ pod instalar
$ abre tu proyecto.xcworkspace
```

4- Descargue un archivo GoogleService-Info.plist de [plist](#) e [inclúyalo](#) en su aplicación.

5- Cargue el certificado APNs a Firebase. [APN Cert](#)

6- Agregue "Importar Firebase" en su archivo de proyecto de aplicación

7- Agregue este "FIRApp.configure ()" en su "aplicación: didFinishLaunchingWithOptions"

8- registrarse para notificación remota

```
if #available(iOS 10.0, *) {
    let authOptions : UNAuthorizationOptions = [.Alert, .Badge, .Sound]
    UNUserNotificationCenter.currentNotificationCenter().requestAuthorizationWithOptions(
        authOptions,
        completionHandler: {_,_ in })

    // For iOS 10 display notification (sent via APNS)
    UNUserNotificationCenter.currentNotificationCenter().delegate = self
    // For iOS 10 data message (sent via FCM)
    FIRMessaging.messaging().remoteMessageDelegate = self
} else {
    let settings: UIUserNotificationSettings =
```



```
    UIApplicationSettings(forTypes: [.Alert, .Badge, .Sound], categories: nil)
    application.registerUserNotificationSettings(settings)
}

application.registerForRemoteNotifications()
```

9- para obtener registro de uso token

```
let token = FIRInstanceID.instanceID().token()!
```

10- y si desea monitorear el cambio de token, use el siguiente código en el archivo AppDelegate

```
func tokenRefreshNotification(notification: NSNotification) {
    if let refreshedToken = FIRInstanceID.instanceID().token() {
        print("InstanceID token: \(refreshedToken)")
    }

    // Connect to FCM since connection may have failed when attempted before having a token.
    connectToFcm()
}
```

11- para recibir el mensaje de fcm, agregue el siguiente código en AppDelegate

```
func connectToFcm() {
    FIRMessaging.messaging().connectWithCompletion { (error) in
        if (error != nil) {
            print("Unable to connect with FCM. \(error)")
        } else {
            print("Connected to FCM.")
        }
    }
}
```

12- y para desconexión de uso.

```
func applicationDidEnterBackground(application: UIApplication) {
    FIRMessaging.messaging().disconnect()
    print("Disconnected from FCM.")
}
```

en tu AppDelegate.

la inicialización completa y el cliente listo para recibir el mensaje del panel fcm o enviarlo por un token desde un servidor de terceros

Lea Mensajería FCM en Swift en línea: <https://riptutorial.com/es/ios/topic/7326/mensajeria-fcm-en-swift>

Capítulo 97: Métodos personalizados de selección de UITableViewCells.

Introducción

Avanzar en las formas de gestionar las selecciones de UITableViewCell. Ejemplos cuando simple `didSelect...` form `UITableViewDelegate` no es suficiente para lograr algo.

Examples

Distinción entre selección simple y doble en fila.

Un ejemplo de implementación que ofrece la posibilidad de detectar si un usuario toca o toca dos veces en UITableViewCell.

```
override func viewDidLoad() {
    viewDidLoad()

    let doubleTapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleDoubleTap(sender:)))
    doubleTapGestureRecognizer.numberOfTapsRequired = 2
    tableView.addGestureRecognizer(doubleTapGestureRecognizer)

    let tapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleTapGesture(sender:)))
    tapGestureRecognizer.numberOfTapsRequired = 1
    tapGestureRecognizer.require(toFail: doubleTapGestureRecognizer)
    tableView.addGestureRecognizer(tapGestureRecognizer)
}

func handleTapGesture(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}

func handleDoubleTap(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}
```

Lea Métodos personalizados de selección de UITableViewCells. en línea:

<https://riptutorial.com/es/ios/topic/9961/metodos-personalizados-de-seleccion-de-uitableviewcells->

Capítulo 98: Métodos personalizados de selección de UITableViewCells.

Examples

Distinción entre selección simple y doble en fila.

Un ejemplo de implementación de UITableView que permite detectar si la celda se ha tocado una o dos veces.

```
override func viewDidLoad() {
    viewDidLoad()

    let doubleTapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleDoubleTap(sender:)))
    doubleTapGestureRecognizer.numberOfTapsRequired = 2
    tableView.addGestureRecognizer(doubleTapGestureRecognizer)

    let tapGestureRecognizer = UITapGestureRecognizer(target: self, action:
#selector(handleTapGesture(sender:)))
    tapGestureRecognizer.numberOfTapsRequired = 1
    tapGestureRecognizer.require(toFail: doubleTapGestureRecognizer)
    tableView.addGestureRecognizer(tapGestureRecognizer)
}

func handleTapGesture(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}

func handleDoubleTap(sender: UITapGestureRecognizer) {
    let touchPoint = sender.location(in: tableView)
    if let indexPath = tableView.indexPathForRow(at: touchPoint) {
        print(indexPath)
    }
}
```

Lea Métodos personalizados de selección de UITableViewCells. en línea:

<https://riptutorial.com/es/ios/topic/9962/metodos-personalizados-de-seleccion-de-uitableviewcells->

Capítulo 99: MKDistanceFormatter

Examples

Cadena de distancia

Dado un `CLLocationDistance` (simplemente un `Double` representa los medidores), `CLLocationDistance` una cadena legible por el usuario:

```
let distance = CLLocationDistance(42)
let formatter = MKDistanceFormatter()
let answer = formatter.stringFromDistance(distance)
// answer = "150 feet"
```

C objetivo

```
CLLocationDistance distance=42;
MKDistanceFormatter *formatter=[[MKDistanceFormatter alloc]init];
NSString *answer=[formatter stringFromDistance:distance];
// answer = "150 feet"
```

Por defecto, esto respeta la configuración regional del usuario.

Unidades de distancia

import Mapkit units import Mapkit **Set a uno de** `.Default`, `.Metric`, `.Imperial`, `.ImperialWithYards` :

```
formatter.units = .Metric
var answer = formatter.stringFromDistance(distance)
// "40 m"

formatter.units = .ImperialWithYards
answer = formatter.stringFromDistance(distance)
// "50 yards"
```

C objetivo

```
MKDistanceFormatter *formatter=[[MKDistanceFormatter alloc]init];
formatter.units=MKDistanceFormatterUnitsMetric;
NSString *answer=[formatter stringFromDistance:distance];
//40 m

formatter.units=MKDistanceFormatterUnitsImperialWithYards;
NSString *answer=[formatter stringFromDistance:distance];
//50 yards
```

Estilo de unidad

Establezca `unitStyle` en uno de `.Default`, `.Abbreviated`, `.Full` :

```
formatter.unitStyle = .Full
var answer = formatter.stringFromDistance(distance)
// "150 feet"

formatter.unitStyle = .Abbreviated
answer = formatter.stringFromDistance(distance)
// "150 ft"
```

C objetivo

```
formatter.unitStyle=MKDistanceFormatterUnitStyleFull;
NSString *answer=[formatter stringFromDistance:distance];
// "150 feet"

formatter.unitStyle=MKDistanceFormatterUnitStyleAbbreviated;
NSString *answer=[formatter stringFromDistance:distance];
// "150 ft"
```

Lea MKDistanceFormatter en línea: <https://riptutorial.com/es/ios/topic/6677/mkdistanceformatter>

Capítulo 100: MKMapView

Examples

Añadir MKMapView

Rápido

```
let mapView = MKMapView(frame: CGRect(x: 0, y: 0, width: 320, height: 500))
```

Se recomienda almacenar el `mapView` como una propiedad del `ViewController` contiene, ya que es posible que desee acceder a él en implementaciones más complejas.

C objetivo

```
self.map = [[MKMapView alloc] initWithFrame:CGRectMake(0, 0, self.view.frame.size.width, self.view.frame.size.height)];  
[self.view addSubview:self.map];
```

Cambiar tipo de mapa

Hay 5 tipos diferentes ([MKMapType](#)), `MKMapView` puede mostrar.

iPhone OS 3

.estándar

Muestra un mapa de calles que muestra la posición de todas las carreteras y algunos nombres de carreteras.

Swift 2

```
mapView.mapType = .Standard
```

Swift 3

```
mapView.mapType = .standard
```

C objetivo

```
_mapView.mapType = MKMapTypeStandard;
```



iPhone OS 3

.satélite

Muestra imágenes de satélite de la zona.

Swift 2

```
mapView.mapType = .Satellite
```

Swift 3

```
mapView.mapType = .satellite
```

C objetivo

```
_mapView.mapType = MKMapTypeSatellite;
```



iOS 9

.satelliteFlyover

Muestra una imagen satelital del área con datos de sobrevuelo donde estén disponibles.

Swift 2

```
mapView.mapType = .SatelliteFlyover
```

Swift 3

```
mapView.mapType = .satelliteFlyover
```

C objetivo

```
_mapView.mapType = MKMapTypeSatelliteFlyover;
```

iPhone OS 3

.híbrido

Muestra una imagen satelital del área con la información del nombre de la carretera y el nombre de la carretera en la parte superior.

Swift 2

```
mapView.mapType = .Hybrid
```

Swift 3

```
mapView.mapType = .hybrid
```

C objetivo

```
_mapView.mapType = MKMapTypeHybrid;
```



iOS 9

.hybridFlyover

Muestra una imagen satelital híbrida con datos de sobrevuelo cuando estén disponibles.

Swift 2

```
mapView.mapType = .HybridFlyover
```

Swift 3

```
mapView.mapType = .hybridFlyover
```

C objetivo

```
_mapView.mapType = MKMapTypeHybridFlyover;
```

Establecer zoom / región para el mapa

Para establecer un cierto nivel de zoom, digamos que queremos ampliar la ubicación del usuario con la ubicación del usuario como centro y 2 km de área como radio. Entonces, usamos el siguiente código

```
MKUserLocation *userLocation = _mapView.userLocation;  
MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance  
(userLocation.location.coordinate, 2000, 2000);  
[_mapView setRegion:region animated:NO];
```

Implementación de búsqueda local utilizando MKLocalSearch

MKLocalSearch permite a los usuarios buscar ubicaciones usando cadenas de lenguaje natural como "gimnasio". Una vez que se completa la búsqueda, la clase devuelve una lista de ubicaciones dentro de una región específica que coinciden con la cadena de búsqueda.

Los resultados de la búsqueda están en forma de MKMapItem dentro del objeto MKLocalSearchResponse.

vamos a probar con el ejemplo

```
MKLocalSearchRequest *request =  
    [[MKLocalSearchRequest alloc] init]; //initialising search request  
request.naturalLanguageQuery = @"Gym"; // adding query  
request.region = _mapView.region; //setting region  
MKLocalSearch *search =  
    [[MKLocalSearch alloc] initWithRequest:request]; //initiate search  
  
[search startWithCompletionHandler:^(MKLocalSearchResponse  
    *response, NSError *error)  
{  
    if (response.mapItems.count == 0)  
        NSLog(@"No Matches");  
    else  
        for (MKMapItem *item in response.mapItems)  
        {  
            NSLog(@"name = %@", item.name);  
            NSLog(@"Phone = %@", item.phoneNumber);  
        }  
}];
```

OpenStreetMap Tile-Overlay

En algunos casos, es posible que no desee utilizar los mapas predeterminados, proporciona

Apple.

Puede agregar una superposición a su `mapView` que contenga mosaicos personalizados, por ejemplo, de [OpenStreetMap](#) .

Supongamos que `self.mapView` es su `MKMapView` que ya ha agregado a su `ViewController` .

Al principio, su `ViewController` debe cumplir con el protocolo `MKMapViewDelegate` .

```
class MyViewController: UIViewController, MKMapViewDelegate
```

Luego tienes que configurar el `ViewController` como delegado de `mapView`

```
mapView.delegate = self
```

A continuación, configura la superposición para el mapa. Necesitarás una plantilla de URL para esto. La URL debe ser similar a esta en todos los servidores de mosaicos e incluso si almacena los datos del mapa sin conexión: `http://tile.openstreetmap.org/{z}/{x}/{y}.png`

```
let urlTeplate = "http://tile.openstreetmap.org/{z}/{x}/{y}.png"  
let overlay = MKTileOverlay(urlTemplate: urlTeplate)  
overlay.canReplaceMapContent = true
```

Después de configurar la superposición, debe agregarla a su `mapView` .

```
mapView.add(overlay, level: .aboveLabels)
```

Para usar mapas personalizados, se recomienda usar `.aboveLabels` para `level` . De lo contrario, las etiquetas predeterminadas serían visibles en su mapa personalizado. Si desea ver las etiquetas predeterminadas, puede elegir `.aboveRoads` aquí.

Si ejecutara su proyecto ahora, reconocería que su mapa aún mostraría el mapa predeterminado:



Esto se debe a que aún no le hemos dicho a `mapView`, cómo representar la superposición. Esta es la razón por la que tuvo que establecer el delegado antes. Ahora puede agregar `func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer` a su controlador de vista:

```
func mapView(_ mapView: MKMapView, rendererFor overlay: MKOverlay) -> MKOverlayRenderer {
    if overlay is MKTileOverlay {
        let renderer = MKTileOverlayRenderer(overlay: overlay)
        return renderer
    } else {
        return MKTileOverlayRenderer()
    }
}
```

Esto devolverá el `MKOverlayRenderer` correcto a su `mapView`. Si ejecuta su proyecto ahora, debería ver un mapa como este:



Si desea mostrar otro mapa, solo tiene que cambiar la plantilla de URL. Hay una [lista de servidores de azulejos](#) en la Wiki de OSM.

Mostrar ejemplo de UserLocation y UserTracking

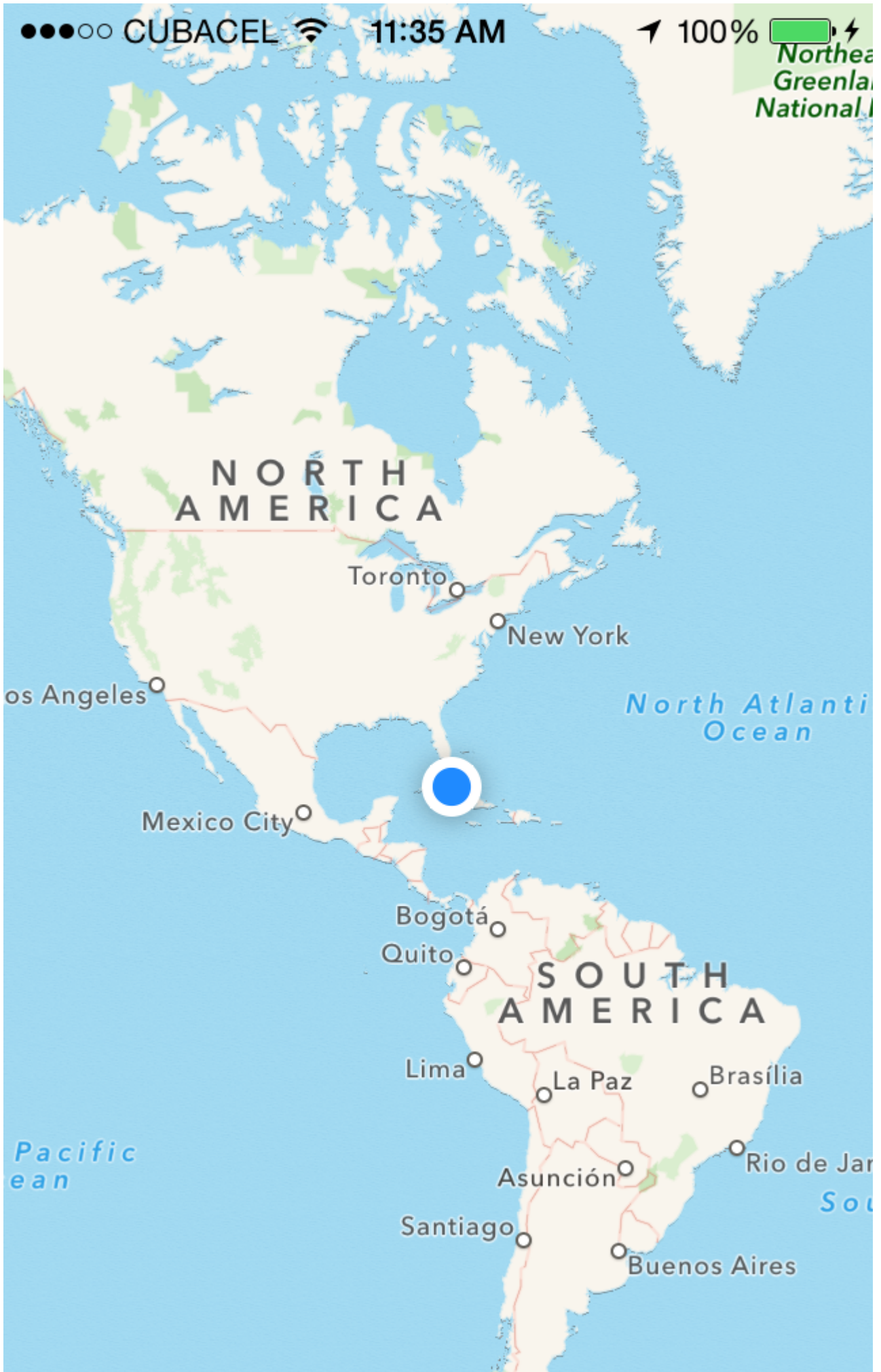
Esto mostrará la ubicación del usuario en el mapa.

C objetivo

```
[self.map setShowsUserLocation:YES];
```

Rápido

```
self.map?.showsUserLocation = true
```



muestre una coordenada en un nivel de zoom en lugar de establecer una región para mostrar. Esta funcionalidad no está implementada de forma predeterminada, por lo que necesita extender `MKMapView` con métodos que realizan el cálculo complejo de una *coordenada y nivel de zoom* a una `MKCoordinateRegion`.

```
let MERCATOR_OFFSET = 268435456.0
let MERCATOR_RADIUS = 85445659.44705395
let DEGREES = 180.0

public extension MKMapView {

    //MARK: Map Conversion Methods

    private func longitudeToPixelSpaceX(longitude:Double)->Double{
        return round(MERCATOR_OFFSET + MERCATOR_RADIUS * longitude * M_PI / DEGREES)
    }

    private func latitudeToPixelSpaceY(latitude:Double)->Double{
        return round(MERCATOR_OFFSET - MERCATOR_RADIUS * log((1 + sin(latitude * M_PI /
DEGREES)) / (1 - sin(latitude * M_PI / DEGREES))) / 2.0)
    }

    private func pixelSpaceXToLongitude(pixelX:Double)->Double{
        return ((round(pixelX) - MERCATOR_OFFSET) / MERCATOR_RADIUS) * DEGREES / M_PI
    }

    private func pixelSpaceYToLatitude(pixelY:Double)->Double{
        return (M_PI / 2.0 - 2.0 * atan(exp((round(pixelY) - MERCATOR_OFFSET) /
MERCATOR_RADIUS))) * DEGREES / M_PI
    }

    private func coordinateSpanWithCenterCoordinate(centerCoordinate:CLLocationCoordinate2D,
zoomLevel:Double)->MKCoordinateSpan{
        // convert center coordinate to pixel space
        let centerPixelX = longitudeToPixelSpaceX(longitude: centerCoordinate.longitude)
        let centerPixelY = latitudeToPixelSpaceY(latitude: centerCoordinate.latitude)
        print(centerCoordinate)
        // determine the scale value from the zoom level
        let zoomExponent:Double = 20.0 - zoomLevel
        let zoomScale:Double = pow(2.0, zoomExponent)
        // scale the map's size in pixel space
        let mapSizeInPixels = self.bounds.size
        let scaledMapWidth = Double(mapSizeInPixels.width) * zoomScale
        let scaledMapHeight = Double(mapSizeInPixels.height) * zoomScale
        // figure out the position of the top-left pixel
        let topLeftPixelX = centerPixelX - (scaledMapWidth / 2.0)
        let topLeftPixelY = centerPixelY - (scaledMapHeight / 2.0)
        // find delta between left and right longitudes
        let minLng = pixelSpaceXToLongitude(pixelX: topLeftPixelX)
        let maxLng = pixelSpaceXToLongitude(pixelX: topLeftPixelX + scaledMapWidth)
        let longitudeDelta = maxLng - minLng
        let minLat = pixelSpaceYToLatitude(pixelY: topLeftPixelY)
        let maxLat = pixelSpaceYToLatitude(pixelY: topLeftPixelY + scaledMapHeight)
        let latitudeDelta = -1.0 * (maxLat - minLat)
        return MKCoordinateSpan(latitudeDelta: latitudeDelta, longitudeDelta: longitudeDelta)
    }

    /**
     Sets the center of the `MKMapView` to a `CLLocationCoordinate2D` with a custom zoom-
     level. There is no need to set a region manually. :-)
```



```

- author: Mylene Bayan (on GitHub)
*/
public func setCenter(_ coordinate:CLLocationCoordinate2D, zoomLevel:Double,
animated:Bool){
    // clamp large numbers to 28
    var zoomLevel = zoomLevel
    zoomLevel = min(zoomLevel, 28)
    // use the zoom level to compute the region
    print(coordinate)
    let span = self.coordinateSpanWithCenterCoordinate(centerCoordinate: coordinate,
zoomLevel: zoomLevel)
    let region = MKCoordinateRegionMake(coordinate, span)
    if region.center.longitude == -180.00000000{
        print("Invalid Region")
    }
    else{
        self.setRegion(region, animated: animated)
    }
}
}
}

```

(La versión original de Swift 2 de [Mylene Bayan](#) se puede encontrar en [GitHub](#))

Después de implementar esta `extension` , puede establecer la coordenada central de la siguiente manera:

```

let centerCoordinate = CLLocationCoordinate2DMake(48.136315, 11.5752901) //latitude, longitude
mapView?.setCenter(centerCoordinate, zoomLevel: 15, animated: true)

```

`zoomLevel` es un valor `Double` , generalmente entre 0 y 21 (que es un nivel de zoom muy alto), pero se permiten valores de hasta 28 .

Trabajando Con Anotación

Obtener toda la anotación

```

//following method returns all annotations object added on map
NSArray *allAnnotations = mapView.annotations;

```

Obtener vista de anotación

```

for (id<MKAnnotation> annotation in mapView.annotations)
{
    MKAnnotationView* annotationView = [mapView viewForAnnotation:annotation];
    if (annotationView)
    {
        // Do something with annotation view
        // for e.g change image of annotation view
        annotationView.image = [UIImage imageNamed:@"SelectedPin.png"];
    }
}

```

Eliminar todas las anotaciones

```
[mapView removeAnnotations:mapView.annotations]
```

Eliminar una sola anotación

```
//getting all Annotation  
NSArray *allAnnotations = self.myMapView.annotations;  
  
if (allAnnotations.count > 0)  
{  
    //getting first annoation  
    id <MKAnnotation> annotation=[allAnnotations firstObject];  
  
    //removing annotation  
    [mapView removeAnnotation:annotation];  
}
```

Ajuste el rectángulo visible de la vista del mapa para mostrar todas las anotaciones

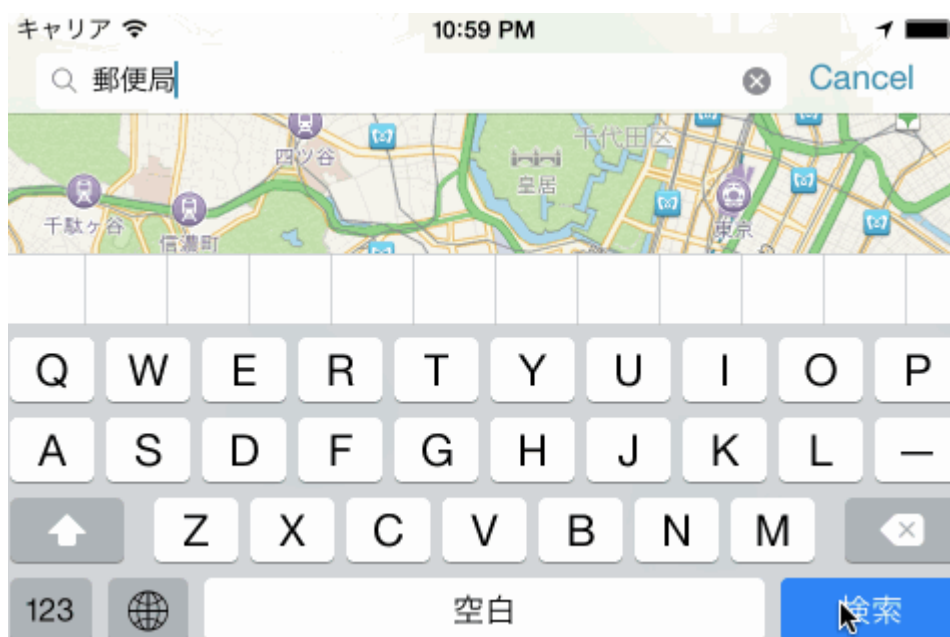
Rápido:

```
mapView.showAnnotations(mapView.annotations, animated: true)
```

C objetivo:

```
[mapView showAnnotations:mapView.annotations animated:YES];
```

Manifestación:



Lea MKMapView en línea: <https://riptutorial.com/es/ios/topic/915/mkmapview>

Capítulo 101: ModeloPresentaciónEstilos

Introducción

Los estilos de presentación modal se utilizan cuando se está haciendo la transición de un controlador de vista a otro. Hay 2 formas de lograr esta personalización. Uno es a través del código y otro a través de Interface Builder (usando segues). Este efecto se logra al establecer la variable `modalPresentationStyle` en una instancia de `UIModalPresentationStyle` enum.

`modalPresentationStyle` propiedad `modalPresentationStyle` es una variable de clase de `UIViewController` y se utiliza para especificar cómo se presenta un `ViewController` en la pantalla.

Observaciones

Siempre recuerda la siguiente mención de Apple.

En un entorno horizontalmente compacto, los controladores de vista modal siempre se presentan en pantalla completa. En un entorno horizontal regular, hay varias opciones de presentación diferentes.

Examples

Explorando ModalPresentationStyle usando Interface Builder

Esta será una aplicación muy básica que ilustrará diferentes `ModalpresentationStyle` en iOS. De acuerdo con la documentación que se encuentra [aquí](#) , hay 9 valores diferentes para `UIModalPresentationStyle` que son los siguientes,

1. `fullScreen`
2. `pageSheet`
3. `formSheet`
4. `currentContext`
5. `custom`
6. `overFullScreen`
7. `overCurrentContext`
8. `popover`
9. `none`

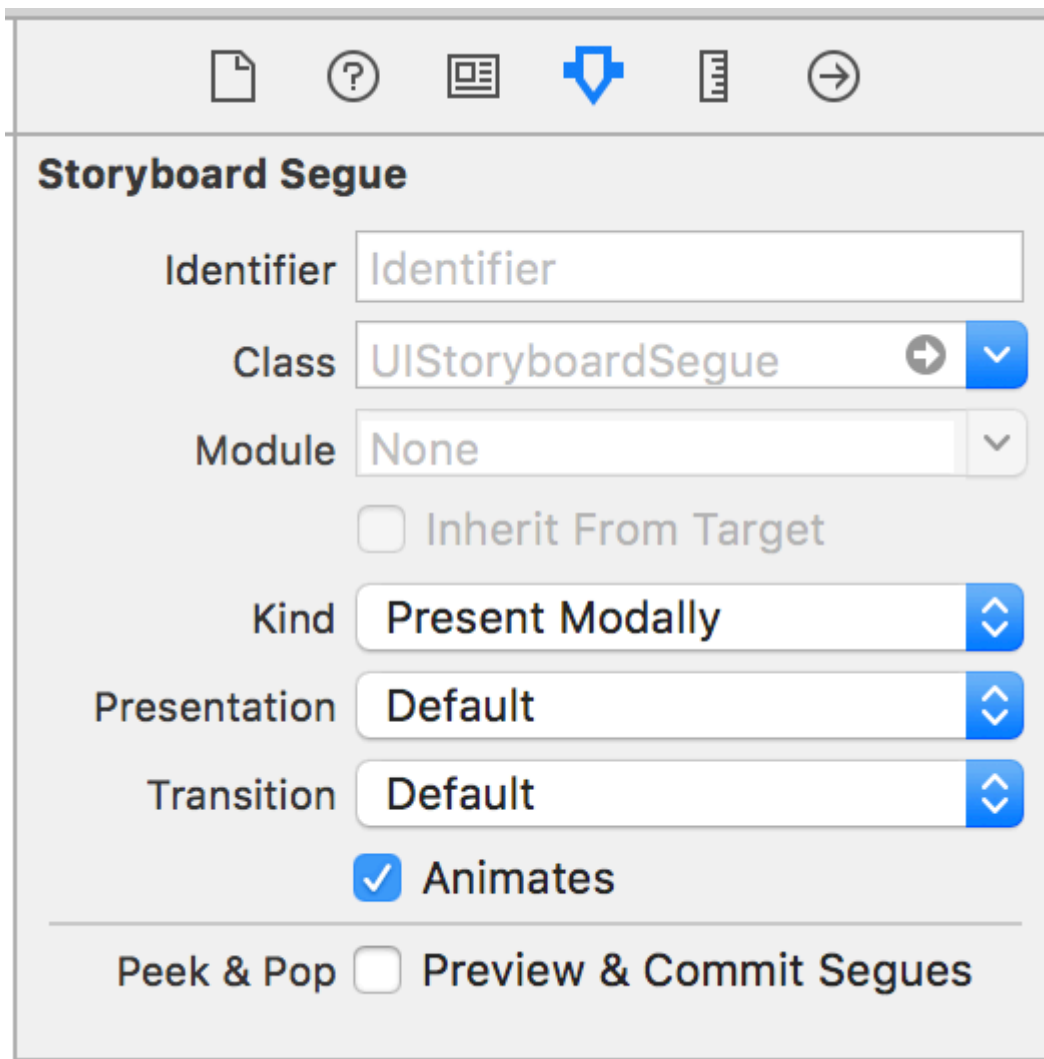
Para configurar un proyecto, simplemente cree un proyecto de iOS normal y agregue 2

`ViewControllers` . Coloque un `UIButton` en su `ViewController` inicial y conéctelo al 2nd `ViewController` través de un mecanismo de `Target -> Action` . Para distinguir ambos `ViewControllers` , establezca la propiedad de fondo de `UIView` en `ViewController` otro color. Si todo va bien, tu Interface Builder debería tener un aspecto similar a esto.



Button

(para obtener más información sobre por qué iPad, consulte la sección de Comentarios). Una vez que haya terminado de configurar su proyecto, seleccione el segmento y vaya al `attributes inspector` . Deberías poder ver algo como esto,



Establezca la propiedad `kind` en `Present Modally` .


Ahora, no veremos todos los efectos en este ejemplo ya que algunos de ellos requieren un poco de código.

Vamos a empezar con `fullscreen` . Este efecto se selecciona de forma predeterminada cuando selecciona `Present Modally` en la pestaña `Kind` . Cuando construyes y ejecutas, el segundo `ViewController` ocuparía la pantalla completa de tu iPad.



. En esta opción, cuando el dispositivo está en modo vertical, el segundo `ViewController` es similar a la pantalla completa, pero en el modo horizontal, el segundo `ViewController` es mucho más estrecho que el ancho del dispositivo. Además, cualquier contenido no cubierto por 2nd `ViewController` se atenuará.



Carrier 

6



se coloca en el centro del dispositivo y el tamaño es más pequeño que el del dispositivo. También cuando el dispositivo está en modo horizontal y el teclado es visible, la posición de la vista se ajusta hacia arriba para mostrar el `ViewController` .



Present as Popover en la pestaña Kind . El 2nd ViewController se presenta como una pequeña ventana emergente (se puede configurar el tamaño). El contenido de fondo está atenuado. Cualquier toque fuera del popover descartaría el popover. Su Attributes Inspector debe verse algo como esto,

Storyboard Segue

Identifier

Class

Module

Inherit From Target

Kind

Directions Up Down
 Left Right

Anchor


Passthrough

Animates

Peek & Pop Preview & Commit Segues

Anchor es el elemento de la interfaz de usuario al que desea que apunte su flecha emergente. Directions son las direcciones en las que permite que su Anchor emergente señale.



Carrier 

6:29 PM



Button

Capítulo 102: Modismos de inicialización

Examples

Ajuste a tuplas para evitar la repetición de código.

Evite la repetición de código en los constructores al establecer una tupla de variables con un liner:

```
class Contact: UIView
{
    private var message: UILabel
    private var phone: UITextView

    required init?(coder aDecoder: NSCoder) {
        (message, phone) = self.dynamicType.setUp()
        super.init(coder: aDecoder)
    }

    override func awakeFromNib() {
        (message, phone) = self.dynamicType.setUp()
        super.awakeFromNib()
    }

    override init(frame: CGRect) {
        (message, phone) = self.dynamicType.setUp()
        super.init(frame: frame)
    }

    private static func setUp(){
        let message = UILabel() // ...
        let phone = UITextView() // ...
        return (message, phone)
    }
}
```

Inicializar con constantes posicionales.

```
let mySwitch: UISwitch = {
    view.addSubview($0)
    $0.addTarget(self, action: "action", forControlEvents: .TouchUpInside)
    return $0
}(UISwitch())
```

Inicializar atributos en didSet

```
@IBOutlet weak var title: UILabel! {
    didSet {
        label.textColor = UIColor.redColor()
        label.font = UIFont.systemFont(ofSize: 20)
        label.backgroundColor = UIColor.blueColor()
    }
}
```

También es posible establecer un valor e inicializarlo:

```
private var loginButton = UIButton() {
    didSet(oldValue) {
        loginButton.addTarget(self, action: #selector(LoginController.didClickLogin),
            forControlEvents: .TouchUpInside)
    }
}
```

Agrupar puntos de venta en un objeto NSO personalizado

Mueva cada salida a un objeto NSO. Luego arrastre un Objeto desde la biblioteca a la escena del controlador del guión gráfico y enganche los elementos allí.

```
class ContactFormStyle: NSObject
{
    @IBOutlet private weak var message: UILabel! {
        didSet {
            message.font = UIFont.systemFont(ofSize: 12)
            message.textColor = UIColor.blackColor()
        }
    }
}

class ContactFormVC: UIViewController
{
    @IBOutlet private var style: ContactFormStyle!
}
```

Inicializar con entonces

Esta es similar en sintaxis al ejemplo que se inicializa usando constantes posicionales, pero requiere la extensión `Then` de <https://github.com/devxoul/Then> (adjunta a continuación).

```
let label = UILabel().then {
    $0.textAlignment = .Center
    $0.textColor = UIColor.blackColor()
    $0.text = "Hello, World!"
}
```

La extensión `Then` :

```
import Foundation

public protocol Then {}

extension Then
{
    public func then(@noescape block: inout Self -> Void) -> Self {
        var copy = self
        block(&copy)
        return copy
    }
}
```

```
extension NSObject: Then {}
```

Método de fábrica con bloque

```
internal fun<Type> Init<Type>(value : Type, block: @noescape (object: Type) -> Void) -> Type  
{  
    block(object: value)  
    return value  
}
```

Uso:

```
Init(UILabel(frame: CGRect.zero)) {  
    $0.backgroundColor = UIColor.blackColor()  
}
```

Lea Modismos de inicialización en línea: <https://riptutorial.com/es/ios/topic/3513/modismos-de-inicializacion>

Capítulo 103: Modos de fondo

Introducción

Ser receptivo es una necesidad para cada aplicación. Los usuarios desean tener aplicaciones que tengan su contenido listo cuando las abran, por lo que los desarrolladores deben usar los modos de fondo para hacer que sus aplicaciones sean más fáciles de usar.

Examples

Activar la capacidad de los modos de fondo

1. Ve a Xcode y abre tu proyecto.
2. En el destino de su aplicación, vaya a la pestaña Capacidades.
3. Activar los modos de fondo.



O. ↕

General

Capabilities

Resource Tags



Background Modes

- Modes:
- Audio, AirPlay, and Picture in Picture
 - Location updates
 - Voice over IP
 - Newsstand downloads
 - External accessory communication
 - Uses Bluetooth LE accessories
 - Acts as a Bluetooth LE accessory
 - Background fetch
 - Remote notifications

Steps: ✓ Add the Required Background Modes

Búsqueda de fondo

La obtención de fondos es un nuevo modo que permite que su aplicación aparezca siempre actualizada con la información más reciente y minimice el impacto en la batería. Puede descargar feeds dentro de intervalos de tiempo fijos con esta capacidad.

Para empezar:

1- Verifique la captura de fondo en la pantalla de capacidades en Xcode.

2- En el método de `application(_:didFinishLaunchingWithOptions:)` en `AppDelegate`, agregue:

Rápido

```
UIApplication.shared.setMinimumBackgroundFetchInterval(UINavigationControllerBackgroundFetchIntervalMinimum)
```

C objetivo

```
[[UIApplication shared]
setMinimumBackgroundFetchInterval:UIApplicationBackgroundFetchIntervalMinimum]
```

En lugar de `UIApplicationBackgroundFetchIntervalMinimum` , puede usar cualquier valor `CGFloat` para establecer los intervalos de búsqueda.

3- Debe implementar la `application(_:performFetchWithCompletionHandler:)` . Agregue eso a su `AppDelegate` :

Rápido

```
func application(_ application: UIApplication, performFetchWithCompletionHandler
completionHandler: @escaping (UIBackgroundFetchResult) -> Void) {
    // your code here
}
```

Prueba de búsqueda de fondo

- 1- Ejecute la aplicación en un dispositivo real y adjúntela al depurador de Xcode.
- 2- En el menú Depurar, seleccione **Simular búsqueda de fondo** :

Pause

⌘ Y

Continue To Current Line

⌘ C

Step Over

F6

Step Into

F7

Step Out

F8

Step Over Instruction

⌘ F6

Step Over Thread

⌘ ↑ F6

Step Into Instruction

⌘ F7

Step Into Thread

⌘ ↑ F7

Capture GPU Frame

GPU Overrides



Simulate Location



Simulate Background Fetch

Simulate UI Snapshot

iCloud



View Debugging



Deactivate Breakpoints

⌘ Y

Breakpoints



Debug Workflow



Capítulo 104: Modos de fondo y eventos

Examples

Reproducir audio en segundo plano

Agregue una clave llamada **Modos de fondo requeridos** en el archivo de lista de propiedades (.plist).

como la siguiente imagen ..

Key	Type	Value
▼ Information Property List	Dictionary	(16 items)
Localization native development r...	String	en
Bundle display name	String	
Executable file	String	\${EXECUTABLE_NAME}
▶ Icon files	Array	(14 items)
Bundle identifier	String	
InfoDictionary version	String	6.0
Bundle name	String	\${PRODUCT_NAME}
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.1
Bundle creator OS Type code	String	????
Bundle version	String	1.1
Application requires iPhone envir...	Boolean	YES
▼ Required background modes	Array	(1 item)
Item 0	String	App plays audio or streams audio/video using AirPlay
Icon already includes gloss effects	Boolean	YES
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations	Array	(1 item)

Y añade el siguiente código en

AppDelegate.h

```
#import <AVFoundation/AVFoundation.h>
#import <AudioToolbox/AudioToolbox.h>
```

AppDelegate.m

en la aplicación didFinishLaunchingWithOptions

```
[[AVAudioSession sharedInstance] setDelegate:self];
[[AVAudioSession sharedInstance] setCategory:AVAudioSessionCategoryPlayback error:nil];
[[AVAudioSession sharedInstance] setActive:YES error:nil];
```

```
[[UIApplication sharedApplication] beginReceivingRemoteControlEvents];

UInt32 size = sizeof(CFStringRef);
CFStringRef route;
AudioSessionGetProperty(kAudioSessionProperty_AudioRoute, &size, &route);
NSLog(@"route = %@", route);
```

Si desea cambios según los eventos, debe agregar el siguiente código en AppDelegate.m

```
- (void)remoteControlReceivedWithEvent:(UIEvent *)theEvent {

    if (theEvent.type == UIEventTypeRemoteControl)    {
        switch(theEvent.subtype)    {
            case UIEventSubtypeRemoteControlPlay:
                [[NSNotificationCenter defaultCenter] postNotificationName:@"TogglePlayPause"
object:nil];
                break;
            case UIEventSubtypeRemoteControlPause:
                [[NSNotificationCenter defaultCenter] postNotificationName:@"TogglePlayPause"
object:nil];
                break;
            case UIEventSubtypeRemoteControlStop:
                break;
            case UIEventSubtypeRemoteControlTogglePlayPause:
                [[NSNotificationCenter defaultCenter] postNotificationName:@"TogglePlayPause"
object:nil];
                break;
            default:
                return;
        }
    }
}
```

Basado en la notificación hay que trabajar en ello ..

Lea Modos de fondo y eventos en línea: <https://riptutorial.com/es/ios/topic/3515/modos-de-fondo-y-eventos>

Capítulo 105: MPMediaPickerDelegate

Observaciones

Consulte la [documentación de Apple](#) para obtener más información sobre la privacidad.

Asegúrese de que la aplicación de música esté disponible en su iPhone. No funcionará en el simulador.

Examples

Cargue música con MPMediaPickerControllerDelegate y reproduzca con AVAudioPlayer

Ir a través de los pasos:

- Agregue 'NSAppleMusicUsageDescription' a su Info.plist para la autoridad de privacidad.
- Asegúrate de que tu música esté disponible en tu iPhone. No funcionará en el simulador.

iOS 10.0.1

```
import UIKit
import AVFoundation
import MediaPlayer

class ViewController: UIViewController, MPMediaPickerControllerDelegate {

    var avMusicPlayer: AVAudioPlayer!
    var mpMediaPicker: MPMediaPickerController!
    var mediaItems = [MPMediaItem]()
    let currentIndex = 0

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    func audioPlayerDidFinishPlaying(_ player: AVAudioPlayer, successfully flag: Bool){
        //What to do?
    }

    func mediaPicker(_ mediaPicker: MPMediaPickerController, didPickMediaItems
mediaItemCollection: MPMediaItemCollection) {
        mediaItems = mediaItemCollection.items
        updatePlayer()
        self.dismiss(animated: true, completion: nil)
    }

    func updatePlayer(){
        let item = mediaItems[currentIndex]
        // DO-TRY-CATCH try to setup AVAudioPlayer with the path, if successful, sets up the
AVMusicPlayer, and song values.
        if let path: NSURL = item.assetURL as NSURL? {
```

```

do
{
    avMusicPlayer = try AVAudioPlayer(contentsOf: path as URL)
    avMusicPlayer.enableRate = true
    avMusicPlayer.rate = 1.0
    avMusicPlayer.numberOfLoops = 0
    avMusicPlayer.currentTime = 0
}
catch
{
    avMusicPlayer = nil
}
}

@IBAction func Play(_ sender: AnyObject) {
    //AVMusicPlayer.deviceCurrentTime
    avMusicPlayer.play()
}

@IBAction func Stop(_ sender: AnyObject) {
    avMusicPlayer.stop()
}

@IBAction func picker(_ sender: AnyObject) {
    mpMediapicker = MPMediaPickerController.self(mediaTypes:MPMediaType.music)
    mpMediapicker.allowsPickingMultipleItems = false
    mpMediapicker.delegate = self
    self.present(mpMediapicker, animated: true, completion: nil)
}
}

```

Lea MPMediaPickerDelegate en línea:

<https://riptutorial.com/es/ios/topic/7299/mpmediapickerdelegate>

Capítulo 106: MPVolumeView

Introducción

La clase MPVolumeView es vista de volumen para presentar al usuario un control deslizante para configurar el volumen de salida de audio del sistema y un botón para elegir la ruta de salida de audio.

Observaciones

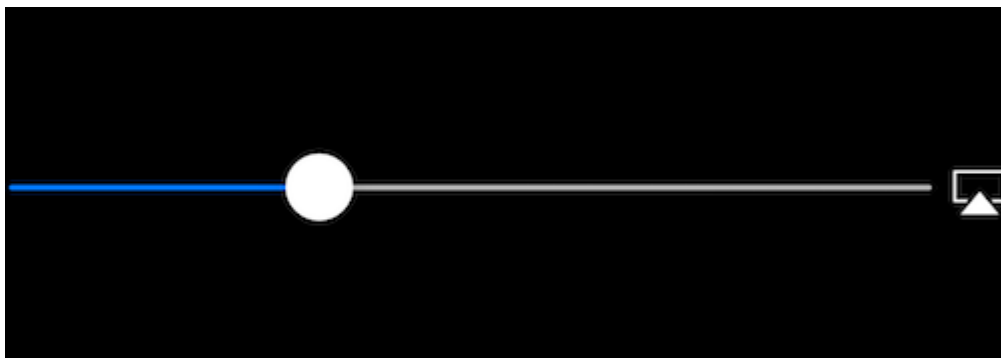
MPVolumeView solo se muestra cuando se construye y ejecuta en un dispositivo iOS real y no funciona en un simulador.

Examples

Añadiendo un MPVolumeView

```
// Add MPVolumeView in a holder view
let mpVolumeHolderView = UIView(frame: CGRect(x: 0, y: view.bounds.midY, width:
view.bounds.width, height: view.bounds.height))
// Set the holder view's background color to transparent
mpVolumeHolderView.backgroundColor = .clear
let mpVolume = MPVolumeView(frame: mpVolumeHolderView.bounds)
mpVolume.showsRouteButton = true
mpVolumeHolderView.addSubview(mpVolume)
view.addSubview(mpVolumeHolderView)
// the volume view is white, set the parent background to black to show it better in this
example
view.backgroundColor = .black
```

!!! Una nota muy importante es que el MPVolumeView solo funciona en un dispositivo real y no en un simulador.



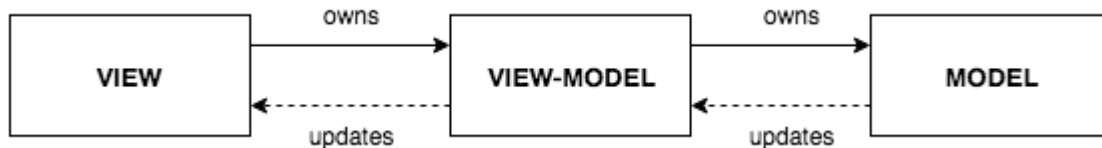
Lea MPVolumeView en línea: <https://riptutorial.com/es/ios/topic/9038/mpvolumeview>

Capítulo 107: MVVM

Examples

MVVM sin programación reactiva

Comenzaré con una explicación muy breve de lo que es y por qué usar el patrón de diseño Model-View-ViewModel (MVVM) en sus aplicaciones iOS. Cuando apareció iOS por primera vez, Apple sugirió usar MVC (Model-View-Controller) como patrón de diseño. Lo mostraron en todos sus ejemplos y todos los primeros desarrolladores se alegraron de usarlo porque separaba muy bien las preocupaciones entre la lógica de negocios y la interfaz de usuario. A medida que las aplicaciones se hacían más grandes y complejas, apareció un nuevo problema llamado Massive View Controllers (MVC). Debido a que toda la lógica de negocios se agregó en el ViewController, con el tiempo por lo general se volvieron demasiado grandes y complejos. Para evitar el problema de MVC, se introdujo un nuevo patrón de diseño en el mundo de iOS: Model-View-ViewModel (MVVM).



El diagrama de arriba muestra cómo se ve MVVM. Tiene un ViewController + View estándar (en el guión gráfico, XIB o Code), que actúa como Vista de MVVM (en el texto posterior: Vista hará referencia a la Vista de MVVM). Una vista tiene una referencia a un ViewModel, donde está nuestra lógica de negocios. Es importante notar que ViewModel no sabe nada sobre la Vista y nunca tiene una referencia a la vista. ViewModel tiene una referencia a un modelo.

Esto es suficiente con una parte teórica de la MVVM. Más sobre esto se puede leer [aquí](#).

Uno de los **principales problemas con MVVM** es cómo actualizar la Vista a través del ViewModel cuando ViewModel no tiene referencias y ni siquiera sabe nada sobre la Vista.

La parte principal de este ejemplo es mostrar cómo usar MVVM (más precisamente, cómo enlazar ViewModel y View) sin ninguna programación reactiva (ReactiveCocoa, ReactiveSwift o RxSwift). Solo como una nota: si desea usar la programación reactiva, incluso mejor, ya que los enlaces MVVM se hacen realmente fáciles de usar. Pero este ejemplo es sobre cómo usar MVVM sin programación reactiva.

Vamos a crear un ejemplo simple para demostrar cómo usar MVVM.

Nuestro `MVVMExampleViewController` es un ViewController simple con una etiqueta y un botón. Cuando se presiona el botón, el texto de la etiqueta debe establecerse en 'Hola'. Desde que decidir qué hacer con la interacción del usuario con el usuario es parte de la lógica empresarial, ViewModel tendrá que decidir qué hacer cuando el usuario presiona el botón. La vista de MVVM no debería hacer ninguna lógica de negocios.

```

class MVVMEexampleViewController: UIViewController {

    @IBOutlet weak var helloLabel: UILabel!

    var viewModel: MVVMEexampleViewModel?

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction func sayHelloButtonPressed(_ sender: UIButton) {
        viewModel?.userTriggeredSayHelloButton()
    }
}

```

MVVMEexampleViewModel es un ViewModel simple.

```

class MVVMEexampleViewModel {

    func userTriggeredSayHelloButton() {
        // How to update View's label when there is no reference to the View??
    }
}

```

Podría preguntarse cómo establecer la referencia de ViewModel en la Vista. Generalmente lo hago cuando se está inicializando ViewController o antes de que se muestre. Para este ejemplo simple, haría algo como esto en el AppDelegate :

```

func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    if let rootVC = window?.rootViewController as? MVVMEexampleViewController {
        let viewModel = MVVMEexampleViewModel()
        rootVC.viewModel = viewModel
    }

    return true
}

```

La verdadera pregunta ahora es: ¿cómo actualizar View from ViewModel sin dar una referencia a View to the ViewModel? (Recuerda, no usaremos ninguna de las bibliotecas de React Programming iOS)

Se podría pensar en usar KVO, pero eso solo complicaría las cosas demasiado. Algunas personas inteligentes han pensado en el tema y han creado la [biblioteca de Bond](#) . La biblioteca puede parecer complicada y un poco más difícil de entender al principio, así que solo tomaré una pequeña parte de ella y haré que nuestra MVVM sea completamente funcional.

Introduzcamos la clase `Dynamic` , que es el núcleo de nuestro patrón MVVM simple pero completamente funcional.

```

class Dynamic<T> {
    typealias Listener = (T) -> Void
    var listener: Listener?

    func bind(_ listener: Listener?) {

```

```

        self.listener = listener
    }

    func bindAndFire(_ listener: Listener?) {
        self.listener = listener
        listener?(value)
    }

    var value: T {
        didSet {
            listener?(value)
        }
    }

    init(_ v: T) {
        value = v
    }
}

```

Dynamic **clase** Dynamic **usa** genéricos y cierres para vincular nuestro ViewModel con nuestra vista. No entraré en detalles sobre esta clase, podemos hacerlo en los comentarios (para hacer este ejemplo más corto). MVVMExampleViewController **ahora** nuestro MVVMExampleViewController y MVVMExampleViewModel **para** usar esas clases.

Nuestro MVVMExampleViewController actualizado

```

class MVVMExampleViewController: UIViewController {

    @IBOutlet weak var helloLabel: UILabel!

    var viewModel: MVVMExampleViewModel?

    override func viewDidLoad() {
        super.viewDidLoad()
        bindViewModel()
    }

    func bindViewModel() {
        if let viewModel = viewModel {
            viewModel.helloText.bind({ (helloText) in
                DispatchQueue.main.async {
                    // When value of the helloText Dynamic variable
                    // is set or changed in the ViewModel, this code will
                    // be executed
                    self.helloLabel.text = helloText
                }
            })
        }
    }

    @IBAction func sayHelloButtonPressed(_ sender: UIButton) {
        viewModel?.userTriggeredSayHelloButton()
    }
}

```

MVVMExampleViewModel **actualizado:**

```
class MVVMExampleViewModel {  
  
    // we have to initialize the Dynamic var with the  
    // data type we want  
    var helloText = Dynamic("")  
  
    func userTriggeredSayHelloButton() {  
        // Setting the value of the Dynamic variable  
        // will trigger the closure we defined in the View  
        helloText.value = "Hello"  
    }  
}
```

Eso es. Su `ViewModel` ahora puede actualizar la `View` sin que tenga una referencia a la `View`.

Este es un ejemplo muy simple, pero creo que tienes una idea de lo poderoso que puede ser esto. No voy a entrar en detalles sobre los beneficios de MVVM, pero una vez que cambie de MVC a MVVM, no volverá. Solo inténtalo y verás por ti mismo.

Lea MVVM en línea: <https://riptutorial.com/es/ios/topic/8775/mvvm>

Capítulo 108: MyLayout

Introducción

MyLayout es un marco simple y fácil de object-c para el diseño de vista de iOS. MyLayout proporciona algunas funciones simples para construir una variedad de interfaces complejas. Integra las funciones que incluyen: Autolayout y SizeClass de iOS, cinco clases de diseño de Android, float y flex-box y bootstrap de HTML / CSS. Se puede visitar desde:

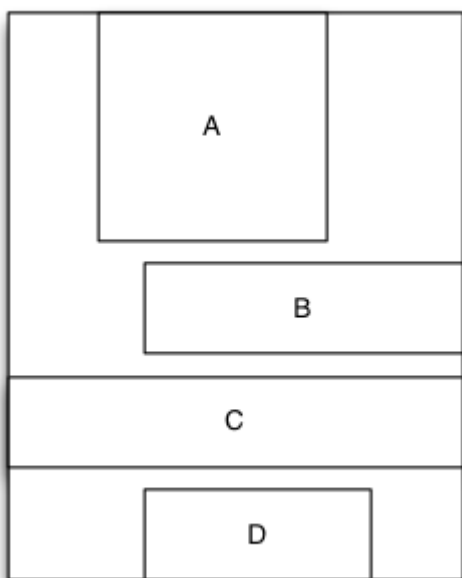
Objective-C: <https://github.com/youngsoft/MyLinearLayout> Swift:
<https://github.com/youngsoft/TangramKit>

Examples

Una demostración simple para usar MyLayout

1. Hay una vista de contenedor S cuyo ancho es 100 y la altura se ajusta a todas las subvistas de altura. hay cuatro subvistas A, B, C, D organizadas de arriba a abajo.
2. El margen izquierdo de la subvista A es el 20% del ancho de S, el margen derecho es el 30% del ancho de S, la altura es igual al ancho de A.
3. El margen izquierdo de la subvista B es 40, el ancho se llena hasta el ancho residual de S, la altura es 40. El ancho de la subvista C se llena hasta S, la altura es 40.
4. El margen derecho de la subvista D es 20, el ancho es 50% el ancho de S, la altura es 40

como la figura de abajo:



```
MyLinearLayout *S = [MyLinearLayout  
linearLayoutWithOrientation:MyLayoutViewOrientation_Vert];
```

```
S.subviewSpace = 10;
S.widthSize.equalTo(@100);

UIView *A = UIView.new;
A.leftPos.equalTo(@0.2);
A.rightPos.equalTo(@0.3);
A.heightSize.equalTo(A.widthSize);
[S addSubview:A];

UIView *B = UIView.new;
B.leftPos.equalTo(@40);
B.widthSize.equalTo(@60);
B.heightSize.equalTo(@40);
[S addSubview:B];

UIView *C = UIView.new;
C.leftPos.equalTo(@0);
C.rightPos.equalTo(@0);
C.heightSize.equalTo(@40);
[S addSubview:C];

UIView *D = UIView.new;
D.rightPos.equalTo(@20);
D.widthSize.equalTo(S.widthSize).multiply(0.5);
D.heightSize.equalTo(@40);
[S addSubview:D];
```

Lea MyLayout en línea: <https://riptutorial.com/es/ios/topic/9692/mylayout>

Capítulo 109: Notificaciones enriquecidas

Introducción

Las Notificaciones enriquecidas le permiten personalizar la apariencia de las notificaciones locales y remotas cuando aparecen en el dispositivo del usuario. Las notificaciones ricas en su mayoría incluyen `UNNotificationServiceExtension` y `UNNotificationContentExtension`, es decir, mostrar la notificación normal de manera extendida

Examples


Creando una extensión simple `UNNotificationContentExtension`

Paso 1


Haciendo el ambiente adecuado para la notificación. Asegúrate de habilitar los **modos de fondo** y la **notificación push**




PROJECT

 TestApplication

TARGETS

 **TestApplication**

 TestAppNotifConten...



Filter



Background Modes



Inter-App Audio



Keychain Sharing



Associated Domains




App Groups




General


Capa

PROJECT

 TestApplication


TARGETS

 **TestApplication**

 TestAppNotifConten...

▶  **iCloud**

▼  **Push Notifications**

▶  **Game Center**

▶  **Wallet**

▶  **Siri**

▶  **Apple Pay**

▶  **In-App Purchase**

▶  **Maps**



Filter

Paso 2: Creando una extensión UNNotificationContentExtension

Haga clic en el icono + en la parte inferior que crea una plantilla de destino y seleccione Extensión del contenido de la notificación -> a continuación -> cree un nombre para la extensión del contenido -> finalice



PROJECT



TARGETS



Choose a template for your new target:

ios watchOS tvOS macOS Cr

Application Extension



Action Extension



Audio Unit Extension



Content Blocker Extension



Custom Keyboard Extension



Intents UI Extension



Message Filter Extension



Click this + icon

- `UNNotificationDefaultContentHidden`: este booleano determina si el cuerpo predeterminado de la notificación debe estar oculto o no
- `UNNotificationCategory`: la categoría se crea en `UNUserNotificationCenter` en su aplicación. Aquí puede ser una cadena o una matriz de cadenas, por lo que cada categoría puede proporcionar diferentes tipos de datos a partir de los cuales podemos crear diferentes UI. La carga útil que enviamos debe contener el nombre de la categoría para mostrar esta extensión en particular en su aplicación. Aquí puede ser una cadena o una matriz de cadenas, por lo que cada categoría puede proporcionar diferentes tipos de datos a partir de los cuales podemos crear diferentes UI. La carga útil que enviamos debe contener el nombre de la categoría para mostrar esta extensión en particular
- `UNNotificationExtensionInitialContentSizeRatio`: el tamaño del contenido inicial, es decir, cuando se muestra la `ContentExtension` por primera vez el tamaño inicial con respecto al ancho del dispositivo. aquí 1 denota la altura será igual al ancho

Paso 4: Crear `UNNotificationAction` y `UNNotificationCategory` en nuestra aplicación

En la aplicación `AppDelegate.swift` `didFinishLaunchingWithOptions` función agregar

```
let userNotificationAction:UNNotificationAction = UNNotificationAction.init(identifier:
"ID1", title: "வணக்கம்", options: .destructive)
let userNotificationAction2:UNNotificationAction = UNNotificationAction.init(identifier:
"ID2", title: "Success", options: .destructive)

let notifCategory:UNNotificationCategory = UNNotificationCategory.init(identifier:
"CATID1", actions: [userNotificationAction,userNotificationAction2], intentIdentifiers:
["ID1","ID2"] , options:.customDismissAction)

UNUserNotificationCenter.current().delegate = self
UNUserNotificationCenter.current().setNotificationCategories([notifCategory])
UIApplication.shared.registerForRemoteNotifications()
```

Creamos dos `UNNotificationAction` con identificadores `ID1` e `ID2` y agregamos esas acciones a `UNNotificationCategory` con el identificador `CATID1` (el ID de categoría en el archivo `info.plist` de `ContentExtension` es el mismo, lo que creamos aquí se debe usar en la carga útil y en el archivo `plist`). Establecemos la categoría en el `UNUserNotificationCenter` nuestra aplicación y en la siguiente línea nos estamos registrando para la notificación que llama a la función `didRegisterForRemoteNotificationsWithDeviceToken` donde obtenemos el token del dispositivo

Nota: no olvide `import UserNotifications` en su `AppDelegate.swift` y agregue `UNUserNotificationCenterDelegate`

Paso 5: Carga útil de muestra para `NotificationContent`

```
'aps': {
  'badge': 0,
  'alert': {
    'title': "Rich Notification",
    'body': "Body of RICH NOTIFICATION",
  },
  'sound' : "default",
  'category': "CATID1",
```

```
'mutable-content':"1",
},
'attachment': "2"
```

Paso 6: Configurando el ContentExtension

Las acciones correspondientes para la categoría se muestran automáticamente mientras se realiza la acción de notificación. Veamos el código de cómo se realiza.

```
import UIKit
import UserNotifications
import UserNotificationsUI

class NotificationViewController: UIViewController, UNNotificationContentExtension {

@IBOutlet var imageView: UIImageView?
override func viewDidLoad() {
    super.viewDidLoad()
}

func didReceive(_ notification: UNNotification) {
    self.title = "Koushik"
    imageView?.backgroundColor = UIColor.clear
    imageView?.image = #imageLiteral(resourceName: "welcome.jpeg")
}

func didReceive(_ response: UNNotificationResponse, completionHandler completion: @escaping
(UNNotificationContentExtensionResponseOption) -> Void) {

    self.title = "Koushik"
    imageView?.image = UIImage.init(named: "Success.jpeg")

    if(response.actionIdentifier == "ID1")
    {
        imageView?.image = UIImage.init(named: "Success.jpeg")
    }
    else
    {
        imageView?.image = UIImage.init(named: "welcome.jpeg")
    }
}
}
```

Paso 7: Resultado

Después de recibir y presionar prolongadamente / hacer clic en Ver notificación, la notificación se verá así



Koushik



Rich Notification

Body of RICH NOTIFICATION

வணக்கம்

Success



Koushik



Rich Notification

Body of RICH NOTIFICATION

வணக்கம்

Success



Koushik



Koushik



`UNNotificationExtensionOverrideDefaultTitle` como YES. En el paso 3 le dimos a `UNNotificationExtensionDefaultContentHidden` como NO si es SÍ, entonces la notificación se verá como las imágenes 3 y 4.

Lea Notificaciones enriquecidas en línea: <https://riptutorial.com/es/ios/topic/10769/notificaciones-enriquecidas>

Capítulo 110: Notificaciones push

Sintaxis

- `UIUserNotificationSettings.types: UIUserNotificationType` // Una máscara de bits de los tipos de notificación que su aplicación puede usar
- `UIUserNotificationSettings.categories: Set` // Grupos de acciones registrados de la aplicación

Parámetros

Parámetro	Descripción
Información de usuario	Un diccionario que contiene información de notificación remota, que puede incluir un número de placa para el icono de la aplicación, sonido de alerta, mensaje de alerta, un identificador de notificación y datos personalizados.

Examples

Dispositivo de registro para notificaciones push

Para registrar su dispositivo para notificaciones push, agregue el siguiente código a su archivo `AppDelegate` en el método `didFinishLaunchingWithOptions`:

Rápido

```
func application(application: UIApplication, didFinishLaunchingWithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
    // Override point for customization after application launch.
    if UIDevice.currentDevice().systemVersion.compare(v, options: .NumericSearch) ==
    NSOrderedAscending {
        // Register for Push Notifications, if running iOS < 8
        if application.respondsToSelector("registerUserNotificationSettings:") {
            let types:UIUserNotificationType = (.Alert | .Badge | .Sound)
            let settings:UIUserNotificationSettings = UIUserNotificationSettings(forTypes:
types, categories: nil)

            application.registerUserNotificationSettings(settings)
            application.registerForRemoteNotifications()
        } else {
            // Register for Push Notifications before iOS 8
            application.registerForRemoteNotificationTypes(.Alert | .Badge | .Sound)
        }
    } else {
        var center = UNUserNotificationCenter.currentNotificationCenter()
        center.delegate = self
        center.requestAuthorizationWithOptions((UNAuthorizationOptionSound |
UNAuthorizationOptionAlert | UNAuthorizationOptionBadge)) {(granted: Bool, error: NSError) ->
Void in
```

```

        if !error {
            UIApplication.sharedApplication().registerForRemoteNotifications()
            // required to get the app to do anything at all about push notifications
            print("Push registration success.")
        } else {
            print("Push registration FAILED")
            print("ERROR: \(error.localizedFailureReason!) -
\(\(error.localizedDescription)")
            print("SUGGESTIONS: \(error.localizedRecoveryOptions) -
\(\(error.localizedRecoverySuggestion)")
        })
    }

    return true
}

```

C objetivo

```

#define SYSTEM_VERSION_LESS_THAN(v) ([[UIDevice currentDevice] systemVersion] compare:v
options:NSNumericSearch] == NSOrderedAscending)

if( SYSTEM_VERSION_LESS_THAN( @"10.0" ) )
{
    if ([application respondsToSelector:@selector(isRegisteredForRemoteNotifications)])
    {
        // iOS 8 Notifications
        [application registerUserNotificationSettings:[UIUserNotificationSettings
settingsForTypes:(UIUserNotificationTypeSound | UIUserNotificationTypeAlert |
UIUserNotificationTypeBadge) categories:nil]];

        [application registerForRemoteNotifications];
    }
    else
    {
        // iOS < 8 Notifications
        [application registerForRemoteNotificationTypes:
(UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeAlert |
UIRemoteNotificationTypeSound)];
    }
}
else
{
    UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];
    center.delegate = self;
    [center requestAuthorizationWithOptions:(UNAuthorizationOptionSound |
UNAuthorizationOptionAlert | UNAuthorizationOptionBadge) completionHandler:^(BOOL granted,
NSError * _Nullable error)
    {
        if( !error )
        {
            [[UIApplication sharedApplication] registerForRemoteNotifications]; // required
to get the app to do anything at all about push notifications
            NSLog( @"Push registration success." );
        }
        else
        {
            NSLog( @"Push registration FAILED" );
            NSLog( @"ERROR: %@ - %@", error.localizedFailureReason,

```

```

error.localizedDescription );
        NSLog( @"SUGGESTIONS: %@ - %@", error.localizedRecoveryOptions,
error.localizedRecoverySuggestion );
    }
    }];
}

//to check if your App lunch from Push notification
//-----
//Handel Push notification
if (launchOptions != nil)
{
    // Here app will open from pushnotification
    //RemoteNotification
    NSDictionary* dictionary1 = [launchOptions
objectForKey:UIApplicationLaunchOptionsRemoteNotificationKey];
    //LocalNotification
    NSDictionary* dictionary2 = [launchOptions
objectForKey:UIApplicationLaunchOptionsLocalNotificationKey];
    if (dictionary1 != nil)
    {
        //RemoteNotification Payload
        NSLog(@"Launched from push notification: %@", dictionary1);
        //here handle your push notification
    }
    if (dictionary2 != nil)
    {
        NSLog(@"Launched from dictionary2dictionary2dictionary2 notification: %@",
dictionary2);
        double delayInSeconds = 7;
        dispatch_time_t popTime = dispatch_time(DISPATCH_TIME_NOW,
(int64_t)(delayInSeconds * NSEC_PER_SEC));
        dispatch_after(popTime, dispatch_get_main_queue(), ^(void){
            // [self addMessageFromRemoteNotification:dictionary2 updateUI:NO];
        });
    }
}
else
{}
//-----

```

El código anterior intentará comunicarse con el servidor de APN para obtener el token del dispositivo (los requisitos previos son los APN habilitados en su perfil de aprovisionamiento de iOS).

Una vez que establece una conexión confiable con el servidor de APN, el servidor le proporciona un token de dispositivo.

Después de agregar el código anterior, agregue estos métodos a la clase `AppDelegate` :

Rápido

```

func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    print("DEVICE TOKEN = \(deviceToken)")
}

```

```

}

func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    print(error)
}

```

C objetivo

```

- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
    NSString * deviceTokenString = [NSString stringWithFormat:@"%<" withString: @""]
        stringByReplacingOccurrencesOfString:@">" withString:@""]
        stringByReplacingOccurrencesOfString:@" " withString:@""];

    NSLog(@"The generated device token string is : %@",deviceTokenString);
}

- (void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error
{
    NSLog(@"Failed to get token, error: %@", error.description);
}

```

Los métodos anteriores se llaman de acuerdo con el escenario de éxito o fracaso del registro.

Llamadas de escenarios de éxito:

Rápido

```

func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    print("DEVICE TOKEN = \(deviceToken)")
}

```

En Swift3:

```

@objc(userNotificationCenter:willPresentNotification:withCompletionHandler:) @available(iOS
10.0, *)
func userNotificationCenter(_ center: UNUserNotificationCenter, willPresent notification:
UNNotification, withCompletionHandler completionHandler: @escaping
(UNNotificationPresentationOptions) -> Void)
{
    //To show notifications in foreground.
    print("Userinfo2 \(notification.request.content.userInfo)")
}

```

C objetivo

```

- (void)application:(UIApplication *)application

```

```

didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
if(application.applicationState == UIApplicationStateInactive) {
    NSLog(@"Inactive - the user has tapped in the notification when app was closed or in
background");
    //do some tasks
    [self handelPushNotification:userInfo];
}
else if (application.applicationState == UIApplicationStateBackground) {
    NSLog(@"application Background - notification has arrived when app was in background");
    [self handelPushNotification:userInfo];
}
else {
    NSLog(@"application Active - notication has arrived while app was opened");
    //Show an in-app banner
    //do tasks
}
}
}

```

Llamadas de escenario de falla:

Rápido

```

func application(application: UIApplication, didFailToRegisterForRemoteNotificationsWithError
error: NSError) {
    print(error)
}

```

C objetivo

```

- (void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error

```

Nota

Si no se llama a ninguno de los métodos anteriores, su dispositivo no puede crear una conexión confiable con el servidor de APN, lo que podría deberse a problemas de acceso a Internet.

Comprobando si su aplicación ya está registrada para la Notificación Push

Rápido

```

let isPushEnabled = UIApplication.sharedApplication().isRegisteredForRemoteNotifications()

```

Registro para notificaciones push (no interactivas)

Se recomienda agregar la lógica de registro para la notificación de inserción en `AppDelegate.swift`

ya que las funciones de devolución de llamada (éxito, falla) se llamarán su. Para registrarse solo haga lo siguiente:

```
let application = UIApplication.sharedApplication()
let settings = UIUserNotificationSettings(forTypes: [.Alert, .Badge, .Sound], categories: nil)
application.registerUserNotificationSettings(settings)
```

Luego se `didRegisterUserNotificationSettings` función de devolución de llamada

`didRegisterUserNotificationSettings` y, en ese caso, solo activará el registro de esta manera:

```
func application(application: UIApplication, didRegisterUserNotificationSettings
notificationSettings: UIUserNotificationSettings) {
    application.registerForRemoteNotifications()
}
```

Y, en ese caso, se mostrará una alerta del sistema solicitando la confirmación para recibir una notificación de inserción. Se llamará una de las siguientes funciones de devolución de llamada:

```
func application(application: UIApplication, didRegisterForRemoteNotificationsWithDeviceToken
deviceToken: NSData) {
    let tokenChars = UnsafePointer<CChar>(deviceToken.bytes)
    var tokenString = ""

    for i in 0..
```

En casos muy raros, no se llama a las funciones de devolución de llamada de éxito o fracaso. Esto sucede cuando tiene problemas con la conexión a Internet o el APNS Sandbox está inactivo. El sistema realiza una llamada de API a APNS para realizar alguna verificación, de lo contrario, no se realizará ninguna de las dos funciones de devolución de llamada. Visita [el estado del sistema de Apple](#) para asegurarte de que está bien.

Manejo de notificaciones push

Una vez que el usuario haga clic en una notificación de inserción, se llamará a la siguiente función de devolución de llamada. Puede analizar el JSON para obtener cualquier información específica enviada desde el backend que lo ayude a vincularse en profundidad:

Rápido

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo: [NSObject
: AnyObject]) {
```

```
    print("Received notification: \(userInfo)")
}
```

C objetivo

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo
{
    NSLog(@"Received notification: %@", userInfo);
}
```

iOS 10

```
#define SYSTEM_VERSION_GREATER_THAN_OR_EQUAL_TO(v) ([[UIDevice currentDevice] systemVersion]
compare:v options:NSNumericSearch] != NSOrderedAscending)

- (void) application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo
fetchCompletionHandler:(void (^) (UIBackgroundFetchResult)) completionHandler
{
    // iOS 10 will handle notifications through other methods
    NSLog(@"Received notification: %@", userInfo);

    if( SYSTEM_VERSION_GREATER_THAN_OR_EQUAL_TO( @"10.0" ) )
    {
        NSLog( @"iOS version >= 10. Let NotificationCenter handle this one." );
        // set a member variable to tell the new delegate that this is background
        return;
    }
    NSLog( @"HANDLE PUSH, didReceiveRemoteNotification: %@", userInfo );

    // custom code to handle notification content

    if( [UIApplication sharedApplication].applicationState == UIApplicationStateInactive )
    {
        NSLog( @"INACTIVE" );
        completionHandler( UIBackgroundFetchResultNewData );
    }
    else if( [UIApplication sharedApplication].applicationState == UIApplicationStateBackground )
    {
        NSLog( @"BACKGROUND" );
        completionHandler( UIBackgroundFetchResultNewData );
    }
    else
    {
        NSLog( @"FOREGROUND" );
        completionHandler( UIBackgroundFetchResultNewData );
    }
}

- (void)userNotificationCenter:(UNUserNotificationCenter *)center
willPresentNotification:(UNNotification *)notification
withCompletionHandler:(void (^) (UNNotificationPresentationOptions options))completionHandler
{
    NSLog( @"Handle push from foreground" );
    // custom code to handle push while app is in the foreground
}
```

```
    NSLog(@"%@", notification.request.content.userInfo);
}

- (void)userNotificationCenter:(UNUserNotificationCenter *)center
didReceiveNotificationResponse:(UNNotificationResponse *)response
    completionHandler:(void (^)(void))completionHandler
{

    NSLog( @"Handle push from background or closed" );
    // if you set a member variable in didReceiveRemoteNotification, you will know if this is
    from closed or background
    NSLog(@"%@", response.notification.request.content.userInfo);
}
```

Registro de ID de aplicación para usar con notificaciones automáticas

Cosas que necesitas

- Una membresía pagada del Programa de Desarrolladores de Apple
- Un ID de aplicación válido y un identificador para su aplicación (como com.example.MyApp) que no se usa antes en ninguna parte
- Acceso a developer.apple.com y Centro de Miembros
- Un dispositivo iOS para probar (ya que las notificaciones push no funcionan en el simulador)

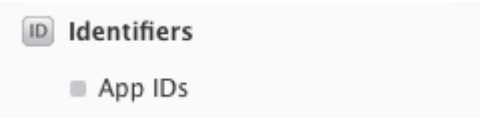
Habilitar el acceso de APN para ID de aplicación en el Centro de desarrolladores de Apple

1- Inicie sesión en developer.apple.com Member Center (el enlace de la Cuenta en la página de inicio)

Account

2- Ir a "Certificados"

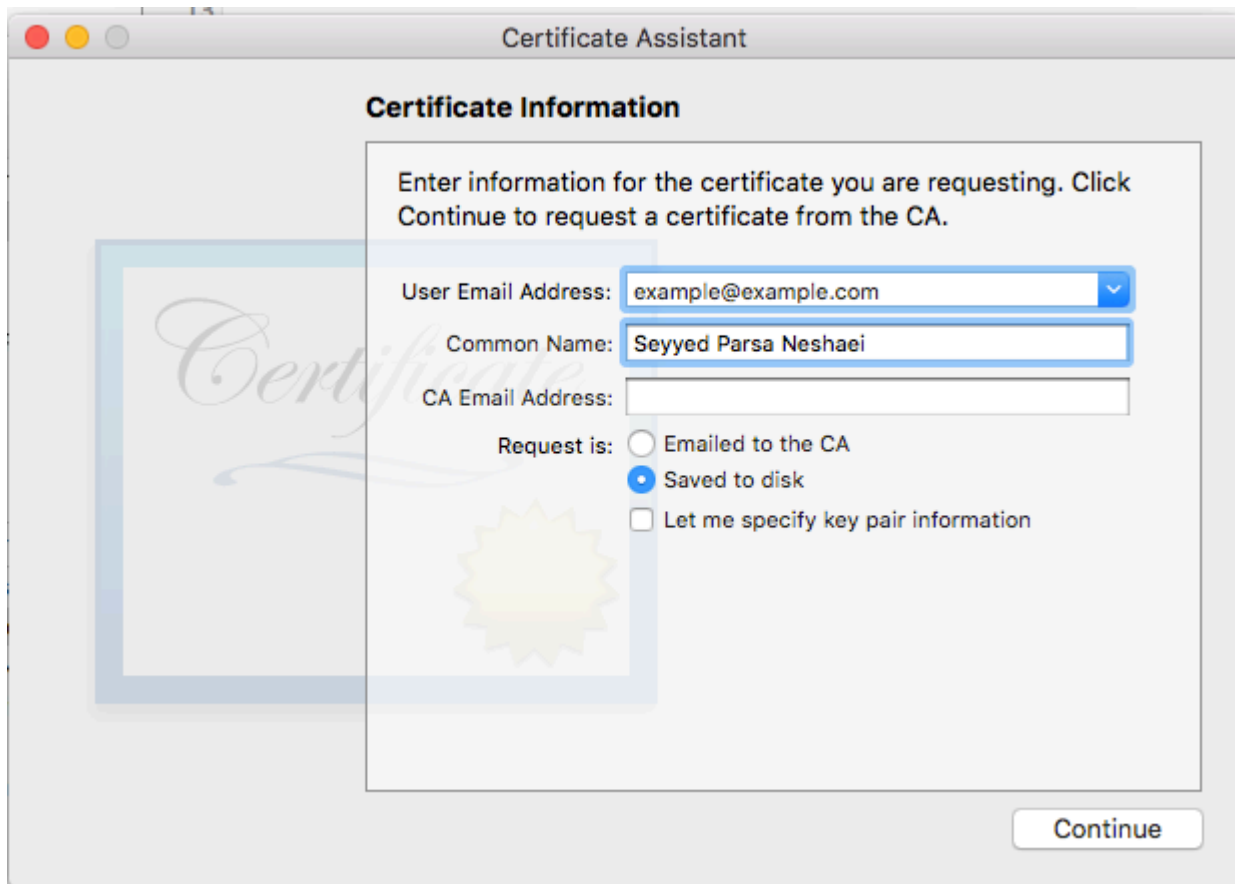
3- Seleccione "App ID" en el panel izquierdo

ID Identifiers
■ App IDs

4- Haga clic en "+" en la parte superior derecha

+

- 5- Añadir la ID de la aplicación con la opción Notificaciones push marcada
- 6- Haga clic en la ID de la aplicación creada y seleccione Editar
- 7- Haga clic en Configurar en el panel de Notificaciones Push
- 8- Abre la aplicación Keychain Access en tu Mac
- 9- En el menú de Acceso a Llaveros, haga clic en Asistente de certificados -> Solicitar un certificado de una autoridad de certificación
- 10- Ingrese su correo en el primer campo de texto
- 11- Ingrese su nombre en el segundo campo de texto



- 12- Dejar la dirección de correo electrónico de CA vacía
- 13- Seleccione Guardado en el disco en lugar de enviarlo por correo electrónico a la CA
- 14- Haga clic en Continuar y cargue el archivo generado.
- 15- Descargue el archivo generado por Apple y ábralo mientras Keychain Access está abierto

Habilitando el acceso APNs en Xcode

- 1- Seleccione su proyecto

2- Abrir la pestaña de capacidades

3- Encuentra Notificaciones Push y enciéndelo

4- Encuentra modos de fondo, enciéndelo y verifica las notificaciones remotas

Anular el registro de notificaciones push

Para cancelar el registro de las notificaciones remotas programáticamente puede utilizar

C objetivo

```
[[UIApplication sharedApplication] unregisterForRemoteNotifications];
```

Rápido

```
UIApplication.sharedApplication().unregisterForRemoteNotifications()
```

Esto es similar a entrar en la configuración de su teléfono y apagar manualmente las notificaciones para la aplicación.

NOTA: puede haber casos raros en los que necesite esto (por ejemplo, cuando su aplicación ya no admita las notificaciones push)

Si solo quieres permitir que el usuario desactive temporalmente las notificaciones. Debe implementar un método para eliminar el token del dispositivo en la base de datos de su servidor. de lo contrario, si solo deshabilita la Notificación localmente en su dispositivo, su servidor seguirá enviando mensajes.

Configuración del icono de la aplicación número de placa

Use el siguiente fragmento de código para establecer el número de distintivo desde su aplicación (suponga que se haya declarado `someNumber` anteriormente):

C objetivo

```
[[UIApplication sharedApplication].applicationIconBadgeNumber = someNumber;
```

Rápido

```
UIApplication.shared.applicationIconBadgeNumber = someNumber
```

Para *eliminar* la credencial por completo, simplemente configure `someNumber = 0`.

Prueba de notificaciones push

Siempre es una buena práctica probar cómo funcionan las notificaciones push, incluso antes de que el servidor esté listo para ellas, solo para asegurarse de que todo esté configurado

correctamente de su lado. Es muy fácil enviarte una notificación de inserción usando un script PHP siguiente.

1. Guarde el script como un archivo (send_push.php por ejemplo) en la misma carpeta que su certificado (desarrollo o producción)
2. Edítelo para poner el token de su dispositivo, la contraseña del certificado.
3. Elija la ruta correcta para abrir una conexión, dev_path o prod_path (aquí es donde 'Abrir una conexión con el servidor APNS' ocurre en el script)
4. Cd a la carpeta en la Terminal y ejecute el comando 'php send_push'
5. Recibe la notificación en tu dispositivo.

```
<?php

// Put your device token here (without spaces):
$deviceToken = '20128697f872d7d39e48c4a61f50cb11d77789b39e6fc6b4cd7ec80582ed5229';
// Put your final pem cert name here. it is supposed to be in the same folder as this script
$cert_name = 'final_cert.pem';
// Put your private key's passphrase here:
$passphrase = '1234';

// sample point
$alert = 'Hello world!';
$event = 'new_incoming_message';

// You can choose either of the paths, depending on what kind of certificate you are using
$dev_path = 'ssl://gateway.sandbox.push.apple.com:2195';
$prod_path = 'ssl://gateway.push.apple.com:2195';

////////////////////////////////////

$ctx = stream_context_create();
stream_context_set_option($ctx, 'ssl', 'local_cert', $cert_name);
stream_context_set_option($ctx, 'ssl', 'passphrase', $passphrase);

// Open a connection to the APNS server
$fp = stream_socket_client(
    $dev_path, $err,
    $errstr, 60, STREAM_CLIENT_CONNECT|STREAM_CLIENT_PERSISTENT, $ctx);

if (!$fp)
    exit("Failed to connect: $err $errstr" . PHP_EOL);

echo 'Connected to APNS' . PHP_EOL;

// Create the payload body
// it should be as short as possible
// if the notification doesnt get delivered that is most likely
// because the generated message is too long
$body['aps'] = array(
    'alert' => $alert,
    'sound' => 'default',
    'event' => $event
);

// Encode the payload as JSON
$payload = json_encode($body);

// Build the binary notification
```

```

$msg = chr(0) . pack('n', 32) . pack('H*', $deviceToken) . pack('n', strlen($payload)) .
$payload;

// Send it to the server
$result = fwrite($fp, $msg, strlen($msg));

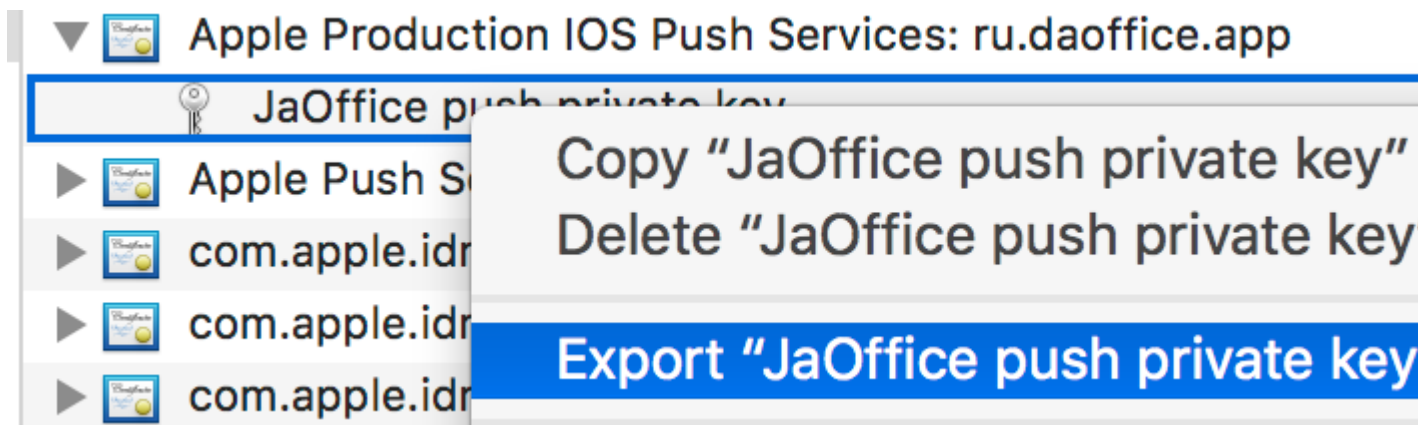
if (!$result)
    echo 'Message not delivered' . PHP_EOL;
else
    echo 'Message successfully delivered' . PHP_EOL;

// Close the connection to the server
fclose($fp);

```

Generar un certificado .pem de su archivo .cer, para pasarlo al desarrollador del servidor

1. Guardar aps.cer en una carpeta
2. Abra "Acceso a llavero" y exporte la clave que está bajo ese certificado a un archivo .p12 (llámelo key.p12). Para hacerlo haz click derecho sobre él y elige Exportar. Guárdelo en la misma carpeta que el paso 1. Al exportar se le pedirá una contraseña. Inventa algo y memorízalo.



3. cd a esa carpeta en la Terminal y ejecute los siguientes comandos:
4. Convertir .cer a un certificado .pem

```
openssl x509 -in aps.cer -inform der -out aps.pem
```

5. Convierte tu clave al formato .pem. Para abrir la clave, ingrese la contraseña con la que la exportó desde el llavero, en el paso 2. Luego, ingrese otra contraseña que protegerá el archivo exportado. Se le pedirá que lo ingrese dos veces para confirmar.

```
openssl pkcs12 -nocerts -out key.pem -in key.p12
```

6. Combina los archivos en un archivo final

```
cat key.pem aps.pem > final_cert.pem
```

7. El final_cert.pem es el resultado final. Páselo a los desarrolladores del servidor con la contraseña del paso 5, para que puedan usar el certificado protegido.

Lea Notificaciones push en línea: <https://riptutorial.com/es/ios/topic/3492/notificaciones-push>

Capítulo 111: NSArray

Introducción

Aquí hay algunas funciones / métodos de utilidad útiles que se pueden usar como con la extensión Array para que el desarrollador pueda realizar ciertas operaciones críticas en una matriz con la ayuda del código de una sola línea.

Observaciones

Una vez que se apruebe el documento actual, también se agregarán muchas mejoras para otras aplicaciones de matriz. Este es mi primer documento y necesito su ayuda y aprobación en mi esfuerzo.

Examples

Convertir Array en cadena json

Llame a esta función con el parámetro argumento como matriz con el tipo 'any'. Te devolverá la cadena json. La cadena Json se utiliza para enviar la matriz en la llamada de servicio web como parámetro de entrada de solicitud en Swift.

// -----

```
let array = [{"one" : 1}, {"two" : 2}, {"three" : 3}, {"four" : 4}]

let jsonString = convertIntoJSONString(arrayObject: array)
print("jsonString - \(jsonString)")
```

// -----

```
func convertIntoJSONString(arrayObject: [Any]) -> String? {

    do {
        let jsonData: Data = try JSONSerialization.data(withJSONObject: arrayObject,
options: [])
        if let jsonString = NSString(data: jsonData, encoding:
String.Encoding.utf8.rawValue) {
            return jsonString as String
        }
    } catch let error as NSError {
        print("Array convertIntoJSON - \(error.description)")
    }
    return nil
}
```

Lea NSArray en línea: <https://riptutorial.com/es/ios/topic/9248/nsarray>

Capítulo 112: NSAttributedString

Observaciones

[Establecer color de fuente usando NSAttributedString](#)

Examples

Creación de una cadena que tiene un kerning personalizado (espaciado entre letras)

`NSAttributedString` (y su hermano mutable `NSMutableAttributedString`) le permite crear cadenas que son complejas en su apariencia para el usuario.

Una aplicación común es usar esto para mostrar una cadena y agregar un espaciado de letras / kerning personalizado.

Esto se lograría de la siguiente manera (donde `label` es una `UILabel`), dando un kerning diferente para la palabra "kerning"

Rápido

```
var attributedString = NSMutableAttributedString("Apply kerning")
attributedString.addAttribute(attribute: NSKernAttributeName, value: 5, range: NSRange(6, 7))
label.attributedString = attributedString
```

C objetivo

```
NSMutableAttributedString *attributedString;
attributedString = [[NSMutableAttributedString alloc] initWithString:@"Apply kerning"];
[attributedString addAttribute:NSKernAttributeName value:@5 range:NSMakeRange(6, 7)];
[label setAttributedString:attributedString];
```

Crear una cadena con texto tachado

C objetivo

```
NSMutableAttributedString *attributeString = [[NSMutableAttributedString alloc]
initWithString:@"Your String here"];
[attributeString addAttribute:NSStrikethroughStyleAttributeName
value:@2
range:NSMakeRange(0, [attributeString length])];
```

Rápido

```
let attributeString: NSMutableAttributedString = NSMutableAttributedString(string: "Your
```

```
String here")
attributeString.addAttribute(NSStrikethroughStyleAttributeName, value: 2, range:
NSMakeRange(0, attributeString.length))
```

Entonces puedes agregar esto a tu UILabel:

```
yourLabel.attributedText = attributeString;
```

Anexando cadenas atribuidas y texto en negrita en Swift

```
let someValue : String = "Something the user entered"
let text = NSMutableAttributedString(string: "The value is: ")
text.appendAttributedString(NSAttributedString(string: someValue, attributes:
[NSFontAttributeName:UIFont.boldSystemFontOfSize(UIFont.systemFontSize())]))
```

El resultado se ve como:

El valor es: **Algo introducido por el usuario.**

Cambiar el color de una palabra o cadena

C objetivo

```
UIColor *color = [UIColor redColor];
NSString *textToFind = @"redword";

NSMutableAttributedString *attrsString = [[NSMutableAttributedString alloc]
initWithAttributedString:yourLabel.attributedText];

// search for word occurrence
NSRange range = [yourLabel.text rangeOfString:textToFind];
if (range.location != NSNotFound) {
    [attrsString addAttribute:NSForegroundColorAttributeName value:color range:range];
}

// set attributed text
yourLabel.attributedText = attrsString;
```

Rápido

```
let color = UIColor.red;
let textToFind = "redword"

let attrsString = NSMutableAttributedString(string:yourlabel.text!);

// search for word occurrence
let range = (yourlabel.text! as NSString).range(of: textToFind)
if (range.length > 0) {
    attrsString.addAttribute(NSForegroundColorAttributeName,value:color,range:range)
}

// set attributed text
yourlabel.attributedText = attrsString
```


Nota :

El principal aquí es usar un `NSMutableAttributedString` y el selector `addAttribute:value:range` con el atributo `NSForegroundColorAttributeName` para cambiar el color de un rango de cadena:

```
NSMutableAttributedString *attrsString = [[NSMutableAttributedString alloc]
initWithAttributedString:label.attributedString];
[attrsString addAttribute:NSForegroundColorAttributeName value:color range:range];
```

Podría usar otra forma de obtener el rango, por ejemplo: `NSRegularExpression`.

Eliminando todos los atributos

C objetivo

```
NSMutableAttributedString *mutAttString = @"string goes here";
NSRange range = NSMakeRange(0, mutAttString.length);
[mutAttString setAttributes:@{} range:originalRange];
```

Según la documentación de Apple, usamos `setAttributes` y no `addAttribute` .

Rápido

```
mutAttString.setAttributes([], range: NSRange(0..
```

Lea `NSAttributedString` en línea: <https://riptutorial.com/es/ios/topic/979/nsattributedString>

Capítulo 113: NSBundle

Examples

Obtener el paquete principal

1. Obteniendo una referencia al paquete principal utilizando Cocoa.

Para obtener el paquete principal de aplicación Cocoa, llame al método de clase de la clase **mainBundle** **NSBundle**.

```
NSBundle *mainBundle;
// Get the main bundle for the app;
mainBundle = [NSBundle mainBundle];
```

2. Obtención de una referencia al paquete principal mediante Core Foundation.

Utilice la función **CFBundleGetMainBundle** para recuperar el paquete principal para su aplicación basada en C.

```
CFBundleRef mainBundle;
// Get the main bundle for the app
mainBundle = CFBundleGetMainBundle();
```

Obtener paquete por ruta

1. Localizando un paquete de cacao usando su camino

Para obtener el paquete en una ruta específica usando Cocoa, llame al método **bundleWithPath:** class del **NSBundle**

```
NSBundle *myBundle;
// obtain a reference to a loadable bundle
myBundle = [NSBundle bundleWithPath:@"~/Library/MyBundle.bundle"];
```

2. Localizando un paquete de la Fundación Cocoa usando su Sendero

Para obtener el paquete en una ruta específica utilizando Core Foundation, llame a la función **CFBundleCreate** y debe usar el tipo **CFURLRef**.

```
CFURLRef bundleURL;
CFBundleRef myBundle;
// Make a CFURLRef from the CFString representation of the bundle's path.
bundleURL = CFURLCreateWithFileSystemPath(kCFAllocatorDefault,
CFSTR("~/Library/MyBundle.bundle"), kCFURLPOSIXPathStyle, true);
// Make a bundle instance using the URLRef.
myBundle = CFBundleCreate(kCFAllocatorDefault, bundleURL);
// You can release the URL now.
```

```
CFRelease(bundleURL);  
// Use the bundle ...  
// Release the bundle when done.  
CFRelease(myBundle);
```

Lea NSBundle en línea: <https://riptutorial.com/es/ios/topic/5862/nsbundle>

Capítulo 114: NSData

Observaciones

Recursos utiles

[Documentación de Apple \(NSData\)](#)

[NSData.dataWithContentsOfFile \(\)](#)

[NSData.bytes](#)

Examples

Creando objetos NSData

Usando un archivo

Rápido

```
let data = NSData(contentsOfFile: filePath) //assuming filePath is a valid path
```

C objetivo

```
NSData *data = [NSData dataWithContentsOfFile:filePath]; //assuming filePath is a valid path
```

Usando un objeto String

Rápido

```
let data = (string as NSString).dataUsingEncoding(NSUTF8StringEncoding) //assuming string is a String object
```

C objetivo

```
NSData *data = [string dataUsingEncoding:NSUTF8StringEncoding]; //assuming string is a String object
```

Convertir NSData a otros tipos

Encadenar

Rápido

```
let string = String(NSString(data: data, encoding: NSUTF8StringEncoding)) //assuming data is a valid NSData object
```

C objetivo

```
NSString *string = [[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];  
//assuming data is a valid NSData object  
[string release];
```

A Array

Rápido

```
let array = data.bytes as! NSMutableArray //assuming data is a valid NSData object
```

C objetivo

```
NSMutableArray *array = (NSMutableArray *)[data bytes]; //assuming data is a valid NSData object
```

A la matriz de bytes

Rápido

```
let byteArray = data.bytes as! UInt8 //assuming data is a valid NSData object
```

C objetivo

```
UInt8 *byteArray = (UInt8 *)data.bytes; //assuming data is a valid NSData object
```

Convertir NSData a cadena HEX

NSData

se puede representar como una cadena hexadecimal, similar a lo que produce en su método de `description`.

Rápido

```
extension NSData {  
  
    func hexString() -> String {  
        return UnsafeBufferPointer<UInt8>(start: UnsafePointer<UInt8>(bytes), count: length)  
            .reduce("") { $0 + String(format: "%02x", $1) }  
    }  
  
}
```

C objetivo

```
@implementation NSData (HexRepresentation)  
  
- (NSString *)hexString {  
    const unsigned char *bytes = (const unsigned char *)self.bytes;  
    NSMutableString *hex = [NSMutableString new];  
    for (NSUInteger i = 0; i < self.length; i++) {  
        [hex appendFormat:@"%02x", bytes[i]];  
    }  
    return [hex copy];  
}  
  
@end
```

Lea NSData en línea: <https://riptutorial.com/es/ios/topic/5084/nsdata>

Capítulo 115: NSDate

Sintaxis

- NSDate () // NSDate object init a la fecha y hora actuales
- NSDate (). TimeIntervalSince1970 // Fecha y hora actuales en número de segundos desde las 00:00:00 UTC del 1 de enero de 1970.
- NSDate (). Compare (other: NSDate) // Devuelve una comparación de la fecha actual con otra fecha devuelve un NSDateComparisonResult

Observaciones

Hay diferentes tipos de formato de fecha que puede establecer: Aquí está la lista completa de ellos.

Formato	Significado / Descripción	Ejemplo 1	Ejemplo 2
y	Un año con al menos 1 dígito.	175 dC → "175"	2016 DC → "2016"
yy	Un año con exactamente 2 dígitos.	5 dC → "05"	2016 DC → "16"
yyy	Un año con al menos 3 dígitos.	5 dC → "005"	2016 DC → "2016"
aaaa	Un año con al menos 4 dígitos.	5 dC → "0005"	2016 DC → "2016"
METRO	Un mes con al menos 1 dígito.	Julio → "7"	"Noviembre" → "11"
MM	Un mes con al menos 2 dígitos.	Julio → "07"	"Noviembre" → "11"
MMM	Abreviatura de tres letras del mes.	Julio → "Julio"	"Noviembre" → "noviembre"
MMMM	Nombre completo del mes.	Julio → "Julio"	"Noviembre" → "Noviembre"
MMMMM	Una abreviatura de un mes (enero, junio, julio, todos tendrán 'J').	Julio → "J"	"Noviembre" → "N"
re	Día con al menos un dígito.	8 → "8"	29 → "29"

Formato	Significado / Descripción	Ejemplo 1	Ejemplo 2
dd	Día con al menos dos dígitos.	8 → "08"	29 → "29"
"E", "EE" o "EEE"	Abreviatura de día de 3 letras del nombre del día.	Lunes → "lun"	Jueves → "Jue"
EEEE	Nombre del día completo.	Lunes → "lunes"	Jueves → "jueves"
EEEEE	Abreviatura de un día de una letra del nombre del día. (Thu y Tue serán 'T')	Lunes → "M"	Jueves → "T"
EEEEEE	Abreviatura de 2 letras del nombre del día.	Lunes → "Mo"	Jueves → "th"
una	Período del día (AM / PM).	10 PM → "PM"	2 AM → "AM"
h	Una hora basada en 1-12 con al menos 1 dígito.	10 PM → "10"	2 AM → "2"
S.S	Una hora basada en 1-12 con al menos 2 dígitos.	10 PM → "10"	2 a.m. → "02"
H	Una hora basada en 0-23 con al menos 1 dígito.	10 p.m. → "14"	2 AM → "2"
S.S	Una hora basada en 0-23 con al menos 2 dígitos.	10 p.m. → "14"	2 a.m. → "02"
metro	Un minuto con al menos 1 dígito.	7 → "7"	29 → "29"
mm	Un minuto con al menos 2 dígitos.	7 → "07"	29 → "29"
s	Un segundo con al menos 1 dígito.	7 → "7"	29 → "29"
ss	Un segundo con al menos 2 dígitos.	7 → "07"	29 → "29"

Hay muchos más, para obtener un tiempo diferente según la zona (z), para obtener el tiempo con detalles de milisegundos (S), etc.

Examples

Obtener fecha actual

Obtener la fecha actual es muy fácil. Obtendrá el objeto NSDate de la fecha actual en una sola

línea de la siguiente manera:

Rápido

```
var date = NSDate()
```

Swift 3

```
var date = Date()
```

C objetivo

```
NSDate *date = [NSDate date];
```

Obtenga NSDate Object N segundos desde la fecha actual

El número de segundos a partir de la fecha y hora actuales para la nueva fecha. Use un valor negativo para especificar una fecha antes de la fecha actual.

Para hacer esto tenemos un método llamado `dateWithTimeIntervalSinceNow(seconds: NSTimeInterval) -> NSDate` (Swift) o `(NSDate*)dateWithTimeIntervalSinceNow:(NSTimeInterval)seconds` (Objective-C).

Ahora, por ejemplo, si necesita una fecha de una semana desde la fecha actual y una semana hasta la fecha actual, entonces podemos hacerlo como.

Rápido

```
let totalSecondsInWeek:NSTimeInterval = 7 * 24 * 60 * 60;
//Using negative value for previous date from today
let nextWeek = NSDate().dateWithTimeIntervalSinceNow(totalSecondsInWeek)

//Using positive value for future date from today
let lastWeek = NSDate().dateWithTimeIntervalSinceNow(-totalSecondsInWeek)
```

Swift 3

```
let totalSecondsInWeek:TimeInterval = 7 * 24 * 60 * 60;

//Using positive value to add to the current date
let nextWeek = Date(timeIntervalSinceNow: totalSecondsInWeek)

//Using negative value to get date one week from current date
let lastWeek = Date(timeIntervalSinceNow: -totalSecondsInWeek)
```

C objetivo

```
NSTimeInterval totalSecondsInWeek = 7 * 24 * 60 * 60;
//Using negative value for previous date from today
NSDate *lastWeek = [NSDate dateWithTimeIntervalSinceNow:-totalSecondsInWeek];

//Using positive value for future date from today
NSDate *nextWeek = [NSDate dateWithTimeIntervalSinceNow:totalSecondsInWeek];

NSLog(@"Last Week: %@", lastWeek);
NSLog(@"Right Now: %@", now);
NSLog(@"Next Week: %@", nextWeek);
```

Comparación de fechas

Hay 4 métodos para comparar fechas:

Rápido

- `isEqualToDate(anotherDate: NSDate) -> Bool`
- `earlierDate(anotherDate: NSDate) -> NSDate`
- `laterDate(anotherDate: NSDate) -> NSDate`
- `compare(anotherDate: NSDate) -> NSComparisonResult`

C objetivo

- - (BOOL)isEqualToDate:(NSDate *)anotherDate
- - (NSDate *)earlierDate:(NSDate *)anotherDate
- - (NSDate *)laterDate:(NSDate *)anotherDate
- - (NSComparisonResult)compare:(NSDate *)anotherDate

Digamos que tenemos 2 fechas:

Rápido

```
let date1: NSDate = ... // initialized as July 7, 2016 00:00:00
let date2: NSDate = ... // initialized as July 2, 2016 00:00:00
```

C objetivo

```
NSDate *date1 = ... // initialized as July 7, 2016 00:00:00
NSDate *date2 = ... // initialized as July 2, 2016 00:00:00
```

Luego, para compararlos, probamos este código:

Rápido

```

if date1.isEqualToDate(date2) {
    // returns false, as both dates aren't equal
}

earlierDate: NSDate = date1.earlierDate(date2) // returns the earlier date of the two (date 2)
laterDate: NSDate = date1.laterDate(date2) // returns the later date of the two (date1)

result: NSComparisonResult = date1.compare(date2)

if result == .OrderedAscending {
    // true if date1 is earlier than date2
} else if result == .OrderedSame {
    // true if the dates are the same
} else if result == .OrderedDescending {
    // true if date1 is later than date1
}

```

C objetivo

```

if ([date1 isEqualToDate:date2]) {
    // returns false, as both date are not equal
}

NSDate *earlierDate = [date1 earlierDate:date2]; // returns date which comes earlier from both
date, here it will return date2
NSDate *laterDate = [date1 laterDate:date2]; // returns date which comes later from both date,
here it will return date1

NSComparisonResult result = [date1 compare:date2];
if (result == NSOrderedAscending) {
    // fails
    // comes here if date1 is earlier then date2, in our case it will not come here
} else if (result == NSOrderedSame){
    // fails
    // comes here if date1 is same as date2, in our case it will not come here
} else{ // NSOrderedDescending
    // succeeds
    // comes here if date1 is later than date2, in our case it will come here
}

```

Si desea comparar fechas y manejar segundos, semanas, meses y años:

Swift 3

```

let dateStringUTC = "2016-10-22 12:37:48 +0000"
let dateFormatter = DateFormatter()
dateFormatter.locale = Locale(identifier: "en_US_POSIX")
dateFormatter.dateFormat = "yyyy-MM-dd HH:mm:ss X"
let date = dateFormatter.date(from: dateStringUTC)!

let now = Date()

let formatter = DateComponentsFormatter()
formatter.unitsStyle = .full
formatter.maximumUnitCount = 2
let string = formatter.string(from: date, to: Date())! + " " + NSLocalizedString("ago",

```

```
comment: "added after elapsed time to say how long before")
```

O puedes usar esto para cada componente:

```
// get the current date and time
let currentDateTime = Date()

// get the user's calendar
let userCalendar = Calendar.current

// choose which date and time components are needed
let requestedComponents: Set<Calendar.Component> = [
    .year,
    .month,
    .day,
    .hour,
    .minute,
    .second
]

// get the components
let dateTimeComponents = userCalendar.dateComponents(requestedComponents, from:
currentDateTime)

// now the components are available
dateTimeComponents.year
dateTimeComponents.month
dateTimeComponents.day
dateTimeComponents.hour
dateTimeComponents.minute
dateTimeComponents.second
```

Obtener tiempo de época de Unix

Para obtener [Unix Epoch Time](#) , use la constante `timeIntervalSince1970` :

Rápido

```
let date = NSDate() // current date
let unixtime = date.timeIntervalSince1970
```

C objetivo

```
NSDate *date = [NSDate date]; // current date
int unixtime = [date timeIntervalSince1970];
```

NSDateFormatter

Convertir un objeto `NSDate` en cadena es solo 3 pasos.

1. Crea un objeto `NSDateFormatter`

Rápido

```
let dateFormatter = NSDateFormatter()
```

Swift 3

```
let dateFormatter = DateFormatter()
```

C objetivo

```
NSDateFormatter *dateFormatter = [[NSDateFormatter alloc] init];
```

2. Establece el formato de fecha en el que quieres tu cadena

Rápido

```
dateFormatter.dateFormat = "yyyy-MM-dd 'at' HH:mm"
```

C objetivo

```
dateFormatter.dateFormat = @"yyyy-MM-dd 'at' HH:mm";
```

3. Obtener la cadena con formato

Rápido

```
let date = NSDate() // your NSDate object
let dateString = dateFormatter.stringFromDate(date)
```

Swift 3

```
let date = Date() // your NSDate object
let dateString = dateFormatter.stringFromDate(date)
```

C objetivo

```
NSDate *date = [NSDate date]; // your NSDate object
NSString *dateString = [dateFormatter stringFromDate:date];
```

Esto dará salida a algo como esto: 2001-01-02 at 13:00

Nota

Crear una instancia de `NSDateFormatter` es una operación costosa, por lo que se recomienda crearla una vez y reutilizarla cuando sea posible.

Extensión útil para convertir la fecha en cadena.

```
extension Date {
    func toString() -> String {
        let dateFormatter = DateFormatter()
        dateFormatter.dateFormat = "MMMM dd yyyy"
        return dateFormatter.string(from: self)
    }
}
```

Enlaces útiles para una rápida formación de fechas que [se obtienen rápidamente para ser humano legibles en la fecha](#) .

Para construir formatos de fecha ver [patrones de formato de fecha](#) .

Convierta NSDate que se compone de hora y minuto (solo) a un NSDate completo

Hay muchos casos en los que uno ha creado una NSDate a partir de un formato de solo una hora y minuto, es decir: 08:12 que regresa de un servidor como una cadena e inicia una instancia de NSDate **solo** con estos **valores**.

El inconveniente de esta situación es que su NSDate está casi completamente "desnudo" y lo que debe hacer es crear: día, mes, año, segundo y huso horario para que este objeto "siga el ritmo" con otros tipos de NSDate.

En aras del ejemplo, digamos que `hourAndMinute` es el tipo NSDate que se compone de formato de hora y minuto:

C objetivo

```
NSDateComponents *hourAndMinuteComponents = [calendar components:NSCalendarUnitHour |
NSCalendarUnitMinute
                                                fromDate:hourAndMinute];
```

```

NSDateComponents *componentsOfDate = [[NSCalendar currentCalendar]
components:NSCalendarUnitDay | NSCalendarUnitMonth | NSCalendarUnitYear
                                               fromDate:[NSDate date]];

NSDateComponents *components = [[NSDateComponents alloc] init];
[components setDay: componentsOfDate.day];
[components setMonth: componentsOfDate.month];
[components setYear: componentsOfDate.year];
[components setHour: [hourAndMinuteComponents hour]];
[components setMinute: [hourAndMinuteComponents minute]];
[components setSecond: 0];
[calendar setTimeZone: [NSTimeZone defaultTimeZone]];

NSDate *yourFullNSDateObject = [calendar dateFromComponents:components];

```

Ahora tu objeto es el opuesto total de estar "desnudo".

UTC Time offset from NSDate with TimeZone

Aquí, esto calculará el desplazamiento de tiempo UTC de los datos actuales en la zona horaria deseada.

```

+(NSTimeInterval) getUTCOffsetIntervalWithCurrentTimeZone: (NSTimeZone *) current forDate: (NSDate *) date {
    NSTimeZone *utcTimeZone = [NSTimeZone timeZoneWithAbbreviation:@"UTC"];
    NSInteger currentGmtoffset = [current secondsFromGMTForDate:date];
    NSInteger gmtoffset = [utcTimeZone secondsFromGMTForDate:date];
    NSTimeInterval gmtoffsetInterval = currentGmtoffset - gmtoffset;
    return gmtoffsetInterval;
}

```

Obtener el tipo de ciclo de tiempo (12 horas o 24 horas)

Comprobando si la fecha actual contiene el símbolo para AM o PM

C objetivo

```

NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
[formatter setLocale:[NSLocale currentLocale]];
[formatter setDateStyle:NSDateFormatterNoStyle];
[formatter setTimeStyle:NSDateFormatterShortStyle];
NSString *dateString = [formatter stringFromDate:[NSDate date]];
NSRange amRange = [dateString rangeOfString:[formatter AMSymbol]];
NSRange pmRange = [dateString rangeOfString:[formatter PMSymbol]];
BOOL is24h = (amRange.location == NSNotFound && pmRange.location == NSNotFound);

```

Solicitando el tipo de ciclo de tiempo de

C objetivo

```
NSString *formatStringForHours = [NSDateFormatter dateFormatFromTemplate:@"j" options:0
locale:[NSLocale currentLocale]];
NSRange containsA = [formatStringForHours rangeOfString:@"a"];
BOOL is24h = containsA.location == NSNotFound;
```

Esto utiliza una cadena de plantilla de fecha especial llamada "j" que, de acuerdo con las [especificaciones de la UCI](#) ...

[...] solicita el formato de hora preferido para la configuración regional (h, H, K o k), según lo determinado por el atributo preferido del elemento horas en los datos complementarios. [...] Tenga en cuenta que el uso de 'j' en un esqueleto que se pasa a una API es la única forma de que el esqueleto solicite el tipo de ciclo de tiempo preferido de una localidad (12 horas o 24 horas).

Esa última oración es importante. Es "la única forma de que un esqueleto solicite el tipo de ciclo de tiempo preferido de un local". Dado que `NSDateFormatter` y `NSCalendar` se basan en la biblioteca de la UCI, lo mismo se aplica aquí.

Referencia

La segunda opción se deriva de [esta respuesta](#) .

Obtenga NSDate del formato de fecha JSON `"/ Date (1268123281843) /"`

Antes de Json.NET, las fechas se escribían con el formato de Microsoft: `"/ Date (1198908717056) /"`. Si su servidor envía la fecha en este formato, puede usar el siguiente código para serializarlo a NSDate:

C objetivo

```
(NSDate*) getDateFromJSON:(NSString *)dateString
{
    // Expect date in this format "/Date(1268123281843)/"
    int startPos = [dateString rangeOfString:@"("].location+1;
    int endPos = [dateString rangeOfString:@")"].location;
    NSRange range = NSMakeRange(startPos,endPos-startPos);
    unsigned long long milliseconds = [[dateString substringWithRange:range] longLongValue];
    NSLog(@"%llu",milliseconds);
    NSTimeInterval interval = milliseconds/1000;
    NSDate *date = [NSDate dateWithTimeIntervalSince1970:interval];
    // add code for date formatter if need NSDate in specific format.
    return date;
}
```


Obtenga tiempo histórico de NSDate (por ejemplo: hace 5s, hace 2m, hace 3h)

Esto se puede usar en varias aplicaciones de chat, feeds rss y aplicaciones sociales donde necesita tener las últimas feeds con marcas de tiempo:

C objetivo

```
- (NSString *)getHistoricTimeText:(NSDate *)since
{
    NSString *str;
    NSTimeInterval interval = [[NSDate date] timeIntervalSinceDate:since];
    if(interval < 60)
        str = [NSString stringWithFormat:@"%is ago", (int)interval];
    else if(interval < 3600)
    {
        int minutes = interval/60;
        str = [NSString stringWithFormat:@"%im ago", minutes];
    }
    else if(interval < 86400)
    {
        int hours = interval/3600;
        str = [NSString stringWithFormat:@"%ih ago", hours];
    }
    else
    {
        NSDateFormatter *dateFormatter=[[NSDateFormatter alloc]init];
        [dateFormatter setLocale:[NSLocale currentLocale]];
        NSString *dateFormat = [NSDateFormatter dateFormatFromTemplate:@"MMM d, YYYY"
options:0 locale:[NSLocale currentLocale]];
        [dateFormatter setDateFormat:dateFormat];
        str = [dateFormatter stringFromDate:since];
    }
    return str;
}
```

Lea NSDate en línea: <https://riptutorial.com/es/ios/topic/1502/nsdate>

Capítulo 116: NSHTTPCookieStorage

Examples

Almacena y lee las cookies de NSUserDefaults

```
import Foundation

class CookiesSingleton {

static let instance : CookiesSingleton = CookiesSingleton()
static var enableDebug = true

func loadCookies() {
    if let cookiesDetails =
NSUserDefaults.standardUserDefaults().objectForKey("customeWebsite") {
        for (keys,_) in cookiesDetails as! NSDictionary{
            if let cookieDict = NSUserDefaults.standardUserDefaults().objectForKey(keys
as! String){
                if let cookie = NSHTTPCookie(properties:cookieDict as! [String:AnyObject])
{
                    NSHTTPCookieStorage.sharedHTTPCookieStorage().setCookie(cookie)
                    if(CookiesSingleton.enableDebug){
                        print("Each Cookies",cookieDict)
                    }
                }
            }
        }
    }
}

func removeCookies(){
    NSURLCache.sharedURLCache().removeAllCachedResponses()
    NSURLCache.sharedURLCache().diskCapacity = 0
    NSURLCache.sharedURLCache().memoryCapacity = 0

    let storage : NSHTTPCookieStorage = NSHTTPCookieStorage.sharedHTTPCookieStorage()
    for cookie in storage.cookies! {
        storage.deleteCookie(cookie as NSHTTPCookie)
    }

    NSUserDefaults.standardUserDefaults().setValue("", forKey: "customeWebsite")
    NSUserDefaults.standardUserDefaults().synchronize()

    if(CookiesSingleton.enableDebug){
        print("Cookies Removed")
    }
}

func saveCookies() {

    let cookieArray = NSMutableArray()
    let savedC = NSHTTPCookieStorage.sharedHTTPCookieStorage().cookies

    let allCookiesDic:NSMutableDictionary = NSMutableDictionary()
```

```

for c : NSHTTPCookie in savedC! {

    let cookieProps = NSMutableDictionary()
    cookieArray.addObject(c.name)
    cookieProps.setValue(c.name, forKey: NSHTTPCookieName)
    cookieProps.setValue(c.value, forKey: NSHTTPCookieValue)
    cookieProps.setValue(c.domain, forKey: NSHTTPCookieDomain)
    cookieProps.setValue(c.path, forKey: NSHTTPCookiePath)
    cookieProps.setValue(c.version, forKey: NSHTTPCookieVersion)
    cookieProps.setValue(NSDate().dateByAddingTimeInterval(2629743), forKey:
NSHTTPCookieExpires)

    allCookiesDic.setValue(cookieProps, forKey: c.name)

}
NSUserDefaults.standardUserDefaults().setValue(allCookiesDic, forKey: "customeWebsite")
NSUserDefaults.standardUserDefaults().synchronize()

if(CookiesSingleton.enableDebug){
    print("Cookies Saved")
}
}
}

```

Lea NSHTTPCookieStorage en línea: <https://riptutorial.com/es/ios/topic/7312/nshttpcookiestorage>

Capítulo 117: NSInvocación

Examples

NSInvocation Objective-C

Refiérase a este [Post original](#) por [e.James](#)

Según [la referencia de la clase NSInvocation de Apple](#) :

Una `NSInvocation` es un mensaje Objective-C que se vuelve estático, es decir, es una acción que se convierte en un objeto.

Y, con un *poco* más de detalle:

El concepto de mensajes es central para la filosofía del objetivo c. Cada vez que llama a un método, o accede a una variable de algún objeto, le está enviando un mensaje. `NSInvocation` es útil cuando desea enviar un mensaje a un objeto en un momento diferente en el tiempo, o enviar el mismo mensaje varias veces. `NSInvocation` permite *describir* el mensaje que va a enviar y luego *invocarlo* (en realidad, enviarlo al objeto de destino) más adelante.

Por ejemplo, supongamos que desea agregar una cadena a una matriz. Normalmente enviarías el mensaje `addObject:` siguiente manera:

```
[myArray addObject:myString];
```

Ahora, supongamos que desea utilizar `NSInvocation` para enviar este mensaje en algún otro momento:

En primer lugar, debería preparar un `NSInvocation` objeto para su uso con `NSMutableArray` 's `addObject:` Selector:

```
NSMethodSignature * mySignature = [NSMutableArray  
    instanceMethodSignatureForSelector:@selector(addObject:)];  
NSInvocation * myInvocation = [NSInvocation  
    invocationWithMethodSignature:mySignature];
```

A continuación, debe especificar a qué objeto enviar el mensaje:

```
[myInvocation setTarget:myArray];
```

Especifique el mensaje que desea enviar a ese objeto:

```
[myInvocation setSelector:@selector(addObject:)];
```

Y complete cualquier argumento para ese método:

```
[myInvocation setArgument:&myString atIndex:2];
```

Tenga en cuenta que los argumentos del objeto deben ser pasados por el puntero. Gracias a [Ryan McCuaig](#) por señalarlo, y consulte [la documentación de Apple](#) para obtener más detalles.

En este punto, `myInvocation` es un objeto completo, que describe un mensaje que se puede enviar. Para enviar el mensaje, deberías llamar:

```
[myInvocation invoke];
```

Este último paso hará que se envíe el mensaje, básicamente ejecutando `[myArray addObject:myString];`.

Piense en ello como enviar un correo electrónico. Abre un nuevo correo electrónico (objeto `NSInvocation`), `NSInvocation` la dirección de la persona (objeto) a quien desea enviarlo, escribe un mensaje para el destinatario (especifique un `selector` y argumentos) y luego haga clic en "enviar" (llamar a `invoke`).

Consulte [Uso de NSInvocation](#) para obtener más información.

`NSUndoManager` utiliza los objetos `NSInvocation` para que pueda *revertir los comandos*.

Esencialmente, lo que estás haciendo es crear un objeto `NSInvocation` para decir: "Oye, si quieres deshacer lo que acabo de hacer, envía este mensaje a ese objeto, con estos argumentos". Le da el objeto `NSInvocation` al `NSUndoManager`, y agrega ese objeto a una matriz de acciones que se pueden deshacer. Si el usuario llama "Deshacer", `NSUndoManager` simplemente busca la acción más reciente en la matriz e invoca el objeto `NSInvocation` almacenado para realizar la acción necesaria.

Consulte [Registro de operaciones de deshacer](#) para obtener más detalles.

Lea [NSInvocación en línea](https://riptutorial.com/es/ios/topic/8276/nsinvocacion): <https://riptutorial.com/es/ios/topic/8276/nsinvocacion>

Capítulo 118: NSNotificationCenter

Introducción

Las notificaciones de iOS son una forma simple y poderosa de enviar datos de manera flexible. Es decir, el remitente de una notificación no tiene que preocuparse por quién (si alguien) recibe la notificación, simplemente la publica en el resto de la aplicación y puede ser recogida por muchas cosas o nada dependiendo de Estado de su aplicación.

Fuente : - [HACKING con Swift](#)

Parámetros

Parámetro	Detalles
nombre	El nombre de la notificación para la cual se registra al observador; es decir, solo se usan notificaciones con este nombre para agregar el bloque a la cola de operaciones. Si pasa nulo, el centro de notificaciones no usa el nombre de una notificación para decidir si agregar el bloque a la cola de operaciones.
obj	El objeto cuyas notificaciones el observador quiere recibir; es decir, solo las notificaciones enviadas por este remitente se envían al observador. Si pasa nulo, el centro de notificaciones no utiliza un remitente de notificación para decidir si se lo entrega al observador.
cola	La cola de operaciones a la que se debe agregar el bloque. Si pasa nulo, el bloqueo se ejecuta de forma síncrona en el hilo de publicación.
bloquear	El bloque a ejecutar cuando se recibe la notificación. El bloque es copiado por el centro de notificaciones y (la copia) se mantiene hasta que se elimina el registro del observador.

Observaciones

Un objeto NSNotificationCenter (o simplemente, centro de notificación) proporciona un mecanismo para transmitir información dentro de un programa. Un objeto NSNotificationCenter es esencialmente una tabla de despacho de notificaciones.

Para más información, echa un vistazo a la documentación de Apple [aquí](#)

[NSNotification & NSNotificationCenter en Swift](#)

Examples

Convenio de denominación

Las notificaciones son identificadas por objetos NSString globales cuyos nombres se componen de esta manera:

Name of associated class + Did | Will + UniquePartOfName + Notification

Por ejemplo:

- UIApplicationDidBecomeActiveNotification
- UIWindowDidMiniaturizeNotification
- NSTextViewDidChangeSelectionNotification
- NSColorPanelColorDidChangeNotification

Swift 2.3

```
NSNotificationCenter.defaultCenter().addObserver(self,
                                                selector:
#selector(self.testNotification(_:)),
                                                name: "TestNotification",
                                                object: nil)
```

Swift 3

```
NSNotificationCenter.default.addObserver(self,
                                        selector: #selector(self.testNotification(_:)),
                                        name: NSNotification.Name(rawValue:
"TestNotification"),
                                        object: nil)
```

C objetivo

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                        selector:@selector(testNotification:)
                                        name:@"TestNotification"
                                        object:nil];
```

PD: También vale la pena señalar que el número de veces que se ha agregado un observador tiene que ser exactamente el número de veces que se elimina el observador. Un error de novato es agregar el observador en el `viewWillAppear:` de un `UIViewController`, pero eliminar el observador en `viewDidUnload:` provocará un número desigual de empujes y, por lo tanto, se perderán el observador y el selector de notificaciones de forma superflua.

Removiendo observadores

Swift 2.3

```
//Remove observer for single notification
NSNotificationCenter defaultCenter().removeObserver(self, name: "TestNotification", object: nil)

//Remove observer for all notifications
NSNotificationCenter defaultCenter().removeObserver(self)
```

Swift 3

```
//Remove observer for single notification
NotificationCenter.default.removeObserver(self, name: NSNotification.Name(rawValue: "TestNotification"), object: nil)

//Remove observer for all notifications
NotificationCenter.default.removeObserver(self)
```

C objetivo

```
//Remove observer for single notification
[[NSNotificationCenter defaultCenter] removeObserver:self name:@"TestNotification" object:nil];

//Remove observer for all notifications
[[NSNotificationCenter defaultCenter] removeObserver:self];
```

Publicar una notificación

Rápido

```
NSNotificationCenter defaultCenter().postNotificationName("TestNotification", object: self)
```

C objetivo

```
[[NSNotificationCenter defaultCenter] postNotificationName:@"TestNotification" object:nil];
```

Publicar una notificación con datos

Rápido

```
let userInfo: [String: AnyObject] = ["someKey": myObject]
NSNotificationCenter defaultCenter().postNotificationName("TestNotification", object: self,
userInfo: userInfo)
```

C objetivo

```
NSDictionary *userInfo = [NSDictionary dictionaryWithObject:myObject forKey:@"someKey"];
[[NSNotificationCenter defaultCenter] postNotificationName: @"TestNotification" object:nil
userInfo:userInfo];
```

Observando una Notificación

Rápido

```
func testNotification(notification: NSNotification) {
    let userInfo = notification.userInfo
    let myObject: MyObject = userInfo["someKey"]
}
```

C objetivo

```
- (void)testNotification:(NSNotification *)notification {
    NSDictionary *userInfo = notification.userInfo;
    MyObject *myObject = [userInfo objectForKey:@"someKey"];
}
```

Agregar / eliminar un observador con un bloque

En lugar de agregar un observador con un selector, se puede usar un bloque:

```
id testObserver = [[NSNotificationCenter defaultCenter] addObserverForName:@"TestNotification"
                                                                    object:nil
                                                                    queue:nil
                                                                    usingBlock:^(NSNotification*
notification) {
    NSDictionary *userInfo = notification.userInfo;
    MyObject *myObject = [userInfo objectForKey:@"someKey"];
}];
```

El observador puede ser eliminado con:

```
[[NSNotificationCenter defaultCenter] removeObserver:testObserver
```

```
name:@"TestNotification"  
object:nil];
```

Añadir y eliminar el observador por nombre

```
// Add observer  
let observer =  
NSNotificationCenter.defaultCenter().addObserverForName("nameOfTheNotification", object: nil,  
queue: nil) { (notification) in  
    // Do operations with the notification in this block  
}  
  
// Remove observer  
NSNotificationCenter.defaultCenter().removeObserver(observer)
```

Lea `NSNotificationCenter` en línea: <https://riptutorial.com/es/ios/topic/1601/nsnotificationcenter>

Capítulo 119: NSPredicate

Sintaxis

- Subestaciones de cadena de formato de predicado
 - Especificadores de cadena de formato C:% d,% s,% f, etc.
 - Sustitución de objetos:% @
 - Sustitución de Keypath:% K
- Operadores de comparación de predicados
 - =, ==: La expresión de la mano izquierda es igual a la expresión de la mano derecha
 - > =, =>: La expresión de la mano izquierda es mayor o igual que la expresión de la mano derecha
 - <=, = <: La expresión de la mano izquierda es menor o igual que la expresión de la mano derecha
 - >: La expresión de la mano izquierda es mayor que la expresión de la mano derecha
 - <: La expresión de la mano izquierda es menos que la expresión de la mano derecha
 - !=, <>: La expresión de la mano izquierda no es igual a la expresión de la mano derecha
 - ENTRE: La expresión de la mano izquierda se encuentra entre o igual a cualquiera de los valores de la expresión de la derecha, que especifica los límites inferior y superior, por ejemplo: ENTRE {0, 5}
- Operadores de compuestos predicados
 - Y, &&: Lógico Y
 - O, ||: O lógico
 - NO,! : NO lógico
- Operadores de comparación de cadenas de predicado
 - BEGINSWITH: La expresión de la mano izquierda comienza con la expresión de la mano derecha
 - ENDSWITH: la expresión de la mano izquierda termina con la de la mano derecha
 - CONTIENE: La expresión de la mano izquierda contiene la expresión de la mano derecha.
 - LIKE: La expresión de la mano izquierda es igual a la expresión de la mano derecha, con sustitución de comodín
 - *: Coincide con cero o más caracteres
 - ?: Empareja un personaje

Examples

Creando un NSPredicate usando predicateWithBlock

C objetivo

```
NSPredicate *predicate = [NSPredicate predicateWithBlock:^(BOOL(id item,  
NSDictionary *bindings) {
```

```
return [item isKindOfClass:[UILabel class]];
];
```

Rápido

```
let predicate = NSPredicate { (item, bindings) -> Bool in
    return item isKindOfClass(UILabel.self)
}
```

En este ejemplo, el predicado coincidirá con los elementos que son de la clase `UILabel`.

Creando un NSPredicate usando predicateWithFormat

C objetivo

```
NSPredicate *predicate = [NSPredicate predicateWithFormat: @"self[SIZE] = %d", 5];
```

Rápido

```
let predicate = NSPredicate(format: "self[SIZE] >= %d", 5)
```

En este ejemplo, el predicado hará coincidir los elementos que son matrices con una longitud de al menos 5.

Creando un NSPredicate con Variables de Sustitución

Un `NSPredicate` puede usar variables de sustitución para permitir que los valores se unan sobre la marcha.

C objetivo

```
NSPredicate *template = [NSPredicate predicateWithFormat: @"self BEGINSWITH $letter"];
NSDictionary *variables = @{@"letter": @"r"};
NSPredicate *beginsWithR = [template predicateWithSubstitutionVariables: variables];
```

Rápido

```
let template = NSPredicate(format: "self BEGINSWITH $letter")
let variables = ["letter": "r"]
let beginsWithR = template.predicateWithSubstitutionVariables(variables)
```

El predicado de la plantilla no se modifica por `predicateWithSubstitutionVariables`. En su lugar, se

crea una copia, y esa copia recibe las variables de sustitución.

Usando NSPredicate para filtrar una matriz

C objetivo

```
NSArray *heroes = @[@"tracer", @"bastion", @"reaper", @"junkrat", @"roadhog"];

NSPredicate *template = [NSPredicate predicateWithFormat:@"self BEGINSWITH $letter"];

NSDictionary *beginsWithRVariables = @{@"letter": @"r"};
NSPredicate *beginsWithR = [template predicateWithSubstitutionVariables:
beginsWithRVariables];

NSArray *beginsWithRHeroes = [heroes filteredArrayUsingPredicate: beginsWithR];
// ["reaper", "roadhog"]

NSDictionary *beginsWithTVariables = @{@"letter": @"t"};
NSPredicate *beginsWithT = [template predicateWithSubstitutionVariables: beginsWithTVariables];

NSArray *beginsWithTHeroes = [heroes filteredArrayUsingPredicate: beginsWithT];
// ["tracer"]
```

Rápido

```
let heroes = ["tracer", "bastion", "reaper", "junkrat", "roadhog"]

let template = NSPredicate(format: "self BEGINSWITH $letter")

let beginsWithRVariables = ["letter": "r"]
let beginsWithR = template.predicateWithSubstitutionVariables(beginsWithRVariables)

let beginsWithRHeroes = heroes.filter { beginsWithR.evaluateWithObject($0) }
// ["reaper", "roadhog"]

let beginsWithTVariables = ["letter": "t"]
let beginsWithT = template.predicateWithSubstitutionVariables(beginsWithTVariables)

let beginsWithTHeroes = heroes.filter { beginsWithT.evaluateWithObject($0) }
// ["tracer"]
```

Validación de formularios utilizando NSPredicate

```
NSString *emailRegex = @"[A-Z0-9a-z]([A-Z0-9a-z._-]{0,64})+[A-Z0-9a-z]+@[A-Z0-9a-z]+([A-Za-z0-9.-]{0,64})+([A-Z0-9a-z])+\.[A-Za-z]{2,4}";
NSString *firstNameRegex = @"[0-9A-Za-z]{2,32}$";
NSString *lastNameRegex = @"[0-9A-Za-z\''-]{2,32}$";
NSString *mobileNumberRegex = @"^[0-9]{10}$";
NSString *zipcodeRegex = @"^[0-9]{5}$";
NSString *SSNRegex = @"^\d{3}-?\d{2}-?\d{4}$";
NSString *addressRegex = @"^[ A-Za-z0-9]{2,32}$";
NSString *cityRegex = @"^[ A-Za-z0-9]{2,25}$";
```

```

NSString *PINRegex = @"^[0-9]{4}$";
NSString *driversLiscRegex = @"^[0-9a-zA-Z]{5,20}$";

-(BOOL)validateEmail {
    //Email address field should give an error when the email address begins with ".", "-", "_"
    .
    NSPredicate *emailPredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
emailRegex];
    return ([emailPredicate evaluateWithObject:self.text] && self.text.length <= 64 &&
([self.text rangeOfString:@".."].location == NSNotFound));
}

- (BOOL)validateFirstName {
    NSPredicate *firstNamePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
firstNameRegex];
    return [firstNamePredicate evaluateWithObject:self.text];
}

- (BOOL)validateLastName {
    NSPredicate *lastNamePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
lastNameRegex];
    return [lastNamePredicate evaluateWithObject:self.text];
}

- (BOOL)validateAlphaNumericMin2Max32 {
    NSPredicate *firstNamePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
firstNameRegex];
    return [firstNamePredicate evaluateWithObject:self.text];
}

- (BOOL)validateMobileNumber {
    NSString *strippedMobileNumber = [[[[self.text stringByReplacingOccurrencesOfString:@"("
withString:@""]
                                stringByReplacingOccurrencesOfString:@")"
withString:@""]
                                stringByReplacingOccurrencesOfString:@"-"
withString:@""]
                                stringByReplacingOccurrencesOfString:@" "
withString:@""];

    NSPredicate *mobileNumberPredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
mobileNumberRegex];

    return [mobileNumberPredicate evaluateWithObject:strippedMobileNumber];
}

- (BOOL)validateZipcode {
    NSPredicate *zipcodePredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
zipcodeRegex];

    return [zipcodePredicate evaluateWithObject:self.text];
}

- (BOOL)validateSSN {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@", SSNRegex];

    return [predicate evaluateWithObject:self.text];
}

- (BOOL)validateAddress {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",

```

```

addressRegex];

    return [predicate evaluateWithObject:self.text];
}

- (BOOL)validateCity {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@", cityRegex];
    return [predicate evaluateWithObject:self.text];
}

- (BOOL)validatePIN {
    NSPredicate *predicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@", PINRegex];

    return [predicate evaluateWithObject:self.text];
}

- (BOOL)validateDriversLiscNumber {
    if([self.text length] > 20) {
        return NO;
    }
    NSPredicate *driversLiscPredicate = [NSPredicate predicateWithFormat:@"SELF MATCHES %@",
driversLiscRegex];

    return [driversLiscPredicate evaluateWithObject:self.text];
}

```

NSPredica con la condición `AND`, `OR` y `NOT`

El predicado condicional será más limpio y seguro al usar la clase `NSCompoundPredicate` que proporciona operadores booleanos básicos para los predicados dados.

C objetivo

Y - Condición

```

NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
NSPredicate *combinedPredicate = [NSCompoundPredicate andPredicateWithSubpredicates:
@[predicate,anotherPredicate]];

```

O - Condición

```

NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];
NSPredicate *combinedPredicate = [NSCompoundPredicate orPredicateWithSubpredicates:
@[predicate,anotherPredicate]];

```

NO - Condición

```

NSPredicate *predicate = [NSPredicate predicateWithFormat:@"samplePredicate"];
NSPredicate *anotherPredicate = [NSPredicate predicateWithFormat:@"anotherPredicate"];

```

```
NSPredicate *combinedPredicate = [NSCompoundPredicate notPredicateWithSubpredicate:  
@[predicate, anotherPredicate]];
```

Lea NSPredicate en línea: <https://riptutorial.com/es/ios/topic/5796/nspredicate>

Capítulo 120: NSTimer

Parámetros

Parámetro	Detalles
<code>interval</code>	El tiempo, en segundos, para esperar antes de encender el temporizador; o, en temporizadores de repetición, el tiempo entre disparos.
<code>target</code>	El objeto para llamar al <code>selector</code> en
<code>selector</code>	En Swift, un objeto <code>Selector</code> que especifica el método para llamar al <code>target</code>
<code>repeats</code>	Si es <code>false</code> , dispara el temporizador solo una vez. Si es <code>true</code> , dispara el temporizador cada <code>interval</code> segundos.

Observaciones

Un `NSTimer` permite enviar un mensaje a un destino una vez transcurrido un período de tiempo específico.

Examples

Creando un temporizador

Esto creará un temporizador para llamar al método `doSomething` en `self` en 5 segundos.

Rápido

```
let timer = NSTimer.scheduledTimerWithTimeInterval(5,
    target: self,
    selector: Selector(doSomething()),
    userInfo: nil,
    repeats: false)
```

Swift 3

```
let timer = Timer.scheduledTimer(timeInterval: 1,
    target: self,
    selector: #selector(doSomething()),
    userInfo: nil,
    repeats: true)
```

C objetivo

```
NSTimer *timer = [NSTimer scheduledTimerWithTimeInterval:5.0 target:self
```

```
selector:@selector(doSomething) userInfo:nil repeats:NO];
```

La configuración se repite en `false/NO` indica que queremos que el temporizador se active solo una vez. Si configuramos esto en `true/YES`, se dispararía cada cinco segundos hasta que se invalide manualmente.

Manualmente disparando un temporizador

Rápido

```
timer.fire()
```

C objetivo

```
[timer fire];
```

Llamar al método de `fire` hace que un `NSTimer` realice la tarea que normalmente habría realizado en un horario.

En un **temporizador que no se repite**, esto invalidará automáticamente el temporizador. Es decir, activar el `fire` antes de que `fire` el intervalo de tiempo dará lugar a una sola invocación.

En un **temporizador de repetición**, esto simplemente invocará la acción sin interrumpir el horario habitual.

Invalidando un temporizador

Rápido

```
timer.invalidate()
```

C objetivo

```
[timer invalidate];
```

Esto detendrá el temporizador de disparar. **Debe llamarse desde el hilo en el que se creó el temporizador**, consulte [las notas de Apple](#):

Debe enviar este mensaje desde el hilo en el que se instaló el temporizador. Si envía este mensaje desde otro hilo, es posible que la fuente de entrada asociada con el temporizador no se elimine de su ciclo de ejecución, lo que podría impedir que el hilo salga correctamente.

Notas: Una vez que se ha invalidado el temporizador, es imposible disparar el mismo temporizador invalidado. En su lugar, debe inicializar nuevamente el temporizador invalidado y activar el método de disparo.

Opciones de frecuencia del temporizador

Evento de temporizador repetido

Rápido

```
class ViewController: UIViewController {

    var timer = NSTimer()

    override func viewDidLoad() {
        NSTimer.scheduledTimerWithTimeInterval(1.0, target: self, selector:
Selector(self.timerMethod()), userInfo: nil, repeats: true)
    }

    func timerMethod() {
        print("Timer method called")
    }

    func endTimer() {
        timer.invalidate()
    }
}
```

Swift 3

```
class ViewController: UIViewController {

    var timer = Timer()

    override func viewDidLoad() {
        Timer.scheduledTimer(timeInterval: 1.0, target: self, selector:
#selector(self.timerMethod()), userInfo: nil, repeats: true)
    }

    func timerMethod() {
        print("Timer method called")
    }

    func endTimer() {
        timer.invalidate()
    }
}
```

Debe ser invalidado manualmente si lo desea.

Rápido

Evento temporizador retardado no repetido

```
NSTimer.scheduledTimerWithTimeInterval(3.0, target: self, selector:
Selector(self.timerMethod()), userInfo: nil, repeats: false)
```

Swift 3

```
Timer.scheduledTimer(timeInterval: 3.0, target: self, selector:
#selector(self.timerMethod()), userInfo: nil, repeats: false)
```

El temporizador se activará una vez, 3 segundos después del tiempo de ejecución. Será invalidado automáticamente, una vez disparado.

Paso de datos utilizando el temporizador

Si desea pasar algunos datos con el disparador del temporizador, puede hacerlo con el parámetro `userInfo`.

Este es el enfoque simple que brinda una breve idea de cómo puede pasar los datos al método activado desde el temporizador.

[*Swift 3*]

```
Timer.scheduledTimer(timeInterval: 1.0, target: self, selector:#selector(iGotCall(sender:)),
userInfo: ["Name": "i am iOS guy"], repeats:true)
```

[*Objetivo - C*]

```
NSTimer* timer = [NSTimer scheduledTimerWithTimeInterval:1.0
                    target:self
                    selector:@selector(iGotCall:)
                    userInfo:@"i am iOS guy" repeats:YES];
```

La línea de código anterior pasa `["Name": "i am iOS guy"]` a la `userInfo` del `userInfo`. Así que ahora, cuando se recibe la llamada de `iGotCall`, puede obtener el valor pasado como se muestra a continuación en el fragmento de código.

[*Swift 3*]

```
func iGotCall(sender: Timer) {
    print((sender.userInfo!))
}
```

[*Objetivo - C*]

```
- (void)iGotCall:(NSTimer*)theTimer {
    NSLog(@"%@", (NSString*)[theTimer userInfo]);
}
```

Lea `NSTimer` en línea: <https://riptutorial.com/es/ios/topic/2624/nstimer>

Capítulo 121: NSURConexión

Examples

Métodos de delegado

// conformar el protocolo NSURLConnectionDelegate.

```
@interface ViewController : UIViewController<NSURLConnectionDelegate>
{
    NSMutableData *_responseData;
}
```

// Implementación de los métodos del protocolo NSURLConnection.

```
#pragma mark NSURLConnection Delegate Methods

- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response
{
    // A response has been received, this is where we initialize the instance var you created
    // so that we can append data to it in the didReceiveData method
    // Furthermore, this method is called each time there is a redirect so reinitializing it
    // also serves to clear it
    _responseData = [[NSMutableData alloc] init];
}

- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data {
    // Append the new data to the instance variable you declared
    [_responseData appendData:data];
}

- (NSCachedURLResponse *)connection:(NSURLConnection *)connection
    willCacheResponse:(NSCachedURLResponse*)cachedResponse {
    // Return nil to indicate not necessary to store a cached response for this connection
    return nil;
}

- (void)connectionDidFinishLoading:(NSURLConnection *)connection {
    // The request is complete and data has been received
    // You can parse the stuff in your instance variable now
}

- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error {
    // The request has failed for some reason!
    // Check the error var
}
```

Solicitud sincrónica

```
NSURLRequest * urlRequest = [NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://google.com"]];
NSURLResponse * response = nil;
```

```
NSError * error = nil;
NSData * data = [NSURLConnection sendSynchronousRequest:urlRequest
                                returningResponse:&response
                                error:&error];

if (error == nil)
{
    // Parse data here
}
```

Solicitud asincrona

```
// Create the request instance.
NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://google.com"]];

// Create url connection and fire request
NSURLConnection *conn = [[NSURLConnection alloc] initWithRequest:request delegate:self];
```

Lea NSURConexión en línea: <https://riptutorial.com/es/ios/topic/6004/nsurconexion>

Capítulo 122: NSURL

Examples

Cómo obtener el último componente de cadena de NSURL String.

```
NSURL *url = [NSURL URLWithString:@"http://www.example.com/images/apple-tree.jpg"];
NSString *fileName = [url lastPathComponent];
// fileName = "apple-tree.jpg"
```

Cómo obtener el último componente de cadena de la URL (NSURL) en Swift

Swift 2.3

```
let url = NSURL(string: "http://google.com/lastPath")
let lastPath = url?.lastPathComponent
```

Swift 3.0

```
let url = URL(string: "http://google.com/lastPath")
let lastPath = url?.lastPathComponent
```

Lea NSURL en línea: <https://riptutorial.com/es/ios/topic/4610/nsurl>

Capítulo 123: NSUserActivity

Introducción

Se puede usar un objeto `NSUserActivity` para coordinar eventos significativos en una aplicación con el sistema. Es la base para el [traspaso](#) entre diferentes dispositivos que ejecutan iOS y macOS. Además, también se puede utilizar para mejorar la indexación pública y aumentar o crear resultados de búsqueda de Spotlight para una aplicación. A partir de iOS 10, también se puede usar para coordinar las interacciones entre su aplicación y Siri utilizando SiriKit.

Observaciones

Tipos de actividad

Los tipos de actividad admitidos deben definirse en el archivo `Info.plist` su aplicación bajo la clave `NSUserActivityTypes`. Las actividades están vinculadas a su ID de equipo de desarrollador, lo que significa que la coordinación de actividades está restringida entre aplicaciones que tienen la misma ID de equipo (por ejemplo, "Safari" no puede aceptar una actividad de transferencia desde "Chrome" o viceversa).

Convertirse / renunciar a la actividad actual

Al marcar una actividad como actual con el uso de `becomeCurrent` está disponible para el traspaso o la indexación de Spotlight. Solo una actividad puede ser actual a la vez. Puede marcar una actividad como inactiva sin invalidar llamando a `resignCurrent`.

Si `invalidate` una actividad, es posible que la misma instancia no se vuelva actual.

No marque una actividad como actual cuando la proporcione para [SiriKit](#).

Indexación de búsqueda

Las actividades **no** deben utilizarse como un mecanismo de indexación de propósito general dentro de su aplicación. En su lugar, solo deben usarse en respuesta a acciones iniciadas por el usuario. Para indexar todo el contenido de su aplicación, use `CoreSpotlight`.

Examples

Creando un `NSUserActivity`

Para crear un objeto `NSUserActivity`, su aplicación debe declarar los tipos de actividades que

admite en su archivo `Info.plist` . Las actividades compatibles están definidas por su aplicación y deben ser únicas. Una actividad se define utilizando un esquema de nombres de estilo de dominio inverso (es decir, "com.companyName.productName.activityName"). Aquí es cómo podría verse una entrada en su `Info.plist`:

Llave	Valor
NSUserActivityTypes	[Formación]
- item0	com.companyName.productName.activityName01
- Objeto 1	com.companyName.productName.activityName02

Una vez que haya definido todos los tipos de actividades compatibles, puede comenzar a acceder y usarlos en el código de su aplicación.

Para crear un objeto `NSUserActivity` debe hacer lo siguiente

```
// Initialize the activity object and set its type from one of the ones specified in your
app's plist
NSUserActivity *currentActivity = [[NSUserActivity alloc]
initWithActivityType:@"com.companyName.productName.activityName01"];

// Set the title of the activity.
// This title may be displayed to the user, so make sure it is localized and human-readable
currentActivity.title = @"Current Activity";

// Configure additional properties like userInfo which will be included in the activity
currentActivity.userInfo = @{@"informationKey" : @"value"};

// Configure the activity so the system knows what may be done with it
// It is important that you only set YES to tasks that your application supports
// In this example, we will only enable the activity for use with Handoff
[currentActivity setEligibleForHandoff:YES];
[currentActivity setEligibleForSearch:NO]; // Defaults to NO
[currentActivity setEligibleForPublicIndexing:NO]; // Defaults to NO

// Set this activity as the current user activity
// Only one activity may be current at a time on a device. Calling this method invalidates any
other current activities.
[currentActivity becomeCurrent];
```

Después de esto, la actividad anterior debería estar disponible para la transferencia (aunque se requiere más trabajo para manejar correctamente la "transferencia").

Lea `NSUserActivity` en línea: <https://riptutorial.com/es/ios/topic/10716/nsuseractivity>

Capítulo 124: NSUserDefaults

Sintaxis

- `UserDefaults.standard.set(dic, forKey: "LoginSession") //Save value inside userdefaults`
 - `UserDefaults.standard.object(forKey: "LoginSession") as? [String:AnyObject] ?? [:]`
`//Get value from UserDefaults`

Observaciones

NSUserDefaults, que se utiliza para almacenar todo tipo de DataType, y puede obtener su valor en cualquier lugar de la clase de aplicación. [NSUserDefaults](#)

Examples

Estableciendo valores

Para establecer un valor en `NSUserDefaults`, puede usar las siguientes funciones:

Vencejo <3

```
setBool(_:forKey:)
setFloat(_:forKey:)
setInteger(_:forKey:)
setObject(_:forKey:)
setDouble(_:forKey:)
setURL(_:forKey:)
```

Swift 3

En Swift 3, los nombres de la función se cambian para `set` del `set` seguido del tipo.

```
set(_:forKey:)
```

C objetivo

```
-(void)setBool:(BOOL)value forKey:(nonnull NSString *)defaultName;
-(void)setFloat:(float)value forKey:(nonnull NSString *)defaultName;
-(void)setInteger:(NSInteger)value forKey:(nonnull NSString *)defaultName;
-(void)setObject:(nullable id)value forKey:(nonnull NSString *)defaultName;
-(void)setDouble:(double)value forKey:(nonnull NSString *)defaultName;
-(void)setURL:(nullable NSURL *)value forKey:(nonnull NSString *)defaultName;
```

Ejemplo de uso sería:

Vencejo <3

```
NSUserDefaults.standardUserDefaults.setObject("Netherlands", forKey: "HomeCountry")
```

Swift 3

```
UserDefaults.standard.set("Netherlands", forKey: "HomeCountry")
```

C objetivo

```
[[NSUserDefaults standardUserDefaults] setObject:@"Netherlands" forKey:@"HomeCountry"];
```

Objetos personalizados

Para guardar objetos personalizados en `NSUserDefaults`, debe hacer que su CustomClass confirme el protocolo de `NSCoding`. Necesitas implementar los siguientes métodos:

Rápido

```
public func encodeWithCoder(aCoder: NSCoder) {
    aCoder.encodeObject(name, forKey:"name")
    aCoder.encodeObject(unitId, forKey: "unitId")
}

required public init(coder aDecoder: NSCoder) {
    super.init()
    name = aDecoder.decodeObjectForKey("name") as? String
    unitId = aDecoder.decodeIntegerForKey("unitId") as? NSInteger
}
```

C objetivo

```
- (id)initWithCoder:(NSCoder *)coder {
    self = [super init];
    if (self) {
        name = [coder decodeObjectForKey:@"name"];
        unitId = [coder decodeIntegerForKey:@"unitId"];
    }
    return self;
}

- (void)encodeWithCoder:(NSCoder*)coder {
    [coder encodeObject:name forKey:@"name"];
    [coder encodeInteger:unitId forKey:@"unitId"];
}
```

Obtención de valores predeterminados

Para obtener un valor en `NSUserDefaults` puede usar las siguientes funciones:

Rápido

```
arrayForKey(_:)
boolForKey(_:)
dataForKey(_:)
dictionaryForKey(_:)
floatForKey(_:)
integerForKey(_:)
objectForKey(_:)
stringArrayForKey(_:)
stringForKey(_:)
doubleForKey(_:)
URLForKey(_:)
```

C objetivo

```
-(nullable NSArray *)arrayForKey:(nonnull NSString *)defaultName;
-(BOOL)boolForKey:(nonnull NSString *)defaultName;
-(nullable NSData *)dataForKey:(nonnull NSString *)defaultName;
-(nullable NSDictionary<NSString *, id> *)dictionaryForKey:(nonnull NSString *)defaultName;
-(float)floatForKey:(nonnull NSString *)defaultName;
-(NSInteger)integerForKey:(nonnull NSString *)defaultName;
-(nullable id)objectForKey:(nonnull NSString *)key;
-(nullable NSArray<NSString *> *)stringArrayForKey:(nonnull NSString *)defaultName;
-(nullable NSString *)stringForKey:(nonnull NSString *)defaultName;
-(double)doubleForKey:(nonnull NSString *)defaultName;
-(nullable NSURL *)URLForKey:(nonnull NSString *)defaultName;
```

Ejemplo de uso sería:

Rápido

```
let homeCountry = NSUserDefaults.standardUserDefaults().stringForKey("HomeCountry")
```

C objetivo

```
NSString *homeCountry = [[NSUserDefaults standardUserDefaults] stringForKey:@"HomeCountry"];
```

Valores de ahorro

`NSUserDefaults` se escriben en el disco periódicamente por el sistema, pero hay ocasiones en las que desea que se guarden sus cambios inmediatamente, como cuando la aplicación pasa al estado de fondo. Esto se hace llamando al `synchronize`.

Rápido

```
NSUserDefaults.standardUserDefaults().synchronize()
```

C objetivo

```
[[NSUserDefaults standardUserDefaults] synchronize];
```

Utilice a los gerentes para guardar y leer datos

Si bien puede usar los métodos de `NSUserDefaults` cualquier lugar, a veces puede ser mejor definir un administrador que `NSUserDefaults` y lea los `NSUserDefaults` por usted y luego use ese administrador para leer o escribir sus datos.

Supongamos que queremos guardar la puntuación de un usuario en `NSUserDefaults`. Podemos crear una clase como la de abajo que tiene dos métodos: `setHighScore` y `highScore`. En cualquier lugar donde desee acceder a las puntuaciones más altas, cree una instancia de esta clase.

Rápido

```
public class ScoreManager: NSObject {

    let highScoreDefaultKey = "HighScoreDefaultKey"

    var highScore = {
        set {
            // This method includes your implementation for saving the high score
            // You can use NSUserDefaults or any other data store like CoreData or
            // SQLite etc.

            NSUserDefaults.standardUserDefaults().setInteger(newValue, forKey:
highScoreDefaultKey)
            NSUserDefaults.standardUserDefaults().synchronize()
        }
        get {
            //This method includes your implementation for reading the high score

            let score =
NSUserDefaults.standardUserDefaults().objectForKey(highScoreDefaultKey)

            if (score != nil) {
                return score.integerValue;
            } else {
                //No high score available, so return -1
                return -1;
            }
        }
    }
}
```

C objetivo

```
#import "ScoreManager.h"

#define HIGHSCORE_KEY @"highScore"

@implementation ScoreManager

- (void)setHighScore:(NSInteger) highScore {
    // This method includes your implementation for saving the high score
    // You can use UserDefaults or any other data store like CoreData or
    // SQLite etc.

    [[NSUserDefaults standardUserDefaults] setInteger:highScore forKey:HIGHSCORE_KEY];
    [[NSUserDefaults standardUserDefaults] synchronize];
}

- (NSInteger)highScore
{
    //This method includes your implementation for reading the high score

    NSNumber *highScore = [[NSUserDefaults standardUserDefaults] objectForKey:HIGHSCORE_KEY];
    if (highScore) {
        return highScore.integerValue;
    }else
    {
        //No high score available, so return -1

        return -1;
    }
}

@end
```

Las ventajas son que:

1. La implementación de su proceso de lectura y escritura es solo en un lugar y puede cambiarla (por ejemplo, cambiar de `NSUserDefaults` a `Core Data`) cuando lo desee y no preocuparse por cambiar todos los lugares en los que está trabajando con la puntuación más alta.
2. Simplemente llame a un solo método cuando desee acceder a la puntuación o escríbalo.
3. Simplemente depúralo cuando veas un error o algo así.

Nota

Si le preocupa la sincronización, es mejor usar una clase singleton que administre la sincronización.

Borrar `NSUserDefaults`

Rápido

```
let bundleIdentifier = NSBundle mainBundle().bundleIdentifier()

NSUserDefaults.standardUserDefaults().removePersistentDomainForName(bundleIdentifier)
```

C objetivo

```
NSString *bundleIdentifier = [[NSBundle mainBundle] bundleIdentifier];

[[NSUserDefaults standardUserDefaults] removePersistentDomainForName: bundleIdentifier];
```

Usos de UserDefaults en Swift 3

Todas las aplicaciones necesitaban almacenar la sesión del usuario o los detalles relacionados con el usuario dentro de la aplicación en UserDefaults. Por lo tanto, creamos toda la lógica dentro de una clase para administrar UserDefaults de una manera mejor.

Swift 3

```
import Foundation

public struct Session {

    fileprivate static let defaults = UserDefaults.standard

    enum userValues: String {
        case auth_token
        case email
        case fname
        case mobile
        case title
        case userId
        case userType
        case OTP
        case isApproved
    }

    //MARK: - Getting here User Details
    static func getSessionDetails()->[String:AnyObject]? {
        let dictionary = defaults.object(forKey: "LoginSession") as? [String:AnyObject]
        return dictionary
    }

    //MARK: - Saving Device Token
    static func saveDeviceToken(_ token:String){
        guard (getSessionDetails() ?? "").isEmpty else {
            return
        }
        defaults.removeObject(forKey: "deviceToken")
        defaults.set(token, forKey: "deviceToken")
        defaults.synchronize()
    }
}
```

```

//MARK: - Getting Token here
static func gettingDeviceToken()->String?{
    let token = defaults.object(forKey: "deviceToken") as? String
    if token == nil{
        return ""
    }else{ return token}
}

//MARK: - Setting here User Details
static func setUserSessionDetails(_ dic :[String : AnyObject]){
    defaults.removeObject(forKey: "LoginSession")
    defaults.set(dic, forKey: "LoginSession")
    defaults.synchronize()
}

//MARK:- Removing here all Default Values
static func userSessionLogout(){
    //Set Activity
    defaults.removeObject(forKey: "LoginSession")
    defaults.synchronize()
}

//MARK: - Get value from session here
static func getUserValues(value: userValues) -> String? {
    let dic = getUserSessionDetails() ?? [:]
    guard let value = dic[value.rawValue] else{
        return ""
    }
    return value as? String
}
}
}

```

Uso de la clase UserDefaults

```

//Saving user Details
Session.setUserSessionDetails(json ?? [:])

//Retriving user Details
let userId = Session.getUserValues(value: .userId) ?? ""

```

Lea UserDefaults en línea: <https://riptutorial.com/es/ios/topic/3150/nsuserdefaults>

Capítulo 125: Objetos asociados a Objective-C

Introducción

Presentado por primera vez en iOS 3.1 como parte del tiempo de ejecución de Objective-C, los objetos asociados proporcionan una manera de agregar variables de instancia a un objeto de clase existente (sin subclasificación).

Esto significa que podrá adjuntar cualquier objeto a cualquier otro objeto sin crear subclases.

Sintaxis

- `void objc_setAssociatedObject (objeto id, void * key, id valor, objc_AssociationPolicy policy)`
- `id objc_getAssociatedObject (id objeto, void * key)`
- `void objc_removeAssociatedObjects (id object)`

Parámetros

Param	Detalles
objeto	El objeto existente que desea modificar.
llave	Básicamente, este puede ser cualquier puntero que tenga una dirección de memoria constante, pero una buena práctica es usar aquí una propiedad computada (getter)
valor	El objeto que quieres agregar.
política	La política de memoria para este nuevo <code>value</code> es decir, debe conservarse / asignarse, copiarse, etc. como cualquier otra propiedad que declararía.

Observaciones

Más detalles aquí:

[NSHipster](#)

[@kostiakoval](#)

[Kingscocoa](#)

Examples

Ejemplo de objeto asociado básico

Supongamos que necesitamos agregar un objeto `SomeClass` a `SomeClass` (no podemos `SomeClass` subclasse).

En este ejemplo, no solo creamos un objeto asociado, sino que también lo envolvemos en una propiedad computada en una categoría para una mayor limpieza.

```
#import <objc/runtime.h>

@interface SomeClass (MyCategory)
// This is the property wrapping the associated object. below we implement the setter and
getter which actually utilize the object association
@property (nonatomic, retain) NSString *associated;
@end

@implementation SomeClass (MyCategory)

- (void)setAssociated:(NSString *)object {
    objc_setAssociatedObject(self, @selector(associated), object,
                            OBJC_ASSOCIATION_RETAIN_NONATOMIC);
}

- (NSString *)associated {
    return objc_getAssociatedObject(self, @selector(associated));
}
```

Ahora sería tan fácil como esto utilizar la propiedad.

```
SomeClass *instance = [SomeClass alloc] init];
instance.associated = @"this property is an associated object under the hood";
```

Lea Objetos asociados a Objective-C en línea: <https://riptutorial.com/es/ios/topic/9102/objetos-asociados-a-objective-c>

Capítulo 126: OpenGL

Introducción

OpenGL ES es una biblioteca de gráficos que iOS usa para hacer renderización 3D.

Examples

Proyecto de ejemplo

Un [proyecto de ejemplo \(Git repo\)](#) que se puede usar como punto de partida para hacer una representación 3D. El código para configurar OpenGL y los sombreadores es bastante largo y tedioso, por lo que no cabe bien en este formato de ejemplo. Partes posteriores de la misma se pueden mostrar en ejemplos separados que detallan qué está sucediendo exactamente con cada pieza de código, pero por ahora aquí está el proyecto Xcode.

Lea OpenGL en línea: <https://riptutorial.com/es/ios/topic/9324/opengl>

Capítulo 127: Operaciones de toda la aplicación

Examples

Obtén lo mejor de UINavigationController

Un enfoque común para obtener la mayoría de `UIViewController` superior es obtener el `RootViewController` de su `UIWindow` activo. Escribí una extensión para esto:

```
extension UIApplication {  
  
    func topViewController(_ base: UIViewController? =  
        UIApplication.shared.keyWindow?.rootViewController) -> UIViewController {  
  
        if let nav = base as? UINavigationController {  
            return topViewController(nav.visibleViewController)  
        }  
  
        if let tab = base as? UITabBarController {  
            if let selected = tab.selectedViewController {  
                return topViewController(selected)  
            }  
        }  
  
        if let presented = base?.presentedViewController {  
            return topViewController(presented)  
        }  
  
        return base!  
    }  
}
```

Interceptar eventos del sistema

Usando el `NotificationCenter` de iOS, que puede ser muy poderoso, puedes interceptar ciertos eventos de toda la aplicación:

```
NotificationCenter.default.addObserver(  
    self,  
    selector: #selector(ViewController.do(_:)),  
    name: NSNotification.Name.UIApplicationDidBecomeActive,  
    object: nil)
```

Puede registrarse para muchos más eventos, solo eche un vistazo a <https://developer.apple.com/reference/foundation/nsnotification.name> .

Lea Operaciones de toda la aplicación en línea:

<https://riptutorial.com/es/ios/topic/7188/operaciones-de-toda-la-aplicacion>

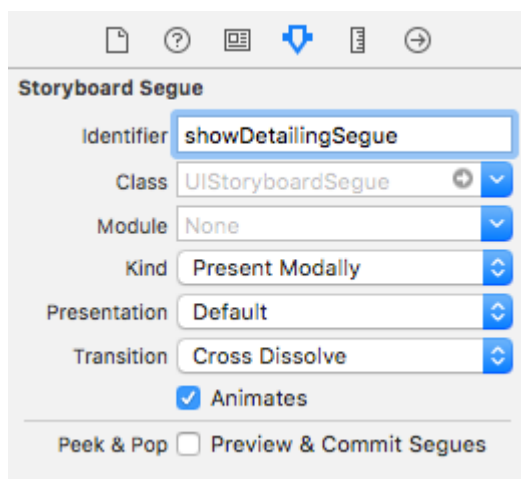
Capítulo 128: Pasando datos entre los controladores de vista

Examples

Usando Segues (pasando datos hacia adelante)

Para pasar los datos del controlador de vista actual al siguiente controlador de vista nuevo (no un controlador de vista anterior) usando segues, primero cree un segmento con un identificador en el guión gráfico relevante. Reemplace el método `prepareForSegue` su controlador de vista actual. Dentro del método, compruebe el segmento que acaba de crear por su identificador. Convierta el controlador de vista de destino y pase los datos al mismo estableciendo las propiedades en el controlador de vista descendente.

Estableciendo un identificador para un segue:



Los segmentos se pueden realizar programáticamente o usando el evento de acción de botón establecido en el guión gráfico con `ctrl + arrastrar` al controlador de vista de destino. Puede solicitar un segue programáticamente, cuando sea necesario, utilizando el identificador de segue en el controlador de vista:

C objetivo

```
- (void) showDetail {
    [self performSegueWithIdentifier:@"showDetailingSegue" sender:self];
}
```

Rápido

```
func showDetail() {
    self.performSegue(withIdentifier: "showDetailingSegue", sender: self)
}
```

Puede configurar la carga útil segue en la versión `prepareForSegue` método `prepareForSegue` . Puede establecer las propiedades requeridas antes de que se cargue el controlador de vista de destino.

C objetivo

```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    if([segue.identifier isEqualToString:@"showDetailingSegue"]){
        DetailViewController *controller = (DetailViewController
*)segue.destinationViewController;
        controller.isDetailingEnabled = YES;
    }
}
```

Rápido

```
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
    if segue.identifier == "showDetailingSegue" {
        let controller = segue.destinationViewController as! DetailViewController
        controller.isDetailingEnabled = true
    }
}
```

`DetailViewController` es el nombre del segundo controlador de vista e `isDetailingEnabled` es una variable pública en ese controlador de vista.

Para expandir este patrón, puede tratar un método público en `DetailViewController` como un pseudoinicializador, para ayudar a inicializar cualquier variable requerida. Esto auto documentará las variables que deben configurarse en `DetailViewController` sin tener que leer su código fuente. También es un lugar práctico para poner valores por defecto.

C objetivo

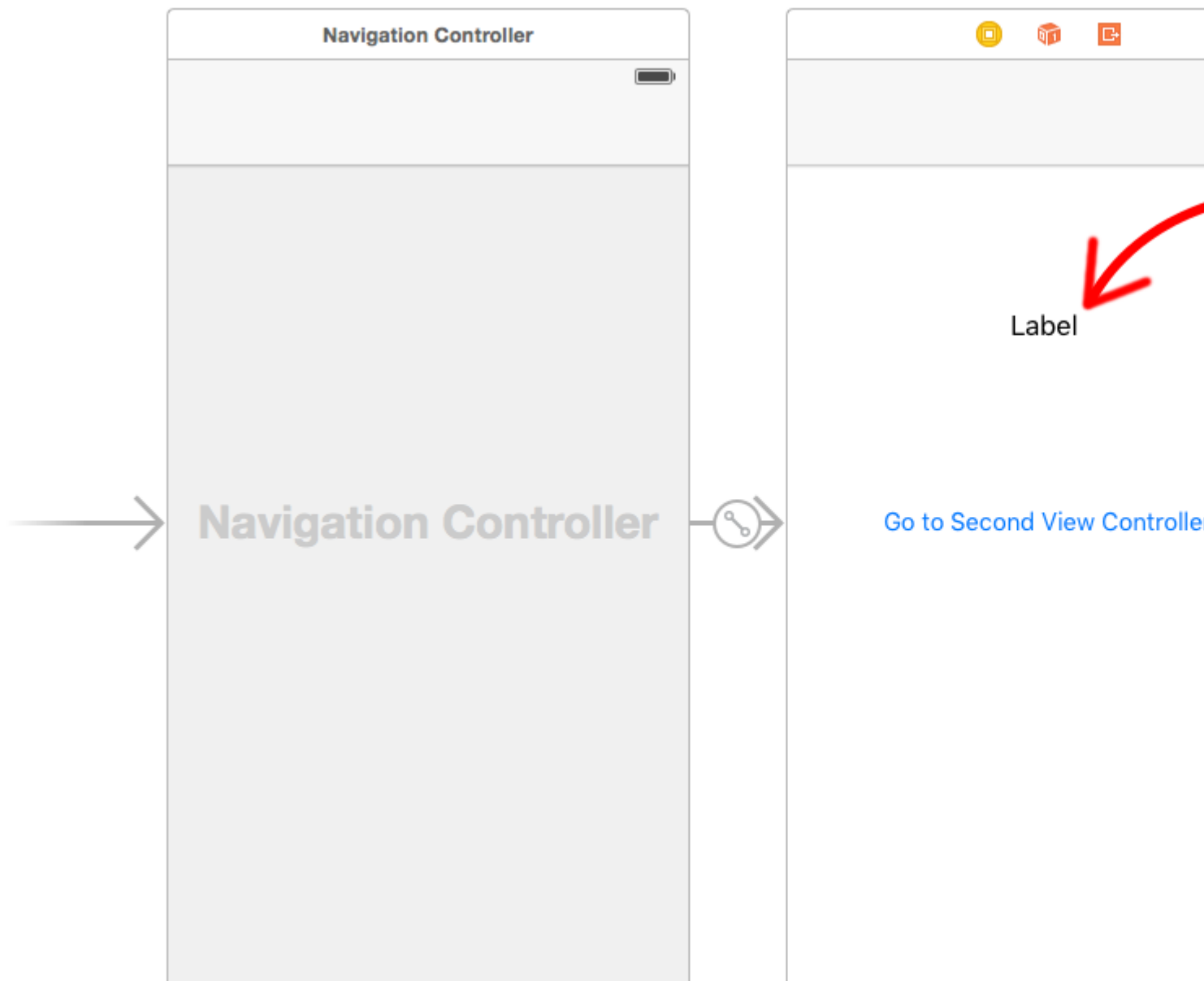
```
- (void)initVC:(BOOL *)isDetailingEnabled {
    self.isDetailingEnabled = isDetailingEnabled
}
```

Rápido

```
func initVC(isDetailingEnabled: Bool) {
    self.isDetailingEnabled = isDetailingEnabled
}
```

Usando el patrón delegado (pasando los datos de vuelta)

Para pasar los datos del controlador de vista actual al controlador de vista anterior, puede usar el patrón delegado.



En este ejemplo, se supone que ha realizado un segmento en el Creador de interfaces y que configuró el identificador de `showSecondViewController` en `showSecondViewController`. Las salidas y acciones también deben estar conectadas a los nombres en el siguiente código.

Primer controlador de vista

El código del First View Controller es

Rápido

```
class FirstViewController: UIViewController, DataEnteredDelegate {

    @IBOutlet weak var label: UILabel!

    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        if segue.identifier == "showSecondViewController", let secondViewController =
        segue.destinationViewController as? SecondViewController {
            secondViewController.delegate = self
        }
    }
}
```

```

    }
}

// required method of our custom DataEnteredDelegate protocol
func userDidEnterInformation(info: String) {
    label.text = info
    navigationController?.popViewControllerAnimated(true)
}
}

```

C objetivo

```

@interface FirstViewController : UIViewController <DataEnteredDelegate>
@property (weak, nonatomic) IBOutlet UILabel *label;
@end

@implementation FirstViewController
- (void)viewDidLoad {
    [super viewDidLoad];
}
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    SecondViewController *secondViewController = segue.destinationViewController;
    secondViewController.delegate = self;
}
-(void)userDidEnterInformation:(NSString *)info {
    _label.text = info
    [self.navigationController popViewControllerAnimated:YES];
}
@end

```

Tenga en cuenta el uso de nuestro protocolo personalizado `DataEnteredDelegate`.

Segundo controlador y protocolo de vista

El código para el segundo controlador de vista es

Rápido

```

// protocol used for sending data back
protocol DataEnteredDelegate: class {
    func userDidEnterInformation(info: String)
}

class SecondViewController: UIViewController {

    // making this a weak variable so that it won't create a strong reference cycle
    weak var delegate: DataEnteredDelegate?

    @IBOutlet weak var textField: UITextField!

    @IBAction func sendTextBackButton(sender: AnyObject) {

        // call this method on whichever class implements our delegate protocol (the first
        view controller)
    }
}

```



```
        delegate?.userDidEnterInformation(textField.text ?? "")
    }
}
```

C objetivo

```
@protocol DataEnteredDelegate <NSObject>
- (void)userDidEnterInformation:(NSString *)info;
@end

@interface SecondViewController : UIViewController
@property (nonatomic) id <DataEnteredDelegate> delegate;
@property (weak, nonatomic) IBOutlet UITextField *textField;
@end

@implementation SecondViewController
- (void)viewDidLoad {
    [super viewDidLoad];
}

- (IBAction) sendTextBackButton:(id)sender{
    [_delegate userDidEnterInformation:textField.text];
}
@end
```

Tenga en cuenta que el `protocol` está fuera de la clase View Controller.

Pasando los datos hacia atrás usando desenrollar para segue

A diferencia de Segue, que le permite pasar datos "hacia adelante" desde el controlador de vista actual al controlador de vista de destino:

(VC1) -> (VC2)

Usando "desenrollar" puede hacer lo contrario, pasar los datos desde el controlador de vista actual o de destino a su controlador de vista actual:

(VC1) <- (VC2)

NOTA : Preste atención a que el uso del desenrollado le permite pasar los datos primero y luego el controlador de vista actual (VC2) se desasignará.

Aquí está cómo hacerlo:

Primero, deberá agregar la siguiente declaración en el controlador de vista de presentación (VC1), que es el controlador de vista al que queremos pasar los datos:

```
@IBAction func unwindToPresentingViewController(segue:UIStoryboardSegue)
```

Lo importante es usar el `unwind` prefijo, esto "informa" a Xcode de que este es un método de desenrollado que le da la opción de usarlo también en el guión gráfico.

Luego, deberá implementar el método, se ve casi igual que un segmento real:

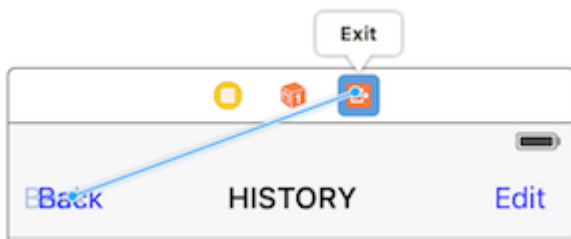
```
@IBAction func unwindToPresentingViewController(segue:UIStoryboardSegue)
{
    if segue.identifier == "YourCustomIdentifer"
    {
        if let VC2 = segue.sourceViewController as? VC2
        {
            // Your custom code in here to access VC2 class member
        }
    }
}
```

Ahora tienes 2 opciones para invocar las llamadas de desvinculación:

1. Usted puede invocar el "código duro":

`self.performSegueWithIdentifier("YourCustomIdentifer", sender: self)` que hará el desenlace por usted cada vez que `performSegueWithIdentifier`.

2. Puede vincular el método de desenrollado utilizando el `storyboard` a su objeto "Salir": presione `Ctrl` + arrastre el botón que desea invocar al método de desenrollado, al objeto "Salir":



Suelte y tendrá la opción de elegir su método de desenrollado personalizado:



Pasar datos utilizando cierres (devolver datos)

En lugar de usar el **patrón delegado**, que divide la implementación en varias partes de la clase `UIViewController`, incluso puede usar `closures` para pasar los datos hacia adelante y hacia atrás. Suponiendo que está utilizando `UIStoryboardSegue`, en el método `prepareForSegue` puede configurar fácilmente el nuevo controlador en un solo paso

```
final class DestinationViewController: UIViewController {
    var onComplete: ((success: Bool) -> ())?

    @IBAction func someButtonTapped(sender: AnyObject?) {
        onComplete?(success: true)
    }
}
```

```

final class MyViewController: UIViewController {
    override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {

        guard let destinationController = segue.destinationViewController as?
        DestinationViewController else { return }

        destinationController.onCompletion = { success in
            // this will be executed when `someButtonTapped(_)` will be called
            print(success)
        }
    }
}

```

Este es un ejemplo de uso y es mejor usarlo en Swift, la sintaxis del bloque Objective-C no es tan fácil hacer que el código sea más legible

Usando el cierre de devolución de llamada (bloque) pasando los datos de vuelta

Este tema es un problema clásico en el desarrollo de iOS, y su solución es variada, como lo muestra otro ejemplo. En este ejemplo, mostraré otro uso común diario: pasar datos usando el `closure` adaptando el ejemplo de `delegate pattern` en esta página al `closure` devolución de llamada.

Una cosa es que este método es superior al `delegate pattern` lugar de dividir el código de configuración en dos lugares diferentes (mire el ejemplo de delegado en esta página, `prepareForSegue` , `userDidEnterInformation`) en lugar de reunirlos (solo en `prepareForSegue` , lo mostraré)

Comenzar desde Second View Controller

debemos averiguar cómo usar la devolución de llamada, luego podemos escribirla, es por eso que comenzamos desde el segundo controlador de vista, ya que es donde usamos la devolución de llamada: cuando recibimos la nueva entrada de texto, llamamos a nuestra devolución de llamada, **utilizando el parámetro de devolución de llamada** como medio Para pasar los datos de vuelta al primer `ViewController`, note que dije que usando el parámetro de devolución de llamada, esto es muy importante, los principiantes (como yo) siempre pasan esto por alto y no saben por dónde empezar a escribir el cierre de la devolución de llamada correctamente.

así que en este caso, sabemos que nuestra devolución de llamada solo toma un parámetro: texto y su tipo es `String` , `String` y hagámoslo propiedad ya que necesitamos poblar desde nuestro primer controlador de vista

Solo comento toda la parte del `delegate` y la guardo para comparar.

```

class SecondViewController: UIViewController {

    //weak var delegate: DataEnteredDelegate? = nil
    var callback: ((String?)->())?

    @IBOutlet weak var textField: UITextField!

```

```

@IBAction func sendTextBackButton(sender: AnyObject) {

    //delegate?.userDidEnterInformation(textField.text!)
    callback?(input.text)

    self.navigationController?.popViewControllerAnimated(true)
}
}

```

Termina la primera vista del controlador

todo lo que tiene que hacer es pasar el cierre de devolución de llamada, y hemos terminado, el cierre hará el trabajo futuro por nosotros, ya que ya lo configuramos en el segundo controlador de vista

Mira cómo hace que nuestro código sea más corto en comparación con el `delegate pattern`

```

//no more DataEnteredDelegate
class FirstViewController: UIViewController {

    @IBOutlet weak var label: UILabel!

    override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
        if segue.identifier == "showSecondViewController" {
            let secondViewController = segue.destinationViewController as!
SecondViewController
            //secondViewController.delegate = self
            secondViewController.callback = { text in self.label.text = text }
        }
    }

    // required method of our custom DataEnteredDelegate protocol
    //func userDidEnterInformation(info: String) {
    //    label.text = info
    //}
}

```

y en el último, tal vez alguno de ustedes se confundirá al ver que solo pasamos los datos (cierre en este caso) solo de una manera, desde el primer controlador de vista al segundo, sin regresar directamente del segundo controlador de vista, ¿cómo podemos? ¿Considerarlo como una herramienta de comunicación? tal vez realmente debería ejecutarlo y probarlo usted mismo, todo lo que diré es el **parámetro**, ¡es el **parámetro de cierre de devolución de llamada** que pasa los datos de vuelta!

Asignando propiedad (Pasando datos adelante)

Puede pasar datos directamente asignando la propiedad del siguiente controlador de vista antes de presionarlo o presentarlo.

```

class FirstViewController: UIViewController {

    func openSecondViewController() {

```

```
// Here we initialize SecondViewController and set the id property to 492
let secondViewController = SecondViewController()
secondViewController.id = 492

// Once it was assign we now push or present the view controller
present(secondViewController, animated: true, completion: nil)
}

}

class SecondViewController: UIViewController {

    var id: Int?

    override func viewDidLoad() {
        super.viewDidLoad()

        // Here we unwrapped the id and will get the data from the previous view controller.
        if let id = id {
            print("Id was set: \(id)")
        }
    }
}
```

Lea Pasando datos entre los controladores de vista en línea:

<https://riptutorial.com/es/ios/topic/434/pasando-datos-entre-los-controladores-de-vista>

Capítulo 129: Pasar datos entre los controladores de vista (con MessageBox-Concept)

Introducción

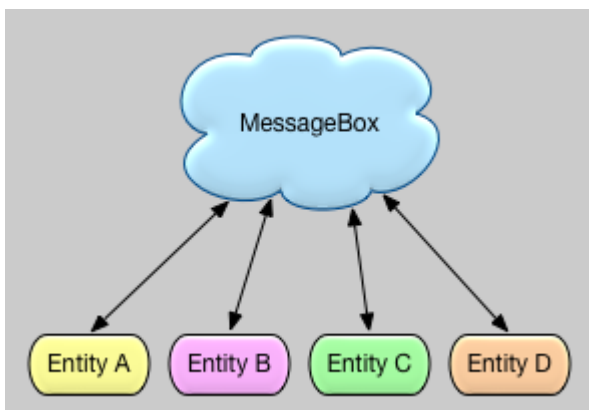
MessageBox es un concepto simple para las entidades de desacoplamiento.

Por ejemplo, la entidad A puede colocar un mensaje que la entidad B puede leer cuando sea apropiado.

Un controlador de vista desea hablar con otro controlador de vista, pero no desea crear una relación fuerte o débil.

Examples

Ejemplo de uso simple



```
let messageBox:MessageBox = MessageBox ()

// set
messageBox.setObject ("TestObject1", forKey:"TestKey1")

// get
// but don't remove it, keep it stored, so that it can still be retrieved later
let someObject:String = messageBox.getObject (forKey:"TestKey1", removeIfFound:false)

// get
// and remove it
let someObject:String = messageBox.getObject (forKey:"TestKey1", removeIfFound:true)
```

Lea [Pasar datos entre los controladores de vista \(con MessageBox-Concept\) en línea: https://riptutorial.com/es/ios/topic/9118/pasar-datos-entre-los-controladores-de-vista--con-messagebox-concept-](https://riptutorial.com/es/ios/topic/9118/pasar-datos-entre-los-controladores-de-vista--con-messagebox-concept-)

Capítulo 130: Perfil con instrumentos

Introducción

Xcode incluye una aplicación de ajuste de rendimiento llamada Instruments que puede usar para perfilar su aplicación utilizando todo tipo de métricas diferentes. Tienen herramientas para inspeccionar el uso de la CPU, el uso de la memoria, las fugas, la actividad de archivos / redes y el uso de la energía, solo por nombrar algunos. Es muy fácil comenzar a perfilar su aplicación desde Xcode, pero a veces no es tan fácil entender lo que ve cuando se perfila, lo que disuade a algunos desarrolladores de poder usar esta herramienta en todo su potencial.

Examples

Perfilador de tiempo

El primer instrumento que verás es el `Time Profiler`. A intervalos medidos, los instrumentos detendrán la ejecución del programa y tomarán un seguimiento de la pila en cada subproceso en ejecución. Piense en ello como presionar el botón de pausa en el depurador de Xcode. Aquí hay una vista previa del Time Profiler:



Running Time	Self	Symbol Name
5838.0ms 46.9%	0.0	▼Main Thread 0xa2db0
5234.0ms 42.0%	0.0	▶ ext.InstrumentsTutorial.ObjectiveC.CIContext
315.0ms 2.5%	0.0	▶ top_level_code InstrumentsTutorial
115.0ms 0.9%	0.0	▶ <Unknown Address>
63.0ms 0.5%	0.0	▶ InstrumentsTutorial.FlickrPhoto
15.0ms 0.1%	0.0	▶ @!objc ext.UIKit.ObjectiveC.CIContext
15.0ms 0.1%	0.0	▶ InstrumentsTutorial.ViewContro
12.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
10.0ms 0.0%	0.0	▶ @!objc ObjectiveC.UImage.init
10.0ms 0.0%	0.0	▶ @!objc ObjectiveC.CIContext.____
8.0ms 0.0%	0.0	▶ InstrumentsTutorial.FlickrPhoto
8.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
3.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
3.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
3.0ms 0.0%	0.0	▶ InstrumentsTutorial.ViewContro
3.0ms 0.0%	0.0	▶ InstrumentsTutorial.ViewContro
2.0ms 0.0%	0.0	▶ InstrumentsTutorial.Flickr.searc
2.0ms 0.0%	0.0	▶ InstrumentsTutorial.FlickrPhoto
1.0ms 0.0%	0.0	▶ swift_getEnumCaseSinglePaylo
1.0ms 0.0%	1.0	▶ Swift.HeapBufferStorage.__dea
1.0ms 0.0%	0.0	▶ swift_getExistentialTypeMetada
1.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
1.0ms 0.0%	1.0	▶ _swift_retain_(swift::HeapObjec
1.0ms 0.0%	1.0	▶ swift_unknownRelease libswif
1.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
1.0ms 0.0%	0.0	▶ swift_getGenericClassObjCNan
1.0ms 0.0%	1.0	▶ _swift_release_(swift::HeapObjec
1.0ms 0.0%	0.0	▶ InstrumentsTutorial.ViewContro

muestra la cantidad de tiempo empleado en la ejecución de varios métodos dentro de una aplicación. Cada fila es un método diferente que ha seguido la ruta de ejecución del programa. El tiempo empleado en cada método se puede determinar a partir del número de veces que se detiene el generador de perfiles en cada método. Por ejemplo, si **100** muestras se realizan a **intervalos de 1 milisegundo** y se encuentra que un método en particular está en la parte superior de la pila en 10 muestras, entonces puede deducir que se gastó aproximadamente el **10%** del tiempo total de ejecución (**10 milisegundos**) en ese método. Es una aproximación bastante cruda, ¡pero funciona!

Desde Xcode's barra de menú Xcode's , seleccione Product\Profile o press `⌘I` . Esto construirá la aplicación y lanzará Instruments. Serás recibido con una ventana de selección que se ve así:

Choose a profiling template for: iPhone 6 (8.2 Simu

Standard

Custom

Recent



Leaks



Multicore



Network



System Usage



Time Profiler



UI Recorder



Time Profiler

Performs low-overhead time-based sampling of proces

Estas son todas las diferentes plantillas que vienen con los instrumentos.

Seleccione el instrumento `Time Profiler` y haga clic en Elegir. Esto abrirá un nuevo documento de instrumentos. Haga clic en el **botón** rojo de **grabación** en la parte superior izquierda para comenzar a grabar e iniciar la aplicación. Es posible que se le solicite su contraseña para autorizar a **Instruments** a analizar otros procesos. No dude, es seguro proporcionarlos aquí. En

la **ventana de Instrumentos** , puede ver el tiempo contando y una pequeña flecha moviéndose de izquierda a derecha sobre el **gráfico** en el centro de la pantalla. Esto indica que la aplicación se está ejecutando.

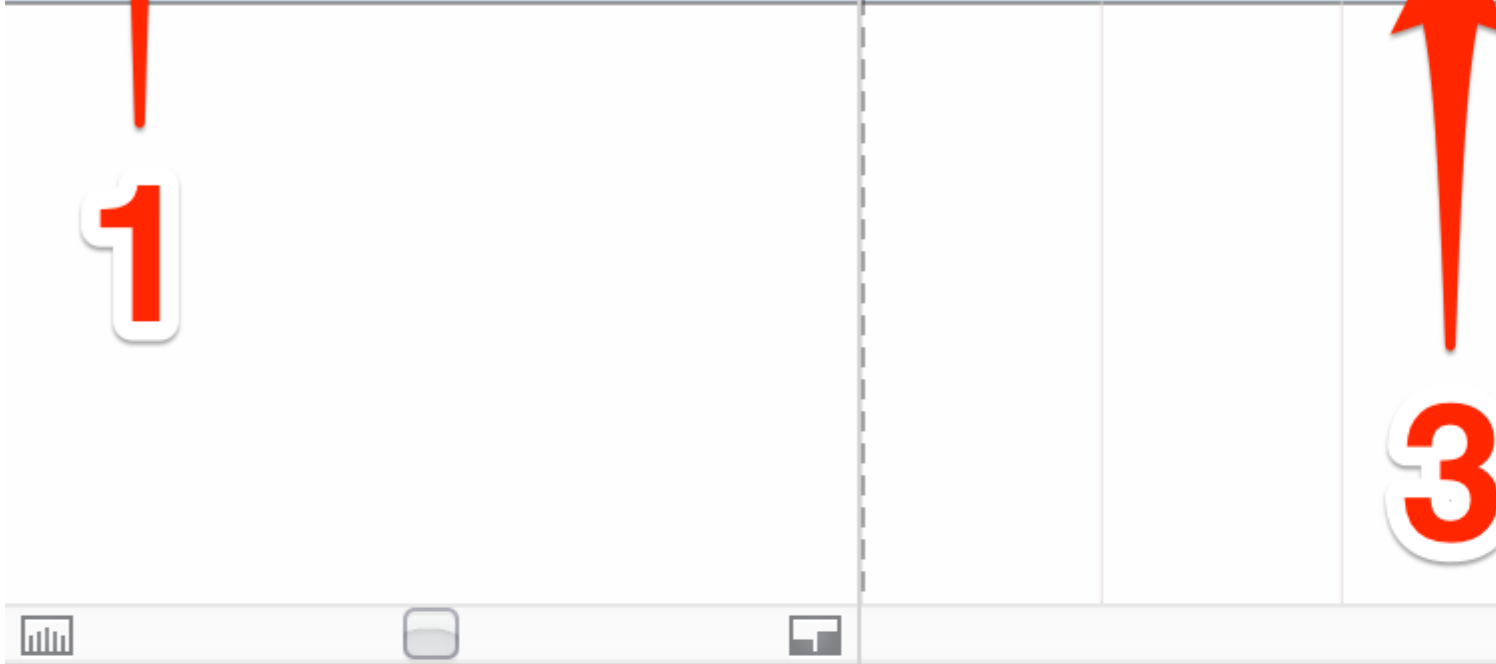
Ahora, empieza a usar la aplicación. Busque algunas imágenes y profundice en uno o más de los resultados de búsqueda. Probablemente haya notado que entrar en un resultado de búsqueda es tediosamente lento, y desplazarse por una lista de resultados de búsqueda también es increíblemente molesto: ¡es una aplicación terriblemente torpe!

Bueno, estás de suerte, ¡porque estás a punto de embarcarte en arreglarlo! Sin embargo, primero vas a ver rápidamente lo que estás viendo en **Instruments** . Primero, asegúrese de que el selector de vista en el lado derecho de la barra de herramientas tenga ambas opciones seleccionadas, así:



Eso asegurará que todos los paneles estén abiertos. Ahora estudie la captura de pantalla a continuación y la explicación de cada sección debajo de ella:

Time Profiler

1

3

Running Time	Self	Symbol Name
4500.0ms 48.6%	0.0	▼ Main Thread 0xa4f45
3903.0ms 42.1%	0.0	▶ ext.InstrumentsTutorial.ObjectiveC.CIContext.
387.0ms 4.1%	0.0	▶ top_level_code InstrumentsTutorial
76.0ms 0.8%	0.0	▶ <Unknown Address>
39.0ms 0.4%	0.0	▶ InstrumentsTutorial.FlickrPhoto
16.0ms 0.1%	0.0	▶ @!objc ext.UIKit.ObjectiveC.CIContext.
10.0ms 0.1%	0.0	▶ InstrumentsTutorial.ViewContro
7.0ms 0.0%	0.0	▶ @!objc ObjectiveC.CIContext.
6.0ms 0.0%	0.0	▶ @!objc ObjectiveC.UILImage.init
5.0ms 0.0%	0.0	▶ InstrumentsTutorial.SearchResu
3.0ms 0.0%	0.0	▶ InstrumentsTutorial SearchRes

. El botón rojo 'grabar' detendrá e iniciará la aplicación que actualmente se está perfilando cuando se hace clic (se alterna entre un icono de grabación y de detención). El botón de pausa hace exactamente lo que cabría esperar y detiene la ejecución actual de la aplicación.

2. Este es el temporizador de ejecución. El temporizador cuenta cuánto tiempo se ha estado ejecutando la aplicación que se está perfilando y cuántas veces se ha ejecutado. Si detiene y luego reinicia la aplicación usando los controles de grabación, se iniciará una nueva ejecución y la pantalla mostrará la Corrida 2 de 2.

3. Esto se llama una pista. En el caso de la plantilla de Time Profiler que seleccionó, solo hay un instrumento, así que solo hay una pista. Aprenderá más sobre los detalles del gráfico que se muestra aquí más adelante en el tutorial.

4. Este es el panel de detalle. Muestra la información principal sobre el instrumento en particular que está utilizando. En este caso, muestra los métodos que son "los más calientes", es decir, los que han consumido más tiempo de CPU. Si hace clic en la barra en la parte superior que dice Árbol de llamadas (la izquierda) y selecciona Lista de muestras, se le presenta una vista diferente de los datos. Esta vista muestra cada una de las muestras. Haga clic en algunas muestras y verá que la traza de la pila capturada aparece en el inspector de detalles extendidos.

5. Este es el panel de inspectores. Hay tres inspectores: Configuración de registro, Configuración de pantalla y Detalle extendido. En breve aprenderás más sobre algunas de estas opciones.

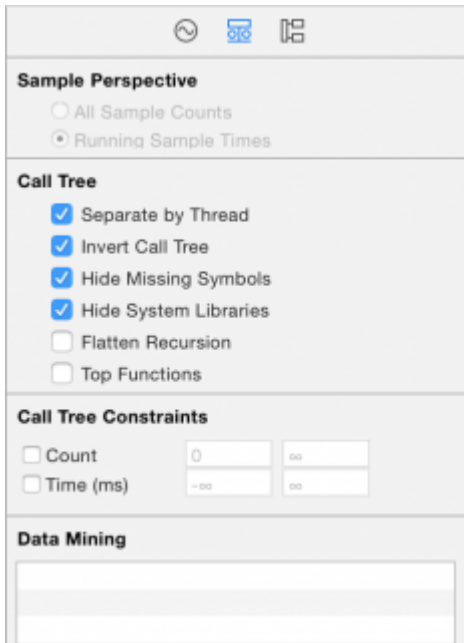
Perforación profunda

Realice una búsqueda de imágenes y profundice en los resultados. Personalmente, me gusta buscar "perro", pero elige lo que desees, ¡puedes ser una de esas personas gato!

Ahora, desplácese hacia arriba y hacia abajo en la lista unas cuantas veces para obtener una buena cantidad de datos en el `Time Profiler`. Debes notar que los números en el centro de la pantalla cambian y el **gráfico se completa**; Esto le indica que se están utilizando **ciclos de CPU**.

¡Realmente no esperarías que una UI fuera tan torpe como esta, ya que ninguna `table view` está lista para enviarse hasta que se desplace como la mantequilla! Para ayudar a identificar el problema, necesita establecer algunas opciones.

En el lado derecho, seleccione el **inspector de configuración de pantalla** (or press `⌘+2`). En el **inspector**, en la sección `Call Tree`, seleccione **Separar por hilo**, **Invertir Call Tree**, **Ocultar símbolos que faltan** y **Ocultar bibliotecas del sistema**. Se verá así:



Esto es lo que hace cada opción con los datos que se muestran en la tabla de la izquierda:

Separar por hilo: cada hilo debe considerarse por separado. Esto le permite comprender qué subprocesos son responsables de la mayor cantidad de uso de la **CPU** .

Invertir árbol de llamadas: con esta opción, el `stack trace` la `stack trace` se considera de arriba a abajo. Esto suele ser lo que desea, ya que desea ver los métodos más profundos en los que la **CPU** está gastando su tiempo.

Ocultar símbolos que faltan: Si no se puede encontrar el archivo `dsym` para su aplicación o un `system framework` , en lugar de ver los nombres de los métodos (símbolos) en la tabla, verá valores hexadecimales correspondientes a las direcciones dentro del binario. Si se selecciona esta opción, solo se muestran los símbolos totalmente resueltos y los valores **hexadecimales** sin resolver se ocultan. Esto ayuda a despejar los datos presentados.

Ocultar bibliotecas del sistema: cuando se selecciona esta opción, solo se muestran los símbolos de su propia aplicación. A menudo es útil seleccionar esta opción, ya que generalmente solo le importa dónde la **CPU** está gastando tiempo en su propio código. ¡No puede hacer mucho sobre la cantidad de **CPU** que utilizan las `system libraries` del `system libraries` !

Aplanar la recursión: esta opción trata las funciones **recursivas** (las que se llaman a sí mismas) como una entrada en cada `stack trace` , en lugar de múltiples.

Funciones principales: habilitar esto hace que los `Instruments` consideren el tiempo total empleado en una función como la suma del tiempo directamente dentro de esa función, así como el tiempo empleado en las funciones llamadas por esa función.

Entonces, si la función A llama a B, entonces el tiempo de A se informa como el tiempo empleado en A **MÁS** que el tiempo empleado en B. Esto puede ser realmente útil, ya que te permite elegir la cifra de tiempo más grande cada vez que descienes en la pila de llamadas, poniendo a cero. en sus métodos más lentos.

Si está ejecutando una aplicación Objective-C , también hay una opción de Mostrar solo **Obj-C** : si se selecciona esta opción, solo se muestran los métodos Objective-C , en lugar de las funciones C o C++ . No hay ninguno en su programa, pero si estaba buscando una aplicación OpenGL , podría tener algún C++ , por ejemplo.

Aunque algunos valores pueden ser ligeramente diferentes, el orden de las entradas debe ser similar al de la siguiente tabla una vez que haya habilitado las opciones anteriores:

Running Time	Self	Symbol Name
12682.0ms	48.0%	0.0
11858.0ms	44.8%	11858.0 ▶ ext.InstrumentsTutorial.ObjectiveC.UImage.applyTonalFilter (ObjectiveC.UImage) -> ObjectiveC.UImage?
428.0ms	1.6%	0.0 ▶ top_level_code InstrumentsTutorial
186.0ms	0.7%	0.0 ▶ <Unknown Address>
120.0ms	0.4%	0.0 ▶ InstrumentsTutorial.FlickrPhoto.isFavourite.getter : Swift.Bool InstrumentsTutorial
23.0ms	0.0%	0.0 ▶ @objc ext.UIKit.ObjectiveC.CIImage.init (ObjectiveC.CIImage.Type)(image : ObjectiveC.UImage) -> Objectiv
12.0ms	0.0%	0.0 ▶ @objc ObjectiveC.CIContext.__allocating_init (ObjectiveC.CIContext.Type)(options : [ObjectiveC.NSObject : :
9.0ms	0.0%	0.0 ▶ InstrumentsTutorial.SearchResultsController.collectionView (InstrumentsTutorial.SearchResultsController :
7.0ms	0.0%	0.0 ▶ InstrumentsTutorial.ViewController.init (InstrumentsTutorial.ViewController.Type)(coder : ObjectiveC.NSCoder)
5.0ms	0.0%	0.0 ▶ InstrumentsTutorial.SearchResultsControllerCollectionViewCell.init (InstrumentsTutorial.SearchResultsCollectionVi
4.0ms	0.0%	0.0 ▶ @objc ObjectiveC.UImage.init (ObjectiveC.UImage.Type)(data : ObjectiveC.NSData) -> ObjectiveC.UImage'
4.0ms	0.0%	0.0 ▶ InstrumentsTutorial.SearchResultsControllerCollectionViewCell.prepareForReuse (InstrumentsTutorial.SearchResultsCo
4.0ms	0.0%	0.0 ▶ InstrumentsTutorial.SearchResultsController.collectionView (InstrumentsTutorial.SearchResultsController :
2.0ms	0.0%	0.0 ▶ InstrumentsTutorial.FlickrPhoto.loadImageFromURL (InstrumentsTutorial.FlickrPhoto)(URL : ObjectiveC.NSUF
2.0ms	0.0%	0.0 ▶ swift_getGenericMetadata libswiftCore.dylib
2.0ms	0.0%	0.0 ▶ @objc ObjectiveC.CIFilter.__allocating_init (ObjectiveC.CIFilter.Type)(name : Swift.String) -> ObjectiveC.CIFil
2.0ms	0.0%	0.0 ▶ InstrumentsTutorial.SearchResultsControllerCollectionViewCell.prepareForReuse (InstrumentsTutorial.SearchResultsCo
2.0ms	0.0%	0.0 ▶ InstrumentsTutorial.ViewController.tableView (InstrumentsTutorial.ViewController)(ObjectiveC.UITableView, cel
1.0ms	0.0%	1.0 ▶ Swift.Optional.init <A>(A?.Type)(nilLiteral : ()) -> A? libswiftCore.dylib
1.0ms	0.0%	0.0 ▶ InstrumentsTutorial.ViewController(searchBarSearchButtonClicked (InstrumentsTutorial.ViewController) -> (OI
1.0ms	0.0%	1.0 ▶ llvm::hashing::detail::hash_short(char const', unsigned long, unsigned long long) libswiftCore.dylib
1.0ms	0.0%	1.0 ▶ @objc Swift_NSContiguousString.copy (Swift_NSContiguousString) -> Swift.AnyObject libswiftCore.dylib
1.0ms	0.0%	0.0 ▶ InstrumentsTutorial.Flickr.searchFlickrForTerm (InstrumentsTutorial.Flickr)(Swift.String, completion : {results :

Bueno, eso ciertamente no se ve muy bien. La gran mayoría del tiempo se invierte en el método que aplica el filtro 'tonal' a las fotos en miniatura. Eso no debería ser una gran sorpresa para usted, ya que la carga y el desplazamiento de la tabla eran las partes más clásicas de la interfaz de usuario, y es cuando las celdas de la tabla se actualizan constantemente.

Para obtener más información sobre lo que sucede dentro de ese método, haga doble clic en su fila en la tabla. Al hacerlo aparecerá la siguiente vista:

```

15
16 class var sharedCache: ImageCache {
17     return _sharedCache
18 }
19
20 func setImage(image: UIImage, forKey key: String) {
21     images[key] = image
22 }
23
24 func imageForKey(key: String) -> UIImage? {
25     return images[key]
26 }
27
28
29 extension UIImage {
30     func applyTonalFilter() -> UIImage? {
31         let context = CIContext(options:nil)
32         let filter = CIFilter(name:"CIPhotoEffectTonal")
33         let input = CoreImage.CIImage(image: self)
34         filter.setValue(input, forKey: kCIInputImageKey)
35         let outputImage = filter.outputImage
36
37         let outImage = context.createCGImage(outputImage, fromRect: outputImage.extent())
38         let returnImage = UIImage(CGImage: outImage)
39         return returnImage
40     }
41 }
42

```

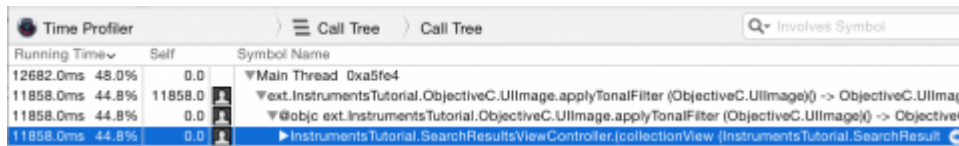
Bueno, eso es interesante, ¿no? applyTonalFilter() es un método agregado a UIImage en una extensión, y casi el 100 % del tiempo invertido se dedica a crear la salida CGImage después de aplicar el filtro de imagen.

Realmente no hay mucho que se pueda hacer para acelerar esto: crear la imagen es un proceso bastante intensivo, y toma todo el tiempo necesario. Intentemos retroceder y ver desde dónde se

llama a `applyTonalFilter()` . Haga clic en `Call Tree` en la ruta de navegación en la parte superior de la vista de código para volver a la pantalla anterior:



Ahora haga clic en la flecha pequeña a la izquierda de la fila de `applyTonalFilter` en la parte superior de la tabla. Esto desplegará el Árbol de llamadas para mostrar al llamante de `applyTonalFilter`. Puede que necesites desplegar la siguiente fila también; al perfilar Swift, a veces habrá filas duplicadas en el árbol de llamadas, con el prefijo `@objc`. Está interesado en la primera fila que tiene como prefijo el nombre de destino de su aplicación (`InstrumentsTutorial`):

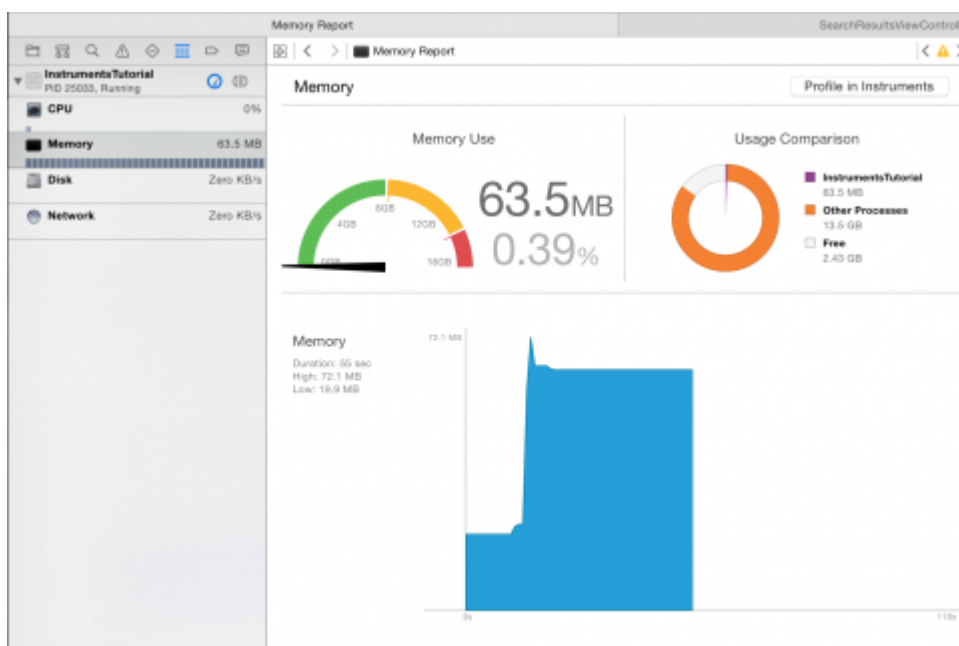


En este caso, esta fila se refiere a `cellForItemAtIndexPath` la vista de colección de `cellForItemAtIndexPath` . Haga doble clic en la fila para ver el código asociado del proyecto.

Ahora puedes ver cual es el problema. El método para aplicar el filtro tonal tarda mucho tiempo en ejecutarse, y se llama directamente desde `cellForItemAtIndexPath`, que bloqueará el `main thread` (y, por lo tanto, la IU completa) cada vez que solicite una imagen filtrada.

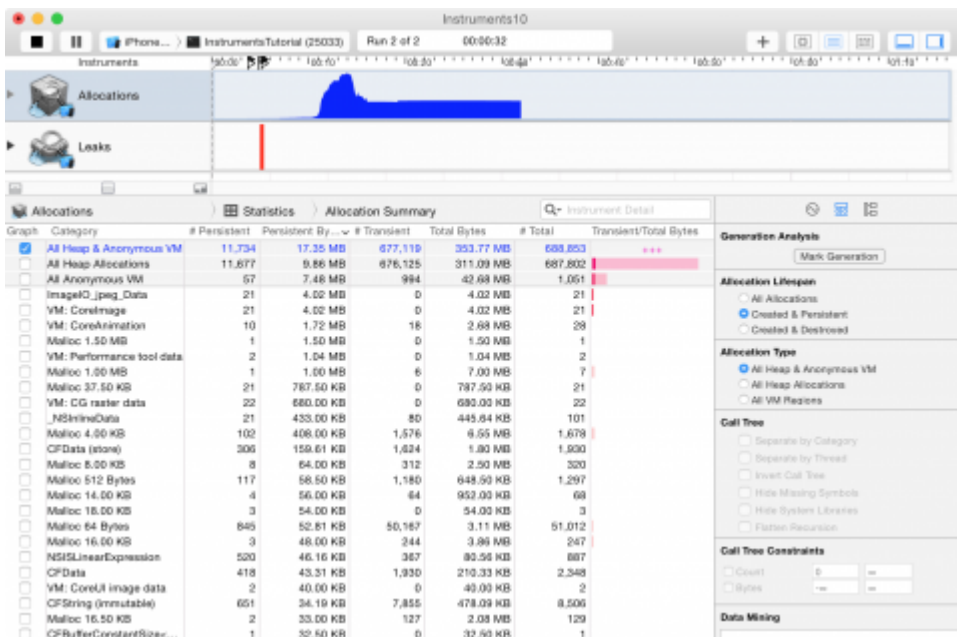
Asignaciones

Hay información detallada sobre todos los **objetos** que se están creando y la memoria que los respalda; También muestra que `retain counts` de cada objeto. Para comenzar de nuevo con un nuevo `instruments profile` , salga de la aplicación Instrumentos. Esta vez, genere y ejecute la aplicación, y abra Debug Navigator en el área de Navegadores. Luego haga clic en **Memoria** para mostrar gráficos de uso de memoria en la ventana principal:



Estos gráficos son útiles para obtener una idea rápida sobre el rendimiento de su aplicación. Pero vas a necesitar un poco más de poder. Haga clic `Profile in Instruments` botón `Profile in Instruments` y luego en `Transferir` para llevar esta sesión a **Instrumentos** . El instrumento de

asignaciones se iniciará automáticamente.



Esta vez notarás dos pistas. Uno se llama Asignaciones, y el otro se llama Fugas. La pista de Asignaciones se discutirá en detalle más adelante; La pista de Fugas es generalmente más útil en Objective-C, y no se tratará en este tutorial. Entonces, ¿qué error vas a encontrar a continuación? Hay algo oculto en el proyecto que probablemente no sabes que está ahí. Es probable que haya oído hablar de las fugas de memoria. Pero lo que quizás no sepa es que en realidad hay dos tipos de fugas:

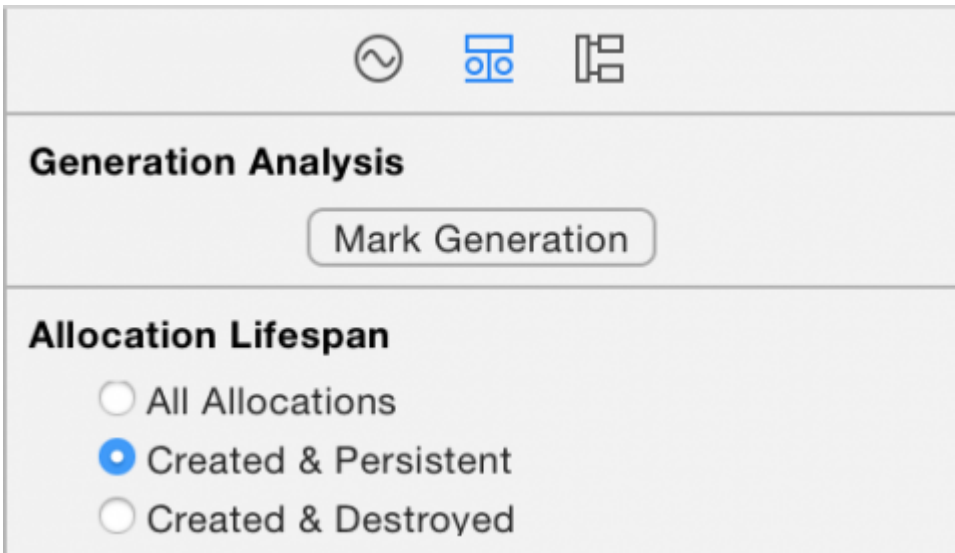
Las verdaderas fugas de memoria son aquellas en las que un objeto ya no está referenciado por nada sino que aún está asignado, lo que significa que la memoria nunca se puede reutilizar. Incluso con Swift y ARC ayudando a administrar la memoria, el tipo más común de pérdida de memoria es un `retain cycle` or `strong reference cycle`. Esto ocurre cuando dos objetos mantienen fuertes referencias entre sí, de modo que cada objeto evita que el otro sea desasignado. Esto significa que su memoria nunca se libera!

El crecimiento de la memoria sin límites es donde la memoria sigue siendo asignada y nunca se le da la oportunidad de ser **desasignada**. Si esto continúa para siempre, en algún momento se llenará `system's memory` y tendrá un gran problema de memoria en sus manos. En iOS, esto significa que la aplicación será eliminada por el sistema.

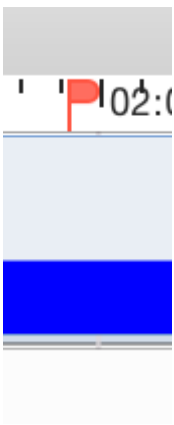
Con el **instrumento** de Asignaciones ejecutándose en la aplicación, realice cinco búsquedas diferentes en la aplicación pero aún no profundice en los resultados. ¡Asegúrate de que las búsquedas tengan algunos resultados! Ahora deja que la aplicación se asiente un poco esperando unos segundos.

Debes haber notado que el **gráfico** en la pista de Asignaciones ha aumentado. Esto te dice que la memoria está siendo asignada. Es esta característica la que lo guiará para encontrar `unbounded memory growth`.

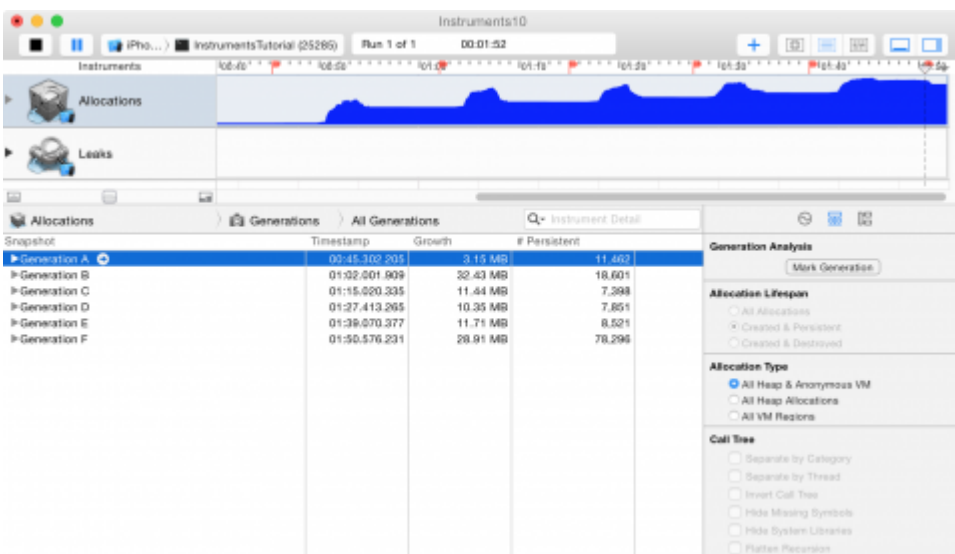
Lo que vas a realizar es un `generation analysis`. Para ello, presione el botón llamado Mark Generation. Encontrará el botón en la parte superior del inspector de configuración de pantalla:



Presiónelo y verá que aparece una bandera roja en la pista, así:



El propósito del `generation analysis` de `generation analysis` es realizar una acción varias veces y ver si la memoria está creciendo de manera `unbounded fashion`. **Profundice** en una búsqueda, espere unos segundos a que se carguen las imágenes y luego regrese a la página principal. Entonces marca la generación otra vez. Haga esto repetidamente para diferentes búsquedas. Después de **profundizar** en algunas búsquedas, los **instrumentos** se verán así:



En este punto, deberías ser sospechoso. Observe cómo va el gráfico azul con cada búsqueda en

la que **profundiza** . Bueno, eso ciertamente no es bueno. Pero espera, ¿qué pasa con las `memory warnings`? Usted sabe acerca de esos, ¿verdad? `Memory warnings` son la forma que tiene iOS de decirle a una aplicación que las cosas se están apretando en el departamento de memoria, y usted necesita borrar algo de memoria.

Es posible que este crecimiento no se deba únicamente a su aplicación; podría ser algo en las profundidades de `UIKit` que se aferra a la memoria. Dé a los marcos del sistema y a su aplicación la oportunidad de borrar su **memoria** antes de señalar con el dedo a cualquiera de ellos.

Simule una `memory warning` seleccionando `Instrument\Simulate Memory Warning` en la barra de menú de Instrumentos, o `Hardware\Simulate Memory Warning` en la barra de menú `simulator's` . Notarás que el uso de la memoria disminuye un poco, o tal vez no. Ciertamente no vuelvo a donde debería estar. Así que todavía hay **un crecimiento ilimitado de memoria** en algún lugar.

La razón para marcar una generación después de cada iteración de profundizar en una búsqueda es que puede ver qué **memoria** se ha asignado entre cada generación. Echa un vistazo en el panel de detalles y verás un montón de generaciones.

Lea Perfil con instrumentos en línea: <https://riptutorial.com/es/ios/topic/9629/perfil-con-instrumentos>

Capítulo 131: plist iOS

Introducción

Plist se utiliza para el almacenamiento de datos en la aplicación iOS. Plist guardar datos en forma de Array y Diccionarios. En plist podemos guardar datos como: 1. Datos estáticos para usar en la aplicación. 2. Datos que vendrán del servidor.

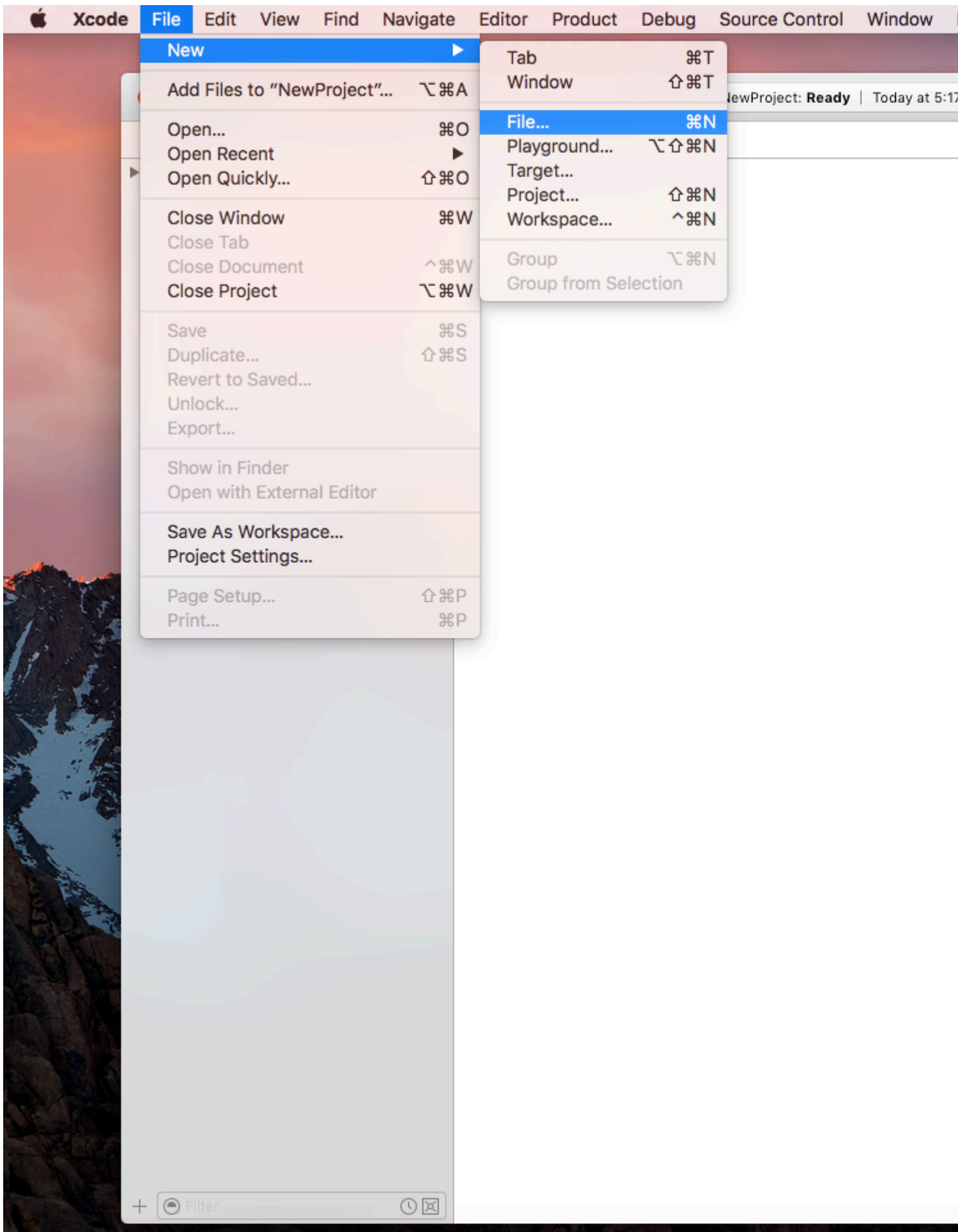
Examples

Ejemplo:

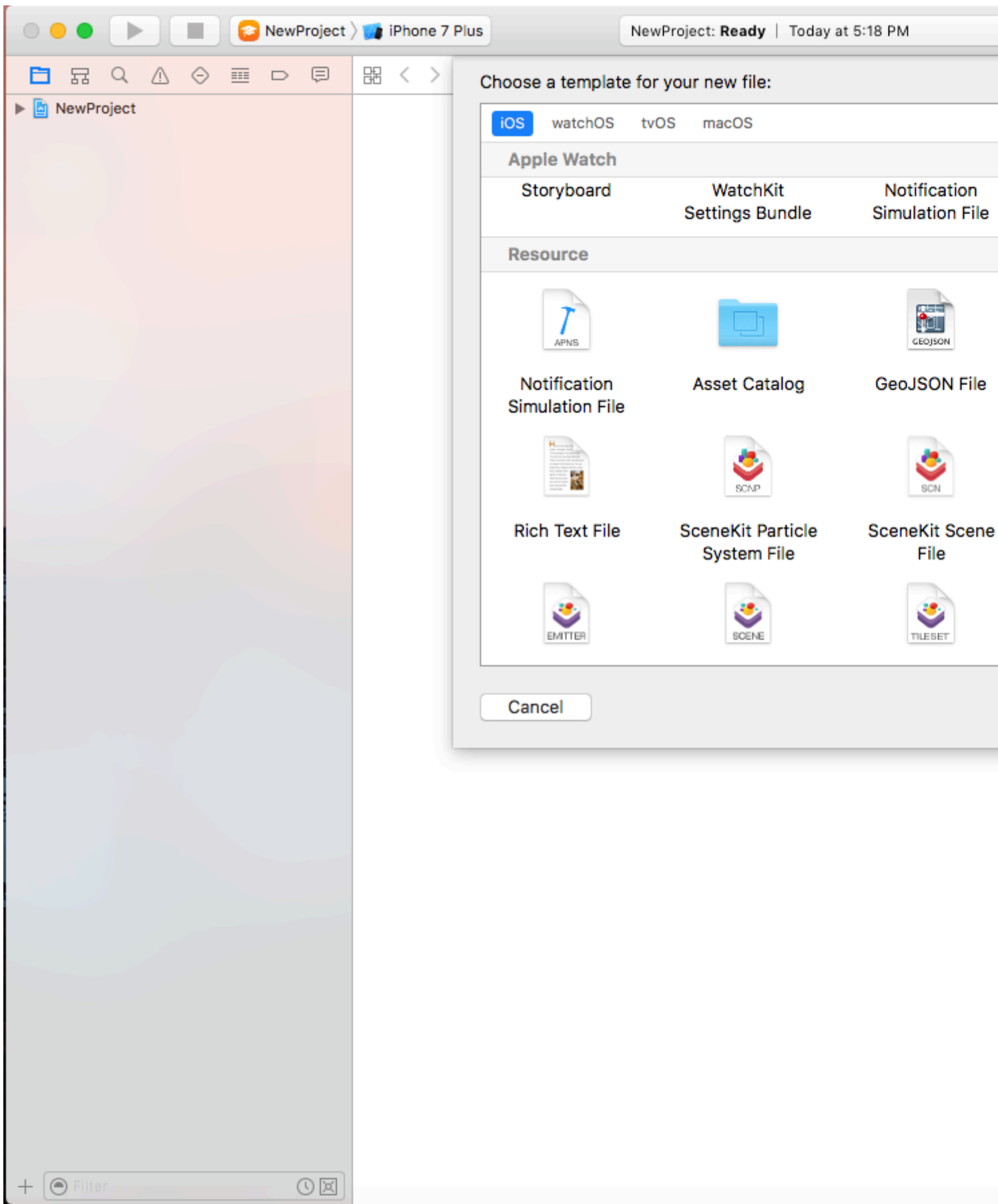
1. Datos estáticos para ser utilizados en la aplicación.

Para guardar datos estáticos en plist siga estos métodos:

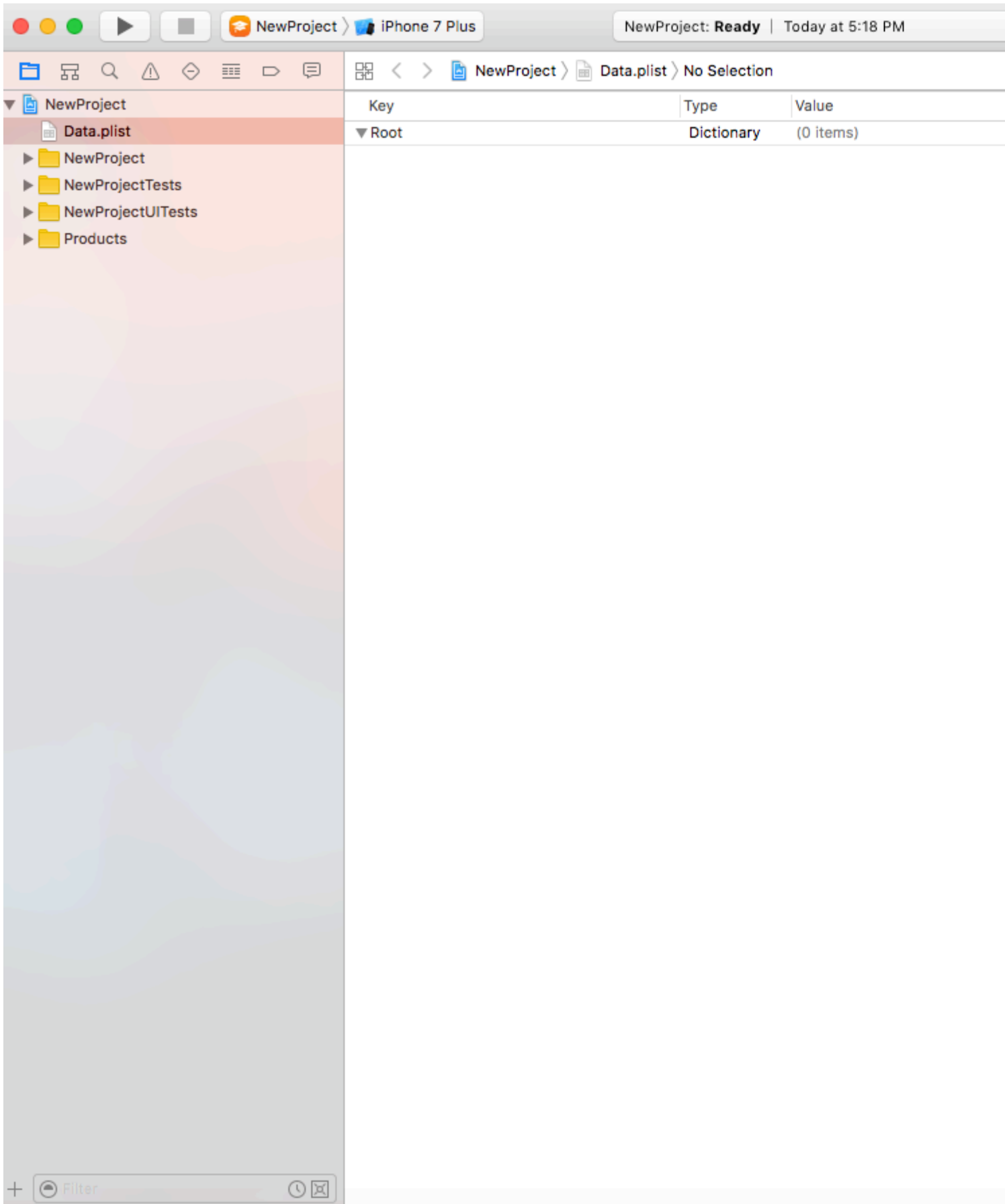
a) Agregar un nuevo archivo



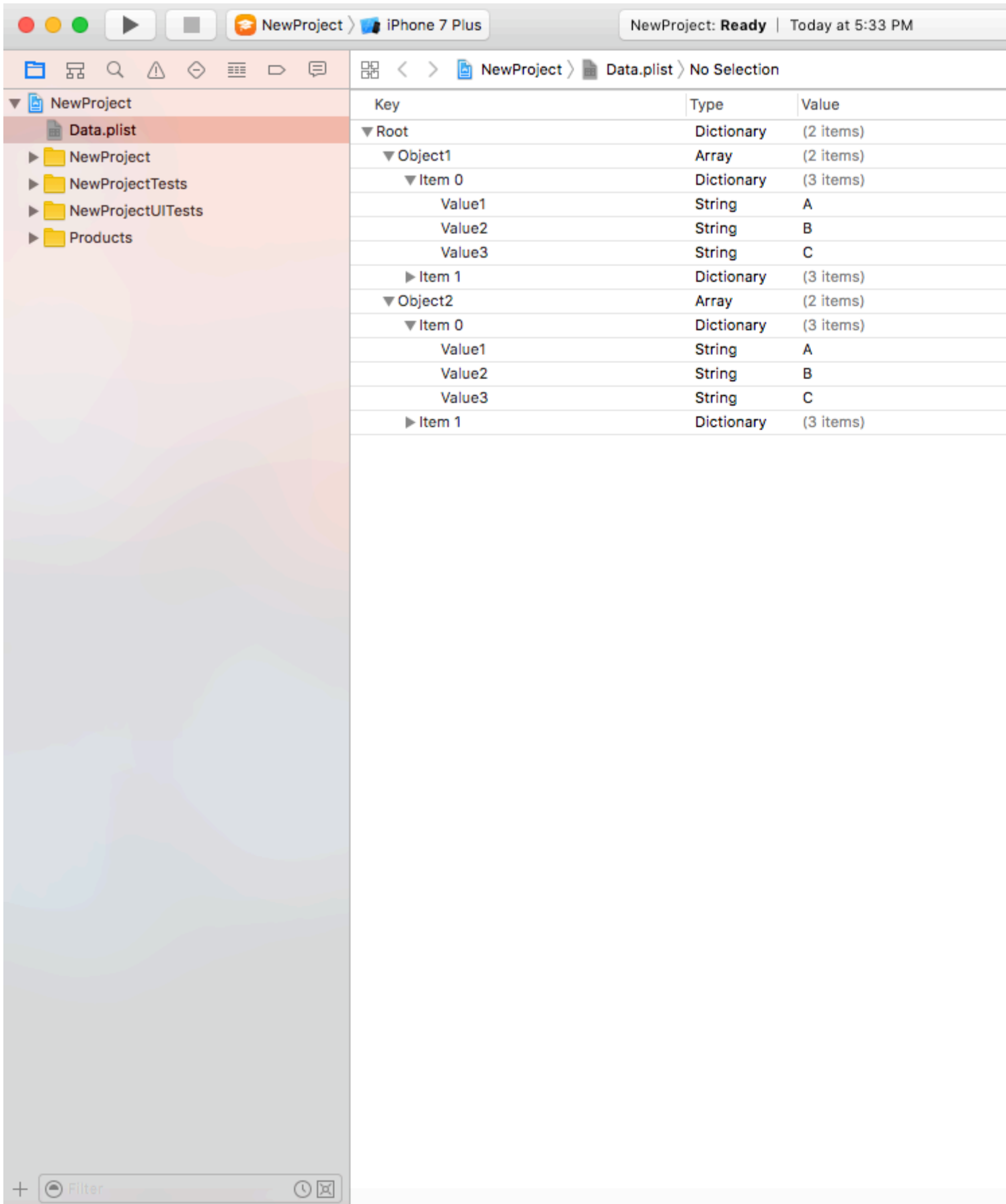
b) Haga clic en Lista de propiedades en Recursos



c) Nombre la lista de propiedades y se creará un archivo como (data.plist aquí)



d) Puedes crear una lista de arrays y diccionarios como:



// Lee plist from bundle y obtén Root Dictionary de él

```
NSDictionary *dictRoot = [NSDictionary dictionaryWithContentsOfFile:[NSBundle mainBundle]
pathForResource:@"Data" ofType:@"plist"]];
```


// Su diccionario contiene una serie de diccionarios // Ahora saque una matriz de él.

```
NSArray *arrayList = [NSArray arrayWithArray:[dictRoot objectForKey:@"Object1"]];

for(int i=0; i< [arrayList count]; i++)
{
    NSMutableDictionary *details=[arrayList objectAtIndex:i];
}
```

Guardar y editar / borrar datos de Plist

Ya has creado un plist. Esta lista seguirá siendo la misma en la aplicación. Si desea editar los datos en este plist, agregar nuevos datos en plist o eliminar datos de plist, no puede realizar cambios en este archivo.

Para este propósito, tendrá que almacenar su lista en el Directorio de documentos. Puede editar su lista guardada en el directorio de documentos.

Guardar plist en el directorio de documentos como:

```
NSString *filePath = [[NSBundle mainBundle] pathForResource:@"Data" ofType:@"plist"];

NSDictionary *dict = [[NSDictionary alloc] initWithContentsOfFile:filePath];

NSDictionary *plistDict = dict;

NSFileManager *fileManager = [NSFileManager defaultManager];

NSString *error = nil;

NSData *plistData = [NSPropertyListSerialization dataFromPropertyList:plistDict
format:NSPropertyListXMLFormat_v1_0 errorDescription:&error];

if (![fileManager fileExistsAtPath: plistPath]) {

    if(plistData)
    {
        [plistData writeToFile:plistPath atomically:YES];
    }
}
else
{
}
}
```

Recupere los datos de Plist como:

```
NSArray *paths = NSSearchPathForDirectoriesInDomains (NSDocumentDirectory,
NSUserDomainMask, YES);
NSString *documentsPath = [paths objectAtIndex:0];
NSString *plistPath = [documentsPath stringByAppendingPathComponent:@"Data.plist"];
NSDictionary *dict = [[NSDictionary alloc] initWithContentsOfFile:plistPath];

NSArray *usersArray = [dict objectForKey:@"Object1"];
```

Puede editar eliminar, agregar nuevos datos según sus requisitos y guardar el plist nuevamente en el Directorio de documentos.

Lea plist iOS en línea: <https://riptutorial.com/es/ios/topic/8141/plist-ios>

Capítulo 132: Proceso de envío de aplicaciones

Introducción

Este tutorial cubre todos los pasos necesarios para cargar una aplicación iOS en la App Store.

Examples

Configurar perfiles de aprovisionamiento

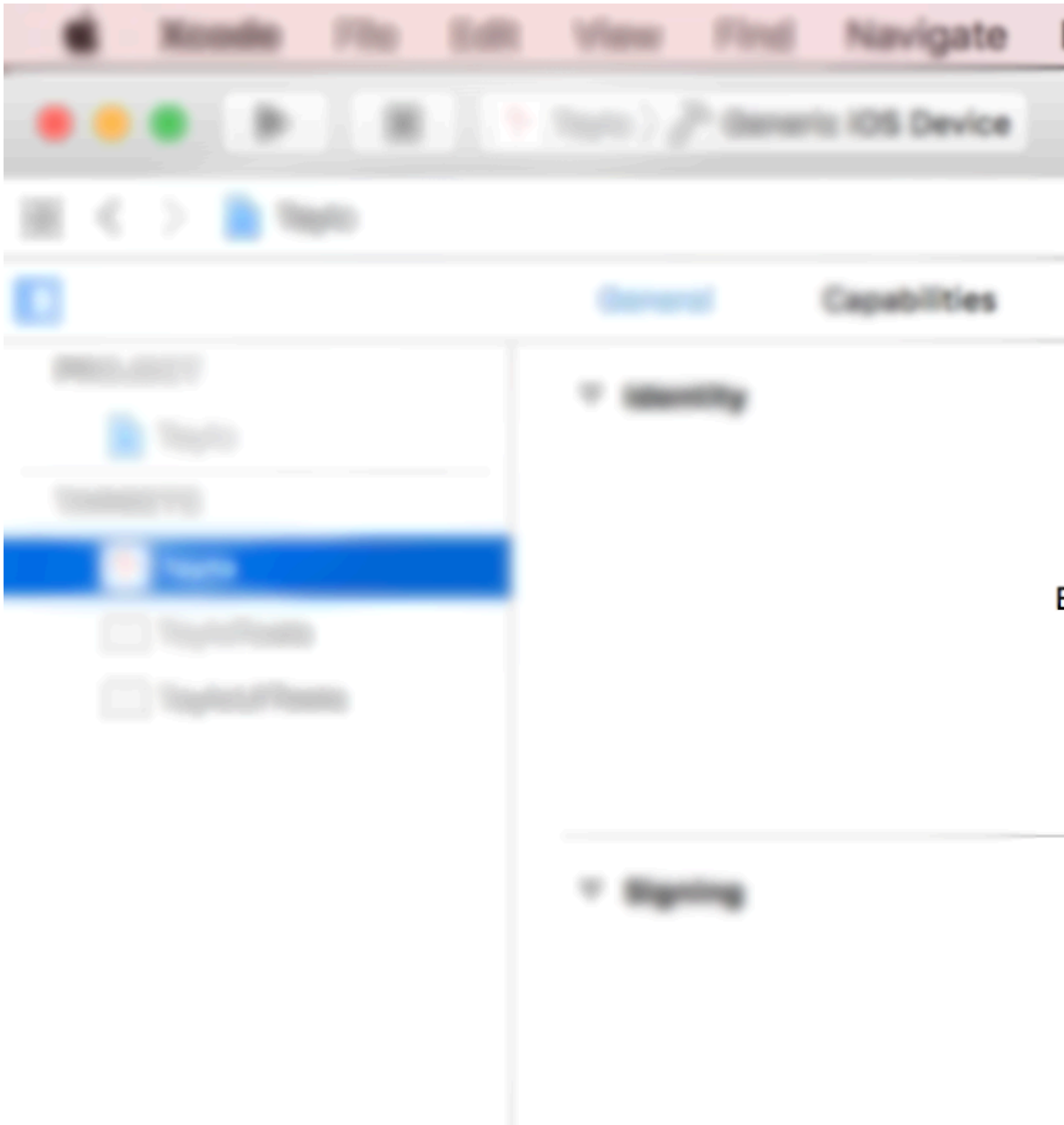
En versiones anteriores, la configuración de los perfiles de aprovisionamiento se hacía manualmente. Usted genera un perfil de distribución, lo descarga y luego distribuye su aplicación. Esto tenía que hacerse para cada máquina de desarrollo que consumía mucho tiempo. Sin embargo, en la mayoría de las situaciones de hoy en día, Xcode 8 hará la mayor parte de este trabajo por usted. Asegúrese de iniciar sesión con la cuenta que se utilizará para distribuir la aplicación y, a continuación, seleccione "Administrar automáticamente la firma de código" en Objetivos -> General.

▼ Signing

Automatically manage signing
Xcode will create and manage certificates for you.

Archivar el código

Una vez que todos los perfiles de aprovisionamiento están configurados, el siguiente paso en el proceso de envío de la aplicación es archivar su código. Desde el menú desplegable de dispositivos y simuladores, seleccione la opción "Dispositivo iOS genérico". Luego, en el menú "Producto" seleccione la opción "Archivo".



En caso de que el envío sea una actualización de la aplicación existente en la tienda, asegúrese de que el número de compilación sea más alto que el actual y que el número de versión sea diferente. Por ejemplo, la aplicación actual tiene el número de compilación 30 y la etiqueta de versión 1.0. La próxima actualización debe tener al menos el número de compilación 31 y la etiqueta de versión 1.0.1. En la mayoría de los casos, debe agregar un tercer decimal a su versión en caso de algunas correcciones de errores urgentes o parches pequeños, el segundo decimal se reserva principalmente para actualizaciones de funciones, mientras que el primer decimal se incrementa en caso de una actualización importante de la aplicación.

Exportar archivo IPA

Una vez hecho esto, puedes encontrar tu archivo en el organizador de Xcode. Aquí es donde se guardan y organizan todas las versiones anteriores y las compilaciones de archivo en caso de que no las elimine. Inmediatamente notará un gran botón azul que dice "Cargar en la App Store ..." sin embargo, en 9/10 casos esto no funcionará debido a varias razones (principalmente errores de Xcode). La solución es exportar su archivo y cargarlo utilizando otra herramienta de Xcode llamada Application Loader. Sin embargo, dado que el cargador de aplicaciones carga archivos IPA en la App Store, el archivo debe exportarse al formato correcto. Esta es una tarea trivial que puede tomar ~ media hora. Haga clic en el botón "Exportar" en el panel lateral derecho.

Select a method for export:

- Save for iOS App Store Deployment**
Sign and package application for distribution in the iOS App Store
- Save for Ad Hoc Deployment**
Sign and package application for Ad Hoc distribution outside of the App Store
- Save for Enterprise Deployment**
Sign and package application for enterprise distribution outside of the App Store
- Save for Development Deployment**
Sign and package application for development distribution outside of the App Store

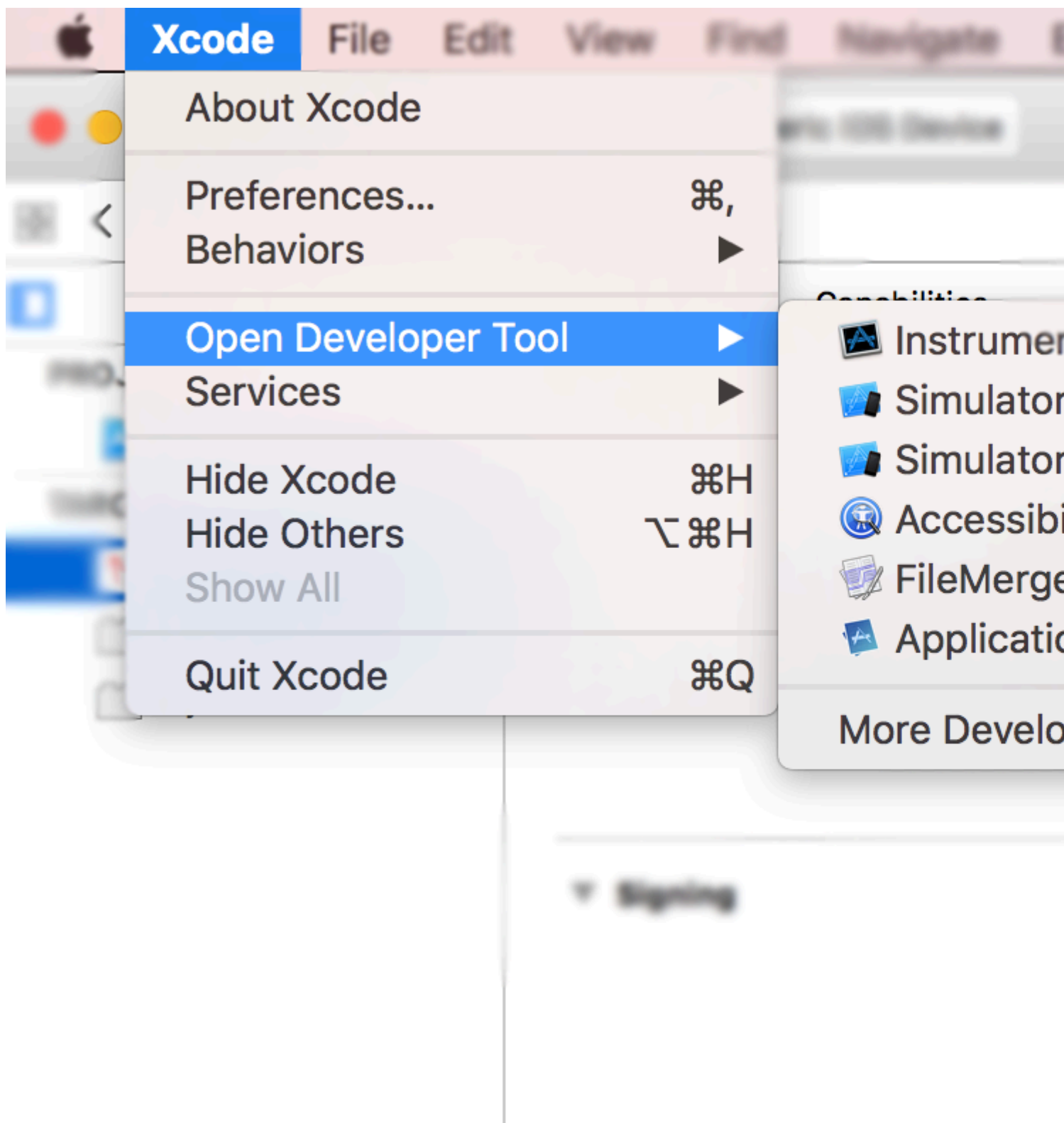
Cancel

Si está cargando una aplicación en la App Store, seleccione la primera opción y haga clic en Siguiente. Inicia sesión y valida tu código una vez más y toma una taza de café. Una vez que se

realiza el proceso de exportación, se le preguntará dónde guardar el archivo IPA generado. Por lo general, el escritorio es la opción más conveniente.

Cargar archivo IPA utilizando el cargador de aplicaciones

Una vez que se genere el archivo IPA, abra Xcode, navegue a las herramientas del desarrollador y abra el Cargador de aplicaciones.



Si tiene varias cuentas en su Xcode, se le pedirá que elija. Naturalmente, elija el que utilizó para

firmar el código en el primer paso. Seleccione "Entregar su aplicación" y cargue el código. Una vez realizada la carga, puede tardar hasta una hora en aparecer en su lista de compilación en iTunes Connect.



Deliver Your App

Open Recent ▾

Import

Lea Proceso de envío de aplicaciones en línea: <https://riptutorial.com/es/ios/topic/8765/proceso-de-envio-de-aplicaciones>

Capítulo 133: Pruebas de interfaz de usuario

Sintaxis

- XCUIApplication () // Proxy para una aplicación. La información que identifica la aplicación se especifica en la configuración de destino de Xcode como la "Aplicación de destino".
- XCUIElement () // Un elemento de interfaz de usuario en una aplicación.

Examples

Agregando archivos de prueba a Xcode Project

Al crear el proyecto.

Debe marcar "Incluir pruebas de IU" en el cuadro de diálogo de creación de proyecto.

Language:

Devices:

Use Core Data

Include Unit Tests

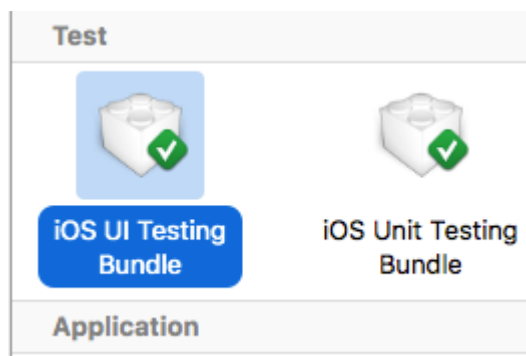
Include UI Tests

Después de crear el proyecto.

Si no pudo comprobar el `UI target` al crear un proyecto, siempre podría agregar un objetivo de prueba más tarde.

Setps:

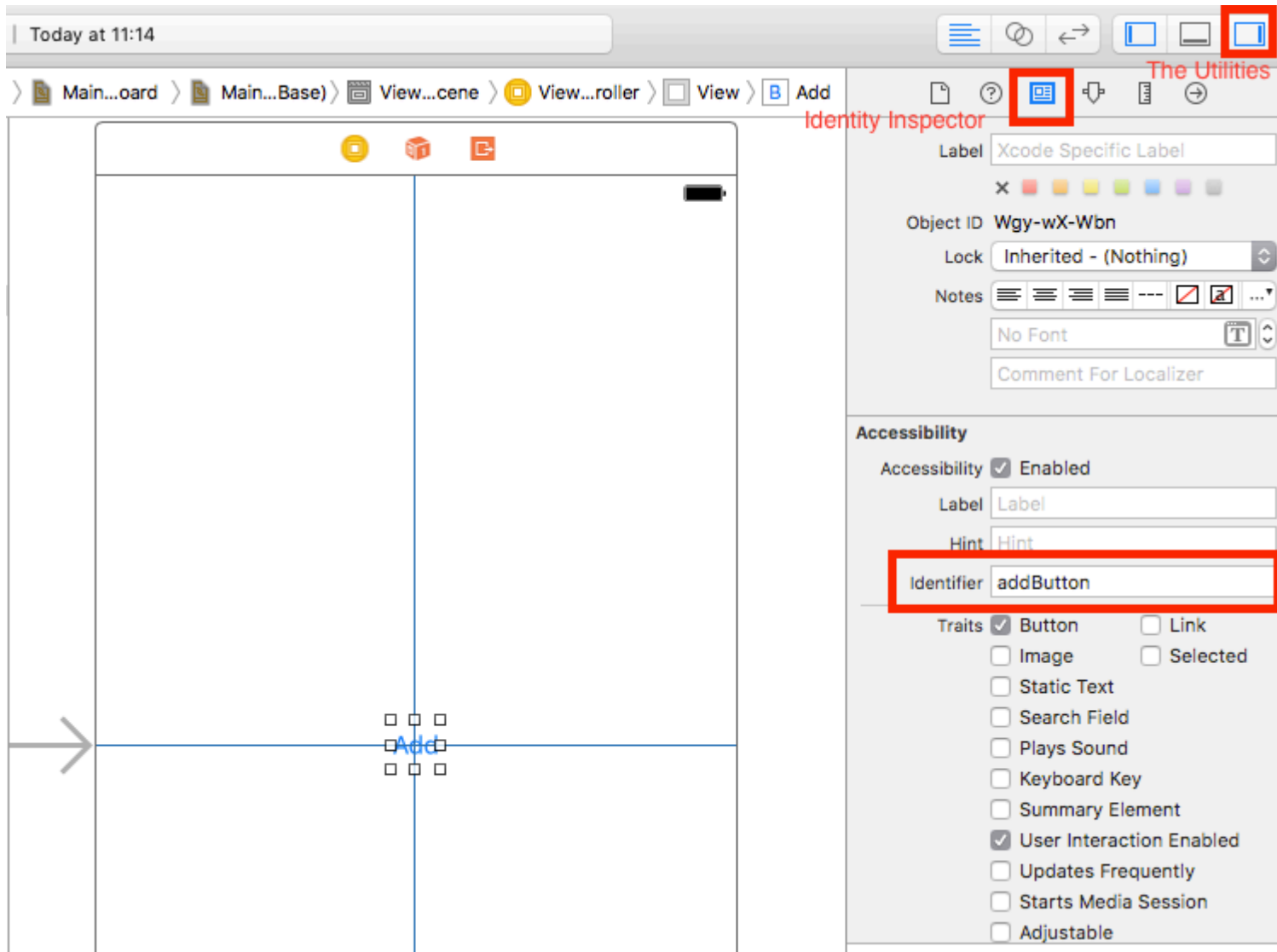
- Mientras el proyecto está abierto, vaya a `File -> New -> Target`
- Encuentra `iOS UI Testing Bundle`



Identificador de accesibilidad

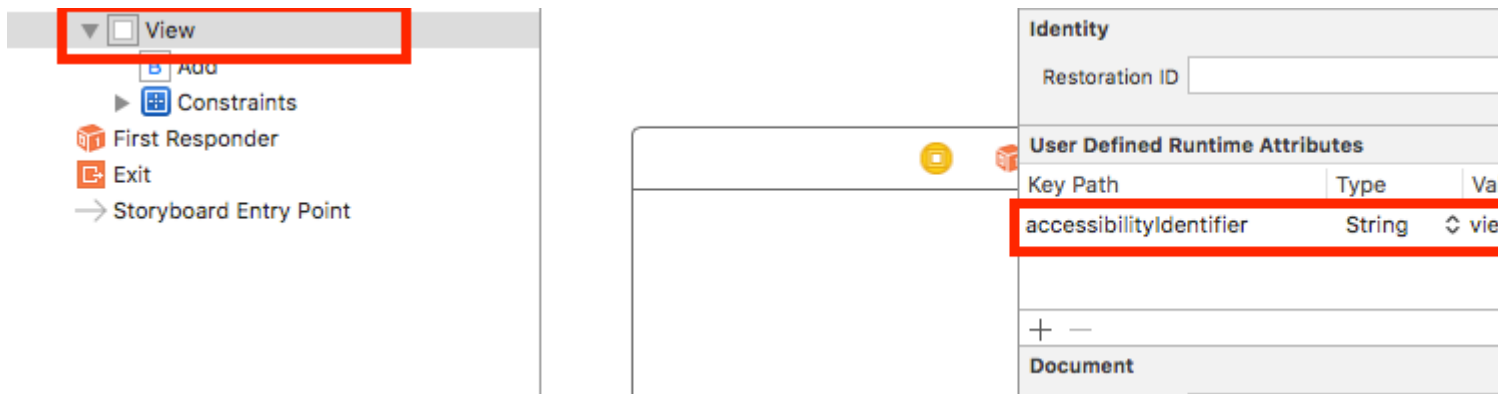
Cuando la accesibilidad habilitada en Utilidades

- Seleccione storyboard .
- Expandir the Utilities
- Seleccione Identity Inspector
- Selecciona tu elemento en el gui3n gr3fico.
- Agregar nuevo identificador de accesibilidad (en el ejemplo addButton)



Cuando la accesibilidad est3 deshabilitada en Utilidades

- Seleccione storyboard .
- Expandir the Utilities
- Seleccione Identity Inspector
- Selecciona tu elemento en el gui3n gr3fico.
- A3adir atributo en los atributos de User Defined Runtime Attributes
- Para el tipo de Key Path accessibilityIdentifier Key Path - accessibilityIdentifier
- Para Type - `Cadena
- Para Value : nuevo identificador de accesibilidad para su elemento (en la view ejemplo)



Configuración en el archivo UITest

```
import XCTest

class StackOverFlowUITests: XCTestCase {

    private let app = XCUIApplication()

    //Views

    private var view: XCUIElement!

    //Buttons

    private var addButton: XCUIElement!

    override func setUp() {
        super.setUp()

        app.launch()

        //Views

        view = app.otherElements["view"]

        //Buttons

        addButton = app.buttons["addButton"]
    }

    func testMyApp() {

        addButton.tap()
        view.tap()
    }
}
```

En [] agregue el identificador de accesibilidad para el elemento.

UIView, UIImageView, UIScrollView

```
let imageView = app.images["imageView"]
let scrollView = app.scrollViews["scrollView"]
```

```
let view = app.otherElements["view"]
```

UILabel

```
let label = app.staticTexts["label"]
```

UIStackView

```
let stackView = app.otherElements["stackView"]
```

UITableView

```
let tableView = app.tables["tableView"]
```

UITableViewCell

```
let tableViewCell = tableView.cells["tableViewCell"]
```

Elementos de UITableViewCell

```
let tableViewCellButton = tableView.cells.element(boundBy: 0).buttons["button"]
```

UICollectionView

```
let collectionView = app.collectionViews["collectionView"]
```

UIButton, UIBarButtonItem

```
let button = app.buttons["button"]  
let barButtonItem = app.buttons["barButtonItem"]
```

UITextField

- normal

```
let textField = app.textFields["textField"]
```

- contraseña UITextField

```
let passwordTextField = app.secureTextFields["passwordTextField"]
```

UITextView

```
let textView = app.textViews["textView"]
```

UISwitch

```
let switch = app.switches["switch"]
```

Las alertas

```
let alert = app.alerts["About yourself"] // Title of presented alert
```

Deshabilitar animaciones durante la prueba de interfaz de usuario

En una prueba puedes deshabilitar animaciones agregando en `setUp` :

```
app.launchEnvironment = ["animations": "0"]
```

Donde la `app` es instancia de `XCUIApplication`.

Almuerzo y finalización de la aplicación mientras se ejecuta.

Aplicación de almuerzo para pruebas.

```
override func setUp() {
    super.setUp()

    let app = XCUIApplication()

    app.launch()
}
```

Aplicación de terminación

```
func testStacOverFlowApp() {

    app.terminate()
}
```

Rotar dispositivos

El dispositivo se puede girar cambiando la `orientation` en `XCUIDevice.shared().orientation` :

```
XCUIDevice.shared().orientation = .landscapeLeft
```

```
XCUIDevice.shared().orientation = .portrait
```

Lea Pruebas de interfaz de usuario en línea: <https://riptutorial.com/es/ios/topic/7526/pruebas-de-interfaz-de-usuario>

Capítulo 134: Red de redes

Examples

Despacho de bloque de finalización en un hilo personalizado

Cuando se utiliza AFNetworking, la llamada se envía en un subproceso personalizado proporcionado por AFNetworking. Cuando la llamada vuelve al bloque de finalización, se ejecuta en el hilo principal.

Este ejemplo establece un hilo personalizado que se envía al bloque de finalización:

AFNetworking 2.xx:

```
// Create dispatch_queue_t with your name and DISPATCH_QUEUE_SERIAL as for the flag
dispatch_queue_t myQueue = dispatch_queue_create("com.CompanyName.AppName.methodTest",
        DISPATCH_QUEUE_SERIAL);

// init AFHTTPRequestOperation of AFNetworking
operation = [[AFHTTPRequestOperation alloc] initWithRequest:request];

// Set the FMDB property to run off the main thread
[operation setCompletionQueue:myQueue];
```

AFNetworking 3.xx:

```
AFHTTPSessionManager *manager = [[AFHTTPSessionManager alloc] init];
[self setCompletionQueue:myQueue];
```

Lea Red de redes en línea: <https://riptutorial.com/es/ios/topic/3002/red-de-redes>

Capítulo 135: Referencia CGContext

Observaciones

El tipo opaco **CGContextRef** representa un destino de dibujo Quartz 2D. Un contexto gráfico contiene parámetros de dibujo y toda la información específica del dispositivo necesaria para representar la pintura en una página al destino, ya sea que el destino sea una ventana en una aplicación, una imagen de mapa de bits, un documento PDF o una impresora.

Examples

Dibujar linea

```
CGContextRef context = UIGraphicsGetCurrentContext();

CGContextSetLineWidth(context, 5.0);
CGColorSpaceRef colorspace = CGColorSpaceCreateDeviceRGB();
CGContextMoveToPoint(context, 200, 400);
CGContextAddLineToPoint(context, 100, 100);
CGContextStrokePath(context);
CGColorSpaceRelease(colorspace);
```



Dibujar texto

Draw To requiere que se agregue el **marco de Core Text** en la fase de compilación

```
[NSString* textToDraw = @"Welcome to the world of IOS";

CFStringRef stringRef = (__bridge CFStringRef)textToDraw;

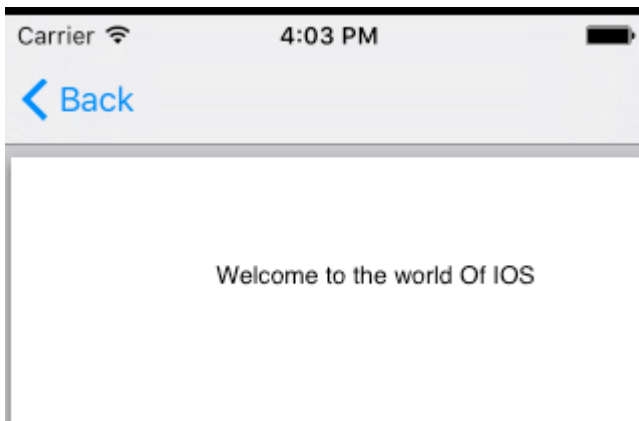
CFAttributedStringRef currentText = CFAttributedStringCreate(NULL, stringRef, NULL);
CTFramesetterRef framesetter = CTFramesetterCreateWithAttributedString(currentText);
CGRect frameRect = CGRectMake(0, 0, 300, 100);
CGMutablePathRef framePath = CGPathCreateMutable();
CGPathAddRect(framePath, NULL, frameRect);

CFRange currentRange = CFRangeMake(0, 0);
CTFrameRef frameRef = CTFramesetterCreateFrame(framesetter, currentRange, framePath,
NULL);
```

```
CGPathRelease(framePath);
CGContextRef currentContext = UIGraphicsGetCurrentContext();

CGContextSetTextMatrix(currentContext, CGAffineTransformIdentity);
CGContextTranslateCTM(currentContext, 200, 300);
CGContextScaleCTM(currentContext, 2, -2);
CTFrameDraw(frameRef, currentContext);

CFRelease(frameRef);
CFRelease(stringRef);
CFRelease(framesetter);
```



Lea Referencia CGContext en línea: <https://riptutorial.com/es/ios/topic/2664/referencia-cgcontext>

Capítulo 136: Reino

Observaciones

Adición de un nuevo objeto RLMO a un dominio existente: esquema y migraciones

Agregar nuevas clases de modelos a un Reino no requiere una migración o un bump de la versión del esquema; solo haciendo cambios a un reino existente.

Examples

Clase de modelo base de RLMObject con clave principal - Objective-C

Un ejemplo de una clase de modelo base de RLMObject que usa una clave principal y algunas propiedades genéricas predeterminadas. Las subclasses pueden establecer metadatos específicos a sus necesidades.

```
@interface BaseModel : RLMObject

@property NSString *uuid;
@property NSString *metadata;

@end

@implementation BaseModel

+ (NSString *)primaryKey
{
    return @"uuid";
}

+ (NSDictionary *)defaultPropertyValues
{
    NSMutableDictionary *defaultPropertyValues = [NSMutableDictionary
dictionaryWithDictionary:[super defaultPropertyValues]];
    NSString *uuid = [[NSUUID UUID] UUIDString];
    [defaultPropertyValues setValue:@"" forKey:@"metadata"];
    [defaultPropertyValues setValue:uuid forKey:@"uuid"];
    return defaultPropertyValues;
}

+ (NSArray *)ignoredProperties
{
    return @[];
}

@end
```

Lea Reino en línea: <https://riptutorial.com/es/ios/topic/4084/reino>

Capítulo 137: Segues

Examples

Una visión general

De la documentación de Apple:

Un objeto `UIStoryboardSegue` es responsable de **realizar la transición visual entre dos controladores de vista** . Además, los objetos segue se utilizan para preparar la transición de un controlador de vista a otro. **Los objetos Segue contienen información sobre los controladores de vista involucrados en una transición** .

Cuando se activa un segmento, pero antes de que se produzca la transición visual, el tiempo de ejecución del guión gráfico llama al método `prepareForSegue(sender:)` del controlador de la vista actual para que pueda pasar los datos necesarios al controlador de la vista que está a punto de mostrarse.

Atributos

Rápido

```
sourceViewController: UIViewController {get}
destinationViewController: UIViewController {get}
identifier: String? {get}
```

Referencias:

- [Referencia de clase UIViewController](#)
- [Referencia de clase UIStoryboardSegue](#)

Preparando su controlador de vista antes de disparar un Segue

PrepareForSegue :

```
func prepareForSegue(_ segue: UIStoryboardSegue, sender sender:AnyObject?)
```

Notifica al controlador de vista que un segmento está a punto de realizarse.

Parámetros

segue : el objeto segue.

remitente : El objeto que inicializó el segue.

Ejemplo en Swift

Realizar una tarea si el identificador del segmento es "SomeSpecificIdentifier"

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
    if segue.identifier == "SomeSpecificIdentifier" {
        //- Do specific task
    }
}
```

Decidir si se debe realizar un Segue invocado.

ShouldPerformSegueWithIdentifier :

```
func shouldPerformSegueWithIdentifier(_ identifier:String, sender sender:AnyObject?) -> Bool
```

Determina si se debe realizar el segmento con el identificador especificado.

Parámetros

Identificador : Cadena que identifica el segmento desencadenado.

Remitente : El objeto que inicializó el segue.

Ejemplo en Swift

Solo realice segue si el identificador es "SomeSpecificIdentifier"

```
override func shouldPerformSegueWithIdentifier(identifier:String, sender:AnyObject?) -> Bool {
    if identifier == "SomeSpecificIdentifier" {
        return true
    }
    return false
}
```

Usando Segues para navegar hacia atrás en la pila de navegación

Desenrollar Segues

Desenrollar los Segues le da una manera de "desenrollar" la pila de navegación y especificar un destino al que volver. La firma de esta función es clave para que Interface Builder la reconozca. **Debe tener un valor de retorno de IBAction y tomar un parámetro de UIStoryboardSegue** . El nombre de la función no importa. De hecho, la función ni siquiera tiene que hacer nada. Simplemente está ahí como un marcador de cuál UIViewController es el destino de la Segue de Desenrollamiento.

[fuente] [1]

Firma requerida de un segue desenrollado

C objetivo:

```
-(IBAction)prepareForUnwind:(UIStoryboardSegue *) segue {  
}
```

Rápido:

```
@IBAction func prepareForUnwind(segue: UIStoryboardSegue) {  
}
```

Segue de disparo programáticamente

PerformSegueWithIdentifier:

```
func performSegueWithIdentifier(_ identifier:String, sender sender:AnyObject?)
```

Inicia el segmento con el identificador especificado del archivo de guión gráfico del controlador de la vista actual

Parámetros

Identificador : Cadena que identifica el segmento desencadenado.

Remitente : El objeto que iniciará el segue.

Ejemplo en Swift

Realización de un segmento con el identificador "SomeSpecificIdentifier" desde una selección de fila de vista de tabla:

```
func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {  
    performSegueWithIdentifier("SomeSpecificIdentifier", sender: indexPath.item)  
}
```

Lea Segues en línea: <https://riptutorial.com/es/ios/topic/5575/segues>

Capítulo 138: Seguridad

Introducción

La seguridad en iOS está relacionada con la seguridad de datos, seguridad de transporte, seguridad de código, etc.

Examples

Seguridad de transporte utilizando SSL

Las aplicaciones de iOS se deben escribir de manera que proporcionen seguridad a los datos que se transportan a través de la red.

SSL es la forma común de hacerlo.

Cada vez que la aplicación intenta llamar a servicios web para extraer o enviar datos a servidores, debe **usar SSL a través de HTTP, es decir, HTTPS**.

Para hacer esto, la **aplicación debe llamar a `https://server.com/part`** dichos servicios web y no a `http://server.com/part`.

En este caso, la aplicación debe confiar en el servidor `server.com` mediante un certificado SSL.

Aquí está el ejemplo de validación de la confianza del servidor

Implementar `URLSessionDelegate` como:

```
func urlSession(_ session: URLSession, didReceive challenge: URLAuthenticationChallenge,
completionHandler: @escaping (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {

    if challenge.protectionSpace.authenticationMethod == NSURLAuthenticationMethodServerTrust
    {
        let serverTrust:SecTrust = challenge.protectionSpace.serverTrust!

        func acceptServerTrust() {
            let credential:URLCredential = URLCredential(trust: serverTrust)
            challenge.sender?.use(credential, for: challenge)
            completionHandler(.useCredential, URLCredential(trust:
challenge.protectionSpace.serverTrust!))
        }

        let success = SSLTrustManager.shouldTrustServerTrust(serverTrust, forCert:
"Server_Public_SSL_Cert")
        if success {
            acceptServerTrust()
            return
        }
    }
    else if challenge.protectionSpace.authenticationMethod ==
NSURLAuthenticationMethodClientCertificate {
        completionHandler(.rejectProtectionSpace, nil);
        return
    }
}
```

```
completionHandler(.cancelAuthenticationChallenge, nil)
}
```

Aquí está el administrador de confianza: (no se pudo encontrar el código Swift)

```
@implementation SSLTrustManager
+ (BOOL)shouldTrustServerTrust:(SecTrustRef)serverTrust forCert:(NSString*)certName {
// Load up the bundled certificate.
NSString *certPath = [[NSBundle mainBundle] pathForResource:certName ofType:@"der"];
NSData *certData = [[NSData alloc] initWithContentsOfFile:certPath];
CFDataRef certDataRef = (__bridge_retained CFDataRef)certData;
SecCertificateRef cert = SecCertificateCreateWithData(NULL, certDataRef);

// Establish a chain of trust anchored on our bundled certificate.
CFArrayRef certArrayRef = CFArrayCreate(NULL, (void *)&cert, 1, NULL);
SecTrustSetAnchorCertificates(serverTrust, certArrayRef);

// Verify that trust.
SecTrustResultType trustResult;
SecTrustEvaluate(serverTrust, &trustResult);

// Clean up.
CFRelease(certArrayRef);
CFRelease(cert);
CFRelease(certDataRef);

// Did our custom trust chain evaluate successfully?
return trustResult == kSecTrustResultUnspecified;
}
@end
```

Server_Public_SSL_Cert.der es la clave pública SSL de los servidores.

Usando este enfoque, nuestra aplicación puede asegurarse de que se está comunicando con el servidor deseado y que nadie está interceptando la comunicación entre la aplicación y el servidor.

Protección de datos en las copias de seguridad de iTunes

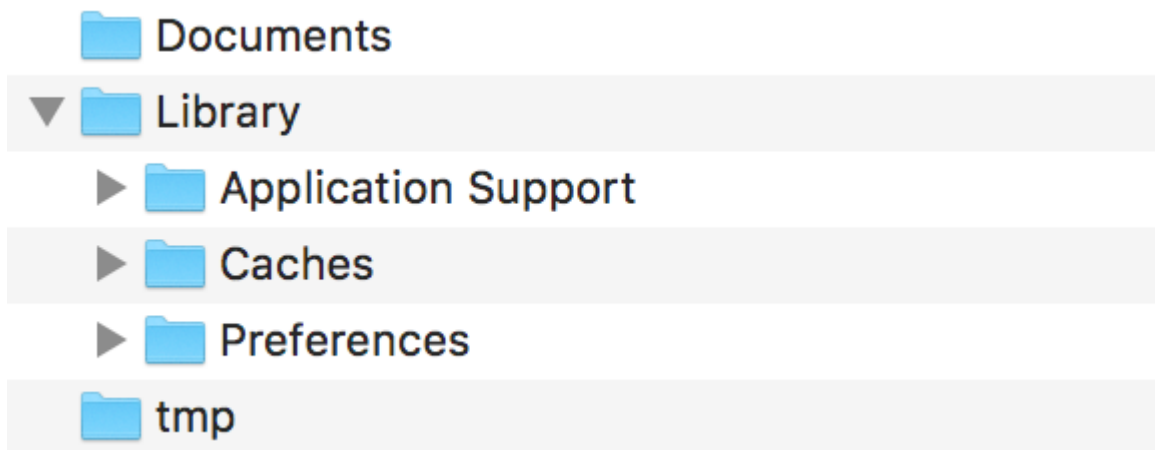
Si queremos que los datos de nuestra aplicación estén protegidos contra las copias de seguridad de iTunes, debemos omitir que los datos de nuestra aplicación se copien en iTunes.

Cada vez que se realiza una copia de seguridad del dispositivo iOS con MacOS, todos los datos almacenados por todas las aplicaciones se copian en esa copia de seguridad y se almacenan en la computadora de respaldo.

Pero podemos excluir los datos de nuestra aplicación de esta copia de seguridad usando la clave

`URLResourceKey.isExcludedFromBackupKey`.

Aquí está la estructura de directorios de nuestra aplicación:



Nota: En general, los datos confidenciales se almacenan en el directorio "Soporte de aplicaciones".

Por ejemplo, si queremos excluir todos nuestros datos almacenados en el directorio de **Soporte de aplicaciones**, podemos usar la clave mencionada anteriormente de la siguiente manera:

```
let urls = FileManager.default.urls(for: .applicationSupportDirectory, in:
.userDomainMask)
let baseURL = urls[urls.count-1];

let bundleIdentifier = Bundle.main.object(forKey: "CFBundleIdentifier") as!
String
let pathURL = baseURL.appendingPathComponent(bundleIdentifier)
let persistentStoreDirectoryPath = pathURL.path
if !FileManager.default.fileExists(atPath: persistentStoreDirectoryPath) {
    do {
        try FileManager.default.createDirectory(atPath: path, withIntermediateDirectories:
true, attributes: nil)
    }catch {
        //handle error
    }
}
let dirURL = URL.init(fileURLWithPath: persistentStoreDirectoryPath, isDirectory: true)
do {
    try (dirURL as NSURL).setResourceValue((true), forKey: .isExcludedFromBackupKey)
} catch {
    //handle error
}
```

Hay muchas herramientas disponibles para ver las copias de seguridad de iTunes de todos los datos respaldados para confirmar si el enfoque anterior funciona o no.

[iExplorer](#) es bueno para explorar las copias de seguridad de iTunes.

Lea Seguridad en línea: <https://riptutorial.com/es/ios/topic/9999/seguridad>

Capítulo 139: Servicios de safari

Examples

Implementar SFSafariViewControllerDelegate

Debe implementar `SFSafariViewControllerDelegate` para que se notifique a su clase cuando el usuario presione el botón Hecho en `SafariViewController` y también puede descartarlo.

Primero declara tu clase para implementar el protocolo.

```
class MyClass: SFSafariViewControllerDelegate {  
  
}
```

Implementar el método de delegado para ser notificado en el despido.

```
func safariViewControllerDidFinish(controller: SFSafariViewController) {  
    // Dismiss the SafariViewController when done  
    controller.dismissViewControllerAnimated(true, completion: nil)  
}
```

No te olvides de establecer tu clase como el delegado de `SafariViewController`.

```
let safariVC = SFSafariViewController(URL: yourURL)  
safariVC.delegate = self
```

Los métodos de delegado adicionales que puede implementar son:

```
// Called when the initial URL load is complete.  
safariViewController(_ controller: SFSafariViewController, didCompleteInitialLoad  
didLoadSuccessfully: Bool) { }  
  
// Called when the user taps an Action button.  
safariViewController(_ controller: SFSafariViewController, activityItemsFor URL: URL, title:  
String?) -> [UIActivity] { }
```

Agregar artículos a la lista de lectura de Safari

Puede agregar elementos a la Lista de lectura de un usuario en Safari llamando al método `addItem` en el singleton `SSReadingList`.

```
let readingList = SSReadingList.default()  
readingList?.addItem(with: yourURL, title: "optional title", previewText: "optional preview  
text")
```

La lista de lectura predeterminada puede ser `nil` si no se permite el acceso a la lista de lectura.

Además, puede verificar si la Lista de lectura admite una URL llamando a `supportsURL` .

```
SSReadingList.default().supportsURL(URL(string: "https://example.com")!)
```

Esto devolverá `true` o `false` indicando si la URL dada es compatible con la Lista de lectura de Safari. Use esto, por ejemplo, para determinar si se muestra un botón para agregar una URL a la Lista de lectura.

Abre una URL con `SafariViewController`

No olvides importar primero el framework necesario.

```
import SafariServices
//Objective-C
@import SafariServices;
```

`SafariViewController` una instancia de `SafariViewController` .

```
let safariVC = SFSafariViewController(URL: URL(string: "your_url")!)
//Objective-C
@import SafariServices;
NSURL *URL = [NSURL URLWithString:[NSString stringWithFormat:@"http://www.google.com"]];
SFSafariViewController *sfvc = [[SFSafariViewController alloc] initWithURL:URL];
```

Opcionalmente, también puede decirle a `SafariViewController` que ingrese al modo de lectura si es posible una vez que haya terminado de cargarse.

```
let safariVC = SFSafariViewController(URL: URL(string: "your_url")!, entersReaderIfAvailable:
true)
//Objective-C
NSURL *URL = [NSURL URLWithString:[NSString stringWithFormat:@"http://www.google.com"]];
SFSafariViewController *sfvc = [[SFSafariViewController alloc] initWithURL:URL
entersReaderIfAvailable:YES];
```

Presentar el controlador de vista.

```
present(safariVC, animated: true, completion: nil)
//Objective-C
[self presentViewController:sfvc animated:YES completion:nil];
```

Lea Servicios de safari en línea: <https://riptutorial.com/es/ios/topic/1371/servicios-de-safari>

Capítulo 140: SESIÓN

Observaciones

La clase **NSURLSession** y las clases relacionadas proporcionan una API para descargar contenido. Esta API proporciona un amplio conjunto de métodos de delegado para admitir la autenticación y le brinda a la aplicación la capacidad de realizar descargas en segundo plano cuando la aplicación no se está ejecutando o, en iOS, mientras la aplicación está suspendida.

En un nivel alto, **NSURLSession** se basa en el concepto de sesiones y tareas. Una tarea representa una solicitud única para una URL única (o una carga única a una URL única). Una sesión es un grupo de solicitudes relacionadas.

El sistema operativo proporciona una única sesión preexistente: la sesión compartida, que básicamente funciona como **NSURLConnection**. Además, puede crear sus propias sesiones en su aplicación según sea necesario.

Diferentes aplicaciones utilizan las sesiones de diferentes maneras. Muchas aplicaciones crean una sola sesión en el inicio y simplemente la reutilizan. Otras aplicaciones se benefician de la posibilidad de cancelar un grupo de tareas relacionadas (por ejemplo, un navegador web que cancela todas las solicitudes pendientes al cerrar una pestaña) y, por lo tanto, crean una sesión para contener cada grupo de solicitudes relacionadas.

El primer paso cuando se usa **NSURLSession** es crear un objeto de configuración de sesión. El objeto (generalmente) reutilizable contiene varias configuraciones de sesión que puede modificar para sus necesidades particulares, como máxima concurrencia, encabezados adicionales para enviar con cada solicitud, ya sea para permitir el envío de solicitudes a través de la radio celular (solo iOS), tiempos de espera, almacenamiento de credenciales, versión mínima de TLS e incluso configuraciones de proxy.

Hay tres tipos de configuraciones de sesión, dependiendo de cómo desea que se comporte la sesión resultante:

- **Las configuraciones predeterminadas** crean sesiones que funcionan de manera muy similar a **NSURLConnection**.
- **Las configuraciones en segundo plano** crean sesiones en las que las solicitudes pasan fuera del proceso, lo que permite que las descargas continúen incluso cuando la aplicación ya no se ejecuta.
- **Las configuraciones efímeras** crean sesiones que no almacenan en caché nada en el disco, no almacenan cookies en el disco, etc. y, por lo tanto, son adecuadas para respaldar cosas como las ventanas de incógnito del navegador.

Cuando crea una configuración en segundo plano, debe proporcionar un identificador de sesión que le permita volver a asociar la sesión en segundo plano más tarde (si su aplicación sale o es suspendida o terminada por el sistema operativo). No debe tener más de una instancia de una sesión con el mismo identificador activo en su aplicación, por lo que, como regla, estas

configuraciones no son reutilizables. Todas las demás configuraciones de sesión se pueden reutilizar para crear tantas sesiones como desee. Por lo tanto, si necesita crear varias sesiones con configuraciones similares, puede crear la configuración una vez y reutilizarla cada vez que cree una nueva sesión.

Después de crear una sesión, puede crear tareas en esa sesión. Hay tres tipos de tareas:

- **Las tareas de datos** devuelven datos como un objeto **NSData** . Estos son adecuados para uso general, pero no se admiten en sesiones de fondo.
- **Las tareas de descarga** devuelven datos como un archivo en el disco. Estos son adecuados para solicitudes más grandes, o para uso en sesiones de fondo.
- **Cargar tareas** cargar datos desde un objeto NSData o desde un archivo en el disco. Proporciona un objeto o archivo de datos que proporciona el cuerpo POST. Los datos / archivos de cuerpo que proporciona en la tarea anulan cualquier archivo / datos de cuerpo proporcionados en el objeto NSURLRequest (si corresponde).

Cada uno de estos tipos le permite obtener los datos de respuesta de un par de formas diferentes, ya sea utilizando devoluciones de llamadas basadas en bloques o proporcionando un delegado en la sesión e implementando métodos de delegado.

Además, NSURLSession le permite proporcionar métodos delegados para manejar la autenticación, realizar el manejo personalizado del certificado TLS (tanto para certificados de cliente y validación del servidor), cambiar el comportamiento de almacenamiento en caché, etc.

Examples

Solicitud GET simple

```
// define url
let url = NSURL(string: "https://urlToGet.com")

//create a task to get data from a url
let task = NSURLSession.sharedSession().dataTaskWithURL(url!)
{
    /*inside this block, we have access to NSData *data, NSURLResponse *response, and
    NSError *error returned by the dataTaskWithURL() function*/
    (data, response, error) in

    if error == nil
    {
        // Data from the request can be manipulated here
    }
    else
    {
        // An error occurred
    }
}

//make the request
task.resume()
```

Objective-C Crear una sesión y tarea de datos

```
NSURL *url = [NSURL URLWithString:@"http://www.example.com/"];
NSURLSessionConfiguration *configuration = [NSURLSessionConfiguration
defaultSessionConfiguration];

// Configure the session here.

NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration];

[[session dataTaskWithURL:url
    completionHandler:^(NSData *data, NSURLResponse *response, NSError *error)
 {
    // The response object contains the metadata (HTTP headers, status code)

    // The data object contains the response body

    // The error object contains any client-side errors (e.g. connection
    // failures) and, in some cases, may report server-side errors.
    // In general, however, you should detect server-side errors by
    // checking the HTTP status code in the response object.
}] resume];
```

Configuración de la configuración de fondo

Para crear una sesión de fondo

```
// Swift:
let mySessionID = "com.example.bgSession"
let bgSessionConfig =
NSURLSessionConfiguration.backgroundSessionConfigurationWithIdentifier(mySessionID)

let session = NSURLSession(configuration: bgSessionConfig)

// add tasks here

// Objective-C:
NSString *mySessionID = @"com.example.bgSession";
NSURLSessionConfiguration *configuration =
    [NSURLSessionConfiguration backgroundSessionConfigurationWithIdentifier: mySessionID];
NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration
    delegate:self]
```

Además, en iOS, debe configurar el soporte para manejar el reinicio de la aplicación en segundo plano. Cuando se `application:handleEventsForBackgroundURLSession:completionHandler:` la `application:handleEventsForBackgroundURLSession:completionHandler:` de su `application:handleEventsForBackgroundURLSession:completionHandler:` método (Objective-C) o `application(_:handleEventsForBackgroundURLSession:completionHandler:)` (Swift), significa que su aplicación se ha relanzado en segundo plano para manejar la actividad de una sesión.

En ese método, debe crear una nueva sesión con el identificador provisto y configurarlo con un delegado para manejar eventos como lo haría normalmente en el primer plano. Además, debe almacenar el controlador de finalización proporcionado en un diccionario, utilizando la sesión

como clave.

Cuando se llama al método `NSURLSessionDidFinishEventsForBackgroundURLSession:` del delegado

`NSURLSessionDidFinishEventsForBackgroundURLSession:` (Obj-C) /

`NSURLSessionDidFinishEventsForBackgroundURLSession` (Swift) para informarle que no hay más eventos que controlar, su aplicación debe buscar el controlador de finalización para esa sesión, eliminar la sesión del diccionario y llame al controlador de finalización y, por lo tanto, le dice al sistema operativo que ya no tiene ningún procesamiento pendiente relacionado con la sesión. (Si aún está haciendo algo por alguna razón cuando recibe la llamada de delegado, espere hasta que termine). Tan pronto como llame a ese método, la sesión en segundo plano se invalidará de inmediato.

Si su aplicación recibe una `application:application:didFinishLaunchingWithOptions:` call (es probable que indique que el usuario `application:application:didFinishLaunchingWithOptions:` primer plano su aplicación mientras estaba ocupado procesando eventos de fondo), es seguro crear una sesión en segundo plano con ese mismo identificador, porque la sesión anterior con ese ID identificador ya no existe.

Si tiene curiosidad acerca de los detalles, a un alto nivel, cuando crea una sesión en segundo plano, está haciendo dos cosas:

- Creando una sesión en un demonio externo (`NSURLSession`) para manejar las descargas
- Creando una sesión dentro de su aplicación que habla con ese demonio externo a través de `NSXPC`

Normalmente, es peligroso crear dos sesiones con el mismo ID de sesión en un solo lanzamiento de la aplicación, ya que ambos intentan hablar con la misma sesión en el daemon de fondo. Por esta razón, la documentación oficial dice que nunca se deben crear varias sesiones con el mismo identificador. Sin embargo, si la primera sesión fue una sesión temporal creada como parte de una llamada `handleEventsForBackgroundURLSession`, ya no existe la asociación entre la sesión dentro de la aplicación ahora invalidada y la sesión en el daemon de fondo.

Enviar una solicitud POST con argumentos utilizando `NSURLSession` en Objective-C

Hay dos formas comunes de codificar un cuerpo de solicitud POST: codificación de URL (`application / x-www-form-urlencoded`) y datos de formulario (`multipart / form-data`). Gran parte del código es similar, pero la forma en que construye los datos del cuerpo es diferente.

Enviando una solicitud usando codificación de URL

Ya sea que tenga un servidor para su pequeña aplicación o que trabaje en un equipo con un ingeniero completo, querrá hablar con ese servidor en un momento dado con su aplicación iOS.

En el siguiente código, vamos a componer una serie de argumentos que la secuencia de comandos del servidor de destino usará para hacer algo que cambia según su caso. Por ejemplo, podemos querer enviar la cadena:

nombre = Brendon y contraseña = abcde

Al servidor cuando un usuario se registra en su aplicación, para que el servidor pueda almacenar esta información en una base de datos.

Empecemos. Querrá crear una solicitud POST de NSURLSession con el siguiente código.

```
// Create the configuration, which is necessary so we can cancel cacheing amongst other things.
NSURLSessionConfiguration * defaultConfigObject = [NSURLSessionConfiguration
defaultSessionConfiguration];
// Disables cacheing
defaultConfigObject.requestCachePolicy = NSURLRequestReloadIgnoringLocalCacheData;
NSURLSession * defaultSession = [NSURLSession sessionWithConfiguration:defaultConfigObject
delegate:self delegateQueue:[NSOperationQueue mainQueue]];

NSString * scriptURL = [NSString stringWithFormat:@"https://server.io/api/script.php"];
//Converts the URL string to a URL usable by NSURLSession
NSMutableURLRequest * urlRequest = [NSMutableURLRequest requestWithURL:[NSURL
URLWithString:scriptURL]];
NSString * postDataString = [NSString stringWithFormat:@"name=%@&password=%@", [self
nameString], [self URLEncode:passwordString]];
[urlRequest setHTTPMethod:@"POST"];
[urlRequest setHTTPBody:[postDataString dataUsingEncoding:NSUTF8StringEncoding]];

NSURLSessionDataTask * dataTask = [defaultSession dataTaskWithRequest:urlRequest];
// Fire the data task.
[dataTask resume];
```

El código anterior acaba de crear y disparó la solicitud POST al servidor. Recuerde que la URL del script y la cadena de datos de la POST cambian según su situación. Si estás leyendo esto, sabrás con qué llenar esas variables.

También deberá agregar un pequeño método que haga la codificación de la URL:

```
- (NSString *)URLEncode:(NSString *)originalString encoding:(NSStringEncoding)encoding
{
    return (__bridge_transfer NSString *)CFURLCreateStringByAddingPercentEscapes(
        kCFAllocatorDefault,
        (__bridge CFStringRef)originalString,
        NULL,
        CFSTR(":/?#[!$&'()*+,-;="),
        CFStringConvertNSStringEncodingToEncoding(encoding));
}
```

Por lo tanto, cuando el servidor haya terminado de procesar estos datos, enviará un retorno a su aplicación iOS. Entonces necesitamos procesar este retorno, pero ¿cómo?

Usamos programación dirigida por eventos y usamos los métodos de delegado de NSURLSession. Esto significa que cuando el servidor envía una respuesta, estos métodos comenzarán a activarse. Los siguientes 5 métodos son los que se activarán a lo largo de la solicitud ENTERA, cada vez que se realice uno:

```
- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
```

```

didReceiveResponse:(NSURLResponse *)response
    completionHandler:(void (^)(NSURLSessionResponseDisposition disposition))completionHandler;

- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveData:(NSData *)data;

- (void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didCompleteWithError:(NSError *)error;

- (void)URLSession:(NSURLSession *)session didReceiveChallenge:(NSURLAuthenticationChallenge
*)challenge completionHandler:(void (^)(NSURLSessionAuthChallengeDisposition, NSURLCredential
*))completionHandler;

- (void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didReceiveChallenge:(NSURLAuthenticationChallenge *)challenge completionHandler:(void
(^)(NSURLSessionAuthChallengeDisposition, NSURLCredential * _Nullable))completionHandler;

```

A continuación verá los métodos anteriores utilizados en contexto. Cada uno de sus propósitos se explica por sí mismo gracias a Apple, pero he comentado sus usos de todos modos:

```

// Response handling delegates
- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveResponse:(NSURLResponse *)response
    completionHandler:(void (^)(NSURLSessionResponseDisposition disposition))completionHandler{
    // Handler allows us to receive and parse responses from the server
    completionHandler(NSURLSessionResponseAllow);
}

- (void)URLSession:(NSURLSession *)session dataTask:(NSURLSessionDataTask *)dataTask
didReceiveData:(NSData *)data{

    // Parse the JSON that came in into an NSDictionary
    NSError * err = nil;
    NSDictionary * jsonDict = [NSJSONSerialization JSONObjectWithData:data
options:NSJSONReadingAllowFragments error:&err];

    if (!err){ // if no error occurred, parse the array of objects as normal
        // Parse the JSON dictionary 'jsonDict' here
    }else{ // an error occurred so we need to let the user know
        // Handle your error here
    }
}

// Error handling delegate
- (void)URLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didCompleteWithError:(NSError *)error{
    if(error == nil){
        // Download from API was successful
        NSLog(@"Data Network Request Did Complete Successfully.");
    }else{
        // Describes and logs the error preventing us from receiving a response
        NSLog(@"Error: %@", [error userInfo]);

        // Handle network error, letting the user know what happened.
    }
}

// When the session receives a challenge (because of iOS 9 App Transport Security blocking
non-valid SSL certificates) we use the following methods to tell NSURLSession "Chill out, I
can trust me".

```

```

// The following is not necessary unless your server is using HTTP, not HTTPS

- (void)NSURLSession:(NSURLSession *)session didReceiveChallenge:(NSURLAuthenticationChallenge
*)challenge completionHandler:(void (^)(NSURLSessionAuthChallengeDisposition, NSURLCredential
*)completionHandler){
    if([challenge.protectionSpace.authenticationMethod
isEqualToString:NSURLAuthenticationMethodServerTrust]){
        if([challenge.protectionSpace.host isEqualToString:@"DomainNameOfServer.io"]){
            NSURLCredential * credential = [NSURLCredential
credentialForTrust:challenge.protectionSpace.serverTrust];
            completionHandler(NSURLSessionAuthChallengeUseCredential,credential);
        }
    }
}

- (void)NSURLSession:(NSURLSession *)session task:(NSURLSessionTask *)task
didReceiveChallenge:(NSURLAuthenticationChallenge *)challenge completionHandler:(void
(^)(NSURLSessionAuthChallengeDisposition, NSURLCredential * _Nullable))completionHandler{
    if([challenge.protectionSpace.authenticationMethod
isEqualToString:NSURLAuthenticationMethodServerTrust]){
        if([challenge.protectionSpace.host isEqualToString:@"DomainNameOfServer.io"]){
            NSURLCredential * credential = [NSURLCredential
credentialForTrust:challenge.protectionSpace.serverTrust];
            completionHandler(NSURLSessionAuthChallengeUseCredential,credential);
        }
    }
}

```

¡Eso es todo! ¡Es todo el código que necesita para enviar, recibir y analizar una solicitud de API en iOS 9! Está bien ... era una especie de código. Pero si se implementa como en el caso anterior, ¡será a prueba de fallas! Asegúrese de manejar siempre los errores donde se sugiere anteriormente.

Enviando una solicitud utilizando codificación de formulario

La codificación de URL es una forma ampliamente compatible de codificar datos arbitrarios. Sin embargo, es relativamente ineficiente para cargar datos binarios (como fotos) porque cada byte no ASCII se convierte en un código de tres caracteres. Tampoco admite archivos adjuntos, por lo que tendría que pasar los nombres de archivo y los datos del archivo como campos separados.

Supongamos que queremos cargar una fotografía de una manera eficiente y que se vea como un archivo en el lado del servidor. Una forma de hacerlo es utilizar la codificación de formularios en su lugar. Para hacer esto, edite el código que crea la NSURLSession de la siguiente manera:

```

UIImage * imgToSend;

// 2nd parameter of UIImageJPEGRepresentation represents compression quality. 0 being most
compressed, 1 being the least
// Using 0.4 likely stops us hitting the servers upload limit and costs us less server space
NSData * imageData = UIImageJPEGRepresentation(imgToSend, 0.4f);

// Alternatively, if the photo is on disk, you can retrieve it with
// [NSData dataWithContentsOfURL:...]
```

// Set up the body of the POST request.


```

// This boundary serves as a separator between one form field and the next.
// It must not appear anywhere within the actual data that you intend to
// upload.
NSString * boundary = @"-----14737809831466499882746641449";

// Body of the POST method
NSMutableData * body = [NSMutableData data];

// The body must start with the boundary preceded by two hyphens, followed
// by a carriage return and newline pair.
//
// Notice that we prepend two additional hyphens to the boundary when
// we actually use it as part of the body data.
//
[body appendData:[NSString stringWithFormat:@"\r\n--%\r\n",boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

// This is followed by a series of headers for the first field and then
// TWO CR-LF pairs.
[body appendData:[NSString stringWithFormat:@"Content-Disposition: form-data;
name=\"%tag_name%\r\n\r\n"] dataUsingEncoding:NSUTF8StringEncoding]];

// Next is the actual data for that field (called "tag_name") followed by
// a CR-LF pair, a boundary, and another CR-LF pair.
[body appendData:[strippedCompanyName dataUsingEncoding:NSUTF8StringEncoding]];
[body appendData:[NSString stringWithFormat:@"\r\n--%\r\n", boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

// Encode the filename and image data as the "userfile" CGI parameter.
// This is similar to the previous field, except that it is being sent
// as an actual file attachment rather than a blob of data, which means
// it has both a filename and the actual file contents.
//
// IMPORTANT: The filename MUST be plain ASCII (and if encoded like this,
// must not include quotation marks in the filename).
//
NSString * picFileName = [NSString stringWithFormat:@"photoName"];
NSString * appendDataString = [NSString stringWithFormat:@"Content-Disposition: form-data;
name=\"userfile\"; filename=\"%@.jpg\"\r\n", picFileName];
[body appendData:[appendDataString dataUsingEncoding:NSUTF8StringEncoding]];
[body appendData:[@"Content-Type: application/octet-stream\r\n\r\n"
dataUsingEncoding:NSUTF8StringEncoding]];
[body appendData:[NSData dataWithData:imageData]];

// Close the request body with one last boundary with two
// additional hyphens prepended **and** two additional hyphens appended.
[body appendData:[NSString stringWithFormat:@"\r\n--%--\r\n", boundary]
dataUsingEncoding:NSUTF8StringEncoding]];

// Create the session
// We can use the delegate to track upload progress and disable cacheing
NSURLSessionConfiguration * defaultConfigObject = [NSURLSessionConfiguration
defaultSessionConfiguration];
defaultConfigObject.requestCachePolicy = NSURLRequestReloadIgnoringLocalCacheData;
NSURLSession * defaultSession = [NSURLSession sessionWithConfiguration: defaultConfigObject
delegate: self delegateQueue: [NSOperationQueue mainQueue]];

// Data uploading task.
NSURL * url = [NSURL URLWithString:@"https://server.io/api/script.php"];
NSMutableURLRequest * request = [NSMutableURLRequest requestWithURL:url];
NSString * contentType = [NSString stringWithFormat:@"multipart/form-data;

```

```
boundary=%@",boundary];
[request addValue:contentType forHTTPHeaderField:@"Content-Type"];
request.HTTPMethod = @"POST";
request.HTTPBody = body;
NSURLSessionDataTask * uploadTask = [defaultSession dataTaskWithRequest:request];
[uploadTask resume];
```

Esto crea y dispara la solicitud `NSURLSession` como antes, y como resultado, los métodos de delegado se comportarán exactamente de la misma manera. Asegúrese de que la secuencia de comandos a la que se envía la imagen (ubicada en la url en la variable `url`) esté esperando una imagen y pueda analizarla correctamente.

Lea **SESIÓN** en línea: <https://riptutorial.com/es/ios/topic/2009/sesion>

Capítulo 141: Simulador

Introducción

Los simuladores de iOS, watchOS y tvOS son excelentes maneras de probar sus aplicaciones sin usar un dispositivo real. Aquí hablaremos de trabajar con simuladores.

Observaciones

Diferentes tipos de simuladores

- simulador de iOS
- simulador de watchOS
- simulador de tvOS
- Simulador de barra táctil

No existe un simulador para macOS, ya que Xcode se ejecuta en macOS y, cuando sea necesario, ejecutará las aplicaciones nativas.

Obteniendo ayuda

Siempre puede visitar la ayuda del simulador en Ayuda -> Ayuda del simulador:



Xcode

File

Edit

View

Find

Navigate



- ▶ Swift
- ▶ Objective-C
- ▶ JavaScript

Abc

Impo
chang

Simulat
Simulat
of the s

Simulat
simulat
These s

Capítulo 142: Simulador construye

Introducción

¿Dónde encontrar la construcción del simulador?

Vaya a ~ / Biblioteca / Desarrollador / CoreSimulator / Dispositivos /

Encontrarás directorios con nombres alfanuméricos.

Luego haga clic en uno de los directorios y haga la siguiente selección.

Datos / Contenedores / Bundle / Aplicación /

Nuevamente encontrará directorios con nombres alfanuméricos si hace clic en el que encontrará Simulador construido por allí

Nota:

Instalar el dispositivo iOS construido en el simulador no funcionará.

simulador de iPhone utiliza la arquitectura i386 simulador de iPad utiliza x8

Examples

Instalando la compilación manualmente en el simulador

```
xcrun simctl install booted *.app
```

Lea Simulador construye en línea: <https://riptutorial.com/es/ios/topic/9813/simulador-construye>

Capítulo 143: Simulando Ubicación Usando archivos GPX iOS

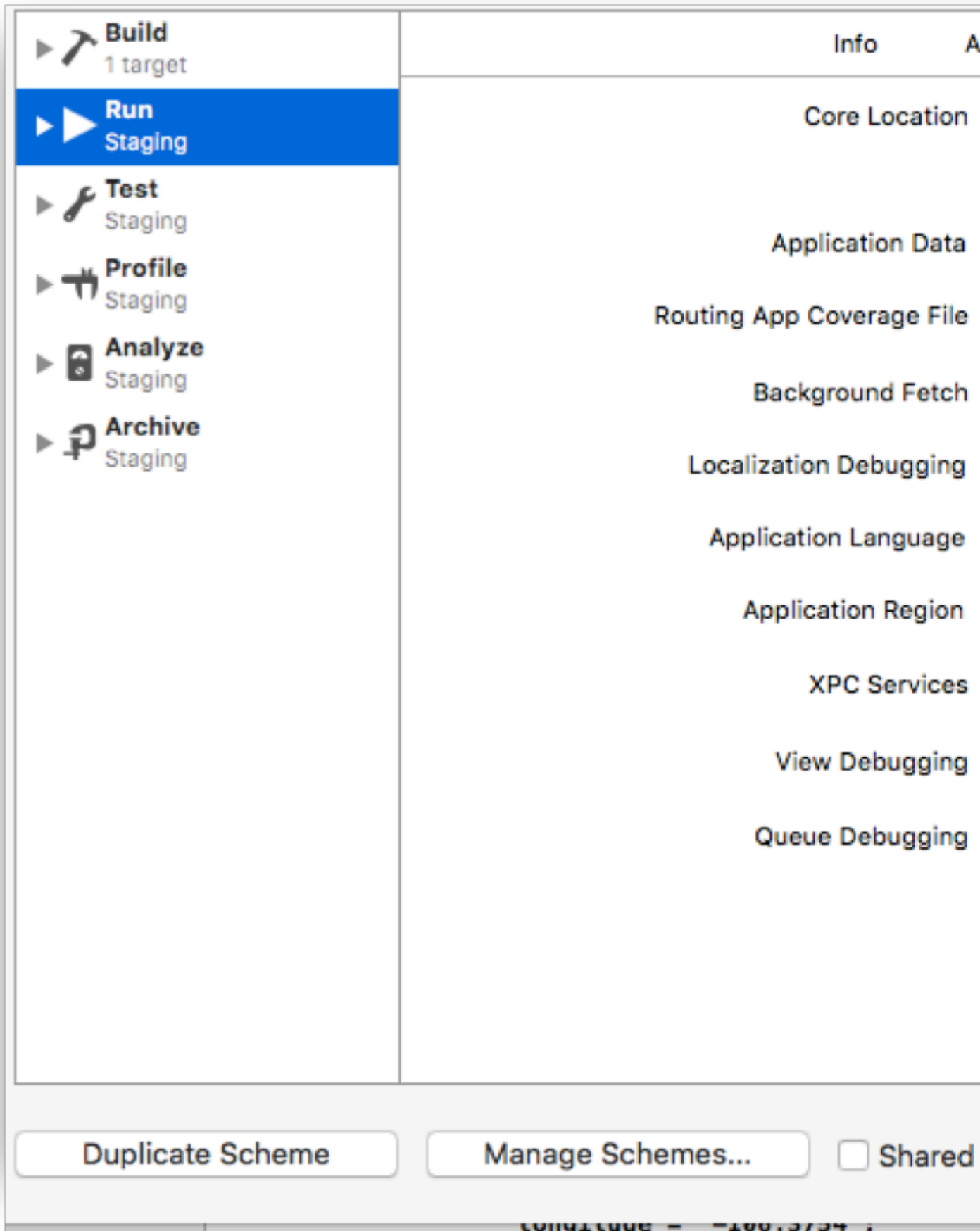
Examples

Su archivo .gpx: MPS_HQ.gpx

```
<gpx xmlns="http://www.topografix.com/GPX/1/1"
  xmlns:gpxx = "http://www.garmin.com/xmlschemas/GpxExtensions/v3"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/1
  http://www.topografix.com/GPX/1/1/gpx.xsd
  http://www.garmin.com/xmlschemas/GpxExtensions/v3
  http://www8.garmin.com/xmlschemas/GpxExtensions/v3/GpxExtensionsv3.xsd"
  version="1.1"
  creator="gpx-poi.com">
<wpt lat="38.9072" lon="77.0369">38.9072/-77.0369
<time>2015-04-16T22:20:29Z</time>
  <name>Washington, DC</name>
  <extensions>
    <gpxx:WaypointExtension>
      <gpxx:Proximity>10</gpxx:Proximity>
      <gpxx:Address>
        <gpxx:StreetAddress>Washington DC</gpxx:StreetAddress>
        <gpxx:City>Washington</gpxx:City>
        <gpxx:State>DC</gpxx:State>
        <gpxx:Country>United States</gpxx:Country>
        <gpxx:PostalCode> 20005 </gpxx:PostalCode>
      </gpxx:Address>
    </gpxx:WaypointExtension>
  </extensions>
```

Para establecer esta ubicación:

1. Ir al esquema de edición.
2. Seleccione Ejecutar -> Opciones.
3. Marque "Permitir simulación de ubicación".
4. Seleccione el nombre del archivo * .GPX en la lista desplegable "Ubicación predeterminada".



Lea Simulando Ubicación Usando archivos GPX iOS en línea:

<https://riptutorial.com/es/ios/topic/9883/simulando-ubicacion-usando-archivos-gpx-ios>

Capítulo 144: Sintetizador AVSpeech

Sintaxis

- AVSpeechSynthesizer () // Crea un sintetizador de voz
- speaker.speakUtterance (voz) // Convierte el texto a voz

Parámetros

Parámetro	Detalles
altavoz	Objeto AVSpeechSynthesizer
habla	Objeto AVSpeechUtterance

Examples

Creación de un texto básico a voz.

Use el método `speakUtterance:` de `AVSpeechSynthesizer` para convertir texto a voz. `AVSpeechUtterance` pasar un objeto `AVSpeechUtterance` a este método, que contiene el texto que desea que se pronuncie.

C objetivo

```
AVSpeechSynthesizer *speaker = [[AVSpeechSynthesizer alloc] init];
AVSpeechUtterance *speech     = [AVSpeechUtterance speechUtteranceWithString:@"Hello World"];
[speaker speakUtterance:speech];
```

Rápido

```
let speaker = AVSpeechSynthesizer()
let speech = AVSpeechUtterance(string: "Hello World")
speaker.speakUtterance(speech)
```

Lea Sintetizador AVSpeech en línea: <https://riptutorial.com/es/ios/topic/1526/sintetizador-avspeech>

Capítulo 145: SiriKit

Observaciones

Diferentes tipos de peticiones Siri.

- Reserva de viajes (p. Ej., Llévame a Nueva York a través de MyApp)
- Mensajería (p. Ej., Enviar un texto a John usando MyApp)
- Búsqueda de fotos (por ejemplo, busque las fotos de playa tomadas el verano pasado en MyApp)
- Pagos (por ejemplo, envíe \$ 20 a John para la cena de anoche usando MyApp)
- Llamadas VoIP (por ejemplo, llame a Mike en mi MyApp)
- Entrenamientos (por ejemplo, iniciar mi entrenamiento diario desde MyApp)
- Clima y radio (diseñado específicamente para CarPlay, por ejemplo, ajuste el calentador a 72 grados)

Examples

Añadiendo la extensión de Siri a la aplicación

Para integrar las capacidades de Siri en su aplicación, debe agregar una extensión como lo haría al crear un Widget de iOS 10 (antigua Extensión de Vista Hoy) o un teclado personalizado.

Añadiendo capacidad

1- En la configuración del proyecto, seleccione el destino de su aplicación iOS y vaya a la pestaña Capacidades

2- Habilitar la capacidad de Siri

Añadiendo la extensión

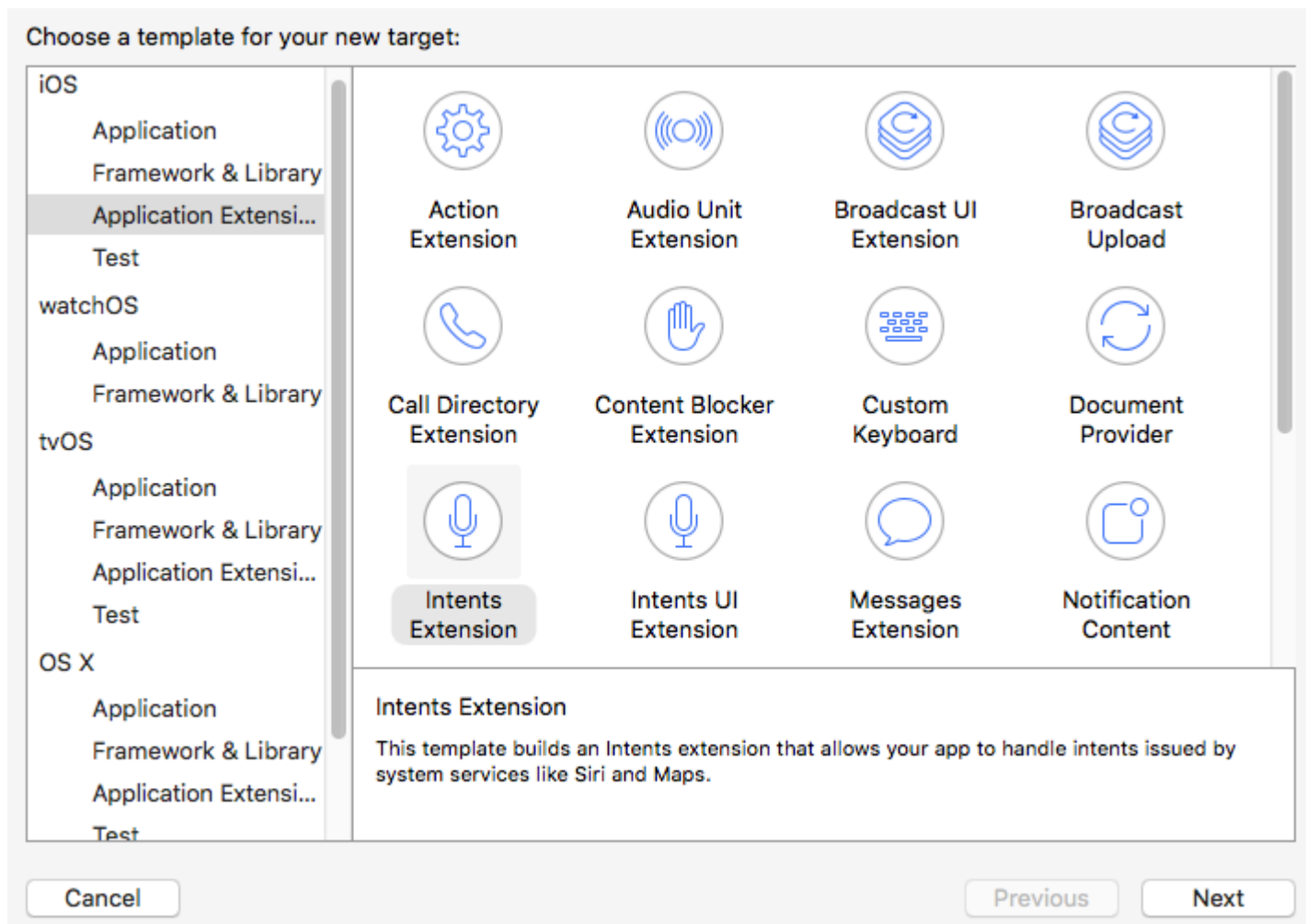
1- Ir a Archivo -> Nuevo -> Objetivo ...

2- Selecciona iOS -> Extensión de aplicación en el panel izquierdo

3- Haz doble clic en Intents Extension desde la derecha

Según Apple:

La plantilla de extensión de intenciones crea una extensión de intenciones que le permite a su aplicación manejar intenciones emitidas por servicios del sistema como Siri y Maps.



4- Elija un nombre y asegúrese de marcar "Incluir extensión de UI"

Language:

Include UI Extension

Al realizar estos pasos, se crean dos nuevos objetivos (la Extensión de intenciones y la Extensión de la interfaz de usuario) y, de forma predeterminada, contienen el Código de Intención de entrenamiento. Para diferentes tipos de solicitudes de Siri, vea Observaciones.

Nota

Cuando quiera depurar su extensión, simplemente seleccione el esquema de Intención de los esquemas disponibles.

Nota

No puedes probar las aplicaciones SiriKit en el simulador. En su lugar, necesita un dispositivo real.

Lea SiriKit en línea: <https://riptutorial.com/es/ios/topic/5869/sirikit>

Capítulo 146: SLComposeViewController

Examples

SLComposeViewController para Twitter, facebook, SinaWeibo y TencentWeibo

C objetivo

Primero agregue el `Social Framework` al proyecto XCode.

Importe la clase `#import "Social/Social.h"` al ViewController requerido

Twitter con texto, imagen y enlace.

```
//- - To Share text on twitter - -
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeTwitter])
{
    //Tweet
    SLComposeViewController *twitterVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTwitter];
    //To send link together with text
    [twitterVC addURL:[NSURL URLWithString:@"https://twitter.com/IbrahimH_ss_n"]];
    //To add a photo to a link
    [twitterVC addImage:[UIImage imageNamed:@"image"]];
    //Sending link and Image with the tweet
    [twitterVC setInitialText:text];
    /* While adding link and images in a tweet the effective length of a tweet i.e.
the number of characters which can be entered by the user decreases.
The default maximum length of a tweet is 140 characters*/
    [self presentViewController:twitterVC animated:YES completion:nil];
}
else
{
    //Shows alert if twitter is not signed in
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to twitter."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay"
style:UIAlertActionStyleDefault handler:nil];
    [alertCont addAction:okay];
}
}
```

Facebook con texto, imagen y enlace

```
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeFacebook])
{
    SLComposeViewController *fbVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeFacebook];
    [fbVC setInitialText:text];
    //To send link together with text
    [fbVC addURL:[NSURL URLWithString:@"https://twitter.com/IbrahimH_ss_n"]];
    //To add a photo to a link
```

```

    [fbVC addImage:[UIImage imageNamed:@"image"]];
    [self presentViewController:fbVC animated:YES completion:nil];
}
else
{//Shows alert if twitter is not signed in
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to twitter."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay"
style:UIAlertActionStyleDefault handler:nil];
    [alertCont addAction:okay];
}
}

```

SinaWeibo

```

// - - SinaWeibo - -
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeSinaWeibo]){

    SLComposeViewController *SinaWeiboVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeSinaWeibo];
    [SinaWeiboVC setInitialText:text];

    [self presentViewController:SinaWeiboVC animated:YES completion:nil];
}
else
{
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to SinaWeibo."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay" style:UIAlertActionStyleDefault
handler:nil];
    [alertCont addAction:okay];
}
}

```

TencentWeibo

```

// - -TencentWeibo text share
if([SLComposeViewController isAvailableForServiceType:SLServiceTypeTencentWeibo])
{
    SLComposeViewController *tencentWeiboVC=[SLComposeViewController
composeViewControllerForServiceType:SLServiceTypeTencentWeibo];
    [tencentWeibo setInitialText:text];
    [self presentViewController:tencentWeibo animated:YES completion:nil];
}
else
{
    UIAlertController *alertCont=[UIAlertController alertControllerWithTitle:@"SocialShare"
message:@"You are not signed in to SinaWeibo."preferredStyle:UIAlertControllerStyleAlert];
    [self presentViewController:alertCont animated:YES completion:nil];
    UIAlertAction *okay=[UIAlertAction actionWithTitle:@"Okay" style:UIAlertActionStyleDefault
handler:nil];
    [alertCont addAction:okay];
}
}

```

Lea [SLComposeViewController](https://riptutorial.com/es/ios/topic/7366/slcomposeviewController) en línea:

<https://riptutorial.com/es/ios/topic/7366/slcomposeviewController>

Capítulo 147: StoreKit

Examples

Obtenga información sobre productos localizados en la App Store

Obtenga información de productos localizada de un conjunto de cadenas de identificador de productos usando `SKProductsRequest` :

```
import StoreKit

let productIdentifierSet = Set(["yellowSubmarine", "pennyLane"])
let productsRequest = SKProductsRequest(productIdentifiers: productIdentifierSet)
```

Para procesar los productos de la solicitud de `productsRequest` , debemos asignar un delegado a la solicitud que maneja la respuesta. El delegado debe cumplir con el protocolo `SKProductsRequestDelegate` , lo que significa que debe heredar de `NSObject` (es decir, cualquier objeto `Foundation`) e implementar el método `productsRequest` :

```
class PaymentManager: NSObject, SKProductsRequestDelegate {

    var products: [SKProduct] = []

    func productsRequest(request: SKProductsRequest,
                        didReceiveResponse response: SKProductsResponse) {

        products = response.products

    }

}
```

Para iniciar `productsRequest` , asignamos `PaymentManager` como delegado de la solicitud de productos y llamamos al método `start()` en la solicitud:

```
let paymentManager = PaymentManager()
productsRequest.delegate = paymentManager
productsRequest.start()
```

Si las solicitudes tienen éxito, los productos estarán en `paymentManager.products` .

Lea StoreKit en línea: <https://riptutorial.com/es/ios/topic/6025/storekit>

Capítulo 148: Swift: cambiando el control rootViewController en AppDelegate para presentar el flujo principal o de inicio de sesión / incorporación

Introducción

A menudo es útil presentar una experiencia de primera ejecución a los nuevos usuarios de su aplicación. Esto podría deberse a varios motivos, como pedirles que inicien sesión (si es necesario para su situación), explicar cómo usar la aplicación o simplemente informarles sobre nuevas funciones en una actualización (como lo hacen Notas, Fotos y Música). iOS11).

Observaciones

En primer lugar, cuando se trata de flujos múltiples, aquí es donde los guiones gráficos se pueden usar de manera efectiva. Por defecto, su aplicación utiliza `Main.storyboard` para su flujo primario. Su flujo de incorporación / alternativa puede estar contenido en un guión gráfico secundario, por ejemplo. `Onboarding.storyboard`

Esto tiene una serie de ventajas:

- en un equipo de desarrolladores, el trabajo en cada flujo de usuario se puede separar
- control de fuente más claro (git)
- separación de intereses

Cuando se inicie su aplicación, puede determinar qué flujo debe presentarse. La lógica para esto puede estar contenida en su AppDelegate:

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    let isFirstRun = true // logic to determine goes here
    if isFirstRun {
        showOnboarding()
    }
    return true
}
```

Para mostrar el flujo de Onboarding, vale la pena considerar cómo le gustaría manejar la experiencia de descartarlo una vez que la persona que lo utiliza ha completado el viaje, y que es semánticamente correcto para lo que está tratando de crear.

Enfoques:

Los dos enfoques principales son:

1. Intercambia el controlador de vista raíz de la ventana principal de la aplicación
2. Presente el flujo de Onboarding como un viaje modal, superponiendo el flujo principal.

La implementación de esto debe estar contenida en una extensión de AppDelegate.

Examples

Opción 1: intercambiar el controlador de vista de raíz (bueno)

El cambio del controlador de vista de raíz tiene ventajas, aunque las opciones de transición están limitadas a las admitidas por `UIViewAnimationOptions`, por lo que, dependiendo de cómo desee hacer la transición entre flujos, puede significar que tiene que implementar una transición personalizada, lo que puede ser engorroso.

Puede mostrar el flujo de Onboarding simplemente configurando

```
UIApplication.shared.keyWindow.rootViewController
```

El despido se maneja utilizando `UIView.transition(with:)` y pasando el estilo de transición como `UIViewAnimationOptions`, en este caso, `Cross Dissolve`. (También se admiten tirones y rizos).

También debe establecer el marco de la vista Principal antes de volver a la misma, ya que lo está creando por primera vez.

```
// MARK: - Onboarding

extension AppDelegate {

    func showOnboarding() {
        if let window = UIApplication.shared.keyWindow, let onboardingViewController =
        UIStoryboard(name: "Onboarding", bundle: nil).instantiateInitialViewController() as?
        OnboardingViewController {
            onboardingViewController.delegate = self
            window.rootViewController = onboardingViewController
        }
    }

    func hideOnboarding() {
        if let window = UIApplication.shared.keyWindow, let mainViewController =
        UIStoryboard(name: "Main", bundle: nil).instantiateInitialViewController() {
            mainViewController.view.frame = window.bounds
            UIView.transition(with: window, duration: 0.5, options: .transitionCrossDissolve,
            animations: {
                window.rootViewController = mainViewController
            }, completion: nil)
        }
    }
}
```

Opción 2: Presentar el flujo alternativo de manera modal (mejor)

En la implementación más sencilla, el flujo de Onboarding se puede presentar simplemente en un

contexto modal, ya que semánticamente el Usuario está en un solo viaje.

[Directrices de la interfaz humana de Apple - Modalidad] [1]:

Considere crear un contexto modal solo cuando sea fundamental captar la atención de alguien, cuándo debe completarse o abandonarse una tarea para continuar usando la aplicación o para guardar información importante.

La presentación moderada permite la opción simple de despido al final del viaje, con poco del recorrido de los controladores de intercambio.

Las transiciones personalizadas también se admiten de forma estándar, ya que utiliza la API

`ViewController.present()` :

```
// MARK: - Onboarding

extension AppDelegate {

    func showOnboarding() {
        if let window = window, let onboardingViewController = UIStoryboard(name:
"Onboarding", bundle: nil).instantiateInitialViewController() as? OnboardingViewController {
            onboardingViewController.delegate = self
            window.makeKeyAndVisible()
            window.rootViewController?.present(onboardingViewController, animated: false,
completion: nil)
        }
    }

    func hideOnboarding() {
        if let window = UIApplication.shared.keyWindow {
            window.rootViewController?.dismiss(animated: true, completion: nil)
        }
    }
}
```

Lea Swift: cambiando el control `rootViewController` en `AppDelegate` para presentar el flujo principal o de inicio de sesión / incorporación en línea:

<https://riptutorial.com/es/ios/topic/10880/swift--cambiando-el-control-rootviewcontroller-en-appdelegate-para-presentar-el-flujo-principal-o-de-inicio-de-sesion---incorporacion>

Capítulo 149: SWRevealViewController

Observaciones

El uso de la clase `SWRevealViewController` como navegación principal no siempre resulta en la mejor experiencia de usuario. Si la barra lateral contiene solo 5 o menos entradas (o el contenido se puede comprimir en 5 o menos entradas), debe considerar el uso de la barra de pestañas predeterminada.

La barra de pestañas es intuitiva y permite al usuario cambiar rápidamente entre vistas / contextos. Por otro lado, la barra lateral de navegación puede realizar más acciones que cambiar la vista / contexto y usa menos espacio cuando se contrae.

Para obtener más información, consulte las [Pautas de interfaz humana de iOS](#) de Apple.

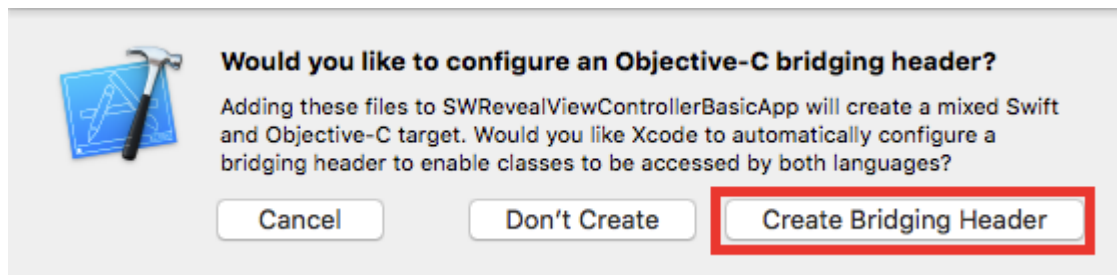
Examples

Configurando una aplicación básica con `SWRevealViewController`

Cree una aplicación básica con una plantilla de aplicación de vista única con swift como idioma

Agregue `SWRevealViewController.h` y `SWRevealViewController.m`

a continuación, haga clic en el botón Crear encabezado puente

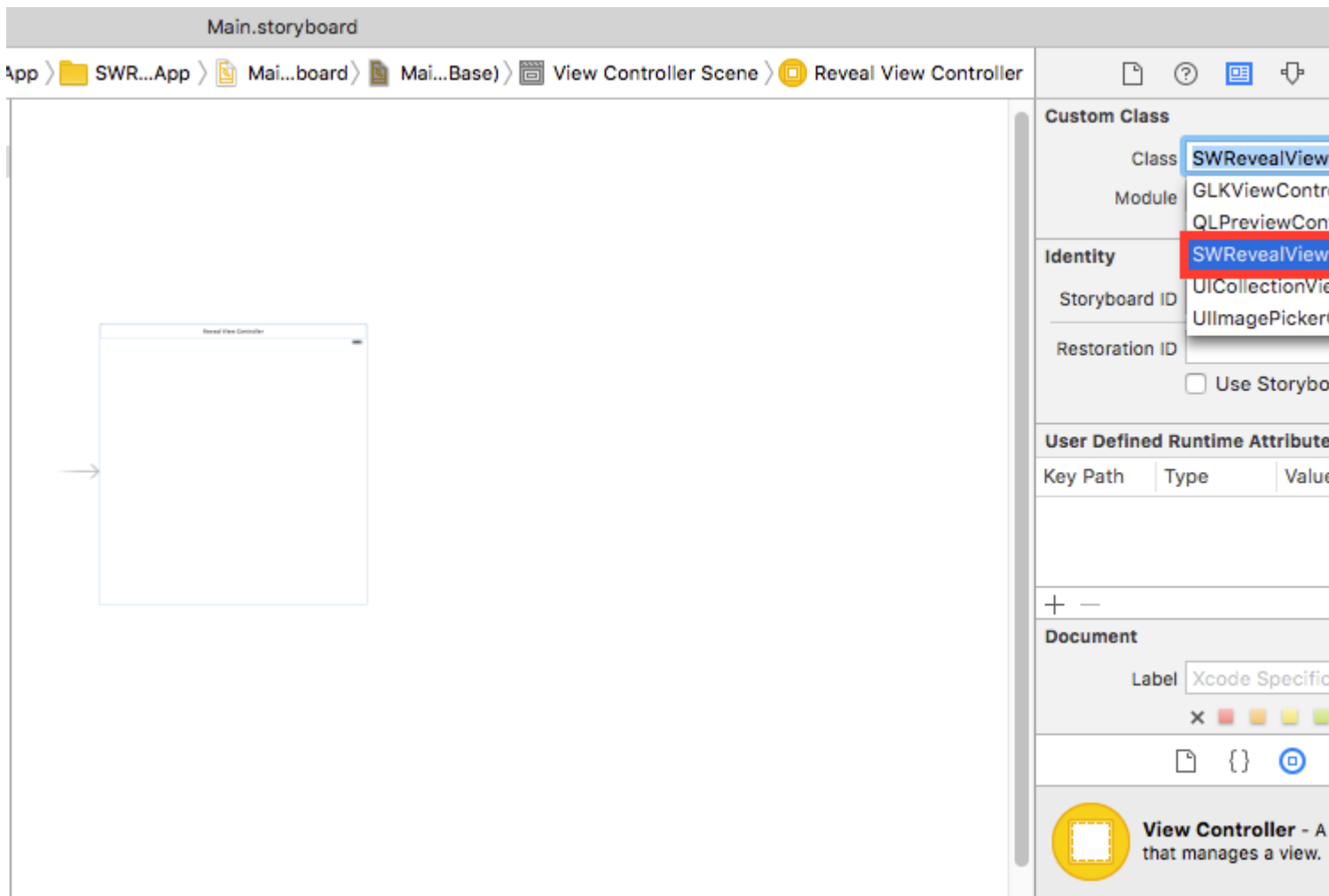


y añadir

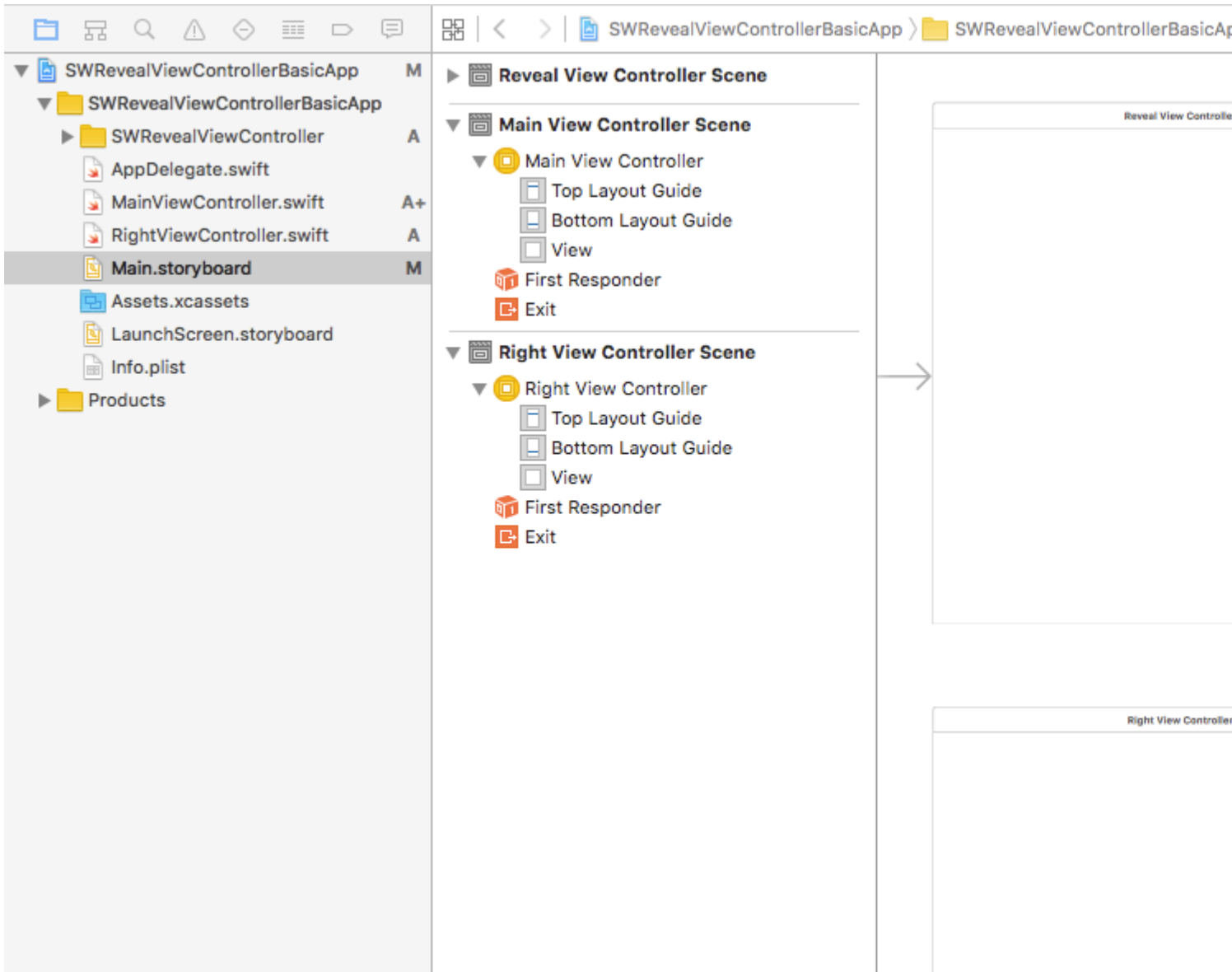
```
#import "SWRevealViewController.h"
```

en el encabezado puente

Luego seleccione `viewController` en el guión gráfico y cambie la clase a `SWRevealViewController`



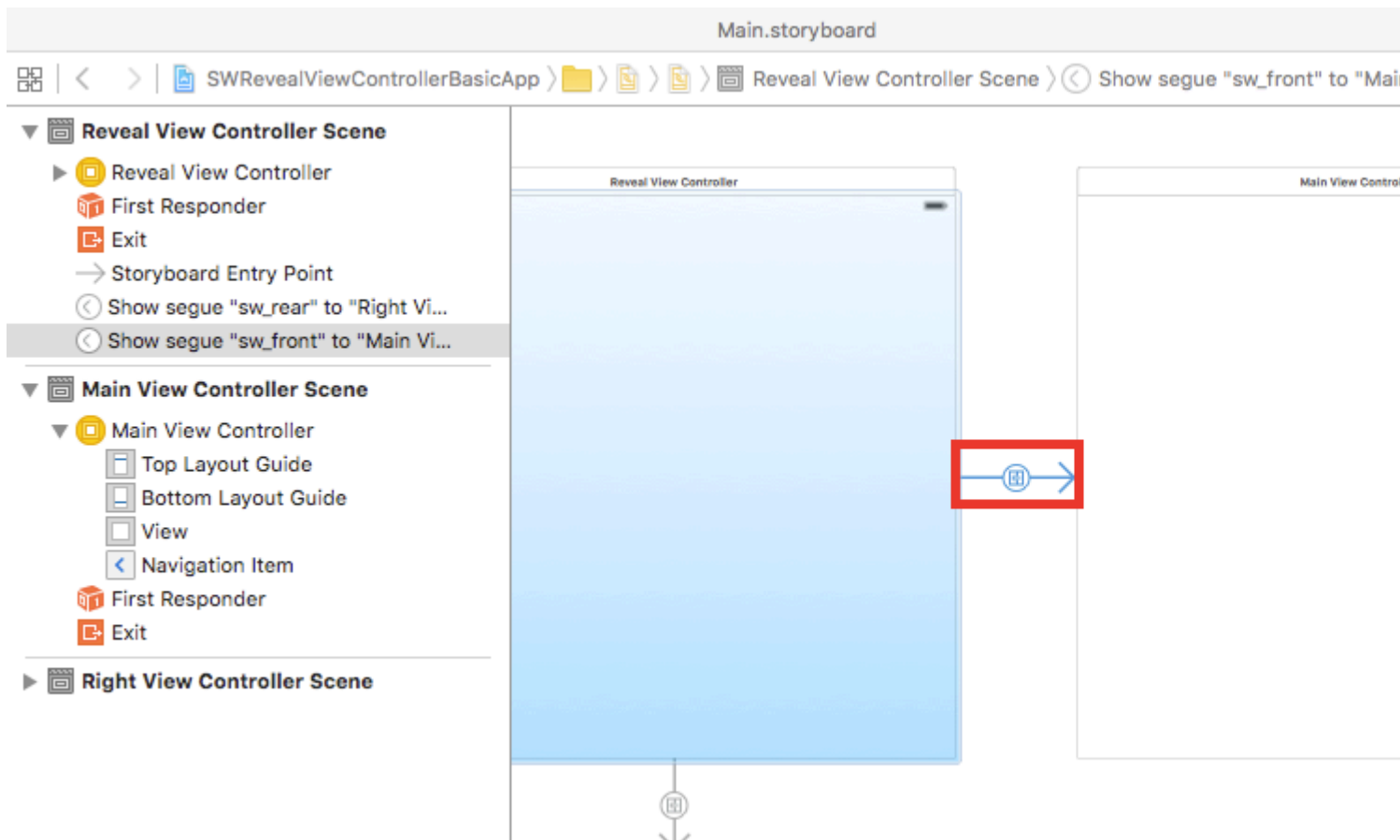
Luego cambie el nombre del viewController en los archivos a MainViewController y agregue el nuevo ViewController con el nombre de RightViewController



luego agregamos dos segmentos de SWRevealViewController a MainViewController y de SWRevealViewController a RightViewController, luego debemos seleccionar el primero (de SWRevealViewController a MainViewController) y editar las propiedades

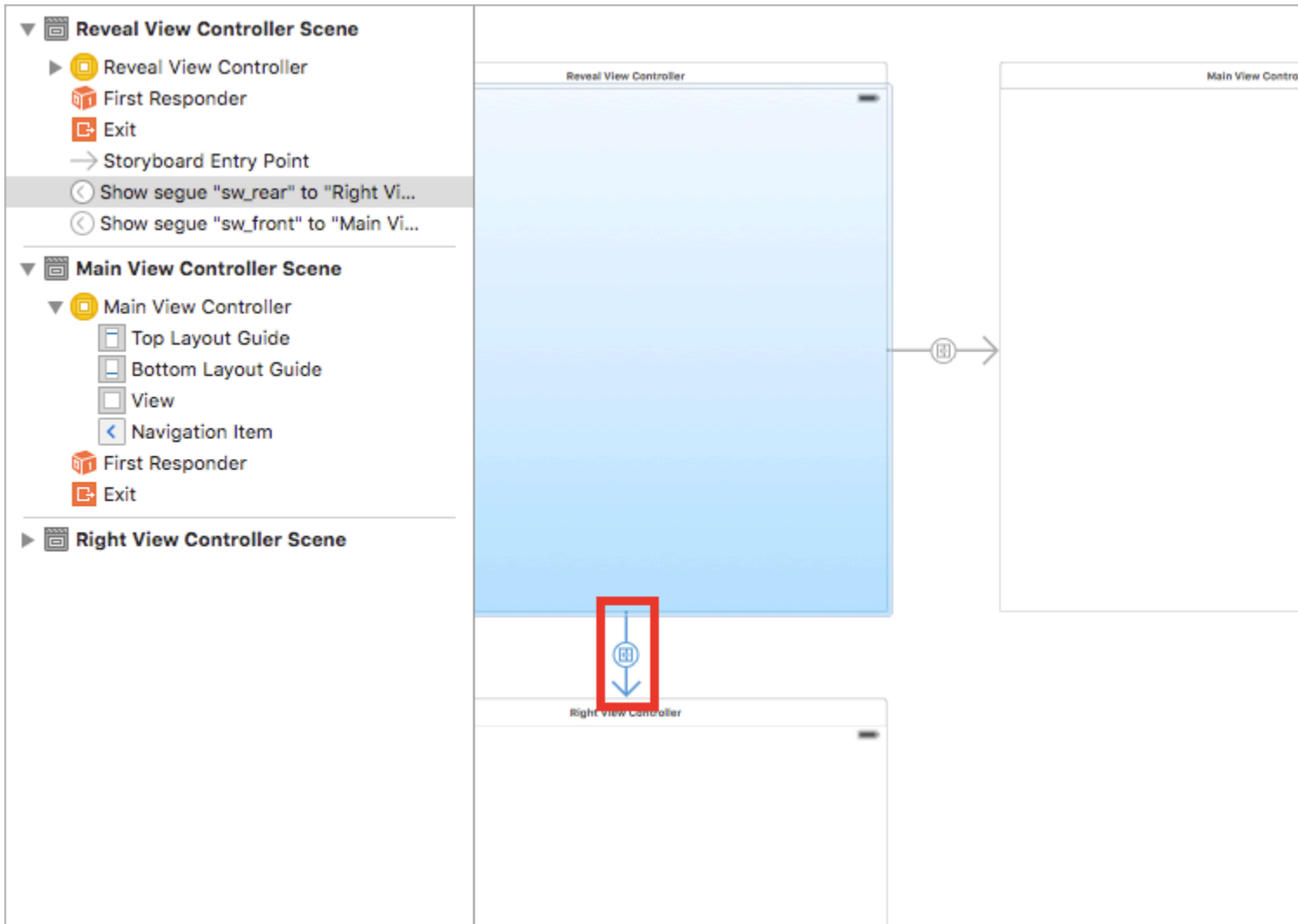
en el conjunto de identificadores `sw_front` en el conjunto de clases

`SWRevealViewControllerSegueSetController`



después de esto, debemos hacer lo mismo con segue (de SWRevealViewController a RightViewController)

en el conjunto de identificadores `sw_rear` en el conjunto de clases `SWRevealViewControllerSegueSetController`



luego en MainViewController agregue esta línea en el método `viewDidLoad`

```
self.view.addGestureRecognizer(self.revealViewController().panGestureRecognizer());
```

Y esto es todo, tiene una aplicación básica con `SWRevealViewController` integrado, puede deslizar hacia la derecha para mostrar `RightViewController` como menú lateral

Lea `SWRevealViewController` en línea:

<https://riptutorial.com/es/ios/topic/4614/swrevealviewController>

Capítulo 150: Tablas de clasificación de GameCenter

Examples

Tablas de clasificación de GameCenter

Requisitos previos:

1. Cuenta de desarrolladores de Apple
2. Configura las tablas de clasificación de GameCenter con iTunesConnect

Configuración de tablas de clasificación de GameCenter:

1. Inicia sesión en *iTunesConnect*
2. Ir a *Mis Aplicaciones* . Crea una aplicación para tu proyecto y luego ve a *Características* .
3. Haga clic en *Game Center*
4. Haga clic en el signo más junto a Tablas de clasificación.
5. Elija *Tabla de clasificación individual* para los tipos de *Tabla de clasificación*.
6. Cree un *nombre de referencia de la tabla de clasificación* para su referencia.
7. Cree una *ID de tabla de clasificación* para que su aplicación se refiera a cuando informe las puntuaciones.
8. Establecer formato de puntuación en *entero*
9. La sumisión del puntaje será la *mejor puntuación*
10. Haga clic en *Agregar idioma* y complete las entradas.

Copia tu `LeaderboardID` que hiciste y te permite ir a Xcode.

Trabajando con Xcode

Hay 4 funciones con las que trabajaremos.

1. Importando el framework y configurando los protocolos.
2. Comprobando si el usuario ha iniciado sesión en GameCenter
3. Reportar los puntajes a GameCenter
4. Viendo tablas de clasificación
5. Importar GameKit `import GameKit` **Protocolos** `GKGameCenterControllerDelegate`
6. Ahora queremos comprobar si el usuario ha iniciado sesión en GameCenter

```
func authenticateLocalPlayer() {  
  
    let localPlayer = GKLocalPlayer.localPlayer()  
  
}
```

```

localPlayer.authenticateHandler = { (viewController, error) -> Void in

    if viewController != nil {
        //If the user is not signed in to GameCenter, we make them sign in
        let vc:UIViewController = self.view!.window!.rootViewController!
        vc.presentViewController(viewController!, animated: true, completion: nil)

    } else {

        //Do something here if you want
    }
}
}

```

3. Ahora el usuario está utilizando la aplicación y, de repente, el usuario tiene una nueva puntuación más alta. Informamos la puntuación más alta llamando a la siguiente función.

La función de abajo hols 2 parámetros.

Identifier que se define como una cadena y se usa para ingresar su ID de tabla de líderes que creó en iTunesConnect.

score que se define como un Int, que será el puntaje que los usuarios enviarán a iTunesConnect

```

func saveHighScore(identifier:String, score:Int) {

    if GKLocalPlayer.localPlayer().authenticated {

        let scoreReporter = GKScore(leaderboardIdentifier: identifier)

        scoreReporter.value = Int64(score)

        let scoreArray:[GKScore] = [scoreReporter]

        GKScore.reportScores(scoreArray, withCompletionHandler: {
            error -> Void in

                if error != nil {
                    print("Error")
                } else {

                }

            })
    }
}
}

```

4. Ahora, si el usuario quiere ver tablas de clasificación, llame a la siguiente función

```

//This function will show GameCenter leaderboards and Achievements if you call this function.
func showGameCenter() {

    let gameCenterViewController = GKGameCenterViewController()
    gameCenterViewController.gameCenterDelegate = self

    let vc:UIViewController = self.view!.window!.rootViewController!
    vc.presentViewController(gameCenterViewController, animated: true, completion:nil)
}

```



```
}  
  
//This function closes gameCenter after showing.  
func gameCenterViewControllerDidFinish(gameCenterViewController:  
GKGameCenterViewController) {  
  
    gameCenterViewController.dismissViewControllerAnimated(true, completion: nil)  
    self.gameCenterAchievements.removeAll()  
  
}
```

Lea Tablas de clasificación de GameCenter en línea:

<https://riptutorial.com/es/ios/topic/6720/tablas-de-clasificacion-de-gamecenter>

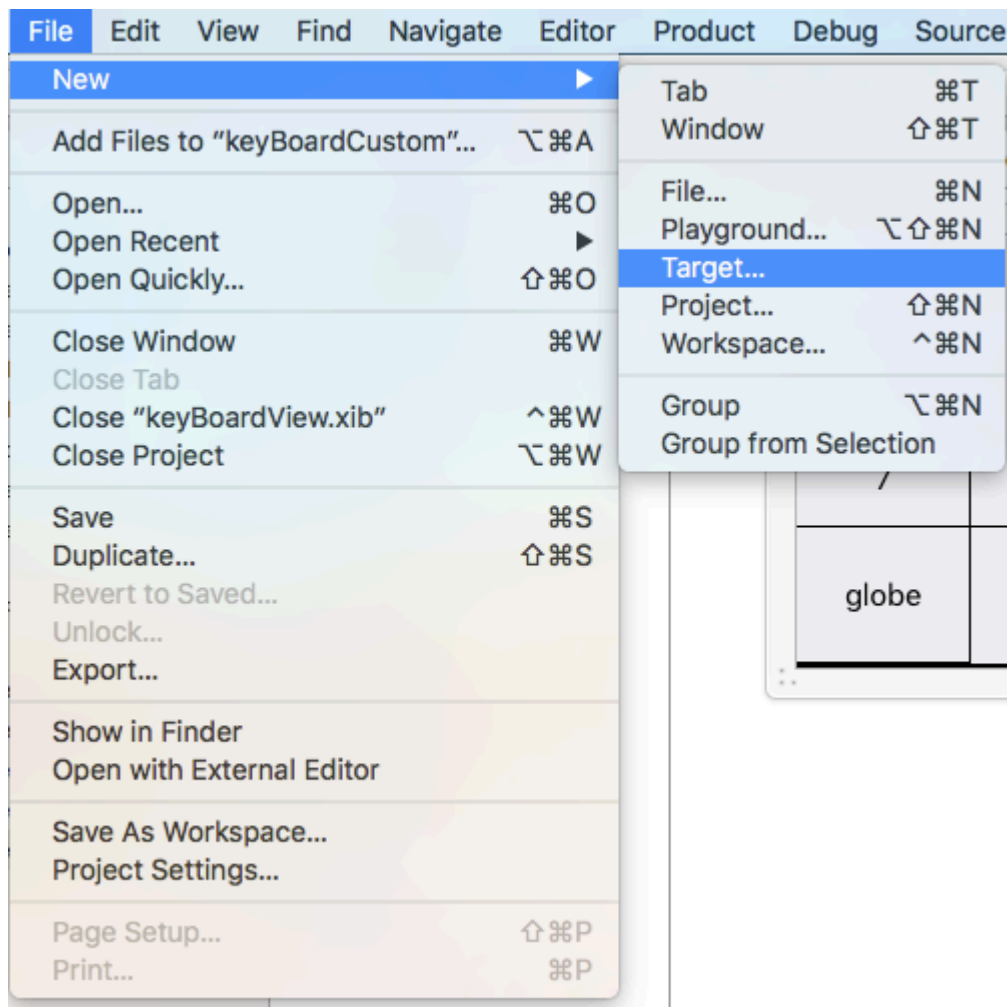
Capítulo 151: Teclado personalizado

Examples

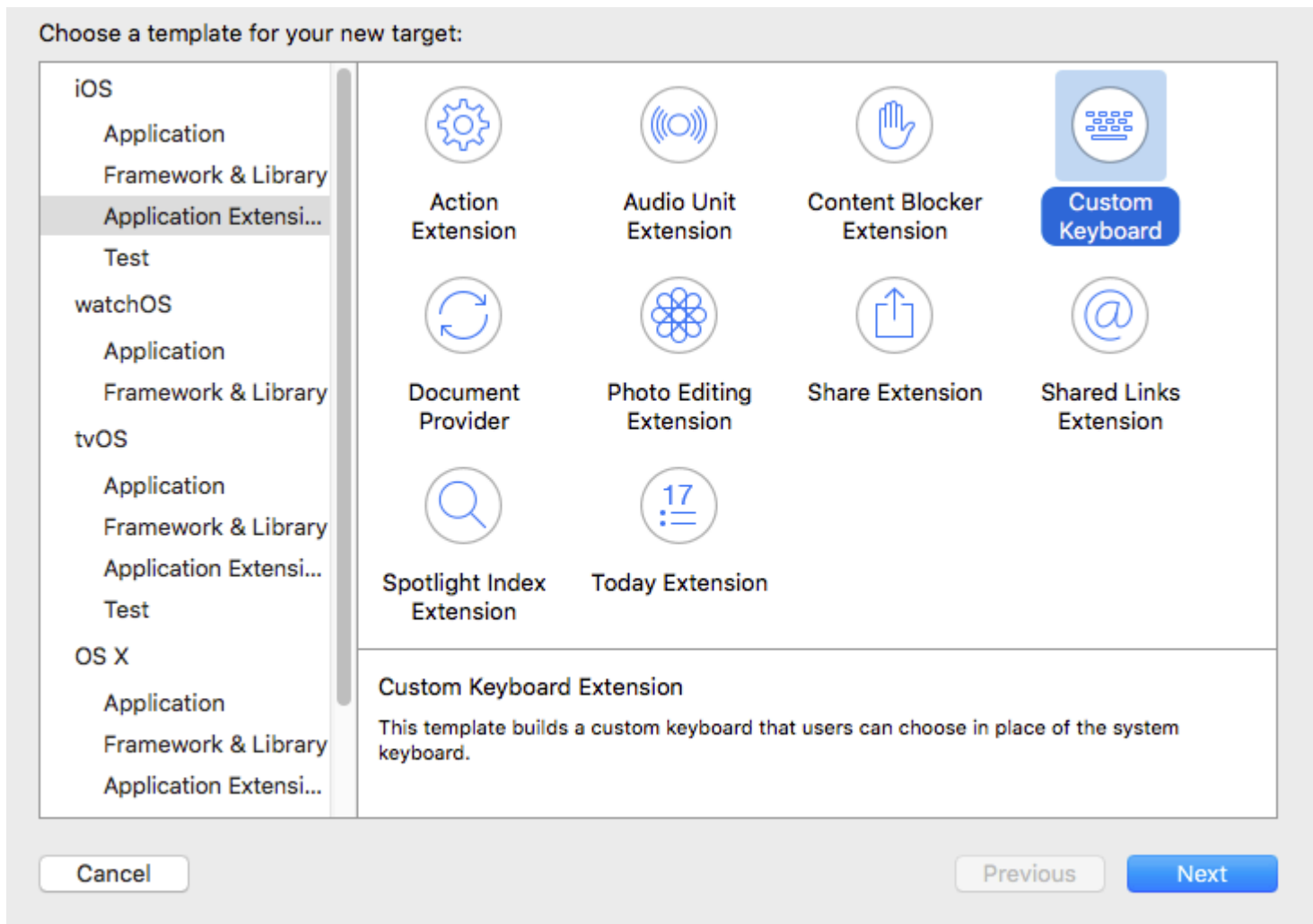
Ejemplo de teclado personalizado

Objective-C y Xib

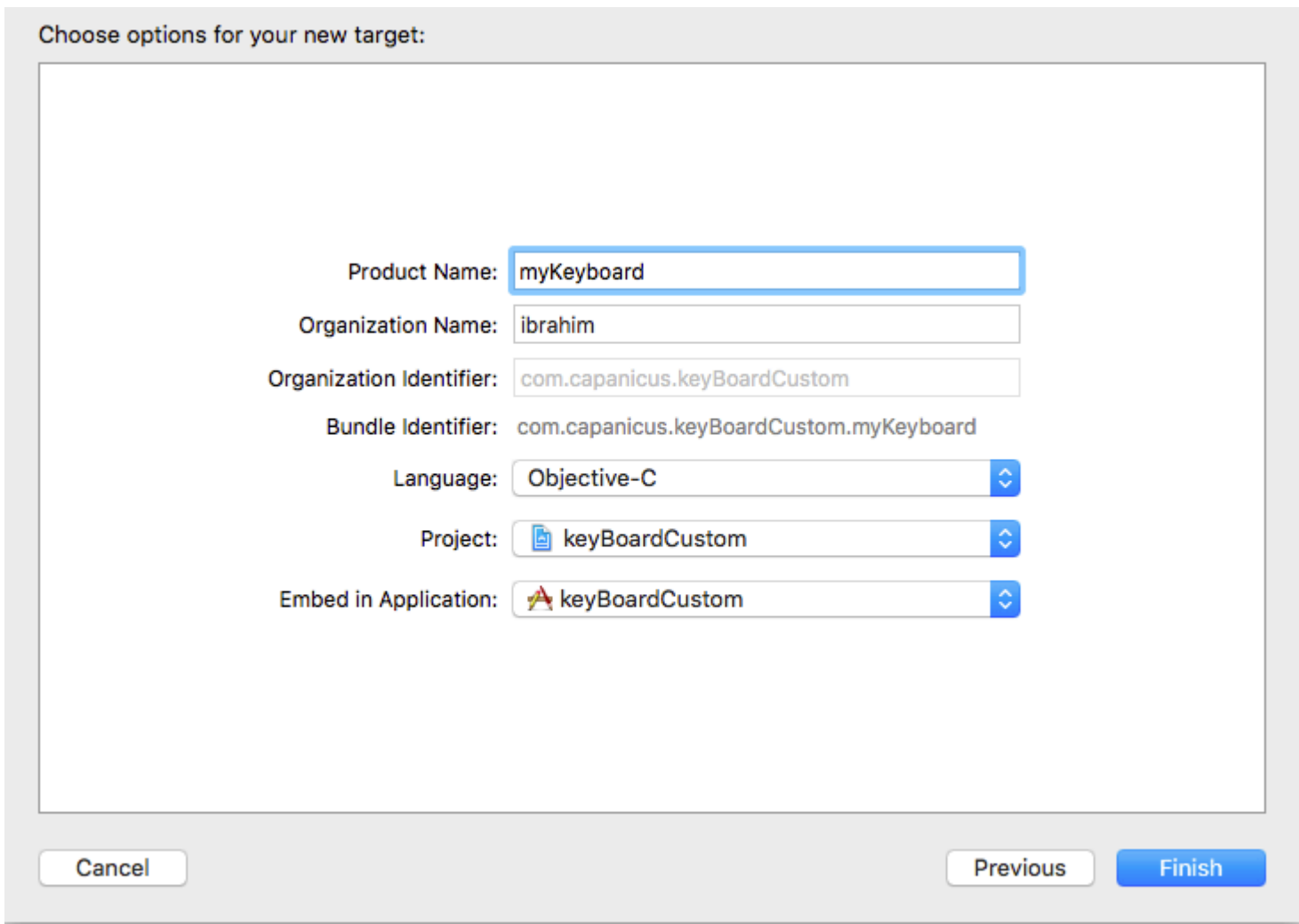
Agregar un objetivo a un proyecto XCode existente



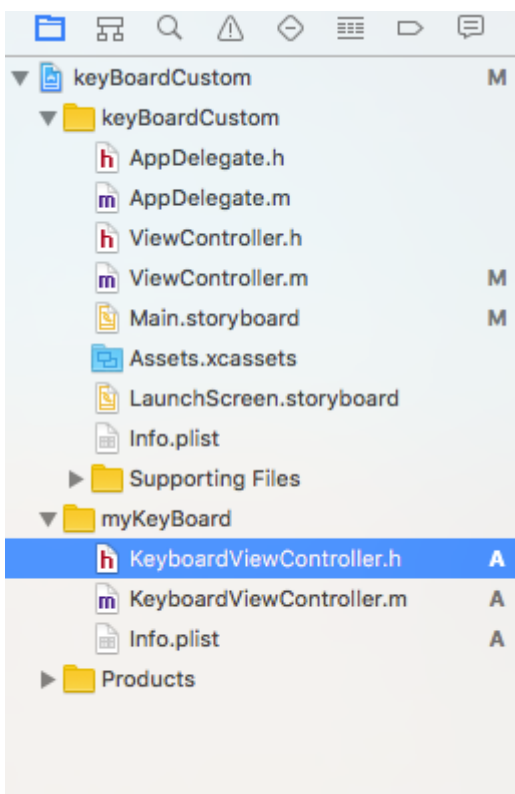
En Add Target, seleccione Custom KeyBoard.



Agrega el objetivo de esta manera:

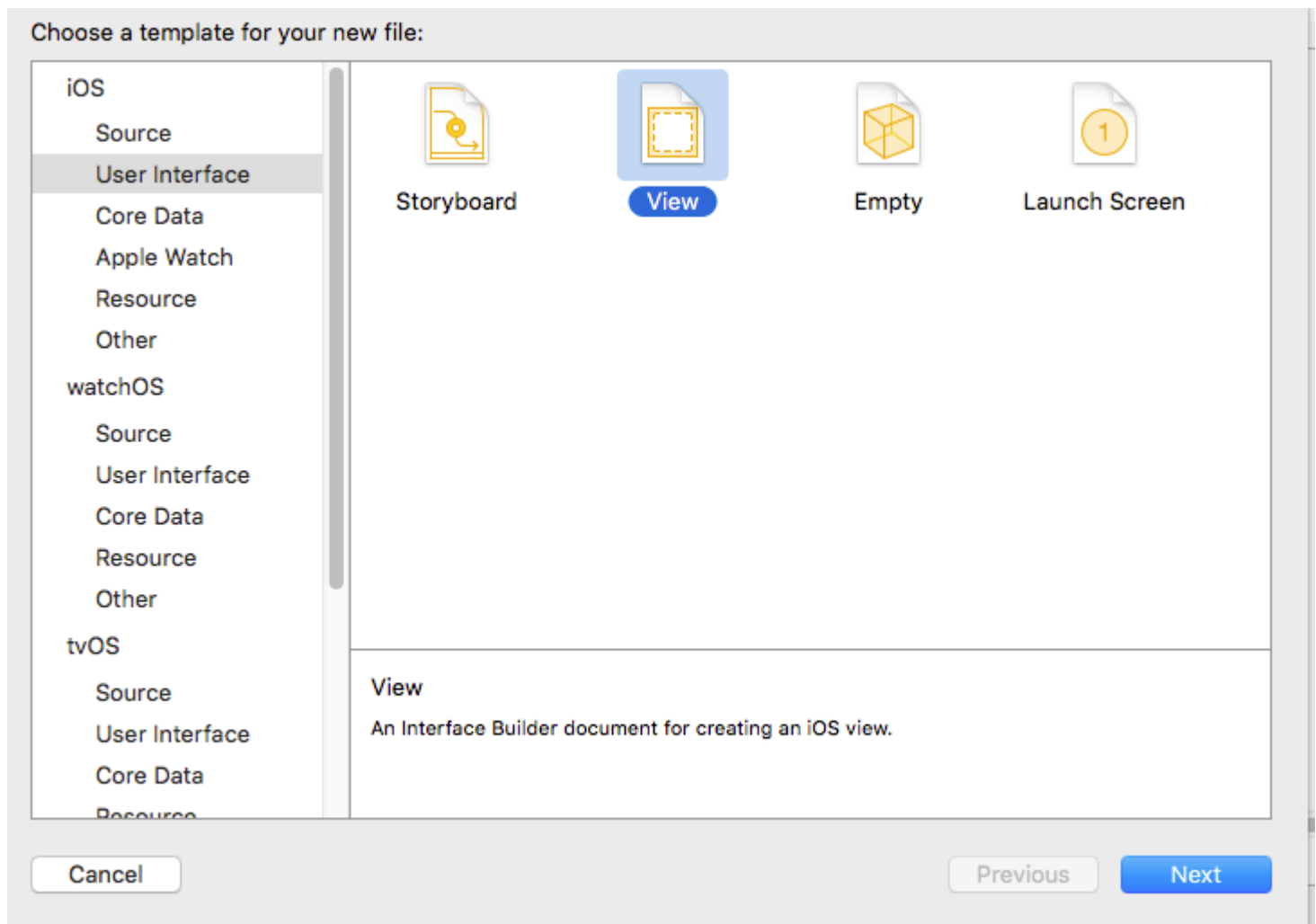


El directorio de archivos de su proyecto debería verse así.

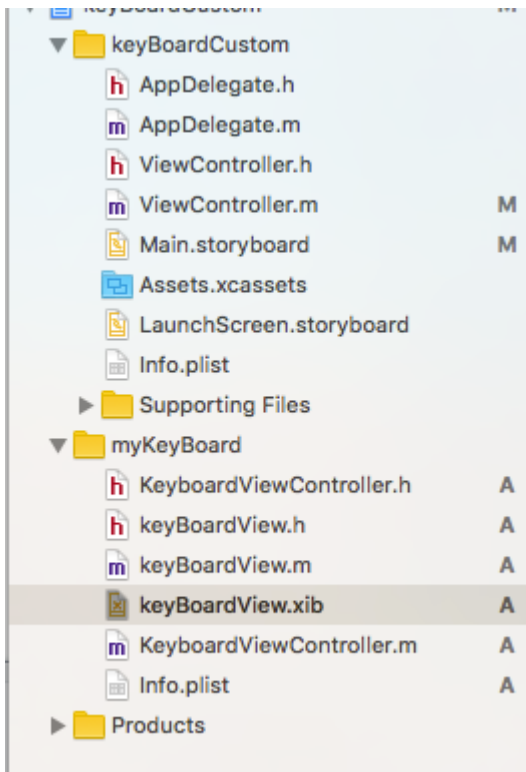


Aquí myKeyBoard es el nombre del objetivo agregado

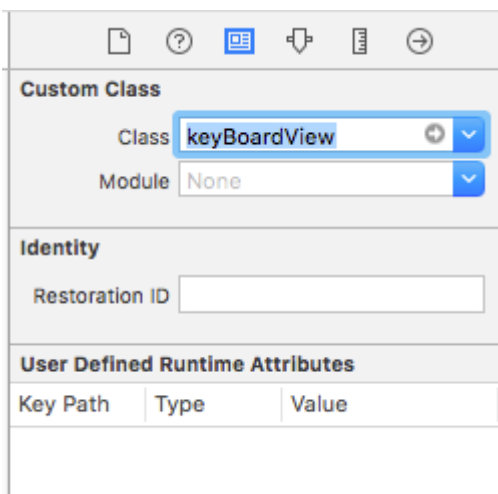
Agregue un nuevo archivo Cocoatouch de tipo de tipo UIView y agregue un archivo de interfaz



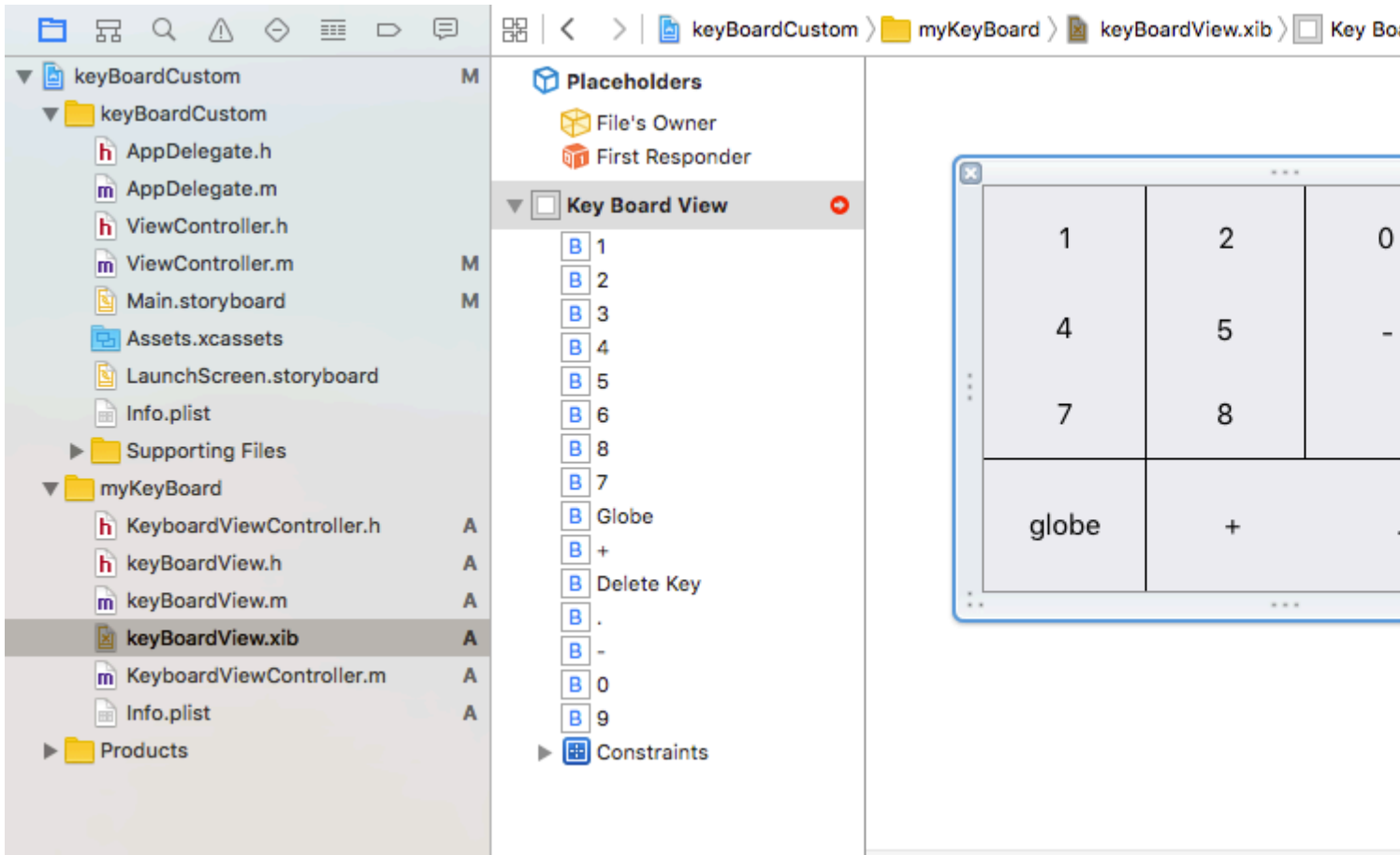
Finalmente el directorio de tu proyecto debería verse así.



convierte a `keyboardView.xib` una subclase de `keyboardView`



Hacer interfaz en el archivo `keyboardView.xib`



Haga las conexiones a partir de la `keyBoardView.xib` a `keyBoardView.h` archivo

`keyBoardView.h` debería verse como

```
#import <UIKit/UIKit.h>

@interface keyBoardView : UIView

@property (weak, nonatomic) IBOutlet UIButton *deleteKey;
//IBOutlet for the delete Key
@property (weak, nonatomic) IBOutlet UIButton *globe;
//Outlet for the key with title globe which changes the keyboard type
@property (strong, nonatomic) IBOutletCollection(UIButton) NSArray *keys;
//Contains a collection of all the keys '0 to 9' '+' '-' and '.'

@end
```

En el archivo `keyBoardViewController.h` , importe `#import "keyBoardView.h"`

Declare una propiedad para teclado `@property (strong, nonatomic)keyBoardView *keyboard;`

Comenta el

```
@property (nonatomic, strong) UIButton *nextKeyboardButton and all the code associated with it
```

La función `viewDidLoad ()` del archivo `KeyboardViewController.m` debería tener este aspecto

```
- (void)viewDidLoad {
```

```

[super viewDidLoad];
self.keyboard=[[NSBundle mainBundle]loadNibNamed:@"keyBoardView" owner:nil
options:nil]objectAtIndex:0];
self.inputView=self.keyboard;
[self addGestureToKeyboard];

// Perform custom UI setup here
// self.nextKeyboardButton = [UIButton buttonWithType:UIButtonTypeSystem];
//
// [self.nextKeyboardButton setTitle:NSString(@"Next Keyboard", @"Title for 'Next
Keyboard' button") forState:UIControlStateNormal];
// [self.nextKeyboardButton sizeToFit];
// self.nextKeyboardButton.translatesAutoresizingMaskIntoConstraints = NO;
//
// [self.nextKeyboardButton addTarget:self action:@selector(advanceToNextInputMode)
forControlEvents:UIControlEventTouchUpInside];
//
// [self.view addSubview:self.nextKeyboardButton];
//
// [self.nextKeyboardButton.leftAnchor constraintEqualToAnchor:self.view.leftAnchor].active
= YES;
// [self.nextKeyboardButton.bottomAnchor
constraintEqualToAnchor:self.view.bottomAnchor].active = YES;
}

```

Las funciones `addGestureToKeyboard` , `pressDeleteKey` , `keyPressed` se definen a continuación

```

-(void) addGestureToKeyboard
{
    [self.keyboard.deleteKey addTarget:self action:@selector(pressDeleteKey)
forControlEvents:UIControlEventTouchUpInside];
    [self.keyboard.globe addTarget:self action:@selector(advanceToNextInputMode)
forControlEvents:UIControlEventTouchUpInside];

    for (UIButton *key in self.keyboard.keys)
    {
        [key addTarget:self action:@selector(keyPressed:)
forControlEvents:UIControlEventTouchUpInside];
    }
}

-(void) pressDeleteKey
{
    [self.textDocumentProxy deleteBackward];
}

-(void)keyPressed:(UIButton *)key
{
    [self.textDocumentProxy insertText:[key currentTitle]];
}

```

Ejecute la aplicación principal y vaya a Configuración-> General-> Teclado-> Agregar nuevo teclado-> y agregue el teclado desde la sección de teclado de terceros (el nombre del teclado mostrado sería `keyBoardCustom`)

El nombre del teclado se puede cambiar agregando una clave llamada `Bundle display name` y en `Value String Value` ingrese el nombre deseado para el teclado del Proyecto principal.

	key	type	value
	Information Property List	Dictionary	(15 items)
	Localization native development re...	String	en
	Executable file	String	\$(EXECUTABLE_NAME)
	Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFI
	InfoDictionary version	String	6.0
	Bundle name	String	\$(PRODUCT_NAME)
	Bundle OS Type code	String	APPL
	Bundle versions string, short	String	1.0
	Bundle display name	String	keyBoardMi
	Bundle creator OS Type code	String	????
	Bundle version	String	1
	Application requires iPhone enviro...	Boolean	YES
	Launch screen interface file base...	String	LaunchScreen
	Main storyboard file base name	String	Main
	Required device capabilities	Array	(1 item)
	Supported interface orientations	Array	(3 items)

También puedes ver este [video de Youtube](#).

Lea Teclado personalizado en línea: <https://riptutorial.com/es/ios/topic/7358/teclado-personalizado>

Capítulo 152: Tiempo de ejecución en Objective-C

Examples

Usando objetos asociados

Los objetos asociados son útiles cuando desea agregar funcionalidad a las clases existentes que requieren un estado de espera.

Por ejemplo, agregando un indicador de actividad a cada UIView:

Implementación de Objective-C

```
#import <objc/runtime.h>

static char ActivityIndicatorKey;

@implementation UIView (ActivityIndicator)

- (UIActivityIndicatorView *)activityIndicator {
    return (UIActivityIndicatorView *)objc_getAssociatedObject(self, &ActivityIndicatorKey);
}

- (void)setActivityIndicator: (UIActivityIndicatorView *)activityIndicator {
    objc_setAssociatedObject(self, &ActivityIndicatorKey, activityIndicator,
OBJC_ASSOCIATION_RETAIN_NONATOMIC);
}

- (void)showActivityIndicator {
    UIActivityIndicatorView *activityIndicator = [[UIActivityIndicatorView alloc]
initWithActivityIndicatorStyle: UIActivityIndicatorViewStyleGray];

    [self setActivityIndicator:activityIndicator];

    activityIndicator.center = self.center;
    activityIndicator.autoresizingMask = UIViewAutoresizingFlexibleTopMargin |
UIViewAutoresizingFlexibleLeftMargin | UIViewAutoresizingFlexibleRightMargin |
UIViewAutoresizingFlexibleBottomMargin;

    [activityIndicator startAnimating];

    [self addSubview: activityIndicator];
}

- (void)hideActivityIndicator {
    UIActivityIndicatorView * activityIndicator = [self activityIndicator];

    if (activityIndicator != nil) {
        [[self activityIndicator] removeFromSuperview];
    }
}

@end
```

También puede acceder al tiempo de ejecución de Objective-C a través de Swift:

código SWIFT

```
extension UIView {
    private struct AssociatedKeys {
        static var activityIndicator = "UIView.ActivityIndicatorView"
    }

    private var activityIndicatorView: UIActivityIndicatorView? {
        get {
            return objc_getAssociatedObject(self, &AssociatedKeys.activityIndicator) as?
                UIActivityIndicatorView
        }
        set (activityIndicatorView) {
            objc_setAssociatedObject(self, &AssociatedKeys.activityIndicator,
                activityIndicatorView, .OBJC_ASSOCIATION_RETAIN_NONATOMIC)
        }
    }

    func showActivityIndicator() {
        activityIndicatorView = UIActivityIndicatorView(activityIndicatorStyle: .gray)
        activityIndicatorView.center = center
        activityIndicatorView.autoresizingMask = [.flexibleLeftMargin, .flexibleRightMargin,
            .flexibleTopMargin, .flexibleBottomMargin]

        activityIndicatorView.startAnimating()

        addSubview(activityIndicatorView)
    }

    func hideActivityIndicator() {
        activityIndicatorView.removeFromSuperview()
    }
}
```

Lea Tiempo de ejecución en Objective-C en línea:

<https://riptutorial.com/es/ios/topic/10120/tiempo-de-ejecucion-en-objective-c>

Capítulo 153: Tipo dinámico

Observaciones

```
// Content size category constants
UIContentSizeCategoryExtraSmall
UIContentSizeCategorySmall
UIContentSizeCategoryMedium
UIContentSizeCategoryLarge
UIContentSizeCategoryExtraLarge
UIContentSizeCategoryExtraExtraLarge
UIContentSizeCategoryExtraExtraExtraLarge

// Accessibility sizes
UIContentSizeCategoryAccessibilityMedium
UIContentSizeCategoryAccessibilityLarge
UIContentSizeCategoryAccessibilityExtraLarge
UIContentSizeCategoryAccessibilityExtraExtraLarge
UIContentSizeCategoryAccessibilityExtraExtraExtraLarge
```

Examples

Obtener el tamaño del contenido actual

Rápido

```
UIApplication.sharedApplication().preferredContentSizeCategory
```

C objetivo

```
[UIApplication sharedApplication].preferredContentSizeCategory;
```

Esto devuelve una constante de categoría de tamaño de contenido, o una constante de categoría de tamaño de contenido de accesibilidad.

Notificación de cambio de tamaño de texto

Puede registrarse para recibir notificaciones de cuándo se modifica el tamaño del texto del dispositivo.

Rápido

```
NSNotificationCenter.defaultCenter().addObserver(self, selector: #selector(updateFont), name:
```

```
name:UIContentSizeCategoryDidChangeNotification, object:nil)
```

C objetivo

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(updateFont)
name:UIContentSizeCategoryDidChangeNotification object:nil];
```

El objeto de notificación `userInfo` contiene el nuevo tamaño bajo `UIContentSizeCategoryNewValueKey`.

Coincidencia de tamaño de fuente de tipo dinámico en WKWebView

WKWebView cambia el tamaño de las fuentes en el contenido web para que una página web de tamaño completo se ajuste al factor de forma del dispositivo. Si desea que el texto web tanto en vertical como en horizontal sea similar en tamaño al tamaño de lectura preferido por el usuario, debe configurarlo explícitamente.

Rápido

```
// build HTML header for dynamic type and responsive design
func buildHTMLHeader() -> String {

    // Get preferred dynamic type font sizes for html styles
    let bodySize = UIFont.preferredFont(forTextStyle: UIFontTextStyle.body).pointSize
    let h1Size = UIFont.preferredFont(forTextStyle: UIFontTextStyle.title1).pointSize
    let h2Size = UIFont.preferredFont(forTextStyle: UIFontTextStyle.title2).pointSize
    let h3Size = UIFont.preferredFont(forTextStyle: UIFontTextStyle.title3).pointSize

    // On iPad, landscape text is larger than preferred font size
    var portraitMultiplier = CGFloat(1.0)
    var landscapeMultiplier = CGFloat(0.5)

    // iPhone text is shrunken
    if UIDevice.current.model.range(of: "iPhone") != nil {
        portraitMultiplier = CGFloat(3.0)
        landscapeMultiplier = CGFloat(1.5)
    }

    // Start HTML header text
    let patternText = "<html> <head> <style> "

    // Match Dynamic Type for this page.
    + "body { background-color: \(backgroundColor); } "
    + "@media all and (orientation:portrait) {img {max-width: 90%; height: auto;} "
    + "p, li { font: -apple-system-body; font-family: Georgia, serif; font-
size:calc(\(bodySize * portraitMultiplier)px + 1.0vw); font-weight: normal; color:
\(fontColor) } "
    + "h1 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h1Size * portraitMultiplier)px + 1.0vw); font-weight: bold; color:
\(headFontColor) } "
    + "h2 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h2Size * portraitMultiplier)px + 1.0vw); font-weight: bold; color:
\(headFontColor) } "
```

```

+ "h3, h4 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h3Size * portraitMultiplier)px + 1.0vw); font-weight: bold; color:
\(\headFontColor) } } "
+ "@media all and (orientation:landscape) {img {max-width: 65%; height: auto;}}"
+ "p, li { font: -apple-system-body; font-family: Georgia, serif; font-
size:calc(\(bodySize * landscapeMultiplier)px + 1.0vw); font-weight: normal; color:
\(\fontColor) }"
+ "h1 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h1Size * landscapeMultiplier)px + 1.0vw); font-weight: bold; color:
\(\headFontColor) } "
+ "h2 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h2Size * landscapeMultiplier)px + 1.0vw); font-weight: bold; color:
\(\headFontColor) } "
+ "h3, h4 { font: -apple-system-headline; font-family: Verdana, sans-serif; font-
size:calc(\(h3Size * landscapeMultiplier)px + 1.0vw); font-weight: bold; color:
\(\headFontColor) } } </style>"
+ "</head><body>"
+ "<meta name=\"viewport\" content=\"width: device-width\">"

return patternText
}

```

Manejo del cambio de tamaño de texto preferido sin notificaciones en iOS 10

UILabel , UITextField , y UITextView tienen una nueva propiedad a partir de iOS 10 para cambiar automáticamente el tamaño de su fuente cuando un usuario cambia su tamaño de lectura preferido llamado `adjustsFontForContentSizeCategory` .

Rápido

```

@IBOutlet var label:UILabel!

if #available(iOS 10.0, *) {
    label.adjustsFontForContentSizeCategory = true
} else {
    // Observe for UIContentSizeCategoryDidChangeNotification and handle it manually
    // since the adjustsFontForContentSizeCategory property isn't available.
}

```

Lea Tipo dinámico en línea: <https://riptutorial.com/es/ios/topic/4466/tipo-dinamico>

Capítulo 154: Toque 3D

Examples

Toque 3D con Swift

El toque 3D ha sido introducido con iPhone 6s Plus. Hay dos comportamientos agregados con esta nueva capa de interfaz: Peek y Pop.

Peek y Pop en pocas palabras

Ojeada - Presione fuerte

Pop - Presiona muy fuerte

Comprobando el soporte 3D

Debe comprobar si el dispositivo tiene un soporte táctil 3D. Puede hacer esto comprobando el valor de la propiedad *forceTouchCapability* de un objeto *UITraitCollection*. *UITraitCollection* describe el entorno de interfaz de iOS para su aplicación.

```
if (traitCollection.forceTouchCapability == .Available) {
    registerForPreviewingWithDelegate(self, sourceView: view)
}
```

Implementando el delegado

Debe implementar los dos métodos de *UIViewControllerPreviewingDelegate* en su clase. Uno de los métodos es para *echar un vistazo* y el otro es para *el comportamiento pop*.

El método que se implementará para el *peek* es *previewingContext*.

```
func previewingContext(previewingContext: UIViewControllerPreviewing,
viewControllerForLocation location: CGPoint) -> UIViewController? {

    guard let indexPath = self.tableView.indexPathForRowAtPoint(location), cell =
self.tableView.cellForRowAtIndex(indexPath) as? <YourTableViewCell> else {
        return nil
    }

    guard let datailVC =
storyboard?.instantiateViewControllerWithIdentifier("<YourViewControllerIdentifier>") as?
<YourViewController> else {
        return nil
    }

    datailVC.peekActive = true
    previewingContext.sourceRect = cell.frame

    // Do the stuff
```

```

return detailVC
}

```

El método a implementar para el *pop* es *previewingContext* . :)

```

func previewingContext (previewingContext: UIViewControllerPreviewing, commitViewController
viewControllerToCommit: UIViewController) {

    let balanceViewController = viewControllerToCommit as! <YourViewController>

    // Do the stuff

    navigationController?.pushViewController(balanceViewController, animated: true)

}

```

Como puedes ver son métodos sobrecargados. Puedes usar 3D touch de cualquier manera implementando estos métodos.

C objetivo

```

//Checking for 3-D Touch availability
if ([self.traitCollection respondsToSelector:@selector(forceTouchCapability)] &&
    (self.traitCollection.forceTouchCapability == UIForceTouchCapabilityAvailable))
{
    [self registerForPreviewingWithDelegate:self sourceView:self.view];
}
//Peek
- (UIViewController *)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
viewControllerForLocation:(CGPoint)location {

    NSIndexPath *indexPath = [self.tableView indexPathForRowAtPoint:location];
    Country *country = [self countryForIndexPath:indexPath];
    if (country) {
        UITableViewCell *cell = [self.tableView cellForRowAtIndexPath:indexPath];
        if (cell) {
            previewingContext.sourceRect = cell.frame;
            UINavigationController *navController = [self.storyboard
instantiateViewControllerWithIdentifier:@"UYLCountryNavController"];
            [self configureNavigationController:navController withCountry:country];
            return navController;
        }
    }
    return nil;
}
//Pop
- (void)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
commitViewController:(UIViewController *)viewControllerToCommit {

    [self showDetailViewController:viewControllerToCommit sender:self];
}

```

3 D Touch Objective-C Ejemplo

C objetivo


```

//Checking for 3-D Touch availability
if ([self.traitCollection respondsToSelector:@selector(forceTouchCapability)] &&
    (self.traitCollection.forceTouchCapability == UIForceTouchCapabilityAvailable))
{
    [self registerForPreviewingWithDelegate:self sourceView:self.view];
}
//Peek
- (UIViewController *)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
    viewControllerForLocation:(CGPoint)location {

    NSIndexPath *indexPath = [self.tableView indexPathForRowAtPoint:location];
    Country *country = [self countryForIndexPath:indexPath];
    if (country) {
        CountryCell *cell = [self.tableView cellForRowAtIndexPath:indexPath];
        if (cell) {
            previewingContext.sourceRect = cell.frame;
            UINavigationController *navController = [self.storyboard
instantiateViewControllerWithIdentifier:@"UYLCountryNavController"];
            [self configureNavigationController:navController withCountry:country];
            return navigationController;
        }
    }
    return nil;
}
//Pop
- (void)previewingContext:(id<UIViewControllerPreviewing>)previewingContext
    commitViewController:(UIViewController *)viewControllerToCommit {

    [self showDetailViewController:viewControllerToCommit sender:self];
}

```

Lea Toque 3D en línea: <https://riptutorial.com/es/ios/topic/6705/toque-3d>

Capítulo 155: Tutorial de AirPrint en iOS

Examples

Impresión de AirPrint Banner Texto

C objetivo

Agregue el delegado y un formateador de texto al archivo `ViewController.h`

```
@interface ViewController : UIViewController <UIPrintInteractionControllerDelegate> {
    UISimpleTextPrintFormatter *_textFormatter;
}
```

En el archivo `ViewController.m` defina las siguientes constantes

```
#define DefaultFontSize 48
#define PaddingFactor 0.1f
```

La función que imprime el texto es la siguiente:

```
-(IBAction)print:(id)sender;
{
    /* Get the UIPrintInteractionController, which is a shared object */
    UIPrintInteractionController *controller = [UIPrintInteractionController
sharedPrintController];
    if(!controller){
        NSLog(@"Couldn't get shared UIPrintInteractionController!");
        return;
    }

    /* Set this object as delegate so you can use the
printInteractionController:cutLengthForPaper: delegate */
    controller.delegate = self;

    UIPrintInfo *printInfo = [UIPrintInfo printInfo];
    printInfo.outputType = UIPrintInfoOutputGeneral;

    /* Use landscape orientation for a banner so the text print along the long side of the
paper. */
    printInfo.orientation = UIPrintInfoOrientationLandscape;

    printInfo.jobName = self.textField.text;
    controller.printInfo = printInfo;

    /* Create the UISimpleTextPrintFormatter with the text supplied by the user in the text
field */
    _textFormatter = [[UISimpleTextPrintFormatter alloc] initWithText:self.textField.text];

    /* Set the text formatter's color and font properties based on what the user chose */
    _textFormatter.color = [self chosenColor];
    _textFormatter.font = [self chosenFontWithSize:DefaultFontSize];
}
```

```

/* Set this UISimpleTextPrintFormatter on the controller */
controller.printFormatter = _textFormatter;

/* Set up a completion handler block. If the print job has an error before spooling, this
is where it's handled. */
void (^completionHandler)(UIPrintInteractionController *, BOOL, NSError *) =
^(UIPrintInteractionController *printController, BOOL completed, NSError *error) {
    if(completed && error)
        NSLog( @"Printing failed due to error in domain %@ with error code %lu. Localized
description: %@, and failure reason: %@", error.domain, (long)error.code,
error.localizedDescription, error.localizedFailureReason );
    };

    if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad)
        [controller presentFromRect:self.printButton.frame inView:self.view animated:YES
completionHandler:completionHandler];
    else
        [controller presentAnimated:YES completionHandler:completionHandler]; // iPhone
}

```

La función de delegado que configura la página de impresión:

```

- (CGFloat)printInteractionController:(UIPrintInteractionController
*)printInteractionController cutLengthForPaper:(UIPrintPaper *)paper {

    /* Create a font with arbitrary size so that you can calculate the approximate
font points per screen point for the height of the text. */
    UIFont *font = _textFormatter.font;
    CGSize size = [self.textField.text sizeWithAttributes:@{NSFontAttributeName: font}];

    float approximateFontPointPerScreenPoint = font.pointSize / size.height;

    /* Create a new font using a size that will fill the width of the paper */
    font = [self.chosenFontWithSize: paper.printableRect.size.width *
approximateFontPointPerScreenPoint];

    /* Calculate the height and width of the text with the final font size */
    CGSize finalTextSize = [self.textField.text sizeWithAttributes:@{NSFontAttributeName:
font}];

    /* Set the UISimpleTextFormatter font to the font with the size calculated */
    _textFormatter.font = font;

    /* Calculate the margins of the roll. Roll printers may have unprintable areas
before and after the cut. We must add this to our cut length to ensure the
printable area has enough room for our text. */
    CGFloat lengthOfMargins = paper.paperSize.height - paper.printableRect.size.height;

    /* The cut length is the width of the text, plus margins, plus some padding */
    return finalTextSize.width + lengthOfMargins + paper.printableRect.size.width *
PaddingFactor;
}

```

Lea Tutorial de AirPrint en iOS en línea: <https://riptutorial.com/es/ios/topic/7395/tutorial-de-airprint-en-ios>

Capítulo 156: Ubicación del núcleo

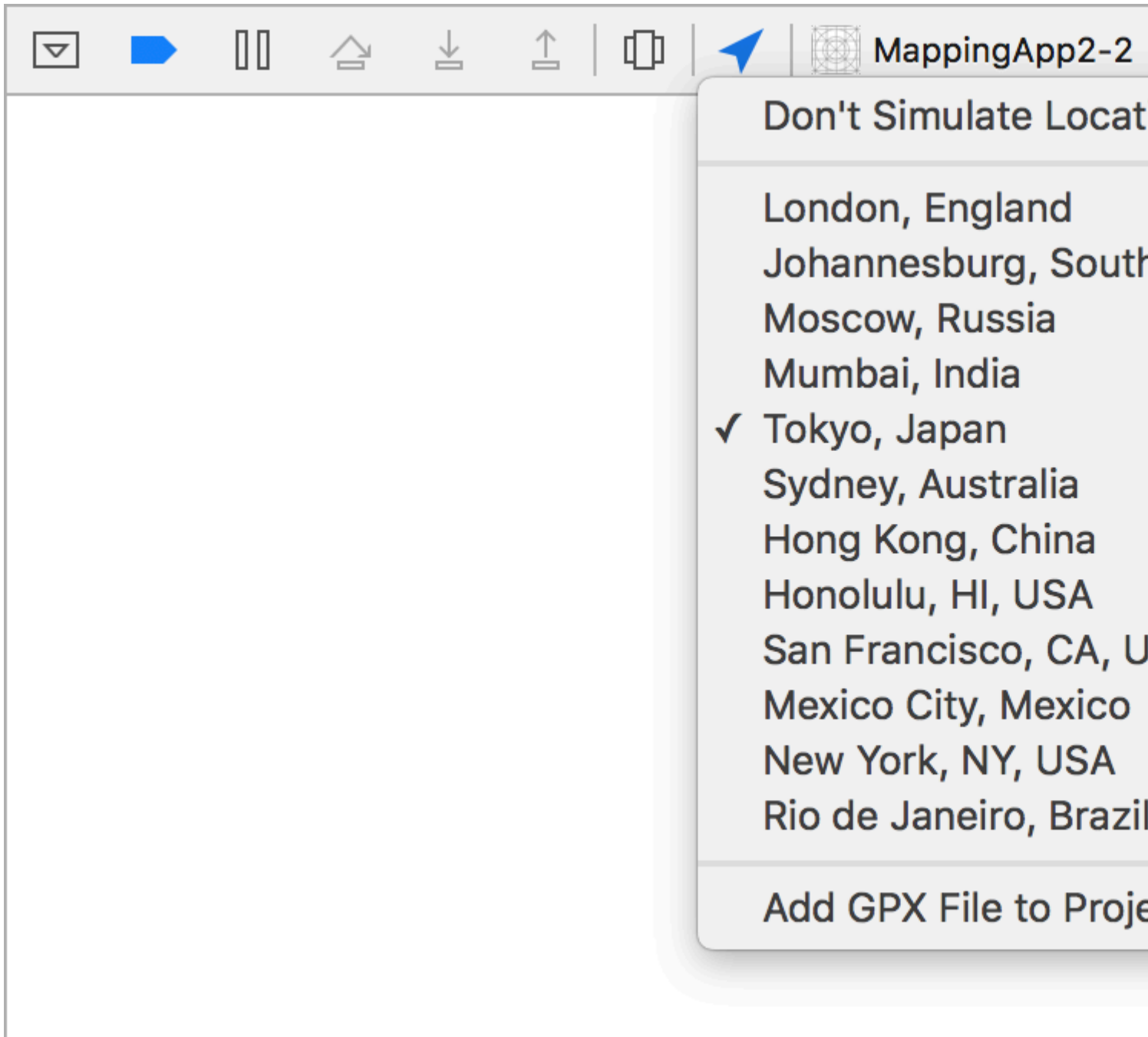
Sintaxis

1. la exactitud deseada
2. distanciafiltro
3. requestLocation ()
4. startUpdatingLocation ()
5. allowDeferredLocationUpdates (untilTraveled: timeout :)
6. startMonitoringSignificantLocationChanges ()
7. allowDeferredLocationUpdates (untilTraveled: timeout :)
8. autorizadosiempre
9. autorizado
10. locationManager (_: didChangeAuthorization :)

Observaciones

Simular una ubicación en tiempo de ejecución

1. Ejecuta la aplicación desde Xcode.
2. En la barra de depuración, haga clic en el botón "Simular ubicación".
3. Elija una ubicación en el menú.



Examples

[Link CoreLocation Framework](#)



MappingApp



General

Cap

PROJECT



MappingApp2-2

TARGETS



MappingApp2-2

Choose frameworks and libraries

CoreL

▼ iOS 9.2



CoreLocation.framework

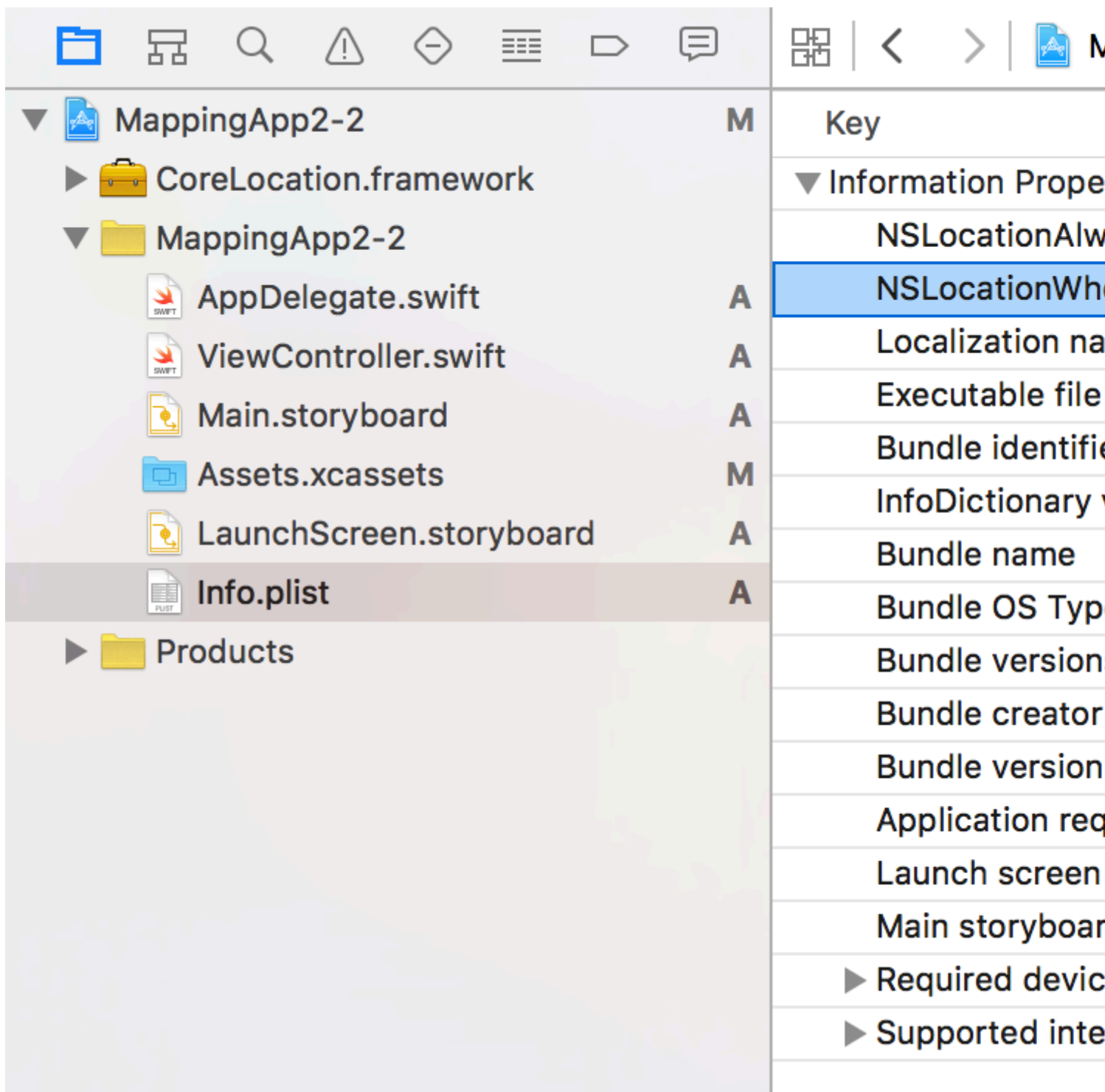


Developer Frameworks

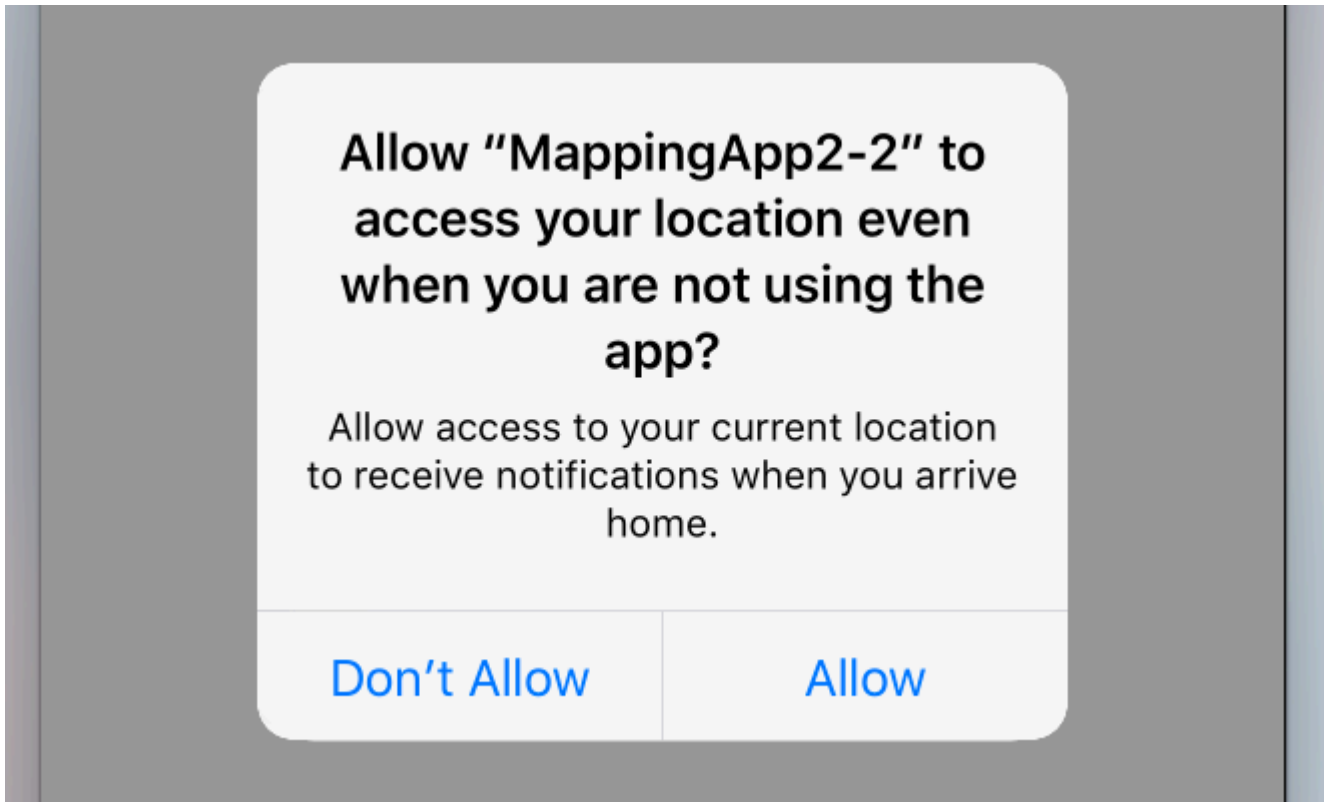
Add Other...



. El valor se utilizará en la etiqueta de `message` del controlador de alerta.



Obtener permiso de servicio de ubicación siempre

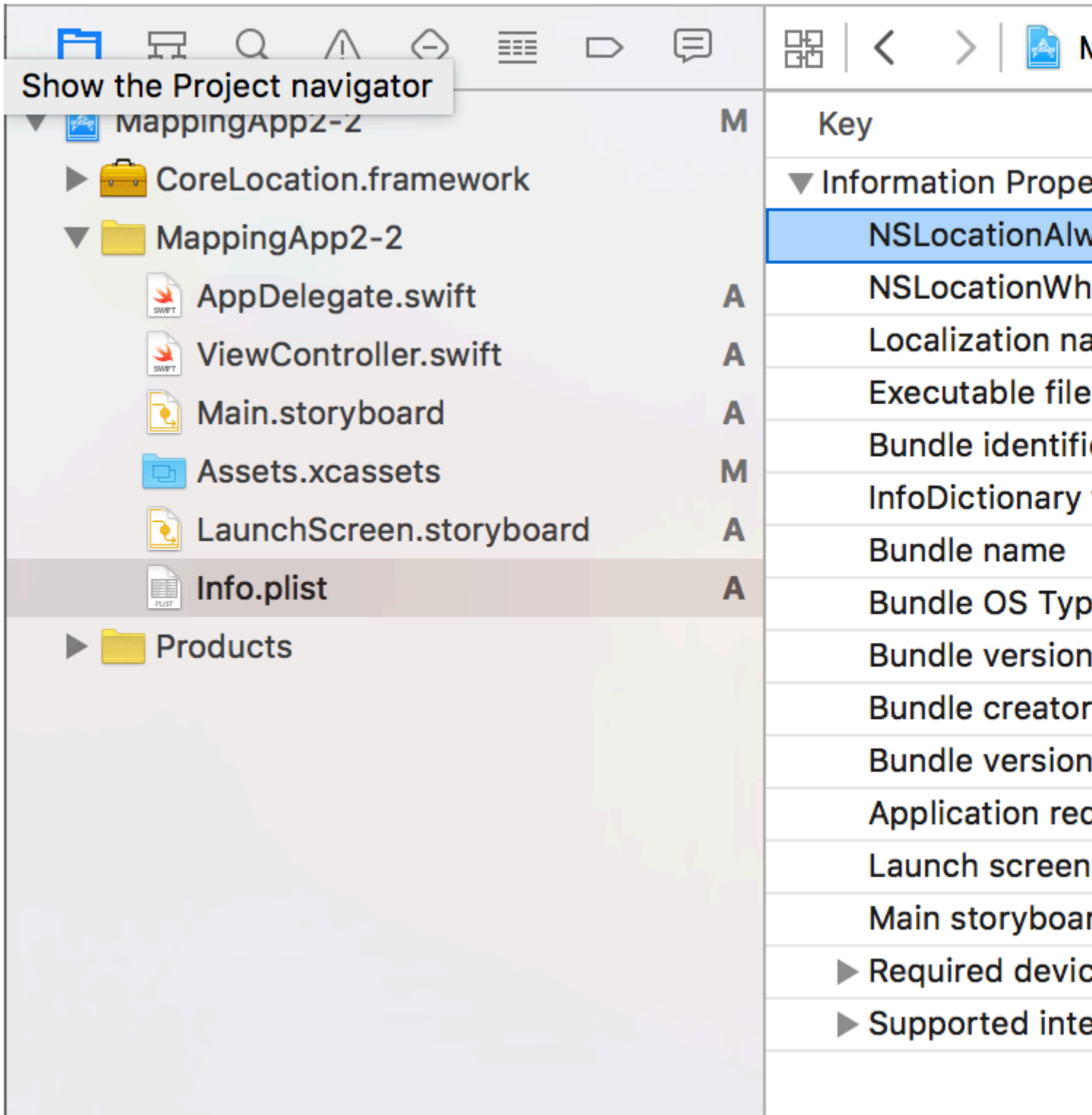


Para solicitar permiso para usar los servicios de ubicación incluso cuando la aplicación no está activa, use la siguiente llamada en su lugar:

```
//Swift
locationManager.requestAlwaysAuthorization()

//Objective-C
[locationManager requestAlwaysAuthorization];
```

Luego, agregue la clave **NSLocationAlwaysUsageDescription** a su *Info.plist*. De nuevo, el valor se utilizará en la etiqueta de `message` del controlador de alerta.



Agrega tu propia ubicación personalizada usando el archivo GPX

Para verificar los servicios de ubicación necesitamos un dispositivo real, pero para propósitos de prueba también podemos usar el simulador y agregar nuestra propia ubicación siguiendo los pasos a continuación:

- Agrega un nuevo archivo GPX a tu proyecto.
- en el archivo GPX agregar puntos de paso como

```
<?xml version="1.0"?>
```

```

<gpx version="1.1" creator="Xcode">
<!--
    Provide one or more waypoints containing a latitude/longitude pair. If you provide one
    waypoint, Xcode will simulate that specific location. If you provide multiple
waypoints,
    Xcode will simulate a route visitng each waypoint.
-->
<wpt lat="52.599878" lon="4.702029">
    <name>location name (eg. Florida)</name>
</wpt>

```

- luego vaya a producto -> Esquema -> Editar esquema y en RUN, establezca la ubicación predeterminada como su nombre de archivo GPX.

Servicios de localización en el fondo

Para usar los servicios de ubicación estándar mientras la aplicación está en segundo plano, primero debe activar los `Background Modes` en la pestaña `Capacidades` de la configuración de `Destino` y seleccionar `Location updates`.

O, agréguelo directamente al `Info.plist`.

```

<key>NSLocationAlwaysUsageDescription</key>
<string>I want to get your location Information in background</string>

<key>UIBackgroundModes</key>
<array>
    <string>location</string>
</array>

```

Entonces necesitas configurar el `CLLocationManager`

C objetivo

```

//The Location Manager must have a strong reference to it.
_locationManager = [[CLLocationManager alloc] init];
_locationManager.delegate = self;

//Request Always authorization (iOS8+)
if ([_locationManager respondsToSelector:@selector(requestAlwaysAuthorization)]) {
    [_locationManager requestAlwaysAuthorization];
}

//Allow location updates in the background (iOS9+)
if ([_locationManager respondsToSelector:@selector(allowsBackgroundLocationUpdates)]) {
    _locationManager.allowsBackgroundLocationUpdates = YES;
}

[_locationManager startUpdatingLocation];

```

Rápido

```
self.locationManager.delegate = self
```

```
if #available (iOS 8.0,*) {
    self.locationManager.requestAlwaysAuthorization()
}

if #available (iOS 9.0,*) {
    self.locationManager.allowsBackgroundLocationUpdates = true
}

self.locationManager.startUpdatingLocation()
```

Lea Ubicación del núcleo en línea: <https://riptutorial.com/es/ios/topic/2937/ubicacion-del-nucleo>

Capítulo 157: UINavigationController

Parámetros

Nombre del parámetro	Descripción
artículos de actividad	Contiene matriz de objeto para realizar la actividad. Esta matriz no debe ser nula y debe contener al menos un objeto.
aplicaciónActividades	Una matriz de objetos UINavigationController que representan los servicios personalizados que admite su aplicación. Este parámetro puede ser nulo.

Examples

Inicializando el Controlador de Vista de Actividades

C objetivo

```
NSString *textToShare = @"StackOverflow Documentation!! Together, we can do for Documentation what we did for Q&A.";
NSURL *documentationURL = [NSURL
    URLWithString:@"http://stackoverflow.com/tour/documentation"];

NSArray *objectsToShare = @[textToShare, documentationURL];

UINavigationController *activityVC = [[UINavigationController alloc]
    initWithActivityItems:objectsToShare applicationActivities:nil];

[self presentViewController:activityVC animated:YES completion:nil];
```

Rápido

```
let textToShare = "StackOverflow Documentation!! Together, we can do for Documentation what we did for Q&A."
let documentationURL = NSURL(string:"http://stackoverflow.com/tour/documentation")

let objToShare : [AnyObject] = [textToShare, documentationURL!]

let activityVC = UINavigationController(activityItems: objToShare, applicationActivities: nil)
self.presentViewController(activityVC, animated: true, completion: nil)
```

Lea UINavigationController en línea:

<https://riptutorial.com/es/ios/topic/2889/uiactivityviewController>

Capítulo 158: UIAlertController

Observaciones

Un objeto `UIAlertController` muestra un mensaje de alerta para el usuario. Esta clase reemplaza las clases `UIActionSheet` y `UIAlertView` para mostrar alertas. Después de configurar el controlador de alertas con las acciones y el estilo que desea,

`presentViewController:animated:completion:` utilizando el `presentViewController:animated:completion:`

De [la documentación de Apple](#)

[UIAlertController en Swift](#)

Examples

AlertViews con UIAlertController

`UIAlertView` y `UIActionSheet` están en `UIActionSheet` en iOS 8 y iOS 8 posteriores. Así que Apple presentó un nuevo controlador para `alertView` y `ActionSheet` llamado `UIAlertController`, cambiando el estilo `preferredStyle`, puede cambiar entre `alertView` y `ActionSheet`. No hay un método de delegado para ello porque todos los eventos de botones se manejan en sus bloques.

alertView simple

Rápido:

```
let alert = UIAlertController(title: "Simple", message: "Simple alertView demo with Cancel and OK.", preferredStyle: .alert)

alert.addAction(UIAlertAction(title: "Cancel", style: .cancel) { _ in
    print("Cancel")
})
alert.addAction(UIAlertAction(title: "OK", style: .default) { _ in
    print("OK")
})

present(alert, animated: true)
```

Objetivo:

```
UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"Simple"
message:@"Simple alertView demo with Cancel and OK."
preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction *cancelAction = [UIAlertAction actionWithTitle:@"Cancel"
style:UIAlertActionStyleCancel handler:^(UIAlertAction * action) {
    NSLog(@"Cancel");
}];

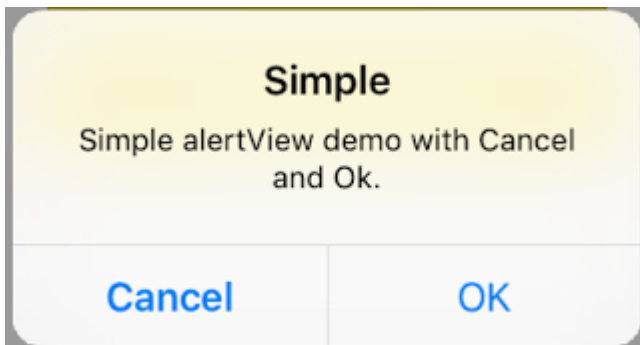
UIAlertAction *okAction = [UIAlertAction actionWithTitle:@"OK"
style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
```

```

        NSLog(@"OK");
    }];

    [alertController addAction:cancelAction];
    [alertController addAction:okAction];
    [self presentViewController:alertController animated: YES completion: nil];

```



Vista de alerta destructiva

Rápido:

```

let alert = UIAlertController(title: "Simple", message: "Simple alertView demo with Cancel and
OK.", preferredStyle: .alert)

alert.addAction(UIAlertAction(title: "Destructive", style: .destructive) { _ in
    print("Destructive")
})
alert.addAction(UIAlertAction(title: "OK", style: .default) { _ in
    print("OK")
})

present(alert, animated: true)

```

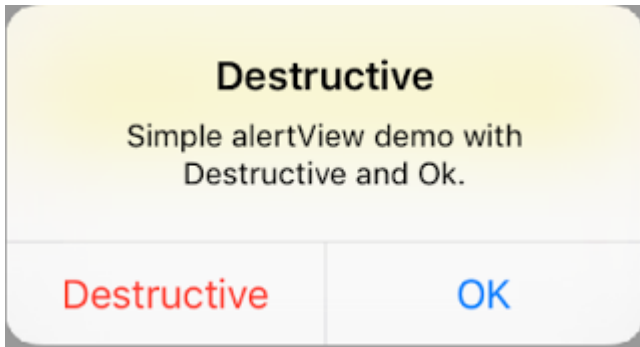
C objetivo:

```

UIAlertController *alertController = [UIAlertController
alertControllerWithTitle:@"Destructive" message:@"Simple alertView demo with Destructive and
OK." preferredStyle:UIAlertControllerStyleAlert];
    UIAlertAction *destructiveAction = [UIAlertAction actionWithTitle:@"Cancel"
style:UIAlertActionStyleDestructive handler:^(UIAlertAction * action) {
    NSLog(@"Destructive");
}];
    UIAlertAction *okAction = [UIAlertAction actionWithTitle:@"OK"
style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
    NSLog(@"OK");
}];

    [alertController addAction:destructiveAction];
    [alertController addAction:okAction];
    [self presentViewController:alertController animated: YES completion: nil];

```



Pop-up temporal como un brindis

Bueno para notificaciones rápidas que no requieren interacción.

Rápido

```
let alert = UIAlertController(title: "Toast", message: "Hello World", preferredStyle: .Alert)

presentViewController(alert, animated: true) {
    let delay_s:Double = 2
    let delayTime = dispatch_time(DISPATCH_TIME_NOW, Int64(delay_s * Double(NSEC_PER_SEC)))
    dispatch_after(delayTime, dispatch_get_main_queue()) {
        alert.dismissViewControllerAnimated(true, completion: nil)
    }
}
```

Agregar campo de texto en UIAlertController como un cuadro de solicitud

Rápido

```
let alert = UIAlertController(title: "Hello",
                            message: "Welcome to the world of iOS",
                            preferredStyle: UIAlertControllerStyle.alert)

let defaultAction = UIAlertAction(title: "OK", style: UIAlertActionStyle.default) { (action)
in
}
defaultAction.isEnabled = false
alert.addAction(defaultAction)

alert.addTextFieldWithConfigurationHandler { (textField) in
    textField.delegate = self
}

present(alert, animated: true, completion: nil)
```

C objetivo

```
UIAlertController* alert = [UIAlertController alertControllerWithTitle:@"Hello"
                                                                    message:@"Welcome to the world
of iOS"
```

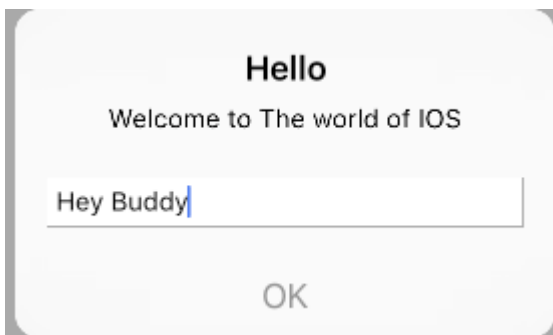
```
preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction* defaultAction = [UIAlertAction actionWithTitle:@"OK"
                                style:UIAlertActionStyleDefault
                                handler:^(UIAlertAction * action) {}];

defaultAction.enabled = NO;
[alert addAction:defaultAction];

[alert addTextFieldWithConfigurationHandler:^(UITextField *textField) {
    textField.delegate = self;
}];

[self presentViewController:alert animated:YES completion:nil];
```



Hojas de acción con UIAlertController

Con `UIAlertController`, las hojas de acción como la `UIActionSheet` obsoleta se crean con la misma API que utiliza para `AlertViews`.

Hoja de acción simple con dos botones

Rápido

```
let alertController = UIAlertController(title: "Demo", message: "A demo with two buttons",
preferredStyle: UIAlertControllerStyle.actionSheet)
```

C objetivo

```
UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"Demo"
message:@"A demo with two buttons" preferredStyle:UIAlertControllerStyleActionSheet];
```

Creando los botones "Cancelar" y "Bien"

Rápido

```
let cancelAction = UIAlertAction(title: "Cancel", style: .cancel) { (result : UIAlertAction) -
```



```
> Void in
    //action when pressed button
}
let okAction = UIAlertAction(title: "Okay", style: .default) { (result : UIAlertAction) ->
Void in
    //action when pressed button
}
```

C objetivo

```
UIAlertAction *cancelAction = [UIAlertAction initWithTitle:@"Cancel"
style:UIAlertActionStyleCancel handler:^(UIAlertAction * action) {
    //action when pressed button
}];

UIAlertAction * okAction = [UIAlertAction initWithTitle:@"Okay"
style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
    //action when pressed button
}];
```

Y añádelos a la hoja de acción:

Rápido

```
alertController.addAction(cancelAction)
alertController.addAction(okAction)
```

C objetivo

```
[alertController addAction:cancelAction];
[alertController addAction:okAction];
```

Ahora presente el `UIAlertController` :

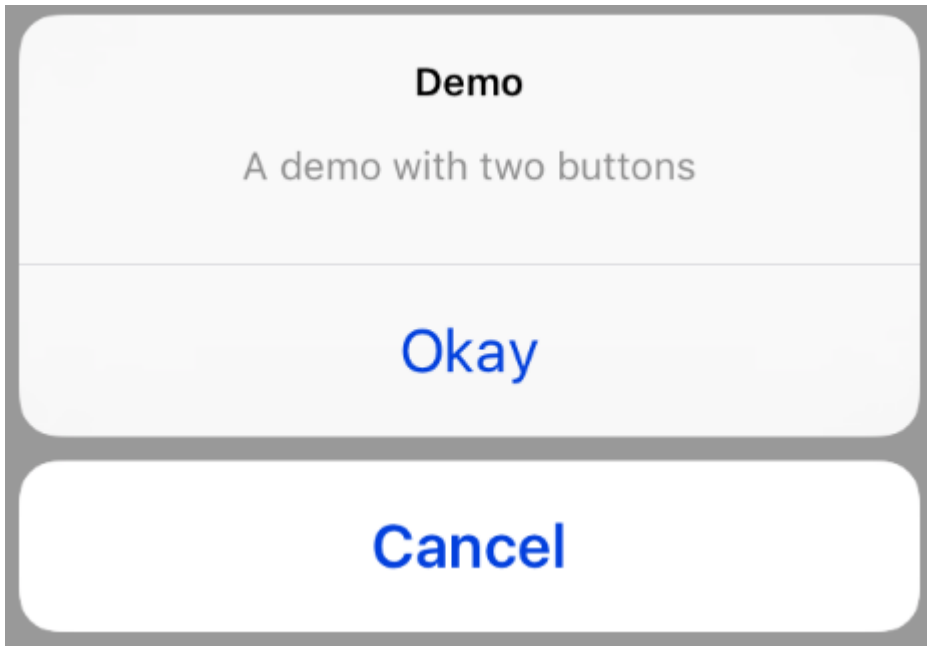
Rápido

```
self.present(alertController, animated: true, completion: nil)
```

C objetivo

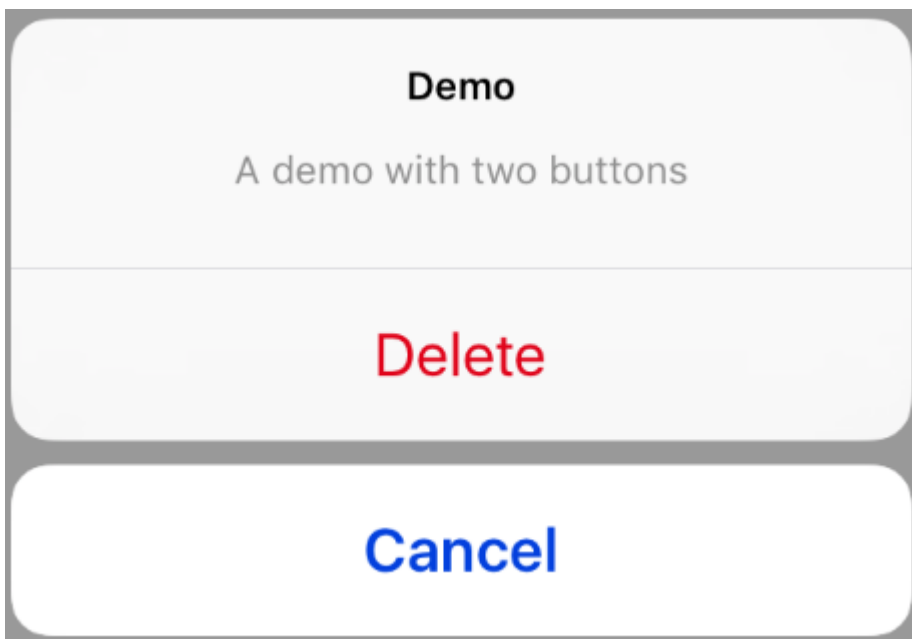
```
[self presentViewController:alertController animated: YES completion: nil];
```

Este debería ser el resultado:



Hoja de acción con botón destructivo

El uso de `UIAlertActionStyle.destructive` para una `UIAlertAction` creará un botón con un color de tinte rojo.



Para este ejemplo, la `okAction` de arriba fue reemplazada por esta `UIAlertAction` :

Rápido

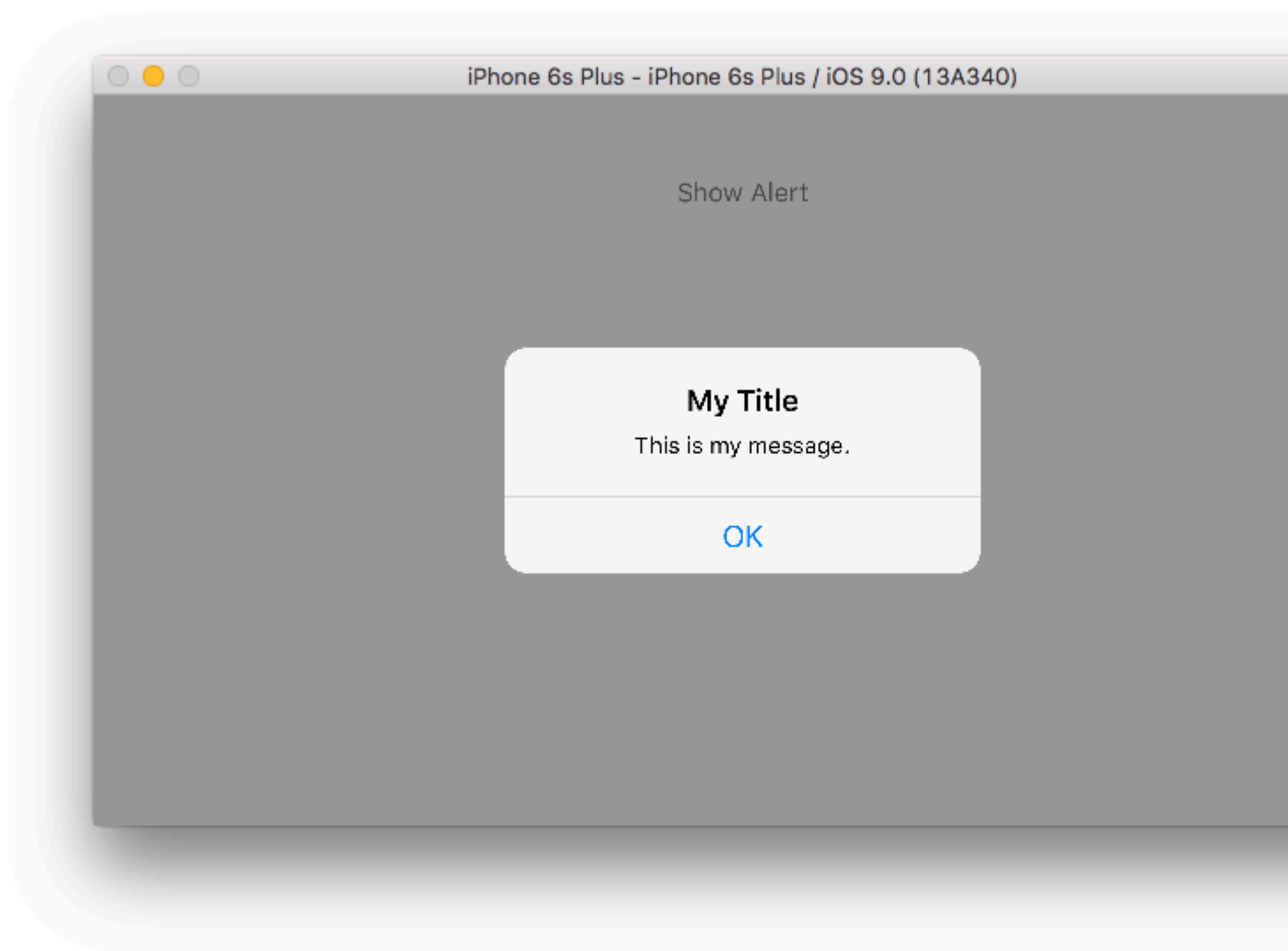
```
let destructiveAction = UIAlertAction(title: "Delete", style: .destructive) { (result :  
UIAlertAction) -> Void in  
    //action when pressed button  
}
```

C objetivo

```
UIAlertAction * destructiveAction = [UIAlertAction actionWithTitle:@"Delete"  
style:UIAlertActionStyleDestructive handler:^(UIAlertAction * action) {  
    //action when pressed button  
}];
```

Visualización y manejo de alertas.

Un botón



Rápido

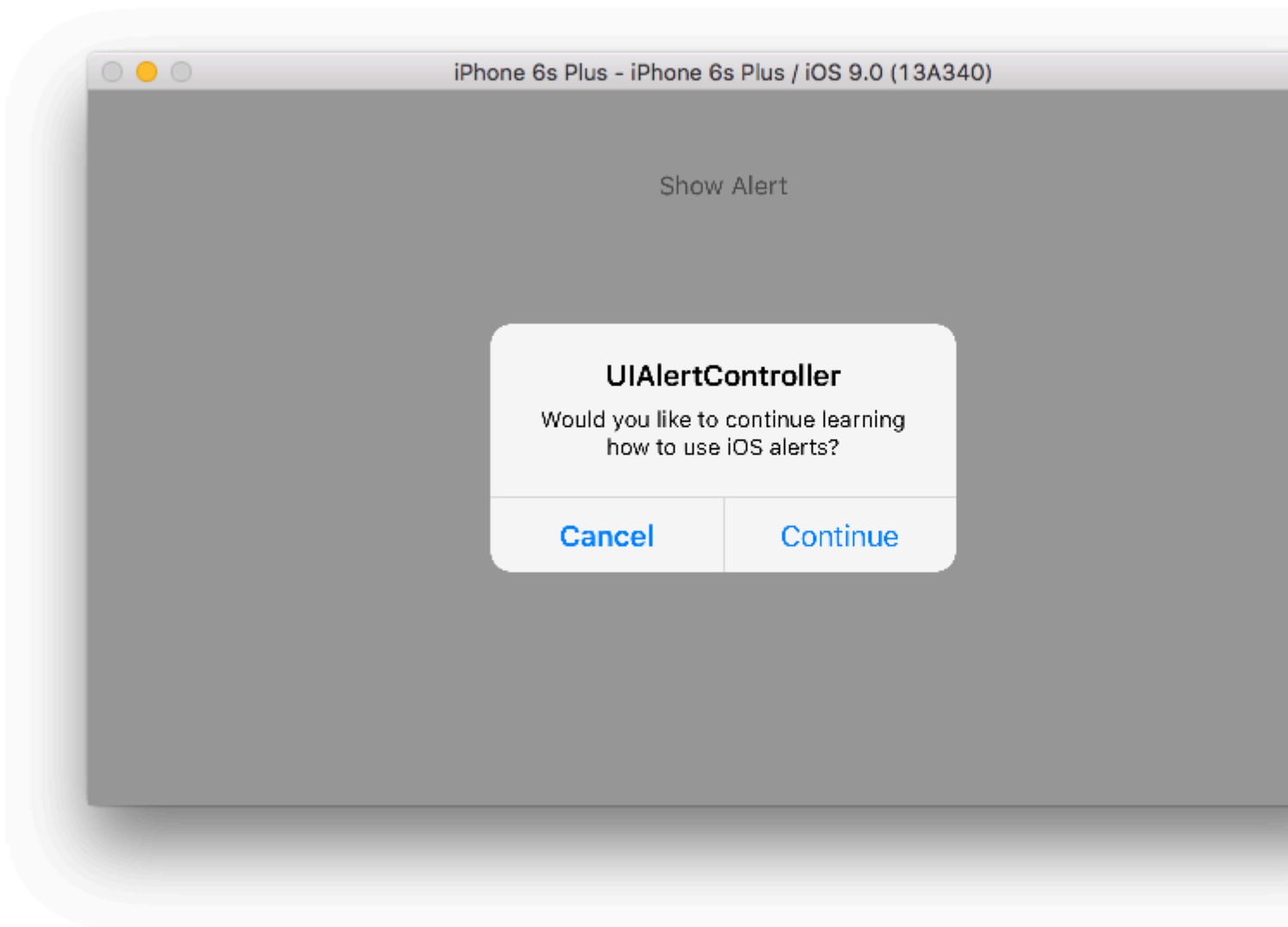
```
class ViewController: UIViewController {  
  
    @IBAction func showAlertButtonTapped(sender: UIButton) {  
  
        // create the alert
```

```
    let alert = UIAlertController(title: "My Title", message: "This is my message.",
preferredStyle: UIAlertControllerStyle.Alert)

    // add an action (button)
    alert.addAction(UIAlertAction(title: "OK", style: UIAlertActionStyle.Default, handler:
nil))

    // show the alert
    self.presentViewController(alert, animated: true, completion: nil)
}
}
```

Dos botones



Rápido

```
class ViewController: UIViewController {

    @IBAction func showAlertButtonTapped(sender: UIButton) {

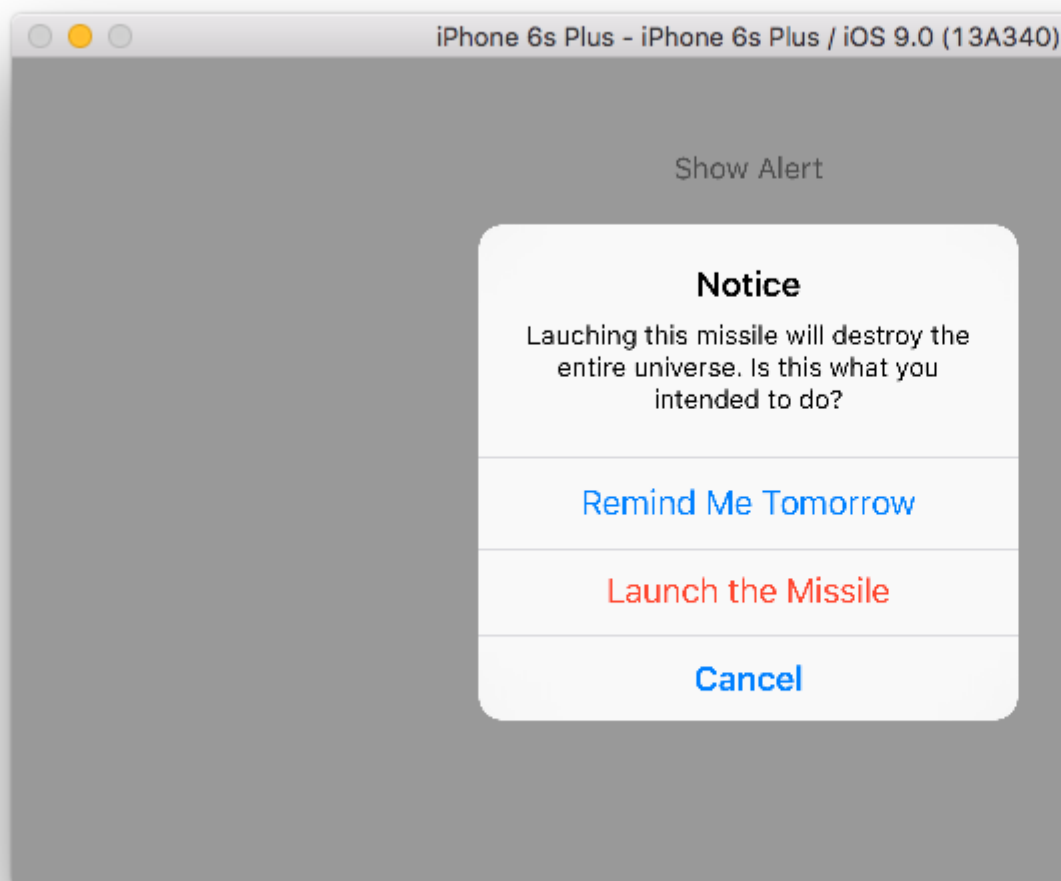
        // create the alert
```

```
let alert = UIAlertController(title: "UIAlertController", message: "Would you like to
continue learning how to use iOS alerts?", preferredStyle: UIAlertControllerStyle.Alert)

// add the actions (buttons)
alert.addAction(UIAlertAction(title: "Continue", style: UIAlertActionStyle.Default,
handler: nil))
alert.addAction(UIAlertAction(title: "Cancel", style: UIAlertActionStyle.Cancel,
handler: nil))

// show the alert
self.presentViewController(alert, animated: true, completion: nil)
}
```

Tres botones



Rápido

```
class ViewController: UIViewController {

    @IBAction func showAlertButtonTapped(sender: UIButton) {
```

```

    // create the alert
    let alert = UIAlertController(title: "Notice", message: "Lauching this missile will
destroy the entire universe. Is this what you intended to do?", preferredStyle:
UIAlertControllerStyle.Alert)

    // add the actions (buttons)
    alert.addAction(UIAlertAction(title: "Remind Me Tomorrow", style:
UIAlertActionStyle.Default, handler: nil))
    alert.addAction(UIAlertAction(title: "Cancel", style: UIAlertActionStyle.Cancel,
handler: nil))
    alert.addAction(UIAlertAction(title: "Launch the Missile", style:
UIAlertActionStyle.Destructive, handler: nil))

    // show the alert
    self.presentViewController(alert, animated: true, completion: nil)
}
}

```

Manipulación de botones

El `handler` fue `nil` en los ejemplos anteriores. Puede reemplazar `nil` con un [cierro](#) para hacer algo cuando el usuario toca un botón, como en el siguiente ejemplo:

Rápido

```

alert.addAction(UIAlertAction(title: "Launch the Missile", style:
UIAlertActionStyle.Destructive, handler: { action in

    // do something like...
    self.launchMissile()

}))

```

Notas

- Múltiples botones no necesariamente necesitan usar diferentes tipos de `UIAlertActionStyle`. Todos ellos podrían ser `.Default`.
- Para más de tres botones considere usar una Hoja de Acción. La configuración es muy similar. [Aquí hay un ejemplo.](#)

Resaltando un botón de acción

El controlador de alertas tiene una propiedad que se utiliza para poner énfasis en una acción agregada en el controlador de alertas. Esta propiedad se puede usar para resaltar una acción particular para la atención del usuario. Para el objetivo C;

```
@property(nonatomic, strong) UIAlertAction *preferredAction
```

Se puede asignar una acción **que ya se agregó en el controlador de alerta** a esta propiedad. El Controlador de alertas destacará esta acción.

Esta propiedad solo se puede utilizar con UIAlertControllerStyleAlert.

El siguiente ejemplo muestra cómo usarlo.

```
UIAlertController *alertController = [UIAlertController alertControllerWithTitle:@"Cancel edit" message:@"Are you really want to cancel your edit?" preferredStyle:UIAlertControllerStyleAlert];

UIAlertAction *cancel = [UIAlertAction actionWithTitle:@"Cancel" style:UIAlertActionStyleCancel handler:^(UIAlertAction * action) {
    NSLog(@"Cancel");
}];

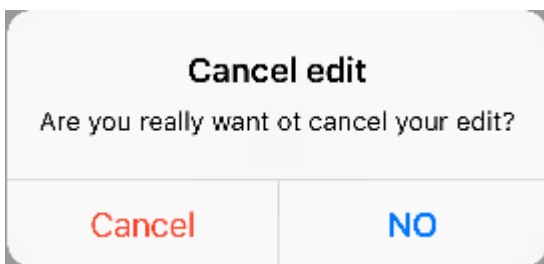
UIAlertAction *no = [UIAlertAction actionWithTitle:@"NO" style:UIAlertActionStyleDefault handler:^(UIAlertAction * action) {
    NSLog(@"Highlighted button is pressed.");
}];

[alertController addAction:cancel];
[alertController addAction:no];

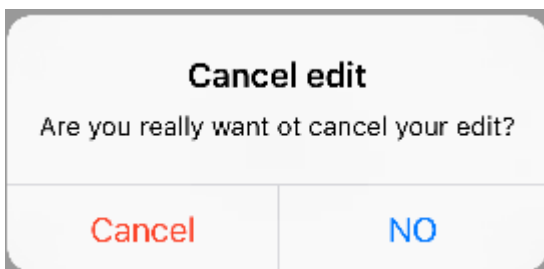
//add no action to preferred action.
//Note
//the action should already be added to alert controller
alertController.preferredAction = no;

[self presentViewController:alertController animated: YES completion: nil];
```

Controlador de alertas con **conjunto de acciones preferidas** . El botón **NO** está resaltado.



Controlador de alertas con **acción preferida no configurada** . El botón **NO** no está resaltado.



Lea UIAlertController en línea: <https://riptutorial.com/es/ios/topic/874/uialertcontroller>

Capítulo 159: UIBarButtonItem

Parámetros

Parámetro	Descripción
título	El título de UIBarButtonItem
estilo	El estilo del UIBarButtonItem.
objetivo	El objeto para recibir la acción UIBarButtonItem.
acción	El selector (método) que se realizará cuando se presione UIBarButtonItem

Observaciones

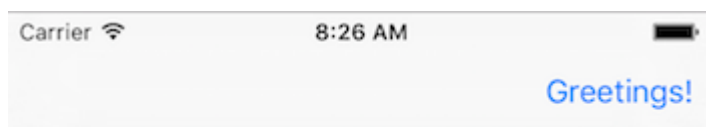
La referencia a `self.navigationItem` asume que UIViewController está incrustado dentro de un UINavigationController.

Examples

Creando un UIBarButtonItem

```
//Swift
let barButtonItem = UIBarButtonItem(title: "Greetings!", style: .Plain, target: self, action:
#selector(barButtonTapped))
self.navigationItem.rightBarButtonItem = barButtonItem

//Objective-C
UIBarButtonItem *barButtonItem = [[UIBarButtonItem alloc] initWithTitle:@"Greetings!"
style:UIBarButtonItemStylePlain target:self action:@selector(barButtonTapped)];
self.navigationItem.rightBarButtonItem = barButtonItem;
```

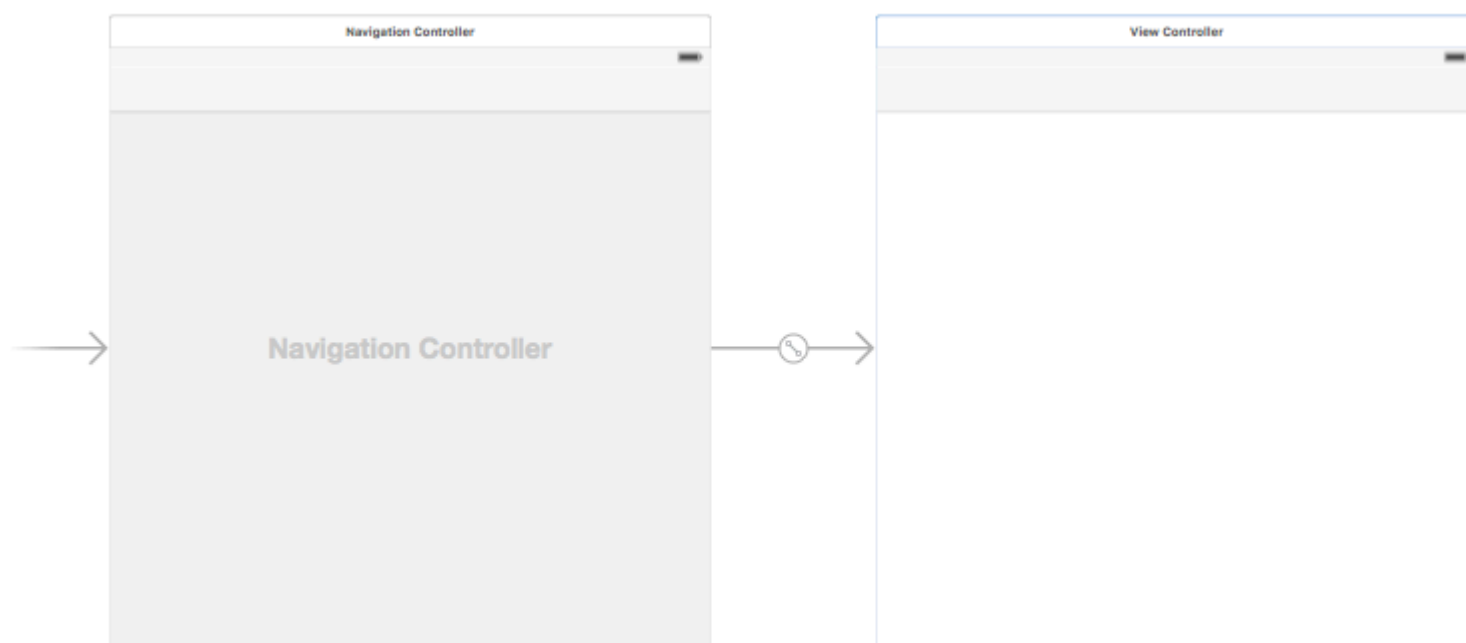


Creando un UIBarButtonItem en el Interface Builder

El siguiente ejemplo muestra cómo agregar un botón de la barra de navegación (denominado UIBarButtonItem) en el Creador de interfaces.

Agrega un controlador de navegación a tu gui3n gr3fico

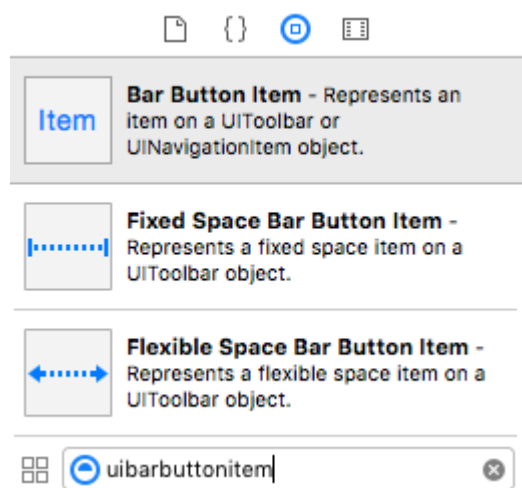
Seleccione su View Controller y luego en el men3 Xcode elija **Editor > Incrustar en > Navigation Controller** .



Alternativamente, puede agregar una `UINavigationController` desde la biblioteca de objetos.

A3adir un elemento de bot3n de barra

Arrastre un `UIBarButtonItem` desde la biblioteca de objetos a la barra de navegaci3n superior.

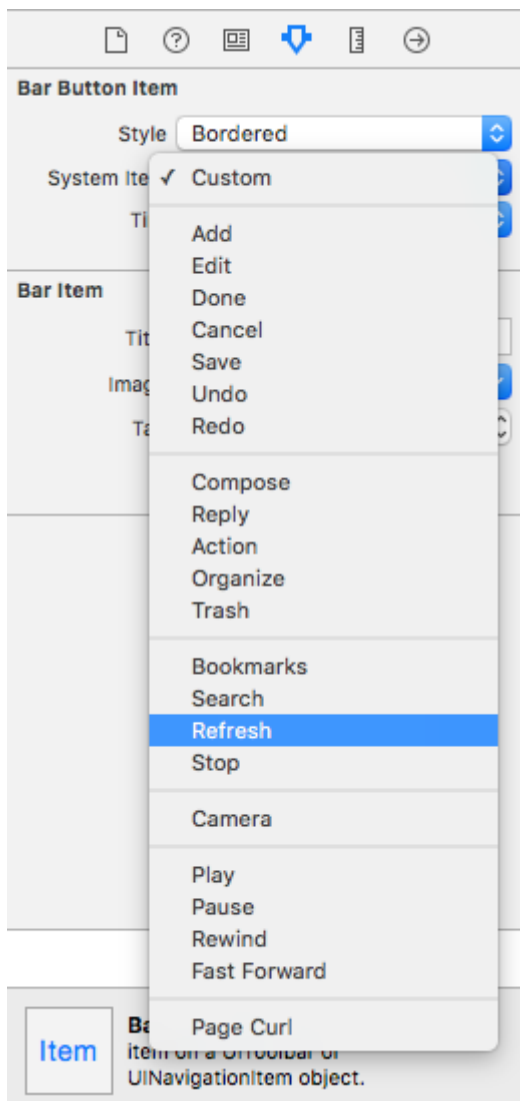


Debe tener un aspecto como este:



Establecer los atributos

Puede hacer doble clic en "Elemento" para cambiar el texto a algo como "Actualizar", pero hay un icono real para *Actualizar* que puede usar. Simplemente seleccione el inspector de atributos para el `UIBarButtonItem` y para el **elemento del sistema** elija **Actualizar**.



Eso te dará el icono de actualización predeterminado.



Añadir una acción IB

Controle el arrastre desde `UIBarButtonItem` al controlador de vista para agregar una `@IBAction`.

```
class ViewController: UIViewController {  
  
    @IBAction func refreshBarButtonItemTap(sender: UIBarButtonItem) {  
  
        print("How refreshing!")  
    }  
  
}
```

Eso es.

Notas

- Este ejemplo proviene originalmente de [esta respuesta de desbordamiento de pila](#).

Elemento de botón de barra Imagen original sin color de tinte

Siempre que `UIBarButtonItem` tenga una propiedad de imagen no nula (por ejemplo, establecida en el Creador de Interfaz).

C objetivo

```
UIBarButtonItem.image = [UIBarButtonItem.image  
imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
```

Lea `UIBarButtonItem` en línea: <https://riptutorial.com/es/ios/topic/1543/uibarbuttonitem>

Capítulo 160: UIBezierPath

Examples

Cómo aplicar el radio de la esquina a los rectángulos dibujados por UIBezierPath

Radio de esquina para los 4 bordes:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) cornerRadius: 11];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Radio de esquina para el borde superior izquierdo:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopLeft cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Radio de esquina para el borde superior derecho:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopRight cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

radio de esquina para el borde inferior izquierdo:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomLeft cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

radio de esquina para el borde inferior derecho:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomRight cornerRadii:
CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

radio de esquina para los bordes inferiores:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerBottomLeft |
UIRectCornerBottomRight cornerRadii: CGSizeMake(11, 11)];
```

```
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

radio de esquina para los bordes superiores:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRoundedRect:
CGRectMake(x,y,width,height) byRoundingCorners: UIRectCornerTopLeft | UIRectCornerTopRight
cornerRadii: CGSizeMake(11, 11)];
[rectanglePath closePath];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Cómo crear formas simples usando UIBezierPath

Para un círculo simple:



```
UIBezierPath* ovalPath = [UIBezierPath bezierPathWithOvalInRect: CGRectMake(0,0,50,50)];
[UIColor.grayColor setFill];
[ovalPath fill];
```

Rápido:

```
let ovalPath = UIBezierPath(ovalInRect: CGRect(x: 0, y: 0, width: 50, height: 50))
UIColor.grayColor().setFill()
ovalPath.fill()
```

Para un rectángulo simple:



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect: CGRectMake(0,0,50,50)];  
[UIColor.grayColor setFill];  
[rectanglePath fill];
```

Rápido:

```
let rectanglePath = UIBezierPath(rect: CGRect(x: 0, y: 0, width: 50, height: 50))  
UIColor.grayColor().setFill()  
rectanglePath.fill()
```

Para una línea simple:



```
UIBezierPath* bezierPath = [UIBezierPath bezierPath];  
[bezierPath moveToPoint: CGPointMake(x1,y1)];  
[bezierPath addLineToPoint: CGPointMake(x2,y2)];  
[UIColor.blackColor setStroke];  
bezierPath.lineWidth = 1;  
[bezierPath stroke];
```

Rápido:

```
let bezierPath = UIBezierPath()  
bezierPath.moveToPoint(CGPoint(x: x1, y: y1))  
bezierPath.addLineToPoint(CGPoint(x: x2, y: y2))  
UIColor.blackColor().setStroke()  
bezierPath.lineWidth = 1  
bezierPath.stroke()
```

Para un semicírculo:



```
CGRect ovalRect = CGRectMake(x,y,width,height);  
UIBezierPath* ovalPath = [UIBezierPath bezierPath];  
[ovalPath addArcWithCenter: CGPointMake(0, 0) radius: CGRectGetWidth(ovalRect) / 2 startAngle:  
180 * M_PI/180 endAngle: 0 * M_PI/180 clockwise: YES];  
[ovalPath addLineToPoint: CGPointMake(0, 0)];  
[ovalPath closePath];
```

```
CGAffineTransform ovalTransform = CGAffineTransformMakeTranslation(CGRectGetMidX(ovalRect),
CGRectGetMidY(ovalRect));
ovalTransform = CGAffineTransformScale(ovalTransform, 1, CGRectGetHeight(ovalRect) /
CGRectGetWidth(ovalRect));
[ovalPath applyTransform: ovalTransform];

[UIColor.grayColor setFill];
[ovalPath fill];
```

Rápido:

```
let ovalRect = CGRect(x: 0, y: 0, width: 50, height: 50)
let ovalPath = UIBezierPath()
ovalPath.addArcWithCenter(CGPoint.zero, radius: ovalRect.width / 2, startAngle: 180 *
CGFloat(M_PI)/180, endAngle: 0 * CGFloat(M_PI)/180, clockwise: true)
ovalPath.addLineToPoint(CGPoint.zero)
ovalPath.closePath()

var ovalTransform = CGAffineTransformMakeTranslation(CGRectGetMidX(ovalRect),
CGRectGetMidY(ovalRect))
ovalTransform = CGAffineTransformScale(ovalTransform, 1, ovalRect.height / ovalRect.width)
ovalPath.applyTransform(ovalTransform)

UIColor.grayColor().setFill()
ovalPath.fill()
```

Para un triángulo simple:



```
UIBezierPath* polygonPath = [UIBezierPath bezierPath];
[polygonPath moveToPoint: CGPointMake(x1, y1)];
[polygonPath addLineToPoint: CGPointMake(x2, y2)];
[polygonPath addLineToPoint: CGPointMake(x3, y2)];
[polygonPath closePath];
[UIColor.grayColor setFill];
[polygonPath fill];
```

Rápido:

```
let polygonPath = UIBezierPath()
polygonPath.moveToPoint(CGPoint(x: x1, y: y1))
polygonPath.addLineToPoint(CGPoint(x: x2, y: y2))
polygonPath.addLineToPoint(CGPoint(x: x3, y: y3))
polygonPath.closePath()
UIColor.grayColor().setFill()
polygonPath.fill()
```

UIBezierPath + AutoLayout

Para que el camino de bezier se redimensione según el marco de vista, anule el drawRect de la

vista que está dibujando el camino de bezier:

```
- (void)drawRect:(CGRect) frame
{
    UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect:
CGRectMake(CGRectGetMinX(frame), CGRectGetMinY(frame), CGRectGetWidth(frame),
CGRectGetHeight(frame))];
    [UIColor.grayColor setFill];
    [rectanglePath fill];
}
```

Cómo aplicar sombras a UIBezierPath

Considere un rectángulo simple que se dibuja por la ruta de Bezier.



```
UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect:
CGRectMake(x,y,width,height)];
[UIColor.grayColor setFill];
[rectanglePath fill];
```

Sombra básica de relleno exterior:



```
CGContextRef context = UIGraphicsGetCurrentContext();

NSShadow* shadow = [[NSShadow alloc] init];
[shadow setShadowColor: UIColor.blackColor];
[shadow setShadowOffset: CGSizeMake(7.1, 5.1)];
[shadow setShadowBlurRadius: 5];

UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect: CGRectMake(x,y,width,height)];
CGContextSaveGState(context);
CGContextSetShadowWithColor(context, shadow.shadowOffset, shadow.shadowBlurRadius,
[shadow.shadowColor CGColor]);
[UIColor.grayColor setFill];
[rectanglePath fill];
CGContextRestoreGState(context);
```

Sombra de relleno interna básica:



```
CGContextRef context = UIGraphicsGetCurrentContext();

NSShadow* shadow = [[NSShadow alloc] init];
[shadow setShadowColor: UIColor.blackColor];
[shadow setShadowOffset: CGSizeMake(9.1, -7.1)];
[shadow setShadowBlurRadius: 6];

UIBezierPath* rectanglePath = [UIBezierPath bezierPathWithRect: CGRectMake(x,y,width,height)];
[UIColor.grayColor setFill];
[rectanglePath fill];

CGContextSaveGState(context);
UIRectClip(rectanglePath.bounds);
CGContextSetShadowWithColor(context, CGSizeZero, 0, NULL);

CGContextSetAlpha(context, CGColorGetAlpha([shadow.shadowColor CGColor]));
CGContextBeginTransparencyLayer(context, NULL);
{
    UIColor* opaqueShadow = [shadow.shadowColor colorWithAlphaComponent: 1];
    CGContextSetShadowWithColor(context, shadow.shadowOffset, shadow.shadowBlurRadius,
[opaqueShadow CGColor]);
    CGContextSetBlendMode(context, kCGBlendModeSourceOut);
    CGContextBeginTransparencyLayer(context, NULL);

    [opaqueShadow setFill];
    [rectanglePath fill];

    CGContextEndTransparencyLayer(context);
}
CGContextEndTransparencyLayer(context);
CGContextRestoreGState(context);
```

Diseñando y dibujando un camino Bézier.

Este ejemplo muestra el proceso desde el diseño de la forma que desea dibujar en una vista. Se utiliza una forma específica, pero los conceptos que aprendes se pueden aplicar a cualquier forma.

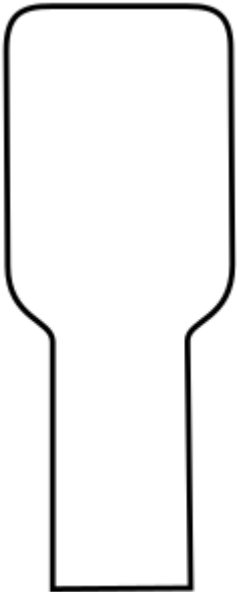
Cómo dibujar una ruta Bézier en una vista personalizada

Estos son los pasos principales:

1. Diseña el contorno de la forma que quieras.
2. Divide el trazado del contorno en segmentos de líneas, arcos y curvas.
3. Construye ese camino programáticamente.
4. Dibuje la ruta en `drawRect` o usando un `CAShapeLayer` .

Diseño de contorno de la forma

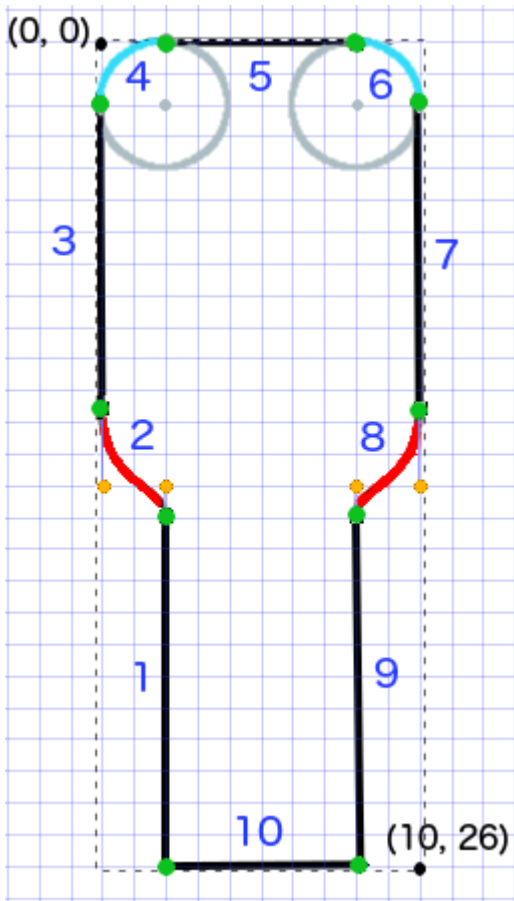
Podrías hacer cualquier cosa, pero como ejemplo he elegido la forma de abajo. Podría ser una tecla emergente en un teclado.



Divide el camino en segmentos.

Mire hacia atrás al diseño de su forma y divídalo en elementos más simples de líneas (para líneas rectas), arcos (para círculos y esquinas redondeadas) y curvas (para cualquier otra cosa).

Aquí es cómo se vería nuestro ejemplo de diseño:



- Negro son segmentos de línea
- Azul claro son segmentos de arco.
- Rojas son curvas
- Los puntos naranjas son los puntos de control de las curvas.
- Los puntos verdes son los puntos entre segmentos de camino
- Las líneas punteadas muestran el rectángulo delimitador.
- Los números de color azul oscuro son los segmentos en el orden en que se agregarán mediante programación

Construye el camino programáticamente

Comenzaremos arbitrariamente en la esquina inferior izquierda y trabajaremos en el sentido de las agujas del reloj. Usaré la cuadrícula en la imagen para obtener los valores x e y para los puntos. Escribiré todo aquí, pero por supuesto que no harías eso en un proyecto real.

El proceso básico es:

1. Crear un nuevo `UIBezierPath`
2. Elija un punto de partida en la ruta con `moveToPoint`
3. Añadir segmentos a la ruta.

- **línea:** `addLineToPoint`
- **arc:** `addArcWithCenter`
- **curva:** `addCurveToPoint`

4. Cierra el camino con `closePath`

Aquí está el código para hacer la ruta en la imagen de arriba.

```
func createBezierPath() -> UIBezierPath {

    // create a new path
    let path = UIBezierPath()

    // starting point for the path (bottom left)
    path.moveToPoint(CGPoint(x: 2, y: 26))

    // *****
    // ***** Left side *****
    // *****

    // segment 1: line
    path.addLineToPoint(CGPoint(x: 2, y: 15))

    // segment 2: curve
    path.addCurveToPoint(CGPoint(x: 0, y: 12), // ending point
        controlPoint1: CGPoint(x: 2, y: 14),
        controlPoint2: CGPoint(x: 0, y: 14))

    // segment 3: line
    path.addLineToPoint(CGPoint(x: 0, y: 2))

    // *****
    // ***** Top side *****
    // *****

    // segment 4: arc
    path.addArcWithCenter(CGPoint(x: 2, y: 2), // center point of circle
        radius: 2, // this will make it meet our path line
        startAngle: CGFloat(M_PI), // π radians = 180 degrees = straight left
        endAngle: CGFloat(3*M_PI_2), // 3π/2 radians = 270 degrees = straight up
        clockwise: true) // startAngle to endAngle goes in a clockwise direction

    // segment 5: line
    path.addLineToPoint(CGPoint(x: 8, y: 0))

    // segment 6: arc
    path.addArcWithCenter(CGPoint(x: 8, y: 2),
        radius: 2,
        startAngle: CGFloat(3*M_PI_2), // straight up
        endAngle: CGFloat(0), // 0 radians = straight right
        clockwise: true)

    // *****
    // ***** Right side *****
    // *****

    // segment 7: line
    path.addLineToPoint(CGPoint(x: 10, y: 12))

    // segment 8: curve
    path.addCurveToPoint(CGPoint(x: 8, y: 15), // ending point
        controlPoint1: CGPoint(x: 10, y: 14),
        controlPoint2: CGPoint(x: 8, y: 14))
}
```

```

// segment 9: line
path.lineTo(CGPoint(x: 8, y: 26))

// *****
// **** Bottom side ****
// *****

// segment 10: line
path.closePath() // draws the final line to close the path

return path
}

```

Nota: parte del código anterior se puede reducir agregando una línea y un arco en un solo comando (ya que el arco tiene un punto de inicio implícito). Vea [aquí](#) para más detalles.

Dibujar el camino

Podemos dibujar el camino ya sea en una capa o en `drawRect`.

Método 1: dibujar ruta en una capa

Nuestra clase personalizada se ve así. `CAShapeLayer` nuestra ruta Bezier a un nuevo `CAShapeLayer` cuando se inicializa la vista.

```

import UIKit
class MyCustomView: UIView {

    override init(frame: CGRect) {
        super.init(frame: frame)
        setup()
    }

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        setup()
    }

    func setup() {

        // Create a CAShapeLayer
        let shapeLayer = CAShapeLayer()

        // The Bezier path that we made needs to be converted to
        // a CGPath before it can be used on a layer.
        shapeLayer.path = createBezierPath().CGPath

        // apply other properties related to the path
        shapeLayer.strokeColor = UIColor.blueColor().CGColor
        shapeLayer.fillColor = UIColor.whiteColor().CGColor
        shapeLayer.lineWidth = 1.0
        shapeLayer.position = CGPoint(x: 10, y: 10)

        // add the new layer to our custom view
        self.layer.addSublayer(shapeLayer)
    }
}

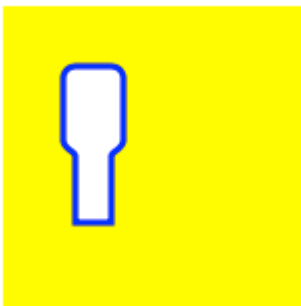
```

```
func createBezierPath() -> UIBezierPath {  
  
    // see previous code for creating the Bezier path  
}  
}
```

Y creando nuestra vista en el Controlador de Vista así.

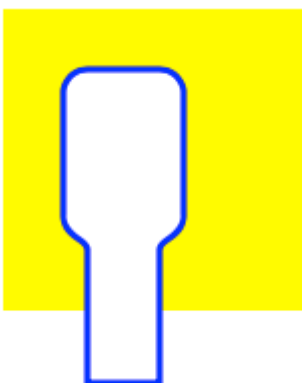
```
override func viewDidLoad() {  
    super.viewDidLoad()  
  
    // create a new UIView and add it to the view controller  
    let myView = MyCustomView()  
    myView.frame = CGRect(x: 100, y: 100, width: 50, height: 50)  
    myView.backgroundColor = UIColor.yellowColor()  
    view.addSubview(myView)  
  
}
```

Obtenemos...



Hmm, eso es un poco pequeño porque codifiqué todos los números. Puedo escalar el tamaño de la ruta, sin embargo, así:

```
let path = createBezierPath()  
let scale = CGAffineTransformMakeScale(2, 2)  
path.applyTransform(scale)  
shapeLayer.path = path.CGPath
```



Método 2: dibujar ruta en `drawRect`

Usar `drawRect` es más lento que dibujar en la capa, por lo que este no es el método recomendado si no lo necesita.

Aquí está el código revisado para nuestra vista personalizada:

```
import UIKit
class MyCustomView: UIView {

    override func drawRect(rect: CGRect) {

        // create path (see previous code)
        let path = createBezierPath()

        // fill
        let fillColor = UIColor.whiteColor()
        fillColor.setFill()

        // stroke
        path.lineWidth = 1.0
        let strokeColor = UIColor.blueColor()
        strokeColor.setStroke()

        // Move the path to a new location
        path.applyTransform(CGAffineTransformMakeTranslation(10, 10))

        // fill and stroke the path (always do these last)
        path.fill()
        path.stroke()

    }

    func createBezierPath() -> UIBezierPath {

        // see previous code for creating the Bezier path
    }

}
```

lo que nos da el mismo resultado ...



Estudio adicional

Excelentes artículos para entender los caminos de Bezier.

- [Pensar como un camino Bézier](#) (todo lo que he leído de este autor es bueno y la inspiración para mi ejemplo anterior vino de aquí).

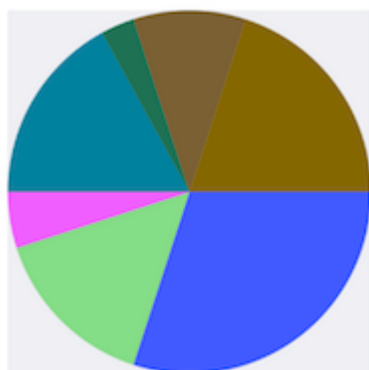
- [Coding Math: Episodio 19 - Curvas de Bezier](#) (divertidas y buenas ilustraciones visuales)
- [Curvas de Bezier](#) (cómo se usan en aplicaciones gráficas)
- [Curvas de Bezier](#) (buena descripción de cómo se derivan las fórmulas matemáticas)

Notas

- Este ejemplo proviene originalmente de [esta respuesta de desbordamiento de pila](#) .
- En sus proyectos reales, probablemente no debería usar números codificados, sino obtener los tamaños desde los límites de su vista.

Vista circular y vista de columna con UIBezierPath

- vista de tarta



```
- (void)drawRect:(CGRect)rect {  
  
    NSArray *data = @[30, 15, 5, 17, 3, 10, 20];  
  
    // 1. context  
    CGContextRef cxtRef = UIGraphicsGetCurrentContext();  
  
    CGPoint center = CGPointMake(150, 150);  
    CGFloat radius = 150;  
    __block CGFloat startAngle = 0;  
    [data enumerateObjectsUsingBlock:^(NSNumber * _Nonnull obj, NSUInteger idx, BOOL *  
    _Nonnull stop) {  
  
        // 2. create path  
        CGFloat endAngle = obj.floatValue / 100 * M_PI * 2 + startAngle;  
        UIBezierPath *circlePath = [UIBezierPath bezierPathWithArcCenter:center radius:radius  
startAngle:startAngle endAngle:endAngle clockwise:YES];  
        [circlePath addLineToPoint:center];  
  
        // 3. add path  
        CGContextAddPath(cxtRef, circlePath.CGPath);  
  
        // set color  
        [[UIColor colorWithRed:((float)arc4random_uniform(256) / 255.0)  
green:((float)arc4random_uniform(256) / 255.0) blue:((float)arc4random_uniform(256) / 255.0)  
alpha:1.0] setFill];  
  
        // 4. render  
        CGContextDrawPath(cxtRef, kCGPathFill);  
    }];  
}
```

```

        // reset angle
        startAngle = endAngle;
    }];
}

```

```

override func draw(_ rect: CGRect) {
    // define data to create pie chart
    let data: [Int] = [30, 15, 5, 17, 3, 10, 20]

    // 1. find center of draw rect
    let center: CGPoint = CGPoint(x: rect.midX, y: rect.midY)

    // 2. calculate radius of pie
    let radius = min(rect.width, rect.height) / 2.0

    var startAngle: CGFloat = 0.0
    for value in data {

        // 3. calculate end angle for slice
        let endAngle = CGFloat(value) / 100.0 * CGFloat.pi * 2.0 + startAngle

        // 4. create UIBezierPath for slide
        let circlePath = UIBezierPath(arcCenter: center, radius: radius, startAngle: startAngle,
endAngle: endAngle, clockwise: true)

        // 5. add line to center to close path
        circlePath.addLine(to: center)

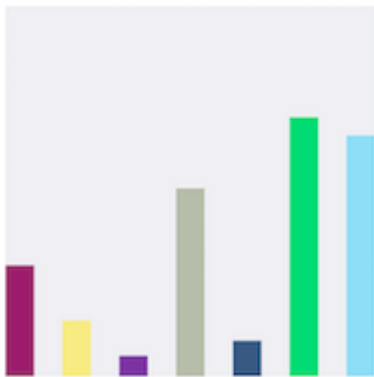
        // 6. set fill color for current slice
        UIColor(red: (CGFloat(arc4random_uniform(256)) / 255.0), green:
(CGFloat(arc4random_uniform(256)) / 255.0), blue: (CGFloat(arc4random_uniform(256)) / 255.0),
alpha: 1.0).setFill()

        // 7. fill slice path
        circlePath.fill()

        // 8. set end angle as start angle for next slice
        startAngle = endAngle
    }
}

```

- vista de columna



```

- (void)drawRect:(CGRect)rect {

```

```

NSArray *data = @[300, 150.65, 55.3, 507.7, 95.8, 700, 650.65];

// 1.
CGContextRef cxtRef = UIGraphicsGetCurrentContext();

NSInteger columnCount = 7;
CGFloat width = self.bounds.size.width / (columnCount + columnCount - 1);
for (NSInteger i = 0; i < columnCount; i++) {

    // 2.
    CGFloat height = [data[i] floatValue] / 1000 * self.bounds.size.height; // floatValue
    CGFloat x = 0 + width * (2 * i);
    CGFloat y = self.bounds.size.height - height;
    UIBezierPath *rectPath = [UIBezierPath bezierPathWithRect:CGRectMake(x, y, width,
height)];
    CGContextAddPath(cxtRef, rectPath.CGPath);

    // 3.
    [[UIColor colorWithRed:((float)arc4random_uniform(256) / 255.0)
green:((float)arc4random_uniform(256) / 255.0) blue:((float)arc4random_uniform(256) / 255.0)
alpha:1.0] setFill];
    CGContextDrawPath(cxtRef, kCGPathFill);
}
}

```

```

override func draw(_ rect: CGRect) {
    // define data for chart
    let data: [CGFloat] = [300, 150.65, 55.3, 507.7, 95.8, 700, 650.65]

    // 1. calculate number of columns
    let columnCount = data.count

    // 2. calculate column width
    let columnWidth = rect.width / CGFloat(columnCount + columnCount - 1)

    for (columnIndex, value) in data.enumerated() {
        // 3. calculate column height
        let columnHeight = value / 1000.0 * rect.height

        // 4. calculate column origin
        let columnOrigin = CGPoint(x: (columnWidth * 2.0 * CGFloat(columnIndex)), y:
(rect.height - columnHeight))

        // 5. create path for column
        let columnPath = UIBezierPath(rect: CGRect(origin: columnOrigin, size: CGSize(width:
columnWidth, height: columnHeight)))

        // 6. set fill color for current column
        UIColor(red: (CGFloat(arc4random_uniform(256)) / 255.0), green:
(CGFloat(arc4random_uniform(256)) / 255.0), blue: (CGFloat(arc4random_uniform(256)) / 255.0),
alpha: 1.0).setFill()

        // 7. fill column path
        columnPath.fill()
    }
}

```

Lea UIBezierPath en línea: <https://riptutorial.com/es/ios/topic/3186/uiBezierPath>

Capítulo 161: UIButton

Introducción

UIButton : **UIControl** intercepta eventos táctiles y envía un mensaje de acción a un objeto de destino cuando se toca. Puede configurar el título, la imagen y otras propiedades de apariencia de un botón. Además, puede especificar una apariencia diferente para cada estado del botón.

Observaciones

Tipos de botones

El tipo de un botón define su apariencia y comportamiento básicos. Después de crear un botón, no puede cambiar su tipo. Los tipos de botones más utilizados son los tipos Personalizado y Sistema, pero use los otros tipos cuando sea apropiado

- UIButtonTypeCustom

```
No button style.
```

- UIButtonTypeSystem

```
A system style button, such as those shown in navigation bars and toolbars.
```

- UIButtonTypeDetailDisclosure

```
A detail disclosure button.
```

- UIButtonTypeInfoLight

```
An information button that has a light background.
```

- UIButtonTypeInfoDark

```
An information button that has a dark background.
```

- UIButtonTypeContactAdd

```
A contact add button.
```

Al crear un botón personalizado, es decir, un botón con el tipo personalizado, el marco del botón se establece inicialmente en (0, 0, 0, 0). Antes de agregar el botón a su interfaz, debe actualizar el marco a un valor más apropiado.

Examples

Creando un UIButton

Los UIButtons se pueden inicializar en un marco:

Rápido

```
let button = UIButton(frame: CGRect(x: x, y: y, width: width, height: height))
```

C objetivo

```
UIButton *button = [[UIButton alloc] initWithFrame:CGRectMake(x, y, width, height)];
```

Un tipo específico de UIButton se puede crear así:

Rápido

```
let button = UIButton(type: .Custom)
```

C objetivo

```
UIButton *button = [UIButton buttonWithType:UIButtonTypeCustom];
```

donde `type` es un `UIButtonType` :

```
enum UIButtonType : Int {
    case Custom
    case System
    case DetailDisclosure
    case InfoLight
    case InfoDark
    case ContactAdd
    static var RoundedRectangle: UIButtonType { get }
}
```

Establecer título

Rápido

```
button.setTitle(titleString, forState: controlState)
```

C objetivo

```
[button setTitle:(NSString *) forState:(UIControlState)];
```

Para configurar el título predeterminado en "¡Hola, mundo!"

Rápido

```
button.setTitle("Hello, World!", forState: .normal)
```

C objetivo

```
[button setTitle:@"Hello, World!" forState:UIControlStateNormal];
```

Establecer el color del título

```
//Swift
button.setTitleColor(color, forState: controlState)

//Objective-C
[button setTitleColor:(nullable UIColor *) forState:(UIControlState)];
```

Para establecer el color del título en azul

```
//Swift
button.setTitleColor(.blue, for: .normal)

//Objective-C
[button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal]
```

Alineación horizontal de contenidos.

Rápido

```
//Align contents to the left of the frame
button.contentHorizontalAlignment = .left

//Align contents to the right of the frame
button.contentHorizontalAlignment = .right

//Align contents to the center of the frame
button.contentHorizontalAlignment = .center

//Make contents fill the frame
button.contentHorizontalAlignment = .fill
```

C objetivo

```
//Align contents to the left
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentLeft;

//Align contents to the right
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentRight;

//Align contents to the center
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentCenter;

//Align contents to fill the frame
button.contentHorizontalAlignment = UIControlContentHorizontalAlignmentFill;
```

Obtener la etiqueta del título

La etiqueta del título subyacente, si existe, puede ser recuperada usando

Rápido

```
var label: UILabel? = button.titleLabel
```

C objetivo

```
UILabel *label = button.titleLabel;
```

Esto se puede usar para establecer la fuente de la etiqueta del título, por ejemplo

Rápido

```
button.titleLabel?.font = UIFont.boldSystemFontOfSize(12)
```

C objetivo

```
button.titleLabel.font = [UIFont boldSystemFontOfSize:12];
```

Deshabilitando un UIButton

Un botón puede ser desactivado por

Rápido

```
myButton.isEnabled = false
```

C objetivo:

```
myButton.enabled = NO;
```

El botón se pondrá gris:

Button

Si no desea que la apariencia del botón cambie cuando está deshabilitado, establezca

`adjustsImageWhenDisabled` a `false` / `NO`

Agregar una acción a un UIButton a través del Código (programáticamente)

Para agregar un método a un botón, primero cree un método de acción:

C objetivo

```
-(void)someButtonAction:(id)sender {
    // sender is the object that was tapped, in this case its the button.
    NSLog(@"Button is tapped");
}
```

Rápido

```
func someButtonAction() {
    print("Button is tapped")
}
```

Ahora para agregar este método de acción a su botón, debe escribir la siguiente línea de código:

C objetivo

```
[yourButtonInstance addTarget:self action:@selector(someButtonAction)
forControlEvents:UIControlEventTouchUpInside];
```

Rápido

```
yourButtonInstance.addTarget(self, action: #selector(someButtonAction), forControlEvents:
.TouchUpInside)
```

Para el parámetro ControlEvents, todos los miembros de `ENUM UIControlEvents` son válidos.

Fuente de ajuste

Rápido

```
myButton.titleLabel?.font = UIFont(name: "YourFontName", size: 20)
```

C objetivo

```
myButton.titleLabel.font = [UIFont fontWithName:@"YourFontName" size:20];
```

Adjuntar un método a un botón

Para agregar un método a un botón, primero cree un método de acción:

C objetivo

```
-(void) someButtonAction{
    NSLog(@"Button is tapped");
}
```

Rápido


```
func someButtonAction() {
    print("Button is tapped")
}
```

Ahora para agregar este método de acción a su botón, debe escribir la siguiente línea de código:

C objetivo

```
[yourButtonInstance addTarget:self action:@selector(someButtonAction)
forControlEvents:UIControlEventTouchUpInside];
```

Rápido

```
yourButtonInstance.addTarget(self, action: #selector(someButtonAction), forControlEvents:
.touchUpInside)
```

Para `ControlEvents`, todos los miembros de `ENUM UIControlEvents` son válidos.

Obtenga el tamaño de UIButton basado estrictamente en su texto y fuente

Para obtener el tamaño exacto de un texto de UIButton basado en su fuente, use la función `intrinsicContentSize`.

Rápido

```
button.intrinsicContentSize.width
```

C objetivo

```
button.intrinsicContentSize.width;
```

Establecer imagen

Rápido

```
button.setImage(UIImage(named:"test-image"), forState: .normal)
```

C objetivo

```
[self.button setImage:[UIImage imageNamed:@"test-image"] forState:UIControlStateNormal];
```

Estados de control multiples

También puede establecer una imagen para varios `UIControlStates`, por ejemplo, para establecer

la misma imagen para el estado `Selected` y `Highlighted` :

Rápido

```
button.setImage(UIImage(named:"test-image"), forState:[.selected, .highlighted])
```

C objetivo

```
[self.button setImage:[UIImage imageNamed:@"test-image"]  
forState:UIControlStateSelected|:UIControlStateHighlighted];
```

Lea `UIButton` en línea: <https://riptutorial.com/es/ios/topic/516/uibutton>

Capítulo 162: UICollectionView

Examples

Crear una vista de colección programáticamente

Rápido

```
func createCollectionView() {
    let layout: UICollectionViewFlowLayout = UICollectionViewFlowLayout()
    let collectionView = UICollectionView(frame: CGRect(x: 0, y: 0, width: view.frame.width,
height: view.frame.height), collectionViewLayout: layout)
    collectionView.dataSource = self
    collectionView.delegate = self
    view.addSubview(collectionView)
}
```

C objetivo

```
- (void)createCollectionView {
    UICollectionViewFlowLayout *layout = [[UICollectionViewFlowLayout alloc] init];
    UICollectionView *collectionView = [[UICollectionView alloc] initWithFrame:CGRectMake(0,
0, self.view.frame.size.width, self.view.frame.size.height) collectionViewLayout:layout];
    [collectionView setDataSource:self];
    [collectionView setDelegate:self];
    [self.view addSubview:collectionView];
}
```

Swift - UICollectionViewDelegateFlowLayout

```
// MARK: - UICollectionViewDelegateFlowLayout
extension ViewController: UICollectionViewDelegateFlowLayout {
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, sizeForItemAtIndexPath indexPath: NSIndexPath) -> CGSize {
        return CGSize(width: 50, height: 50)
    }
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, insetForSectionAtIndex section: Int) -> UIEdgeInsets {
        return UIEdgeInsets(top: 5, left: 5, bottom: 5, right: 5)
    }
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumLineSpacingForSectionAtIndex section: Int) -> CGFloat {
        return 5.0
    }
    func collectionView(collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumInteritemSpacingForSectionAtIndex section: Int) -> CGFloat {
        return 5.0
    }
}
```

Crear un UICollectionView

Inicialice un `UICollectionView` con un marco `CGRect` :

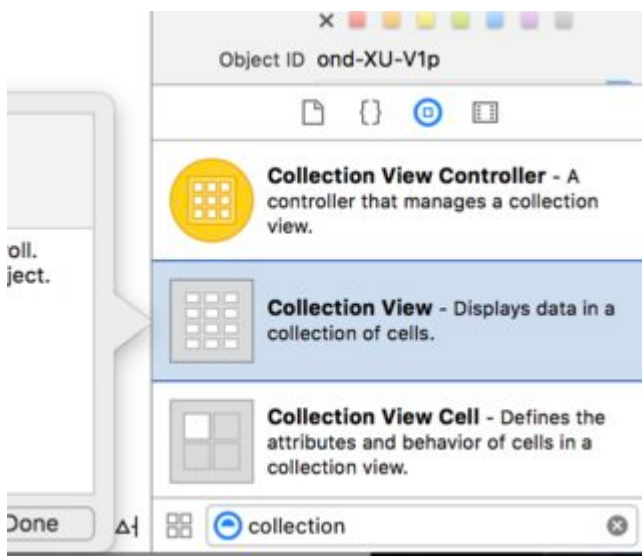
Rápido:

```
let collection = UICollectionView(frame: CGRect(x: 0, y: 0, width: 200, height: 21))
```

C objetivo:

```
UICollectionView *collection = [[UICollectionView alloc] initWithFrame:CGRectMake(0, 0, 200, 21)];
```

También puede crear un `UICollectionView` en Interface Builder



UICollectionView - Fuente de datos

Cada vista de colección debe tener un objeto `DataSource` . El objeto `DataSource` es el contenido que su aplicación mostrará dentro de `UICollectionView` . Como mínimo, todos los objetos `DataSource` deben implementar los métodos `collectionView:numberOfItemsInSection:` y `collectionView:cellForItemAtIndexPath:`

Métodos requeridos

Rápido

```
func collectionView(collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
    // Return how many items in section
    let sectionArray = _data[section]
    return sectionArray.count
}

func collectionView(collectionView: UICollectionView, cellForItemAtIndexPath indexPath: NSIndexPath) -> UICollectionViewCell {

    let cell = collectionView.dequeueReusableCellWithReuseIdentifier(MyCellID)
    // If you use a custom cell class then cast the cell returned, like:
    // as! MyCollectionViewCellClass
```

```

// or you will have errors when you try to use features of that class.

//Customize your cell here, default UICollectionViewCells do not contain any inherent
//text or image views (like UITableView), but some could be added,
//or a custom UICollectionViewCell sub-class could be used
return cell
}

```

C objetivo

```

- (NSInteger)collectionView:(UICollectionView*)collectionView
numberOfItemsInSection:(NSInteger)section {
    // Return how many items in section
    NSArray *sectionArray = [_data objectAtIndex:section];
    return [sectionArray count];
}

- (UICollectionViewCell *)collectionView:(UICollectionView *)collectionView
cellForItemAtIndexPath:(NSIndexPath *)indexPath {
    // Return a cell
    UICollectionViewCell *newCell = [self.collectionView
                                     dequeueReusableCellWithReuseIdentifier:MyCellID
                                     forIndexPath:indexPath];

    //Customize your cell here, default UICollectionViewCells do not contain any inherent
    //text or image views (like UITableView), but some could be added,
    //or a custom UICollectionViewCell sub-class could be used
    return newCell;
}

```

Ejemplo Swift básico de una vista de colección

Crear un nuevo proyecto

Puede ser solo una aplicación de vista única.

Agrega el código

Cree un nuevo archivo Cocoa Touch Class (Archivo > Nuevo > Archivo ... > iOS > Clase Cocoa Touch). `MyCollectionViewCell`. Esta clase contendrá los puntos de venta de las vistas que agregue a su celda en el guión gráfico.

```

import UIKit
class MyCollectionViewCell: UICollectionViewCell {

    @IBOutlet weak var myLabel: UILabel!
}

```

Vamos a conectar esta salida más tarde.

Abra `ViewController.swift` y asegúrese de tener el siguiente contenido:

```

import UIKit
class ViewController: UIViewController, UICollectionViewDataSource, UICollectionViewDelegate {

    let reuseIdentifier = "cell" // also enter this string as the cell identifier in the
    storyboard
    var items = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14",
"15", "16", "17", "18", "19", "20", "21", "22", "23", "24", "25", "26", "27", "28", "29",
"30", "31", "32", "33", "34", "35", "36", "37", "38", "39", "40", "41", "42", "43", "44",
"45", "46", "47", "48"]

    // MARK: - UICollectionViewDataSource protocol

    // tell the collection view how many cells to make
    func collectionView(collectionView: UICollectionView, numberOfItemsInSection section: Int)
-> Int {
        return self.items.count
    }

    // make a cell for each cell index path
    func collectionView(collectionView: UICollectionView, cellForItemAtIndexPath indexPath:
NSIndexPath) -> UICollectionViewCell {

        // get a reference to our storyboard cell
        let cell = collectionView.dequeueReusableCellWithReuseIdentifier(reuseIdentifier,
forIndexPath: indexPath) as! MyCollectionViewCell

        // Use the outlet in our custom class to get a reference to the UILabel in the cell
        cell.myLabel.text = self.items[indexPath.item]
        cell.backgroundColor = UIColor.yellowColor() // make cell more visible in our example
project

        return cell
    }

    // MARK: - UICollectionViewDelegate protocol

    func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath:
NSIndexPath) {
        // handle tap events
        print("You selected cell #\(indexPath.item)!")
    }
}

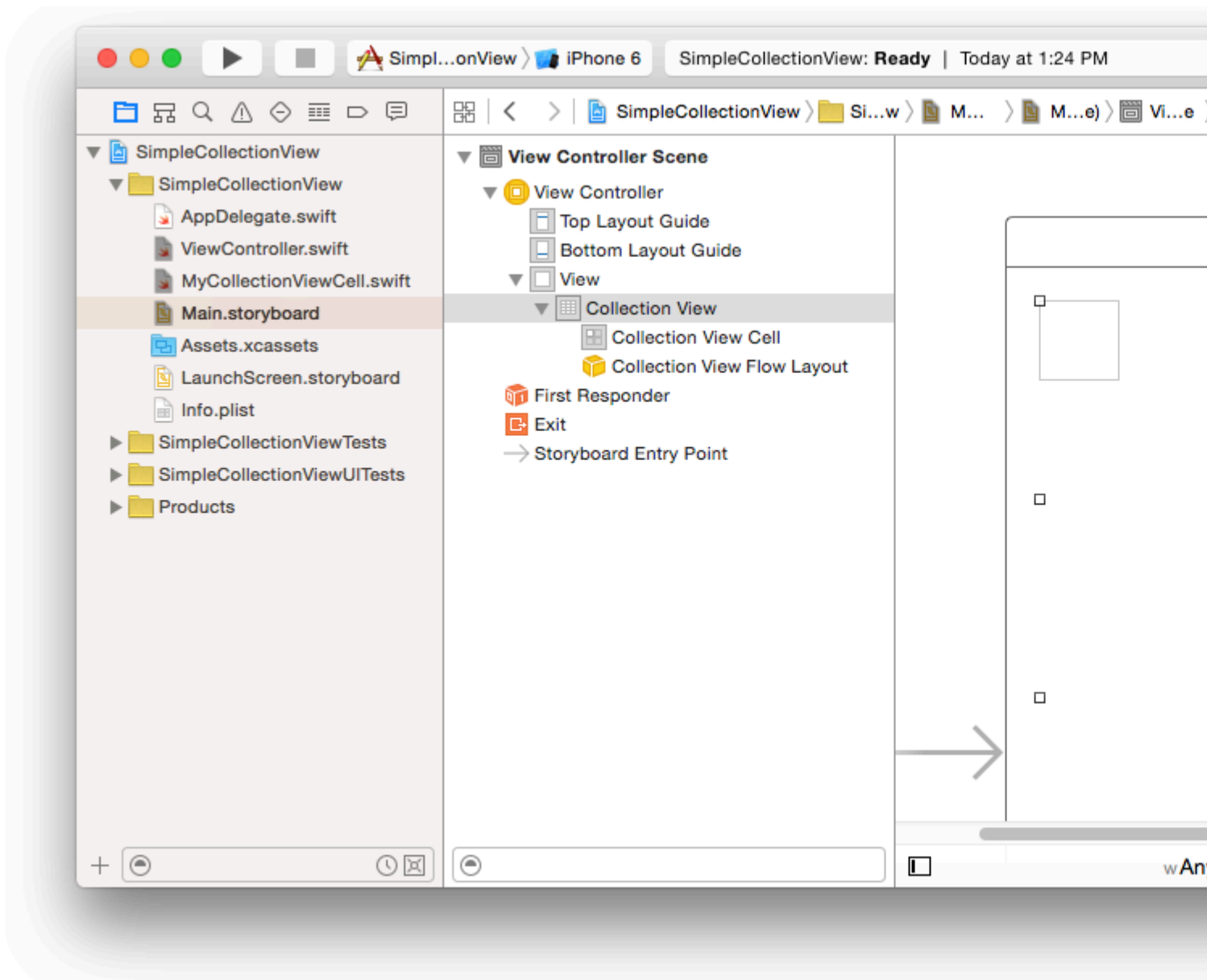
```

Notas

- `UICollectionViewDataSource` y `UICollectionViewDelegate` son los protocolos que sigue la vista de colección. También puede agregar el protocolo `UICollectionViewDelegateFlowLayout` para cambiar el tamaño de las vistas mediante programación, pero no es necesario.
- Solo estamos colocando cadenas simples en nuestra cuadrícula, pero ciertamente podrías hacer imágenes más tarde.

Configurar el guión gráfico

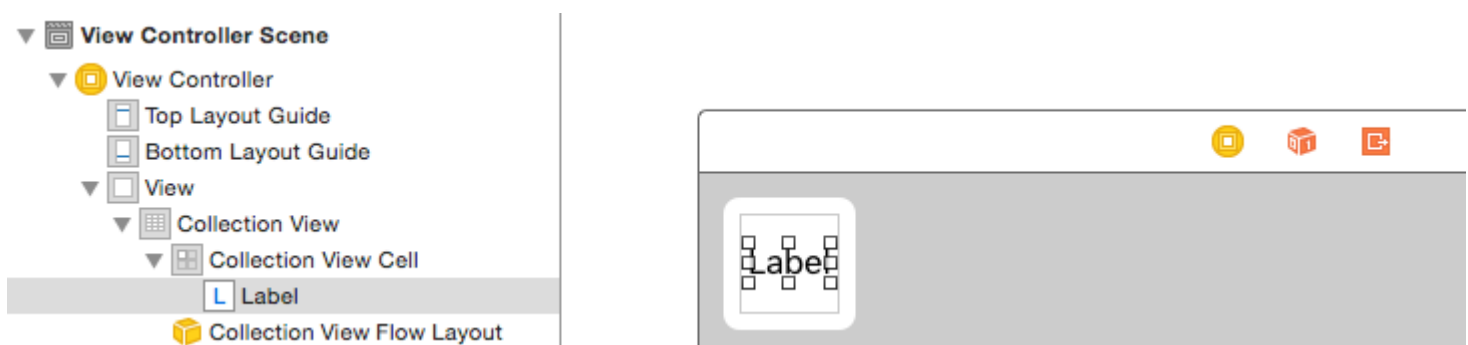
Arrastre una vista de colección al controlador de vista en su guión gráfico. Puede agregar restricciones para que llene la vista principal si lo desea.



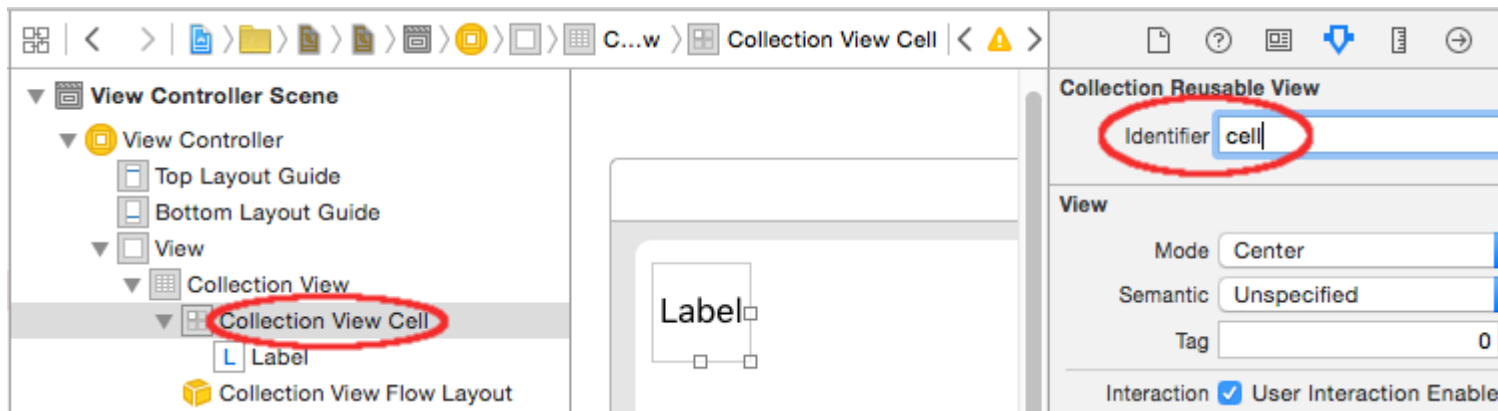
Asegúrese de que sus valores predeterminados en el inspector de atributos también estén

- Artículos: 1
- Diseño: Flujo

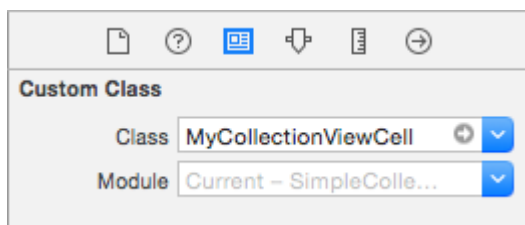
El pequeño cuadro en la parte superior izquierda de la vista de colección es una celda de vista de colección. Lo usaremos como nuestro prototipo de célula. Arrastre una etiqueta a la celda y céntrela. Puede cambiar el tamaño de los bordes de las celdas y agregar restricciones para centrar la etiqueta si lo desea.



Escriba "celda" (sin comillas) en el cuadro Identificador del inspector de atributos para la celda de vista de colección. Tenga en cuenta que este es el mismo valor que `let reuseIdentifier = "cell"` en `ViewController.swift`.

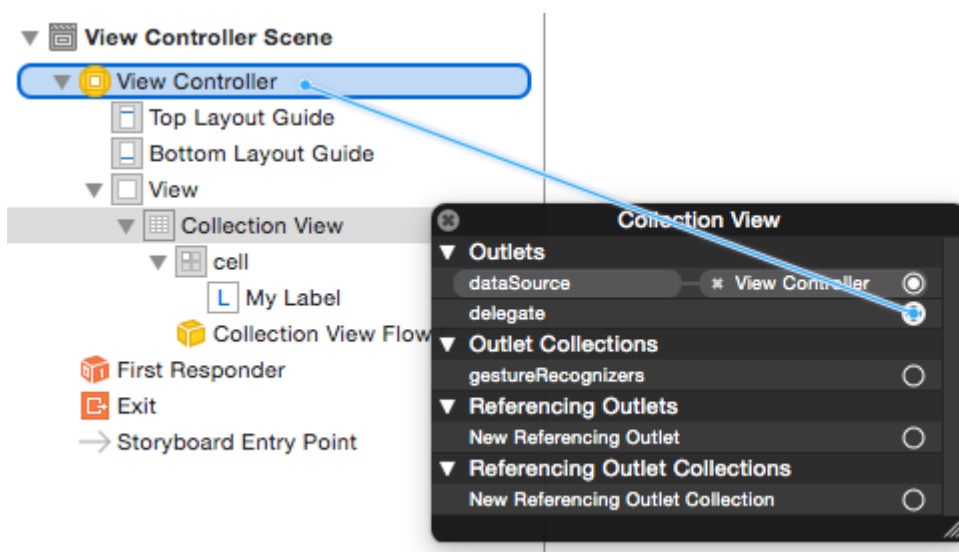


Y en el inspector de identidad para la celda, establezca el nombre de la clase en `MyCollectionViewCell`, nuestra clase personalizada que hicimos.



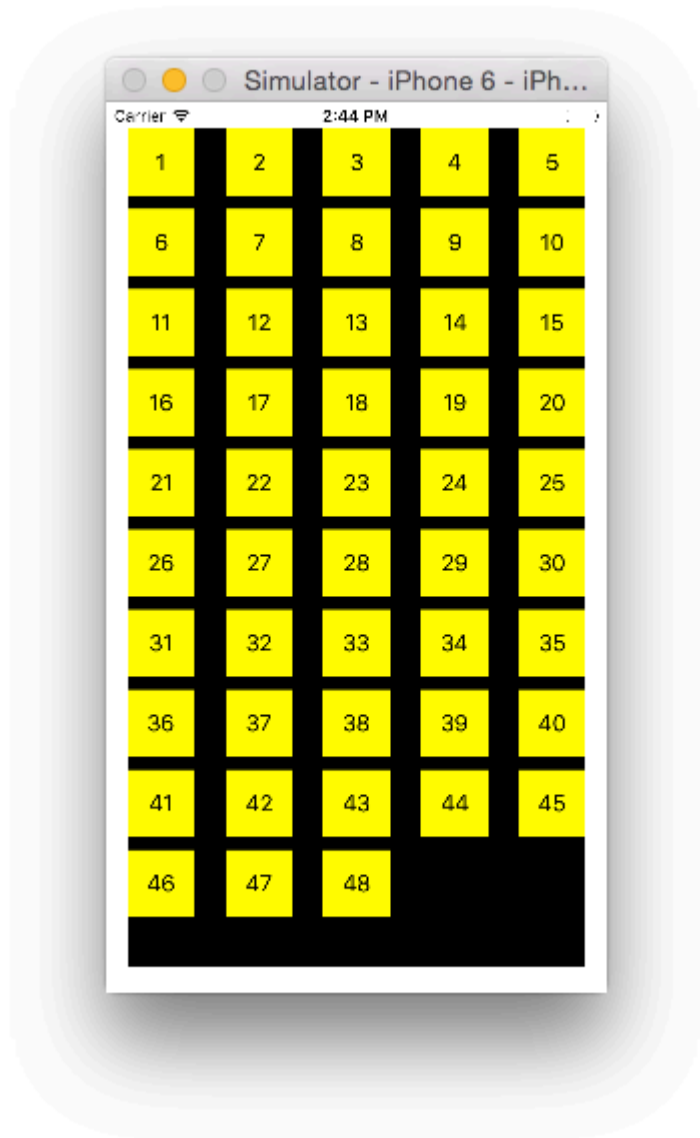
Conectar los puntos de venta

- Enganche la etiqueta en la celda de colección a `myLabel` en la clase `MyCollectionViewCell`. (Puedes presionar **Control y arrastrar**.)
- Enganche el `delegate` y el `dataSource` de `dataSource` de la vista de la `dataSource` al controlador de vista. (Haga clic con el botón derecho en Vista de colección en el Esquema del documento. Luego, haga clic y arrastre la flecha más hasta el Controlador de vista).



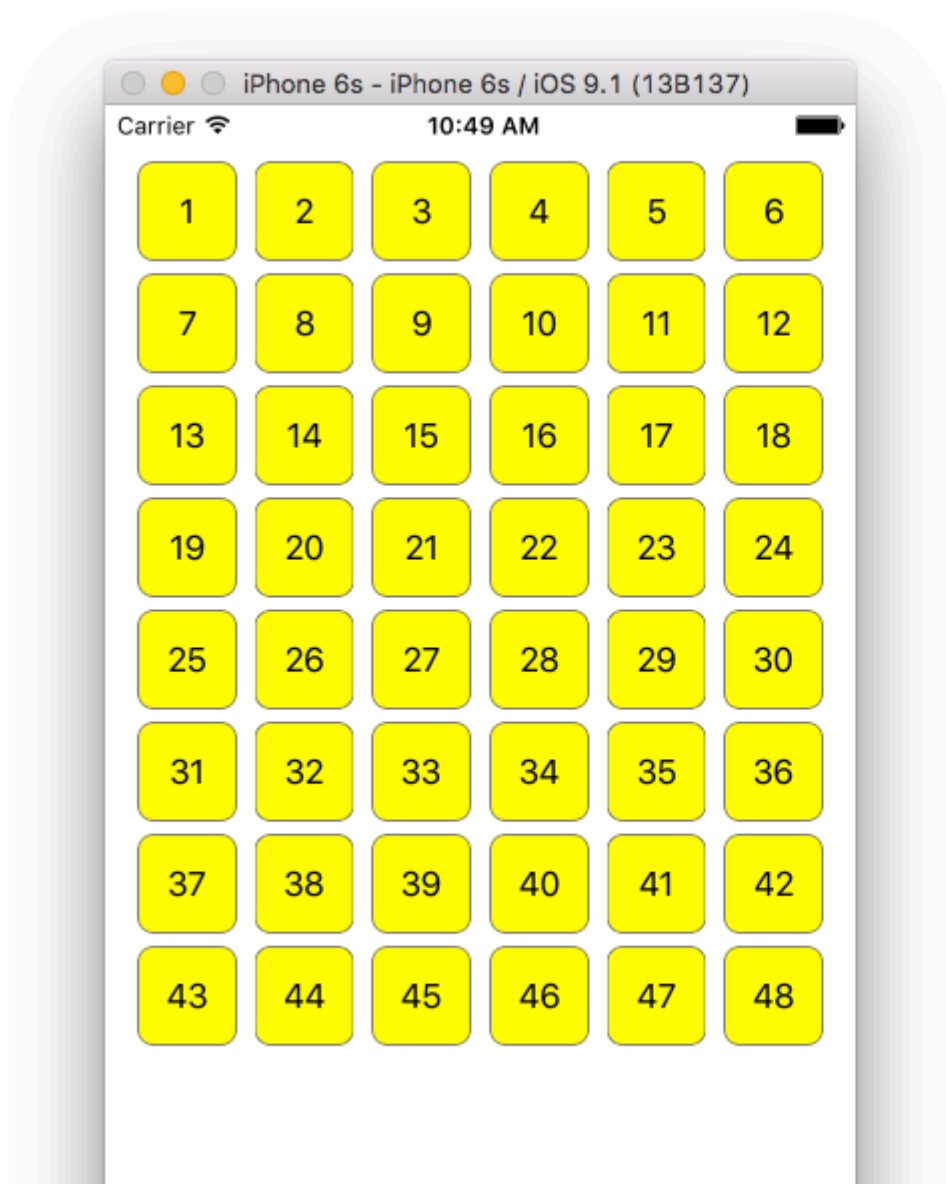
Terminado

Así es como se ve después de agregar restricciones para centrar la Etiqueta en la celda y fijar la Vista de Colección en las paredes del padre.



Haciendo mejoras

Si desea realizar mejoras en la apariencia, [consulte la publicación original de la que proviene este ejemplo](#) .



Estudio adicional

- [Un simple tutorial de UICollectionView](#)
- [UICollectionView Tutorial Part 1: Getting Started](#)
- [UICollectionView Tutorial Parte 2: Vistas reutilizables y selección de celdas](#)

Realización de actualizaciones por lotes

Puede animar cambios complejos a su vista de colección usando el método `performBatchUpdates`. Dentro del bloque de actualización, puede especificar varias modificaciones para que se animen a la vez.

```
collectionView.performBatchUpdates({  
    // Perform updates  
}, nil)
```

Dentro del bloque de actualización, puede realizar inserciones, eliminaciones, movimientos y

recargas. Aquí es cómo determinar qué indexPath utilizar:

Tipo	NSIndexPath
Inserción	Índice en nueva matriz
Supresión	Índice en matriz antigua
Movimiento	desde: matriz antigua, a: nueva matriz
Recargar	ya sea nueva o antigua matriz (no debería importar)

Solo debe llamar a recargar en celdas que no se hayan movido, pero su contenido ha cambiado. Es importante tener en cuenta que un movimiento no actualizará el contenido de una celda, sino que solo moverá su ubicación.

Para verificar que la actualización por lotes se realizará correctamente, asegúrese de que el conjunto de indexPaths para `deletion`, `move-from` y `reload` sea único, y que el conjunto de indexPaths para `insertion`, `move-to` y `reload` sea único.

Aquí hay un ejemplo de una actualización por lotes adecuada:

```
let from = [1, 2, 3, 4, 5]
let to = [1, 3, 6, 4, 5]

collectionView.performBatchUpdates({
    collectionView.insertItemsAtIndexPaths([NSIndexPath(forItem: 2, inSection: 0)])
    collectionView.deleteItemsAtIndexPaths([NSIndexPath(forItem: 1, inSection: 0)])
    collectionView.moveItemAtIndexPath(NSIndexPath(forItem: 2, inSection: 0),
                                        toIndexPath: NSIndexPath(forItem: 1, inSection: 0))
}, nil)
```

UICollectionViewDelegate configuración y selección de elementos

A veces, si una acción debe estar vinculada a la selección de celda de una vista de colección, debe implementar el protocolo `UICollectionViewDelegate`.

Digamos que la vista de colección está dentro de un `UIViewController MyViewController`.

C objetivo

En su `MyViewController.h` declara que implementa el protocolo `UICollectionViewDelegate`, como se muestra a continuación

```
@interface MyViewController : UIViewController <UICollectionViewDelegate, .../* previous existing delegate, as UICollectionViewDataSource */>
```

Rápido

En su `MyViewController.swift` agregue lo siguiente

```
class MyViewController : UICollectionViewDelegate {  
}
```

El método que se debe implementar es

C objetivo

```
-(void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath *)indexPath  
{  
}
```

Rápido

```
func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath: NSIndexPath)  
{  
}
```

A modo de ejemplo, podemos establecer el color de fondo de la celda seleccionada en verde.

C objetivo

```
-(void)collectionView:(UICollectionView *)collectionView didSelectItemAtIndexPath:(NSIndexPath *)indexPath  
{  
    UICollectionViewCell* cell = [collectionView cellForItemAtIndexPath:indexPath];  
    cell.backgroundColor = [UIColor greenColor];  
}
```

Rápido

```
class MyViewController : UICollectionViewDelegate {  
    func collectionView(collectionView: UICollectionView, didSelectItemAtIndexPath indexPath: NSIndexPath)  
    {  
        var cell : UICollectionViewCell = collectionView.cellForItemAtIndexPath(indexPath)!  
        cell.backgroundColor = UIColor.greenColor()  
    }  
}
```

Administrar la vista de colección múltiple con DataSource y Flowlayout

Aquí estamos administrando la colección múltiple, delegando métodos con didselect eventos.

```
extension ProductsVC: UICollectionViewDelegate, UICollectionViewDataSource{  
  
    // MARK: - UICollectionViewDataSource
```

```

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection
section: Int) -> Int {
        guard collectionView == collectionCategory else {
            return arrOfProducts.count
        }
        return arrOfCategory.count
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath:
IndexPath) -> UICollectionViewCell {

        guard collectionView == collectionProduct else {
            let cell = collectionView.dequeueReusableCell(withReuseIdentifier:
"ProductCategoryCell", for: indexPath) as! ProductCategoryCell
            cell.viewBackground.layer.borderWidth = 0.5
            //Do some thing as per use
            return cell
        }

        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: cellIdentifier,
for: indexPath) as! ProductCell
        cell.contentView.layer.borderWidth = 0.5
        cell.contentView.layer.borderColor = UIColor.black.cgColor
        let json = arrOfProducts[indexPath.row]
        //Do something as per use

        return cell
    }

    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath:
IndexPath) {
        guard collectionView == collectionCategory else {
            let json = arrOfProducts[indexPath.row]
            // Do something for collectionProduct here
            return
        }
        let json = arrOfCategory[indexPath.row] as [String: AnyObject]
        let id = json["cId"] as? String ?? ""
        // Do something
    }
}

extension ProductsVC: UICollectionViewDelegateFlowLayout{

    // MARK: - UICollectionViewDelegateFlowLayout
    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {

        let collectionWidth = collectionView.bounds.width
        guard collectionView == collectionProduct else {
            var itemWidth = collectionWidth / 4 - 1;

            if(UI_USER_INTERFACE_IDIOM() == .pad) {
                itemWidth = collectionWidth / 4 - 1;
            }
            return CGSize(width: itemWidth, height: 50)
        }

        var itemWidth = collectionWidth / 2 - 1;
        if(UI_USER_INTERFACE_IDIOM() == .pad) {
            itemWidth = collectionWidth / 4 - 1;
        }
    }
}

```

```

    }
    return CGSize(width: itemWidth, height: 250);
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumInteritemSpacingForSectionAt section: Int) -> CGFloat {
    return 1
}

func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout:
UICollectionViewLayout, minimumLineSpacingForSectionAt section: Int) -> CGFloat {
    return 1
}
}

```

23-01-2017

24-01-2017

25-01-2017

11:00	11:15	11:30	11:45
12:00	12:15	12:30	12:45
13:00	13:15	13:30	13:45
14:00	14:15	14:30	14:45
15:00	15:15	15:30	15:45
16:00	16:15	16:30	16:45

Lea UICollectionView en línea: <https://riptutorial.com/es/ios/topic/2399/UICollectionView>

Capítulo 163: UIColor

Examples

Creando un UIColor

Hay muchas maneras de crear un `UIColor` :

Rápido

- Usando uno de los colores predefinidos:

```
let redColor = UIColor.redColor()
let blueColor: UIColor = .blueColor()

// In Swift 3, the "Color()" suffix is removed:
let redColor = UIColor.red
let blueColor: UIColor = .blue
```

Si el compilador ya sabe que la variable es una instancia de `UIColor` , puede omitir todo el tipo:

```
let view = UIView()
view.backgroundColor = .yellowColor()
```

- Usando el valor de escala de grises y el alfa:

```
let grayscaleColor = UIColor(white: 0.5, alpha: 1.0)
```

- Usando matiz, saturación, brillo y alfa:

```
let hsbColor = UIColor(
    hue: 0.4,
    saturation: 0.3,
    brightness: 0.7,
    alpha: 1.0
)
```

- Usando los valores RGBA:

```
let rgbColor = UIColor(
    red: 30.0 / 255,
    green: 70.0 / 255,
    blue: 200.0 / 255,
    alpha: 1.0
)
```

- Usando una imagen de patrón:

```
let patternColor = UIColor(patternImage: UIImage(named: "myImage")!)
```

C objetivo

- Usando uno de los colores predefinidos:

```
UIColor *redColor = [UIColor redColor];
```

- Usando el valor de escala de grises y el alfa:

```
UIColor *grayscaleColor = [UIColor colorWithWhite: 0.5 alpha: 1.0];
```

- Usando matiz, saturación, brillo y alfa:

```
UIColor *hsbColor = [UIColor  
    colorWithHue: 0.4  
    saturation: 0.3  
    brightness: 0.7  
    alpha: 1.0  
];
```

- Usando los valores RGBA:

```
UIColor *rgbColor = [UIColor  
    colorWithRed: 30.0 / 255.0  
    green: 70.0 / 255.0  
    blue: 200.0 / 255.0  
    alpha: 1.0  
];
```

- Usando una imagen de patrón:

```
UIColor *pattenColor = [UIColor colorWithPatternImage:[UIImage  
    imageNamed:@"myImage.png"]];
```

Métodos no documentados

Hay una variedad de métodos no documentados en `UIColor` que exponen colores o funciones alternativos. Estos se pueden encontrar en el [archivo de encabezado privado UIColor](#) .

Documentaré el uso de dos métodos privados, `styleString()` y `_systemDestructiveTintColor()` .

`styleString`

Desde iOS 2.0, existe un método de instancia privada en `UIColor` llamado `styleString` que devuelve una representación de cadena RGB o RGBA del color, incluso para colores como `whiteColor` fuera del espacio RGB.

C objetivo:


```

@interface UIColor (Private)

- (NSString *)styleString;

@end

// ...

[[UIColor whiteColor] styleString]; // rgb(255,255,255)
[[UIColor redColor] styleString]; // rgb(255,0,0)
[[UIColor lightTextColor] styleString]; // rgba(255,255,255,0.600000)

```

En Swift puedes usar un encabezado de puente para exponer la interfaz. Con Swift puro, deberá crear un protocolo `@objc` con el método privado, y `unsafeBitCast UIColor` con el protocolo:

```

@objc protocol UIColorPrivate {
    func styleString() -> String
}

let white = UIColor.whiteColor()
let red = UIColor.redColor()
let lightTextColor = UIColor.lightTextColor()

let whitePrivate = unsafeBitCast(white, UIColorPrivate.self)
let redPrivate = unsafeBitCast(red, UIColorPrivate.self)
let lightTextColorPrivate = unsafeBitCast(lightTextColor, UIColorPrivate.self)

whitePrivate.styleString() // rgb(255,255,255)
redPrivate.styleString() // rgb(255,0,0)
lightTextColorPrivate.styleString() // rgba(255,255,255,0.600000)

```

`_systemDestructiveTintColor()`

Hay un método de clase no documentado en `UIColor` llamado `_systemDestructiveTintColor` que devolverá el color rojo utilizado por los botones del sistema destructivo:

```

let red = UIColor.performSelector("_systemDestructiveTintColor").takeUnretainedValue()

```

Devuelve un objeto no administrado, al que debe llamar `.takeUnretainedValue()`, ya que la propiedad del color no se ha transferido a nuestro propio objeto.

Al igual que con cualquier API no documentada, debe tener cuidado al intentar usar este método:

```

if UIColor.respondsToSelector("_systemDestructiveTintColor") {
    if let red = UIColor.performSelector("_systemDestructiveTintColor").takeUnretainedValue()
as? UIColor {
        // use the color
    }
}

```

o utilizando un protocolo:

```

@objc protocol UIColorPrivateStatic {
    func _systemDestructiveTintColor() -> UIColor
}

```

```
}  
  
let privateClass = UIColor.self as! UIColorPrivateStatic  
privateClass._systemDestructiveTintColor() // UIDeviceRGBColorSpace 1 0.231373 0.188235 1
```

Color con componente alfa

Puede configurar la opacidad a un determinado `UIColor` sin crear uno nuevo utilizando el `init(red:_, green:_, blue:_, alpha:_)` .

Rápido

```
let colorWithAlpha = UIColor.redColor().colorWithAlphaComponent(0.1)
```

Swift 3

```
//In Swift Latest Version  
_ colorWithAlpha = UIColor.red.withAlphaComponent(0.1)
```

C objetivo

```
UIColor * colorWithAlpha = [[UIColor redColor] colorWithAlphaComponent:0.1];
```


Hacer que los atributos definidos por el usuario apliquen el tipo de datos `CGColor`.

De forma predeterminada, Interface Builder no acepta el tipo de datos `CGColor` , por lo que permite agregar un `CGColor` usando atributos definidos por el usuario en el constructor de interfaces; uno puede querer usar una extensión como esta:

Extensión rápida:

```
extension CALayer {  
    func borderUIColor() -> UIColor? {  
        return borderColor != nil ? UIColor(CGColor: borderColor!) : nil  
    }  
  
    func setBorderUIColor(color: UIColor) {  
        borderColor = color.CGColor  
    }  
}
```

El nuevo atributo definido por el usuario (`borderUIColor`) será reconocido y aplicado sin problemas.

User Defined Runtime Attributes		
Key Path	Type	Value
layer.cornerRadius	Number	6.5
layer.borderWidth	Number	1
layer.clipsToBounds	Boolean	<input checked="" type="checkbox"/>
layer.borderColor	Color	

Creando un UIColor a partir de un número hexadecimal o cadena

Puede crear un `UIColor` partir de un número o cadena hexadecimal, por ejemplo, `0xff00cc`, `"#FFFFFF"`

Rápido

Valor int

```
extension UIColor {
  convenience init(hex: Int, alpha: CGFloat = 1.0) {
    let r = CGFloat((hex >> 16) & 0xff) / 255
    let g = CGFloat((hex >> 08) & 0xff) / 255
    let b = CGFloat((hex >> 00) & 0xff) / 255
    self.init(red: r, green: g, blue: b, alpha: alpha)
  }
}
```

Ejemplo:

```
let color = UIColor(hex: 0xff00cc, alpha: 1.0)
```

Tenga en cuenta que para `alpha` el valor predeterminado de `1.0`, por lo que se puede utilizar de la siguiente manera:

```
let color = UIColor(hex: 0xff00cc)
```

Valor de cadena

```
extension UIColor {
  convenience init(hexCode: String) {
    let hex =
hexCode.stringByTrimmingCharactersInSet(NSCharacterSet.alphanumericCharacterSet().invertedSet)
    var int = UInt32()
    NSScanner(string: hex).scanHexInt(&int)
    let a, r, g, b: UInt32

    switch hex.characters.count {
    case 3:
      (a, r, g, b) = (255, (int >> 8) * 17, (int >> 4 & 0xF) * 17, (int & 0xF) * 17)
    case 6:
      (a, r, g, b) = (255, int >> 16, int >> 8 & 0xFF, int & 0xFF)
    case 8:
      (a, r, g, b) = (int >> 24, int >> 16 & 0xFF, int >> 8 & 0xFF, int & 0xFF)
    default:

```

```

        (a, r, g, b) = (1, 1, 1, 0)
    }

    self.init(red: CGFloat(r) / 255, green: CGFloat(g) / 255, blue: CGFloat(b) / 255,
alpha: CGFloat(a) / 255)
    }
}

```

Ejemplo de uso:

Hex con alfa

```
let color = UIColor("#80FFFFFF")
```

Hex. Sin alfa (el color alfa será igual a 1.0)

```
let color = UIColor("FFFFFF")
let color = UIColor("FFF")
```

C objetivo

Valor int

```

@interface UIColor (Hex)
+ (UIColor *)colorWithHex:(NSUInteger)hex alpha:(CGFloat)alpha;
@end

@implementation UIColor (Hex)
+ (UIColor *)colorWithHex:(NSUInteger)hex alpha:(CGFloat)alpha {
    return [UIColor colorWithRed:((CGFloat)((hex & 0xFF0000) >> 16))/255.0
                green:((CGFloat)((hex & 0xFF00) >> 8))/255.0
                blue:((CGFloat)(hex & 0xFF))/255.0
                alpha:alpha];
}
@end

```

Ejemplo:

```
UIColor *color = [UIColor colorWithHex:0xff00cc alpha:1.0];
```

Valor de cadena

```

- (UIColor*) hex:(NSString*)hexCode {

    NSString *noHashString = [hexCode stringByReplacingOccurrencesOfString:@"#"
withString:@""];
    NSScanner *scanner = [NSScanner scannerWithString:noHashString];
    [scanner setCharactersToBeSkipped:[NSCharacterSet symbolCharacterSet]];

    unsigned hex;
    if (![scanner scanHexInt:&hex]) return nil;
    int a;
    int r;

```

```

int g;
int b;

switch (noHashString.length) {
    case 3:
        a = 255;
        r = (hex >> 8) * 17;
        g = ((hex >> 4) & 0xF) * 17;
        b = ((hex >> 0) & 0xF) * 17;
        break;
    case 6:
        a = 255;
        r = (hex >> 16);
        g = (hex >> 8) & 0xFF;
        b = (hex) & 0xFF;
        break;
    case 8:
        a = (hex >> 24);
        r = (hex >> 16) & 0xFF;
        g = (hex >> 8) & 0xFF;
        b = (hex) & 0xFF;
        break;

    default:
        a = 255.0;
        r = 255.0;
        b = 255.0;
        g = 255.0;
        break;
}

return [UIColor colorWithRed:r / 255.0f green:g / 255.0f blue:b / 255.0f alpha:a / 255];
}

```

Ejemplo de uso:

Hex con alfa

```
UIColor* color = [self hex:@"#80FFFFFF"];
```

Hex. Sin alfa (el color alfa será igual a 1)

```
UIColor* color = [self hex:@"#FFFFFF"];
UIColor* color = [self hex:@"#FFF"];
```

Brillo de color ajustado de UIColor

El ejemplo de código a continuación le dará una versión ajustada de ese color donde un porcentaje más alto será más brillante y un porcentaje más bajo será más oscuro.

C objetivo

```

+ (UIColor *)adjustedColorForColor:(UIColor *)c : (double)percent
{
    if (percent < 0) percent = 0;

```

```

CGFloat r, g, b, a;
if ([c getRed:&r green:&g blue:&b alpha:&a])
    return [UIColor colorWithRed:MAX(r * percent, 0.0)
                green:MAX(g * percent, 0.0)
                blue:MAX(b * percent, 0.0)
                alpha:a];

return nil;
}

```

Rápido

```

func adjustedColorForColor( c: UIColor, var percent: CGFloat) -> UIColor {
    if percent < 0 {
        percent = 0
    }

    var r,g,b,a: CGFloat
    r = 0.0
    g = 0.0
    b = 0.0
    a = 0.0

    if c.getRed(&r, green: &g, blue: &b, alpha: &a) {
        return UIColor(red: max(r * percent, 0.0), green: max(g * percent, 0.0), blue: max(b *
percent, 0.0), alpha: a)
    }

    return UIColor()
}

```

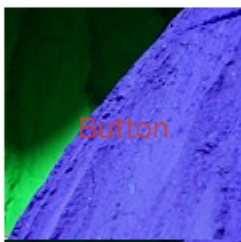
UIColor de un patrón de imagen

Puede crear un objeto `UIColor` usando un patrón de imagen usando el `UIColor(patternImage:_)` .

```

btn.backgroundColor = UIColor(patternImage: UIImage(named: "image")!)

```



Tono más claro y oscuro de un UIColor dado

El siguiente ejemplo de código muestra cómo puede obtener un tono más claro y oscuro de un color determinado, útil en aplicaciones que tienen temas dinámicos.

Para un **color más oscuro**

```
+ (UIColor *)darkerColorForColor:(UIColor *)c
{
    CGFloat r, g, b, a;
    if ([c getRed:&r green:&g blue:&b alpha:&a])
        return [UIColor colorWithRed:MAX(r - 0.2, 0.0)
                                green:MAX(g - 0.2, 0.0)
                                blue:MAX(b - 0.2, 0.0)
                                alpha:a];
    return nil;
}
```

Para un **color más claro**

```
+ (UIColor *)lighterColorForColor:(UIColor *)c
```

```
{
  CGFloat r, g, b, a;
  if ([c getRed:&r green:&g blue:&b alpha:&a])
    return [UIColor colorWithRed:MIN(r + 0.2, 1.0)
                        green:MIN(g + 0.2, 1.0)
                        blue:MIN(b + 0.2, 1.0)
                        alpha:a];
  return nil;
}
```

Vea las diferencias visuales a continuación, considerando que el color dado es `[UIColor orangeColor]`



Lea UIColor en línea: <https://riptutorial.com/es/ios/topic/956/UIColor>

Capítulo 164: UIControl - Manejo de eventos con bloques

Examples

Introducción

Normalmente, cuando se usa `UIControl` o `UIButton`, agregamos un `selector` como una acción de devolución de llamada para cuando ocurre un evento en un botón o control, como cuando el usuario presiona el botón o toca el control.

Por ejemplo, haríamos lo siguiente:

```
import UIKit

class ViewController: UIViewController {
    @IBOutlet weak var button: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()

        let button = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 44))
        button.addTarget(self, action: #selector(self.onButtonPress(_:)), for: .touchUpInside)
        self.view.addSubview(button)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    func onButtonPress(_ button: UIButton!) {
        print("PRESSED")
    }
}
```

Cuando se trata de `selector`, el compilador solo necesita saber que existe. Esto puede hacerse a través de un `protocol` y no implementarse.

Por ejemplo, lo siguiente podría bloquear su aplicación:

```
import UIKit

@objc
protocol ButtonEvent {
    @objc optional func onButtonPress(_ button: UIButton)
}

class ViewController: UIViewController, ButtonEvent {
    @IBOutlet weak var button: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()
    }
}
```

```

        let button = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 44))
        button.addTarget(self, action: #selector(ButtonEvent.onButtonPress(_:)), for:
        .touchUpInside)
        self.view.addSubview(button)
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }
}

```

Esto se debe a que su aplicación NO implementa la función `onButtonPress` .

Ahora, ¿qué pasaría si pudieras hacer todo esto junto con la inicialización del botón? ¿Qué pasaría si no tuviera que especificar devoluciones de llamada y en su lugar podría especificar bloques que se pueden agregar y eliminar en cualquier momento? ¿Por qué preocuparse por implementar selectores?

Solución

```

import Foundation
import UIKit

protocol RemovableTarget {
    func enable();
    func disable();
}

extension UIControl {
    func addEventHandler(event: UIControlEvents, runnable: (control: UIControl) -> Void) -> RemovableTarget {

        class Target : RemovableTarget {
            private var event: UIControlEvents
            private weak var control: UIControl?
            private var runnable: (control: UIControl) -> Void

            private init(event: UIControlEvents, control: UIControl, runnable: (control: UIControl) -> Void) {
                self.event = event
                self.control = control
                self.runnable = runnable
            }

            @objc
            private func run(_ control: UIControl) {
                runnable(control: control)
            }

            private func enable() {
                control?.addTarget(self, action: #selector(Target.run(_:)), for: event)
                objc_setAssociatedObject(self, unsafeAddress(of: self), self,
                .OBJC_ASSOCIATION_RETAIN)
            }

            private func disable() {

```

```

        control?.removeTarget(self, action: #selector(Target.run(_:)), for:
self.event)
        objc_setAssociatedObject(self, unsafeAddress(of: self), nil,
.OBJC_ASSOCIATION_ASSIGN)
    }
}

let target = Target(event: event, control: self, runnable: runnable)
target.enable()
return target
}
}

```

Lo anterior es una extensión simple en `UIControl` . Agrega una clase privada interna que tiene una `func run(_ control: UIControl)` devolución de llamada `func run(_ control: UIControl)` que se utiliza como acción de los eventos.

A continuación, utilizamos la `object association` para agregar y eliminar el destino, ya que no será retenido por `UIControl` .

La función del controlador de eventos devuelve un `Protocol` para ocultar el funcionamiento interno de la clase de `Target` , pero también para permitirle `enable` y `disable` el objetivo en cualquier momento.

Ejemplo de uso:

```

import Foundation
import UIKit

class ViewController: UIViewController {

    override func viewDidLoad() {
        super.viewDidLoad()

        //Create a button.
        let button = UIButton(frame: CGRect(x: 0, y: 0, width: 100, height: 44))

        //Add an event action block/listener -- Handles Button Press.
        let target = button.addHandler(event: .touchUpInside) { (control) in
            print("Pressed")
        }

        self.view.addSubview(button)

        //Example of enabling/disabling the listener/event-action-block.
        DispatchQueue.main.after(when: DispatchTime.now() + 5) {
            target.disable() //Disable the listener.

            DispatchQueue.main.after(when: DispatchTime.now() + 5) {
                target.enable() //Enable the listener.
            }
        }
    }
}

```

```
override func didReceiveMemoryWarning() {  
    super.didReceiveMemoryWarning()  
}  
}
```

Lea UIControl - Manejo de eventos con bloques en línea:

<https://riptutorial.com/es/ios/topic/3180/uicontrol---manejo-de-eventos-con-bloques>

Capítulo 165: UIDatePicker

Observaciones

`UIDatePicker` no hereda de `UIPickerView`, pero administra un objeto de vista de selector personalizado como una subvista.

Examples

Crear un selector de fecha

Rápido

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200))
```

C objetivo

```
UIDatePicker *datePicker = [[UIDatePicker alloc] initWithFrame:CGRectMake(x: 0, y: 0, width: 320, height: 200)];
```

Configuración de fecha mínima-máxima

Puede establecer la fecha mínima y máxima que puede mostrar `UIDatePicker`.

Fecha minima

```
[datePicker setMinimumDate:[NSDate date]];
```

Fecha maxima

```
[datePicker setMaximumDate:[NSDate date]];
```

Modos

`UIDatePicker` tiene varios modos de selección.

```
enum UIDatePickerMode : Int {
    case Time
    case Date
    case DateAndTime
    case CountdownTimer
}
```

- `Time` : el selector de fecha muestra las horas, los minutos y (opcionalmente) una designación de AM / PM.
- `Date` : el selector de fecha muestra los meses, los días del mes y los años.
- `DateAndTime` y `DateAndTime` : el selector de fecha muestra las fechas (como valores de día de la semana, mes y día del mes unificados) más horas, minutos y (opcionalmente) una designación de AM / PM.
- `CountDownTimer` : el selector de fecha muestra los valores de hora y minuto, por ejemplo [1 | 53]. La aplicación debe configurar un temporizador para que se active en el intervalo adecuado y establecer el selector de fecha a medida que los segundos marcan.

Estableciendo la propiedad `datePickerMode`

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.datePickerMode = .Date
```

Ajuste de intervalo de minutos

Puede cambiar la propiedad `minuteInterval` para establecer el intervalo mostrado por la rueda de minutos. El valor predeterminado es 1, el valor máximo es 30.

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.minuteInterval = 15
```

Duración de la cuenta regresiva

El valor `NSTimeInterval` de esta propiedad indica los segundos desde los cuales el selector de fecha en el modo de temporizador de cuenta regresiva cuenta hacia abajo. Si el modo del selector de fecha no es `CountDownTimer`, este valor se ignora. El valor máximo es 86,399 segundos (23:59)

```
let datePicker = UIDatePicker(frame: CGRect(x: 0, y: 0, width: 320, height: 200)
datePicker.countDownDuration = 60 * 60
```

Lea `UIDatePicker` en línea: <https://riptutorial.com/es/ios/topic/5643/uidatepicker>

Capítulo 166: UIDevice

Parámetros

Propiedad	Descripción
nombre	El nombre que identifica al dispositivo.
nombre_sistema: cadena	El nombre del sistema operativo que se ejecuta en el dispositivo representado por el receptor.
modelo: cuerda	El modelo del dispositivo.
systemVersion: String	La versión actual del sistema operativo.

Observaciones

La clase `UIDevice` proporciona una instancia de Singleton que representa el dispositivo actual. Desde esta instancia, puede obtener información sobre el dispositivo, como el nombre asignado, el modelo del dispositivo y el nombre y la versión del sistema operativo.

Examples

Obtener el nombre del modelo del dispositivo iOS

Swift 2

```
import UIKit

extension UIDevice {

    var modelName: String {
        var systemInfo = utsname()
        uname(&systemInfo)
        let machineMirror = Mirror(reflecting: systemInfo.machine)
        let identifier = machineMirror.children.reduce("") { identifier, element in
            guard let value = element.value as? Int8 where value != 0 else { return identifier }
        }

        return identifier + String(UnicodeScalar(UInt8(value)))
    }

    switch identifier {
    case "iPod5,1": return "iPod Touch 5"
    case "iPod7,1": return "iPod Touch 6"
    case "iPhone3,1", "iPhone3,2", "iPhone3,3": return "iPhone 4"
    case "iPhone4,1": return "iPhone 4s"
    case "iPhone5,1", "iPhone5,2": return "iPhone 5"
    case "iPhone5,3", "iPhone5,4": return "iPhone 5c"
    }
}
```

```

    case "iPhone6,1", "iPhone6,2":           return "iPhone 5s"
    case "iPhone7,2":                       return "iPhone 6"
    case "iPhone7,1":                       return "iPhone 6 Plus"
    case "iPhone8,1":                       return "iPhone 6s"
    case "iPhone8,2":                       return "iPhone 6s Plus"
    case "iPhone9,1", "iPhone9,3":         return "iPhone 7"
    case "iPhone9,2", "iPhone9,4":         return "iPhone 7 Plus"
    case "iPhone8,4":                       return "iPhone SE"
    case "iPad2,1", "iPad2,2", "iPad2,3", "iPad2,4":return "iPad 2"
    case "iPad3,1", "iPad3,2", "iPad3,3":   return "iPad 3"
    case "iPad3,4", "iPad3,5", "iPad3,6":   return "iPad 4"
    case "iPad4,1", "iPad4,2", "iPad4,3":   return "iPad Air"
    case "iPad5,3", "iPad5,4":             return "iPad Air 2"
    case "iPad2,5", "iPad2,6", "iPad2,7":   return "iPad Mini"
    case "iPad4,4", "iPad4,5", "iPad4,6":   return "iPad Mini 2"
    case "iPad4,7", "iPad4,8", "iPad4,9":   return "iPad Mini 3"
    case "iPad5,1", "iPad5,2":             return "iPad Mini 4"
    case "iPad6,3", "iPad6,4", "iPad6,7", "iPad6,8":return "iPad Pro"
    case "AppleTV5,3":                      return "Apple TV"
    case "i386", "x86_64":                  return "Simulator"
    default:                                return identifier
  }
}

if UIDevice.currentDevice().modelName == "iPhone 6 Plus" {
  // is an iPhone 6 Plus
}

```

Swift 3

```

import UIKit

public extension UIDevice {

  var modelName: String {
    var systemInfo = utsname()
    uname(&systemInfo)
    let machineMirror = Mirror(reflecting: systemInfo.machine)
    let identifier = machineMirror.children.reduce("") { identifier, element in
      guard let value = element.value as? Int8 , value != 0 else { return identifier
    }

    return identifier + String(UnicodeScalar(UInt8(value)))
  }

  switch identifier {
    case "iPod5,1":           return "iPod Touch 5"
    case "iPod7,1":           return "iPod Touch 6"
    case "iPhone3,1", "iPhone3,2", "iPhone3,3": return "iPhone 4"
    case "iPhone4,1":         return "iPhone 4s"
    case "iPhone5,1", "iPhone5,2": return "iPhone 5"
    case "iPhone5,3", "iPhone5,4": return "iPhone 5c"
    case "iPhone6,1", "iPhone6,2": return "iPhone 5s"
    case "iPhone7,2":         return "iPhone 6"
    case "iPhone7,1":         return "iPhone 6 Plus"
    case "iPhone8,1":         return "iPhone 6s"
    case "iPhone8,2":         return "iPhone 6s Plus"
    case "iPhone9,1", "iPhone9,3": return "iPhone 7"
    case "iPhone9,2", "iPhone9,4": return "iPhone 7 Plus"
    case "iPhone8,4":         return "iPhone SE"
  }
}

```



```

    case "iPad2,1", "iPad2,2", "iPad2,3", "iPad2,4":return "iPad 2"
    case "iPad3,1", "iPad3,2", "iPad3,3":          return "iPad 3"
    case "iPad3,4", "iPad3,5", "iPad3,6":          return "iPad 4"
    case "iPad4,1", "iPad4,2", "iPad4,3":          return "iPad Air"
    case "iPad5,3", "iPad5,4":                      return "iPad Air 2"
    case "iPad2,5", "iPad2,6", "iPad2,7":          return "iPad Mini"
    case "iPad4,4", "iPad4,5", "iPad4,6":          return "iPad Mini 2"
    case "iPad4,7", "iPad4,8", "iPad4,9":          return "iPad Mini 3"
    case "iPad5,1", "iPad5,2":                      return "iPad Mini 4"
    case "iPad6,3", "iPad6,4", "iPad6,7", "iPad6,8":return "iPad Pro"
    case "AppleTV5,3":                              return "Apple TV"
    case "i386", "x86_64":                          return "Simulator"
    default:                                         return identifier
  }
}

if UIDevice.current.modelName == "iPhone 7" {
  // is an iPhone 7
}

```

Obtención del estado de la batería y del nivel de la batería

```

override func viewDidLoad() {
    super.viewDidLoad()
    NotificationCenter.default.addObserver(self, selector:
Selector(("batteryStateDidChange:")), name: NSNotification.Name.UIDeviceBatteryStateDidChange,
object: nil)
    NotificationCenter.default.addObserver(self, selector:
Selector(("batteryLevelDidChange:")), name: NSNotification.Name.UIDeviceBatteryLevelDidChange,
object: nil)

    // Stuff...
}

func batteryStateDidChange(notification: NSNotification){
    // The stage did change: plugged, unplugged, full charge...
}

func batteryLevelDidChange(notification: NSNotification){

    let batteryLevel = UIDevice.current.batteryLevel
    if batteryLevel < 0.0 {
        print(" -1.0 means battery state is UIDeviceBatteryStateUnknown")
        return
    }

    print("Battery Level : \(batteryLevel * 100)%")
    // The battery's level did change (98%, 99%, ...)
}

```

Identificando el dispositivo y operando

```

UIDevice *deviceInfo = [UIDevice currentDevice];
NSLog(@"Device Name %@", deviceInfo.name);
//Ex: myIphone6s
NSLog(@"System Name %@", deviceInfo.systemName);

```

```

//Device Name iPhone OS
NSLog(@"System Version %@", deviceInfo.systemVersion);
//System Version 9.3
NSLog(@"Model %@", deviceInfo.model);
//Model iPhone
NSLog(@"Localized Model %@", deviceInfo.localizedModel);
//Localized Model iPhone
int device=deviceInfo.userInterfaceIdiom;
//UIUserInterfaceIdiomPhone=0
//UIUserInterfaceIdiomPad=1
//UIUserInterfaceIdiomTV=2
//UIUserInterfaceIdiomCarPlay=3
//UIUserInterfaceIdiomUnspecified=-1
NSLog(@"identifierForVendor %@", deviceInfo.identifierForVendor);
//identifierForVendor <__NSConcreteUUID 0x7a10ae20> 556395DC-0EB4-4FD5-BC7E-B16F612ECC6D

```

Obtención de la orientación del dispositivo

```

UIDevice *deviceInfo = [UIDevice currentDevice];
int d = deviceInfo.orientation;

```

`deviceInfo.orientation` devuelve un valor `UIDeviceOrientation` que se muestra a continuación:

```

UIDeviceOrientationUnknown 0
UIDeviceOrientationPortrait 1
UIDeviceOrientationPortraitUpsideDown 2
UIDeviceOrientationLandscapeLeft 3
UIDeviceOrientationLandscapeRight 4
UIDeviceOrientationFaceUp 5
UIDeviceOrientationFaceDown 6

```

Escuchando los cambios de orientación del dispositivo en un controlador de vista:

```

- (void) viewWillAppear:(BOOL) animated
{
    [super viewWillAppear:animated];
    [[UIDevice currentDevice] beginGeneratingDeviceOrientationNotifications];
    [[NSNotificationCenter defaultCenter] addObserver:self
                                             selector:@selector(deviceOrientationDidChange)
                                             name:UIDeviceOrientationDidChangeNotification
                                             object:nil];
}

-(void) deviceOrientationDidChange
{
    UIDeviceOrientation orientation = [[UIDevice currentDevice] orientation];
    if (orientation == UIDeviceOrientationPortrait || orientation ==
UIDeviceOrientationPortraitUpsideDown) {
        [self changedToPortrait];
    } else if (orientation == UIDeviceOrientationLandscapeLeft || orientation ==
UIDeviceOrientationLandscapeRight) {
        [self changedToLandscape];
    }
}

-(void) changedToPortrait
{
    // Function Body
}

```

```

}

-(void) changedToLandscape
{
    // Function Body
}

```

Para desactivar la verificación de cualquier cambio de orientación:

```

- (void) viewWillDisappear:(BOOL) animated {
    [super viewWillDisappear:animated];
    [[UIDevice currentDevice] endGeneratingDeviceOrientationNotifications];
}

```

Obtención del estado de la batería del dispositivo

```

//Get permission for Battery Monitoring
[[UIDevice currentDevice] setBatteryMonitoringEnabled:YES];
UIDevice *myDevice = [UIDevice currentDevice];

[myDevice setBatteryMonitoringEnabled:YES];
double batLeft = (float)[myDevice batteryLevel] * 100;
NSLog(@"%.f", batLeft);

int d = myDevice.batteryState;
//Returns an Integer Value
//UIDeviceBatteryStateUnknown 0
//UIDeviceBatteryStateUnplugged 1
//UIDeviceBatteryStateCharging 2
//UIDeviceBatteryStateFull 3

//Using notifications for Battery Monitoring
-(void) startMonitoringForBatteryChanges
{
    // Enable monitoring of battery status
    [[UIDevice currentDevice] setBatteryMonitoringEnabled:YES];
    // Request to be notified when battery charge or state changes
    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(checkBatteryStatus)
    name:UIDeviceBatteryLevelDidChangeNotification object:nil];
    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(checkBatteryStatus)
    name:UIDeviceBatteryStateDidChangeNotification object:nil];
}

-(void) checkBatteryStatus
{
    NSLog(@"Battery Level is %.f", [[UIDevice currentDevice] batteryLevel]*100);
    int d=[[UIDevice currentDevice] batteryState];
    if (d==0)
    {
        NSLog(@"Unknown");
    }
    else if (d==1)
    {
        NSLog(@"Unplugged");
    }
    else if (d==2)
    {
        NSLog(@"Charging");
    }
}

```

```
else if (d==3)
{
    NSLog(@"Battery Full");
}
}
```

Usando el sensor de proximidad

```
//Enabling the proximity Sensor
- (void)viewWillAppear:(BOOL)animated {

    [super viewWillAppear:animated];
    [[UIDevice currentDevice] setProximityMonitoringEnabled:YES];
    [[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(sensorStateMonitor:) name:@"UIDeviceProximityStateDidChangeNotification"
object:nil];
}

- (void)sensorStateMonitor:(NSNotificationCenter *)notification
{
    if ([[UIDevice currentDevice] proximityState] == YES)
    {
        NSLog(@"Device is close to user.");
    }

    else
    {
        NSLog(@"Device is not closer to user.");
    }
}
```

Lea UIDevice en línea: <https://riptutorial.com/es/ios/topic/4878/uidevice>

Capítulo 167: UIFeedbackGenerator

Introducción

`UIFeedbackGenerator` y sus subclases ofrecen una interfaz pública para el Taptic Engine® que se encuentra en los dispositivos iOS que comienzan con el iPhone 7. Los Haptics, marca Taptics, proporcionan comentarios táctiles para eventos en pantalla. Si bien muchos controles del sistema proporcionan hápticas fuera de la caja, los desarrolladores pueden usar subclases

`UIFeedbackGenerator` para agregar hápticas a controles personalizados y otros eventos.

`UIFeedbackGenerator` es una clase abstracta que no debe usarse directamente, sino que los desarrolladores usan una de sus subclases.

Examples

Impacto de gatillo háptico

El ejemplo muestra cómo activar un háptico de impacto usando `UIImpactFeedbackGenerator` después de presionar un botón.

Rápido

```
class ViewController: UIViewController
{
    lazy var button: UIButton =
    {
        let button = UIButton()
        button.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(button)
        button.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
        button.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        button.setTitle("Impact", for: .normal)
        button.setTitleColor(UIColor.gray, for: .normal)
        return button
    }()

    // Choose between heavy, medium, and light for style
    let impactFeedbackGenerator = UIImpactFeedbackGenerator(style: .heavy)

    override func viewDidLoad()
    {
        super.viewDidLoad()
        button.addTarget(self, action: #selector(self.didPressButton(sender:)), for:
        .touchUpInside)

        // Primes feedback generator for upcoming events and reduces latency
        impactFeedbackGenerator.prepare()
    }

    func didPressButton(sender: UIButton)
    {
        // Triggers haptic
    }
}
```

```

        impactFeedbackGenerator.impactOccurred()
    }
}

```

C objetivo

```

@interface ViewController ()
@property (nonatomic, strong) UIImpactFeedbackGenerator *impactFeedbackGenerator;
@property (nonatomic, strong) UIButton *button;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.button addTarget:self action:@selector(didPressButton:)
    forControlEvents:UIControlEventTouchUpInside];

    // Choose between heavy, medium, and light for style
    self.impactFeedbackGenerator = [[UIImpactFeedbackGenerator alloc]
    initWithStyle:UIImpactFeedbackStyleHeavy];

    // Primes feedback generator for upcoming events and reduces latency
    [self.impactFeedbackGenerator prepare];
}

- (void)didPressButton:(UIButton *)sender
{
    // Triggers haptic
    [self.impactFeedbackGenerator impactOccurred];
}

#pragma mark - Lazy Init
- (UIButton *)button
{
    if (!_button)
    {
        _button = [[UIButton alloc]init];
        _button.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_button];
        [_button.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_button.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
        [_button setTitle:@"Impact" forState:UIControlStateNormal];
        [_button setTitleColor:[UIColor grayColor] forState:UIControlStateNormal];
    }
    return _button;
}

@end

```

Lea `UIFeedbackGenerator` en línea: <https://riptutorial.com/es/ios/topic/10048/uifeedbackgenerator>

Capítulo 168: UIFont

Introducción

UIFont es una clase que se utiliza para obtener y configurar información relacionada con la fuente. Hereda de `NSObject` y se ajusta a `Hashable`, `Equatable`, `CVarArg` y `NSCopying`.

Examples

Declarar e inicializar UIFont

Puede declarar un `UIFont` siguiente manera:

```
var font: UIFont!
```

`UIFont` tiene más métodos `init()`:

- `UIFont.init(descriptor: UIFontDescriptor, size: CGFloat)`
- `UIFont.init(name: String, size: CGFloat)`

Por lo tanto, puedes inicializar un `UIFont` como este:

```
let font = UIFont(name: "Helvetica Neue", size: 15)
```

La fuente predeterminada es `System`, tamaño 17.

Cambiando la fuente de una etiqueta

Para cambiar la fuente de texto de una etiqueta, debe acceder a su propiedad de `font`:

```
label.font = UIFont(name:"Helvetica Neue", size: 15)
```

El código anterior cambiará la fuente de la etiqueta a `Helvetica Neue`, tamaño 15. Tenga en cuenta que debe escribir correctamente el nombre de la fuente, de lo contrario lanzará este error, ya que el valor inicializado arriba es un Opcional, y por lo tanto puede ser nulo:

Se encontró inesperadamente nil mientras se desarrollaba un valor opcional

Lea `UIFont` en línea: <https://riptutorial.com/es/ios/topic/9792/UIFont>

Capítulo 169: UITapGestureRecognizer

Examples

UITapGestureRecognizer

Inicialice el `UITapGestureRecognizer` con un objetivo, `self` en este caso, y una `action` que es un método que tiene un solo parámetro: un `UITapGestureRecognizer`.

Después de la inicialización, agréguelo a la vista donde debería reconocer los toques.

Rápido

```
override func viewDidLoad() {
    super.viewDidLoad()
    let recognizer = UITapGestureRecognizer(target: self,
                                          action: #selector(handleTap(_:)))
    view.addGestureRecognizer(recognizer)
}

func handleTap(recognizer: UITapGestureRecognizer) {
}
```

C objetivo

```
- (void)viewDidLoad {
    [super viewDidLoad];
    UITapGestureRecognizer *recognizer =
        [[UITapGestureRecognizer alloc] initWithTarget:self
                                             action:@selector(handleTap:)];
    [self.view addGestureRecognizer:recognizer];
}

- (void)handleTap:(UITapGestureRecognizer *)recognizer {
}
```

Ejemplo de despido de teclado a través de UITapGestureRecognizer:

Primero, creas la función para descartar el teclado:

```
func dismissKeyboard() {
    view.endEditing(true)
}
```

Luego, agrega un reconocedor de gestos de toque en su controlador de vista, llamando al método que acabamos de hacer.


```
let tap: UITapGestureRecognizer = UITapGestureRecognizer(target: self, action:
"dismissKeyboard")
    view.addGestureRecognizer(tap)
```

Ejemplo de obtención de la ubicación del gesto UITapGestureRecognizer (Swift 3):

```
func handleTap(gestureRecognizer: UITapGestureRecognizer) {
    print("tap working")
    if gestureRecognizer.state == UIGestureRecognizerState.recognized
    {
        print(gestureRecognizer.location(in: gestureRecognizer.view))
    }
}
```

UIPanGestureRecognizer

Los reconocedores de gestos de pan detectan gestos de arrastre. El siguiente ejemplo agrega una imagen a un controlador de vista y le permite al usuario arrastrarla en la pantalla.

C objetivo

```
- (void)viewDidLoad {
    [super viewDidLoad];

    UIImageView *imageView = [[UIImageView alloc] initWithImage:[UIImage
imageNamed:@"imageToDrag"]];
    [imageView sizeToFit];
    imageView.userInteractionEnabled = YES;
    [self.view addSubview:imageView];

    UIPanGestureRecognizer *pan = [[UIPanGestureRecognizer alloc] initWithTarget:self
action:@selector(handlePan:)];
    [imageView addGestureRecognizer:pan];
}

- (void)handlePan:(UIPanGestureRecognizer *)recognizer {
    CGPoint translation = [recognizer translationInView:self.view];
    recognizer.view.center = CGPointMake(recognizer.view.center.x + translation.x,
recognizer.view.center.y + translation.y);
    [recognizer setTranslation:CGPointZero inView:self.view];
}
```

Rápido

```
override func viewDidLoad() {
    super.viewDidLoad()

    let imageView = UIImageView.init(image: UIImage.init(named: "imageToDrag"))
    imageView.sizeToFit()
    imageView.isUserInteractionEnabled = true
    self.view.addSubview(imageView)

    let pan = UIPanGestureRecognizer.init(target: self, action:
#selector(handlePan(recognizer:)))
    imageView.addGestureRecognizer(pan)
}
```

```
func handlePan(recognizer: UIPanGestureRecognizer) {
    let translation = recognizer.translation(in: self.view)
    if let view = recognizer.view {
        view.center = CGPoint(x: view.center.x + translation.x, y: view.center.y +
translation.y)
    }
    recognizer.setTranslation(CGPoint.zero, in: self.view)
}
```

Nota: Aunque `UIPanGestureRecognizer` es útil para detectar gestos de arrastre, si solo desea detectar un gesto básico, como que el usuario arrastre su dedo hacia la izquierda / derecha o hacia arriba / abajo, use `UISwipeGestureRecognizer`.

`UIPanGestureRecognizer` es una mejor opción si necesita acceder a métodos como `translationInView:` o `velocityInView:`

UITapGestureRecognizer (Doble toque)

El doble toque, como un solo toque, también utiliza el `UITapGestureRecognizer`. Simplemente establezca el `numberOfTapsRequired` a 2.

Rápido

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Double Tap
    let doubleTapGesture = UITapGestureRecognizer(target: self, action:
#selector(handleDoubleTap))
    doubleTapGesture.numberOfTapsRequired = 2
    doubleTapView.addGestureRecognizer(doubleTapGesture)
}

// Double tap action
func handleDoubleTap() {
    label.text = "Double tap recognized"
}
```

Notas

- Un proyecto de muestra se puede encontrar [aquí](#).
- Podrías reconocer un toque triple configurando `numberOfTapsRequired` a 3.

UILongPressGestureRecognizer

`UILongPressGestureRecognizer` permite escuchar una pulsación larga en una vista. Puede establecer la duración del retraso antes de llamar al método de acción.

Rápido

```
override func viewDidLoad() {
```

```

super.viewDidLoad()

// Long Press
let longPressGesture = UILongPressGestureRecognizer(target: self, action:
#selector(handleLongPress(_:)))
longPressView.addGestureRecognizer(longPressGesture)
}

// Long press action
func handleLongPress(gesture: UILongPressGestureRecognizer) {
    if gesture.state == UIGestureRecognizerState.Began {
        label.text = "Long press recognized"
    }
}
}

```

Notas

- Un proyecto de muestra más completo se puede encontrar [aquí](#) .
- Cambie el valor `minimumPressDuration` para establecer la duración de la pulsación larga.

UISwipeGestureRecognizer

Los gestos de deslizamiento le permiten escuchar al usuario moviendo su dedo por la pantalla rápidamente en una dirección determinada.

Rápido

```

override func viewDidLoad() {
    super.viewDidLoad()

    // Swipe (right and left)
    let swipeRightGesture = UISwipeGestureRecognizer(target: self, action:
#selector(handleSwipe(_:)))
    let swipeLeftGesture = UISwipeGestureRecognizer(target: self, action:
#selector(handleSwipe(_:)))
    swipeRightGesture.direction = UISwipeGestureRecognizerDirection.Right
    swipeLeftGesture.direction = UISwipeGestureRecognizerDirection.Left
    swipeView.addGestureRecognizer(swipeRightGesture)
    swipeView.addGestureRecognizer(swipeLeftGesture)
}

// Swipe action
func handleSwipe(gesture: UISwipeGestureRecognizer) {
    label.text = "Swipe recognized"

    // example task: animate view off screen
    let originalLocation = swipeView.center
    if gesture.direction == UISwipeGestureRecognizerDirection.Right {
        label.text = "Swipe right"
    } else if gesture.direction == UISwipeGestureRecognizerDirection.Left {
        label.text = "Swipe left"
    }
}
}

```

C objetivo

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    UISwipeGestureRecognizer *swipeLeft = [[UISwipeGestureRecognizer alloc] initWithTarget:self
    action:@selector(handleSwipe:)];
    UISwipeGestureRecognizer *swipeRight = [[UISwipeGestureRecognizer alloc] initWithTarget:self
    action:@selector(handleSwipe:)];

    // Setting the swipe direction.
    [swipeLeft setDirection:UISwipeGestureRecognizerDirectionLeft];
    [swipeRight setDirection:UISwipeGestureRecognizerDirectionRight];

    // Adding the swipe gesture on image view
    [self.view addGestureRecognizer:swipeLeft];
    [self.view addGestureRecognizer:swipeRight];
}

//Handling Swipe Gesture Events

- (void)handleSwipe:(UISwipeGestureRecognizer *)swipe {

    if (swipe.direction == UISwipeGestureRecognizerDirectionLeft) {
        NSLog(@"Left Swipe");
    }

    if (swipe.direction == UISwipeGestureRecognizerDirectionRight) {
        NSLog(@"Right Swipe");
    }
}
```

Notas

- Un ejemplo de proyecto más completo se puede encontrar [aquí](#) .

UIPinchGestureRecognizer

Los pellizcos son un gesto de dos dedos donde los dedos se acercan o alejan uno del otro. Este gesto se utiliza generalmente para cambiar el tamaño de una vista.

Rápido

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Pinch
    let pinchGesture = UIPinchGestureRecognizer(target: self, action:
    #selector(handlePinch(_:)))
    pinchView.addGestureRecognizer(pinchGesture)
}

// Pinch action
func handlePinch(gesture: UIPinchGestureRecognizer) {
```

```
label.text = "Pinch recognized"

if gesture.state == UIGestureRecognizerState.Changed {
    let transform = CGAffineTransformMakeScale(gesture.scale, gesture.scale)
    pinchView.transform = transform
}
}
```

Notas

- Un ejemplo de proyecto más completo se puede encontrar [aquí](#) .

UIRotationGestureRecognizer

Se pueden escuchar dos dedos que giran alrededor de un centro con `UIRotationGestureRecognizer` . Esto se utiliza generalmente para rotar una vista.

Rápido

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Rotate
    let rotateGesture = UIRotationGestureRecognizer(target: self, action:
    #selector(handleRotate(_:)))
    rotateView.addGestureRecognizer(rotateGesture)
}

// Rotate action
func handleRotate(gesture: UIRotationGestureRecognizer) {
    label.text = "Rotate recognized"

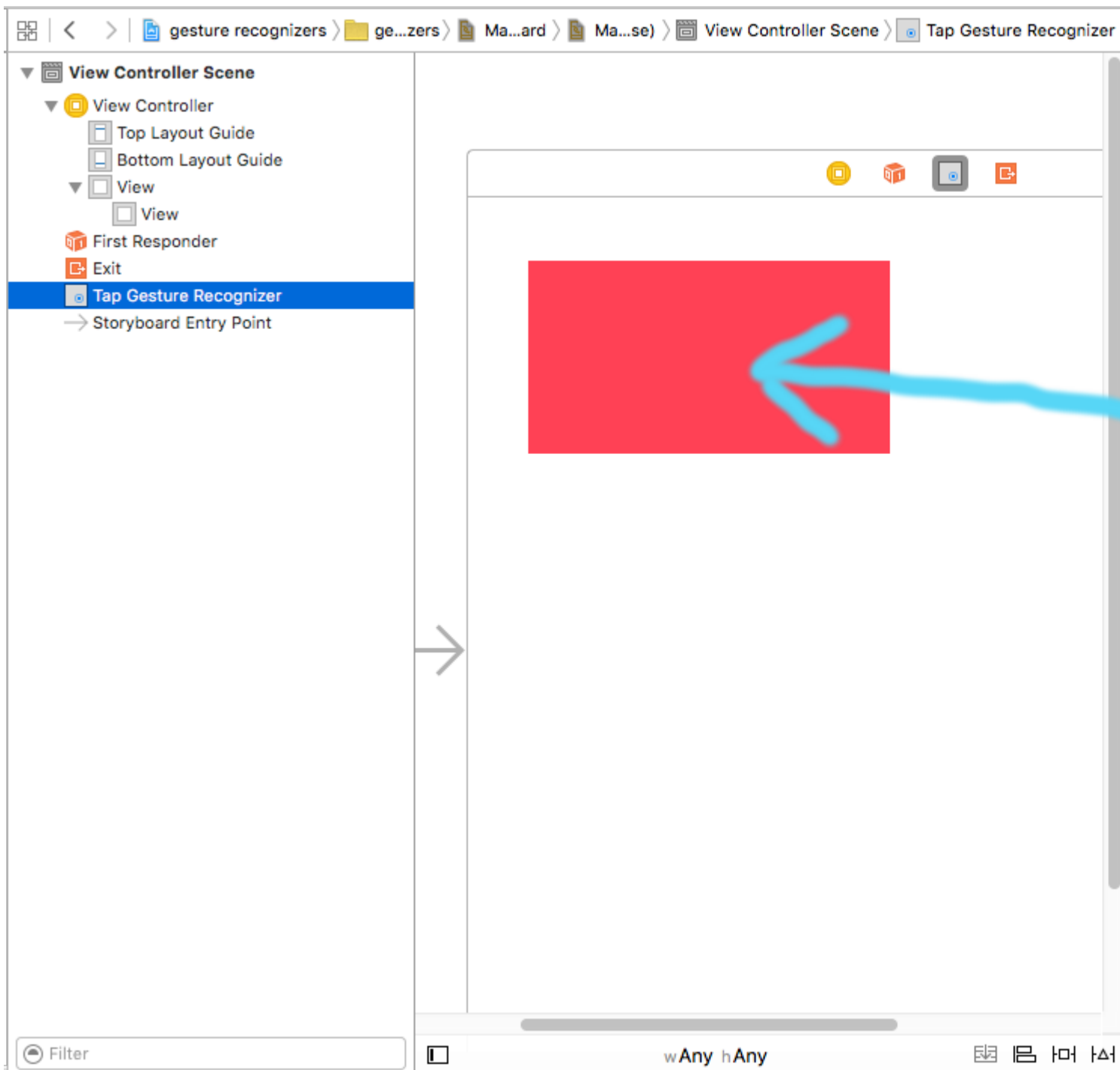
    if gesture.state == UIGestureRecognizerState.Changed {
        let transform = CGAffineTransformMakeRotation(gesture.rotation)
        rotateView.transform = transform
    }
}
```

Notas

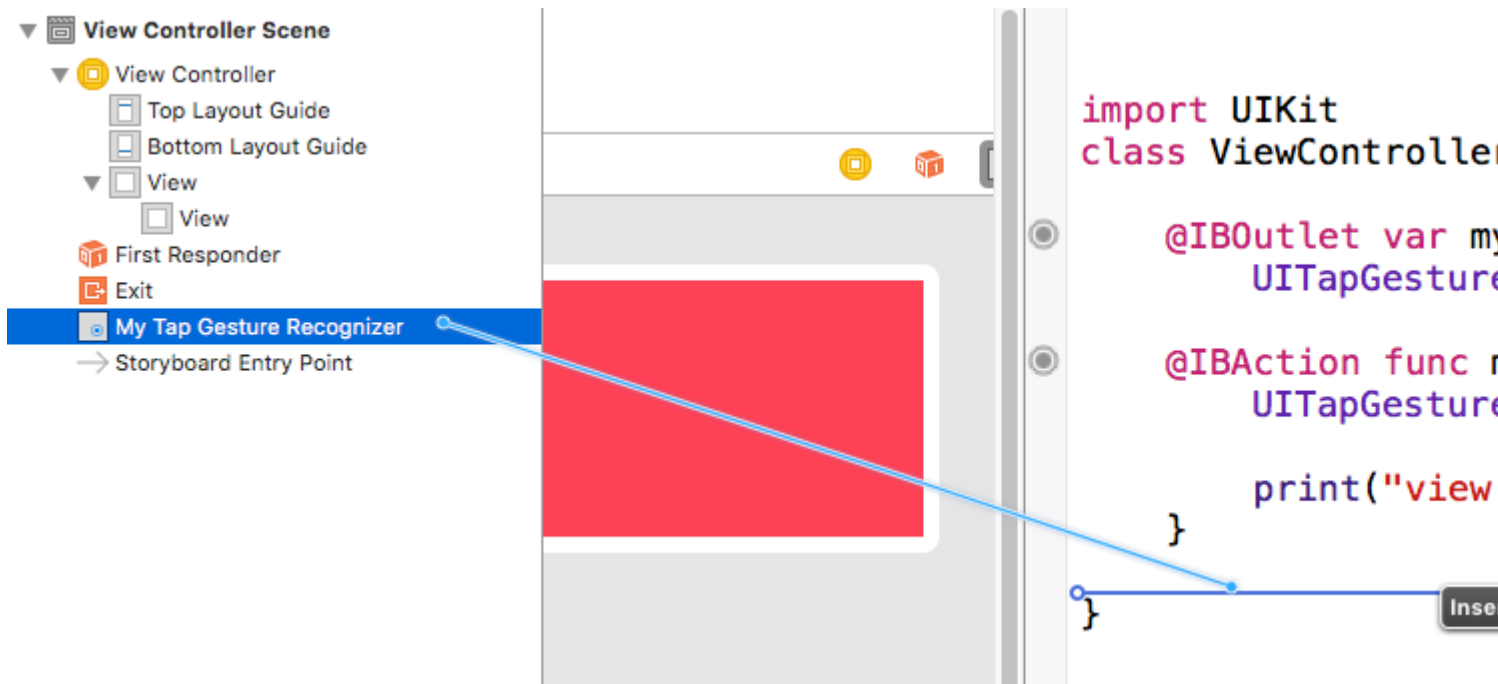
- Un proyecto de muestra se puede encontrar [aquí](#) .

Añadiendo un reconocedor de gestos en el Interface Builder

Arrastre un reconocedor de gestos desde la biblioteca de objetos a su vista.



Controle el arrastre desde el gesto en el Esquema del documento a su código de View Controller para hacer un Outlet y una Acción.



Notas

- Este ejemplo proviene de [este proyecto de ejemplo más completo que muestra los reconocedores de gestos](https://riptutorial.com/es/ios/topic/1289/uigesturerecognizer).

Lea `UIGestureRecognizer` en línea: <https://riptutorial.com/es/ios/topic/1289/uigesturerecognizer>

Capítulo 170: UIImage

Observaciones

Tema desarrollador de Apple para [UIImage](#)

Examples

Creando UIImage

Con imagen local

Rápido

```
let image = UIImage(named: "imageFromBundleOrAsset")
```

C objetivo

```
UIImage *image = [UIImage imageNamed:@"imageFromBundleOrAsset"];
```

Nota

El método `imageNamed` almacena en memoria caché el contenido de la imagen. La carga de muchas imágenes grandes puede provocar advertencias de poca memoria que pueden provocar la finalización de la aplicación. Esto puede solucionarse utilizando el método `imageWithContentsOfFile` de `UIImage`, que no utiliza el almacenamiento en caché.

Con NSData

Rápido

```
let imageData = Data(base64Encoded: imageString, options:
Data.Base64DecodingOptions.ignoreUnknownCharacters)

let image = UIImage(data: imageData!)
```


Con UIColor

Rápido

```
let color = UIColor.red
let size = CGSize(width: 200, height: 200)

 UIGraphicsBeginImageContextWithOptions(size, false, 0.0)
 UIGraphicsGetCurrentContext()!.setFillColor(color.cgColor)
 UIGraphicsGetCurrentContext()!.fill(CGRect(origin: .zero, size: size))
 let colorImage = UIGraphicsGetImageFromCurrentImageContext()
 UIGraphicsEndImageContext()
```

C objetivo

```
UIColor *color=[UIColor redColor];
CGRect frame = CGRectMake(0, 0, 80, 100);
 UIGraphicsBeginImageContext(frame.size);
 CGContextRef context = UIGraphicsGetCurrentContext();
 CGContextSetFillColorWithColor(context, [color CGColor]);
 CGContextFillRect(context, frame);
 UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
 UIGraphicsEndImageContext();
```

Con contenido de archivo

C objetivo

Ejemplo:

```
UIImage *image = [UIImage imageWithContentsOfFile:[NSBundle mainBundle]
pathForResource:@"cellCountry objectForKey:@"Country_Flag" ofType:nil];
```

Usando Array:

Ejemplo:

```
NSMutableArray *imageArray = [[NSMutableArray alloc] init];

for (int imageNumber = 1; self.myPhoto != nil; imageNumber++) {
    NSString *fileName = [NSString stringWithFormat:@"%d.jpg", self.myPhoto];

    // check if a file exists
    if ([UIImage imageNamed:fileName]) {
        // if it exists, add it to the array
        [imageArray addObject:[UIImage imageWithContentsOfFile:[NSBundle
mainBundle]pathForResource:[NSString stringWithFormat:@"%d", fileName] ofType:@""]];
    }
}
```

```
    } else {  
        break;  
    }  
}
```

// Usando matriz de imágenes para animaciones aquí:

```
self.myImageView.animationImages = imageArray;
```

Creando e inicializando objetos de imagen con el contenido del archivo

Crear y devolver un objeto de imagen cargando los datos de imagen del archivo en la ruta especificada.

Ejemplo:

```
UIImage *image = [UIImage imageWithContentsOfFile:[NSBundle mainBundle]  
pathForResource:[cellCountry objectForKey:@"Country_Flag"] ofType:nil];
```

Usando Array:

Ejemplo

```
NSMutableArray *imageArray = [[NSMutableArray alloc] init];  
  
for (int imageNumber = 1; self.myPhoto != nil; imageNumber++) {  
    NSString *fileName = [NSString stringWithFormat:@"%d.jpg", self.myPhoto];  
  
    // check if a file exists  
    if ([UIImage imageNamed:fileName]) {  
        // if it exists, add it to the array  
        [imageArray addObject:[UIImage imageWithContentsOfFile:[NSBundle  
mainBundle]pathForResource:[NSString stringWithFormat:@"%d", fileName] ofType:@""]];  
    } else {  
        break;  
    }  
}  
  
//Using image array for animations here  
self.myImageView.animationImages = imageArray;
```

Imagen redimensionable con gorras.

En el ejemplo de una burbuja de mensaje ilustrada a continuación: las esquinas de la imagen deben permanecer sin cambios, lo que se especifica en `UIEdgeInsets`, pero los bordes y el centro de la imagen deben expandirse para cubrir el nuevo tamaño.





```
let insets = UIEdgeInsetsMake(12.0, 20.0, 22.0, 12.0)
let image = UIImage(named: "test")
image?.resizableImageWithCapInsets(insets, resizingMode: .Stretch)
```

Comparando imágenes

El método `isEqual:` es la única forma confiable de determinar si dos imágenes contienen los mismos datos de imagen. Los objetos de imagen que cree pueden ser diferentes entre sí, incluso cuando los inicializa con los mismos datos de imagen en caché. La única forma de determinar su igualdad es usar el método `isEqual:` que compara los datos de la imagen real. El Listado 1 ilustra las formas correctas e incorrectas de comparar imágenes.

Fuente: [Documentación Apple](#)

Rápido

```
// Load the same image twice.
let image1 = UIImage(named: "MyImage")
let image2 = UIImage(named: "MyImage")

// The image objects may be different, but the contents are still equal
if let image1 = image1, image1.isEqual(image2) {
    // Correct. This technique compares the image data correctly.
}

if image1 == image2 {
    // Incorrect! Direct object comparisons may not work.
}
```

C objetivo

```
// Load the same image twice.
UIImage* image1 = [UIImage imageNamed:@"MyImage"];
UIImage* image2 = [UIImage imageNamed:@"MyImage"];

// The image objects may be different, but the contents are still equal
if ([image1 isEqual:image2]) {
    // Correct. This technique compares the image data correctly.
}

if (image1 == image2) {
    // Incorrect! Direct object comparisons may not work.
}
```

Crea UIImage con UIColor

Rápido

```
let color = UIColor.redColor()
let size = CGSize(width: 200, height: 200)

 UIGraphicsBeginImageContextWithOptions(size, false, 0.0)
 CGContextSetFillColorWithColor(UIGraphicsGetCurrentContext(), color.CGColor)
 CGContextFillRect(UIGraphicsGetCurrentContext(), CGRect(origin: .zero, size: size))
 let colorImage = UIGraphicsGetImageFromCurrentImageContext()
 UIGraphicsEndImageContext()
```

Swift 3

```
let color = UIColor.red()
let size = CGSize(width: 200, height: 200)

 UIGraphicsBeginImageContextWithOptions(size, false, 0.0)
 if let context = UIGraphicsGetCurrentContext() {
     context.setFillColor(color.cgColor)
     context.fill(CGRect(origin: .zero, size: size))
     let colorImage = UIGraphicsGetImageFromCurrentImageContext()
 }
 UIGraphicsEndImageContext()
```

C objetivo:

Agrega este método como una extensión de UIImage :

```
+ (UIImage *)createImageWithColor: (UIColor *)color {
    CGRect rect=CGRectMake(0.0f, 0.0f, 1.0f, 1.0f);
    UIGraphicsBeginImageContext(rect.size);
    CGContextRef context = UIGraphicsGetCurrentContext();
    CGContextSetFillColorWithColor(context, [color CGColor]);
    CGContextFillRect(context, rect);

    UIImage *theImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return theImage;
}
```

Imagen degradada con colores

Creando Gradient UIImage con colores en CGRect

Rápido:

```

extension UIImage {
    static func gradientImageWithBounds(bounds: CGRect, colors: [CGColor]) -> UIImage {
        let gradientLayer = CAGradientLayer()
        gradientLayer.frame = bounds
        gradientLayer.colors = colors

        UIGraphicsBeginImageContext(gradientLayer.bounds.size)
        gradientLayer.render(in: UIGraphicsGetCurrentContext()!)
        let image = UIGraphicsGetImageFromCurrentImageContext()
        UIGraphicsEndImageContext()
        return image!
    }
}

```

Uso:

```

let image = UIImage.gradientImageWithBounds(CGRect(x: 0, y: 0, width: 200, height: 200),
    colors: [UIColor.yellowColor().CGColor, UIColor.blueColor().CGColor])

```

C objetivo:

```

+ (UIImage *)gradientImageWithBounds:(CGRect)bounds colors:(NSArray *)colors {
    CAGradientLayer *gradientLayer = [CAGradientLayer layer];
    gradientLayer.frame = bounds;
    gradientLayer.colors = colors;

    UIGraphicsBeginImageContext(gradientLayer.bounds.size);
    [gradientLayer renderInContext:UIGraphicsGetCurrentContext()];
    UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return image;
}

```

Gradiente de fondo de capa para límites

```

+ (CALayer *)gradientBGLayerForBounds:(CGRect)bounds colors:(NSArray *)colors
{
    CAGradientLayer * gradientBG = [CAGradientLayer layer];
    gradientBG.frame = bounds;
    gradientBG.colors = colors;
    return gradientBG;
}

```

Convertir UIImage a / desde la codificación base64

Codificación

```

//convert the image to NSData first
let imageData:NSData = UIImagePNGRepresentation(image)!
// convert the NSData to base64 encoding
let strBase64:String =
imageData.base64EncodedStringWithOptions(.Encoding64CharacterLineLength)

```

Descodificación

```
let dataDecoded:NSData = NSData(base64EncodedString: strBase64, options:
NSDataBase64DecodingOptions(rawValue: 0))!
let decodedimage:UIImage = UIImage(data: dataDecoded)!
```

Toma una instantánea de una vista

```
//Here self.webView is the view whose screenshot I need to take
//The screenshot is saved in jpg format in the application directory to avoid any loss of
quality in retina display devices i.e. all current devices running iOS 10
 UIGraphicsBeginImageContextWithOptions(self.webView.bounds.size, NO, [UIScreen
 mainScreen].scale);
 [self.webView.layer renderInContext:UIGraphicsGetCurrentContext()];
 UIImage *image = UIGraphicsGetImageFromCurrentImageContext();
 UIGraphicsEndImageContext();
 NSString *jpgPath = [NSHomeDirectory() stringByAppendingPathComponent:@"Documents/Test.jpg"];
 [UIImageJPEGRepresentation(image, 1.0) writeToFile:jpgPath atomically:YES];
 UIImage *pop=[[UIImage alloc] initWithContentsOfFile:jpgPath];
 //pop is the final image in jpg format and high quality with the exact resolution of the view
you selected in pixels and not just points
```

Aplicar UIColor a UIImage

Utilice el mismo UIImage con la aplicación base de múltiples temas simplemente aplicando UIColor a la instancia de UIImage de la siguiente manera.

```
// *** Create an UIImage instance with RenderingMode AlwaysTemplate ***
UIImage *imgMenu = [[UIImage imageNamed:@"iconMenu"]
imageWithRenderingMode:UIImageRenderingModeAlwaysTemplate];

// *** Now Apply `tintColor` to `UIImageView` of UIImageView or UIButton and convert image in
given color ***
[btn setImage:imgMenu forState:UIControlStateNormal]; // Set UIImage in UIButton.

[button.imageView setTintColor:[UIColor blueColor]]; // It changes image color of UIButton to
blue color
```

Ahora digamos que quieres hacer lo mismo con UIImageView y luego usar el siguiente código

```
[imageView setImage:imgMenu]; // Assign UIImage to UIImageView
[imageView setTintColor:[UIColor greenColor]]; // Change imageview image color to green.
[imageView setTintColor:[UIColor redColor]]; // Change imageview image color to red.
```

Cambiar UIImage Color

Swift Añade esta extensión a UIImage:

```
extension UIImage {
    func maskWithColor(color: UIColor) -> UIImage? {

        let maskImage = self.CGImage
        let width = self.size.width
        let height = self.size.height
        let bounds = CGRectMake(0, 0, width, height)
```

```

let colorSpace = CGColorSpaceCreateDeviceRGB()
let bitmapInfo = CGBitmapInfo(rawValue: CGImageAlphaInfo.PremultipliedLast.rawValue)
let bitmapContext = CGContextCreate(nil, Int(width), Int(height), 8, 0,
colorSpace, bitmapInfo.rawValue) //needs rawValue of bitmapInfo

CGContextClipToMask(bitmapContext, bounds, maskImage)
CGContextSetFillColorWithColor(bitmapContext, color.CGColor)
CGContextFillRect(bitmapContext, bounds)

//is it nil?
if let cImage = CGContextCreateImage(bitmapContext) {
    let coloredImage = UIImage(CGImage: cImage)

    return coloredImage
} else {
    return nil
}
}
}

```

Luego, para cambiar el color de tu UIImage.

```
my_image.maskWithColor(UIColor.blueColor())
```

Encontrado [en este enlace](#)

Lea UIImage en línea: <https://riptutorial.com/es/ios/topic/1409/uiimage>

Capítulo 171: UIImagePickerController

Introducción

UIImagePickerController proporciona una solución casi lista para usar que le permite al usuario seleccionar una imagen de su dispositivo o tomar una foto con la cámara y luego presentar esa imagen. Al ajustarse a UIImagePickerControllerDelegate, puede crear una lógica que especifique en su aplicación cómo presentar la imagen y qué hacer con ella (utilizando didFinishPickingMediaWithInfo) y también qué hacer si el usuario se niega a seleccionar una imagen o tomar una foto (mediante UIImagePickerControllerDidCancel).

Examples

Uso genérico de UIImagePickerController

Paso 1: Cree el controlador, establezca el delegado y cumpla con el protocolo

```
//Swift
class ImageUploadViewController: UIViewController, UIImagePickerControllerDelegate,
 UINavigationControllerDelegate {

    let imagePickerController = UIImagePickerController()

    override func viewDidLoad() {
        super.viewDidLoad()
        imagePickerController.delegate = self
    }
}

//Objective-C
@interface ImageUploadViewController : UIViewController
<UIImagePickerControllerDelegate, UINavigationControllerDelegate> {

    UIImagePickerController *imagePickerController;

}

@end

@implementation ImageUploadViewController

- (void)viewDidLoad {

    [super viewDidLoad];

    imagePickerController.delegate = self;

}

@end
```

Nota: en realidad no implementaremos nada definido en UINavigationControllerDelegate, pero

UIImagePickerController hereda de UINavigationController y cambia el comportamiento de UINavigationController . Por lo tanto, aún debemos decir que nuestro controlador de vista se ajusta a UINavigationControllerDelegate .

Paso 2: cuando necesites mostrar UIImagePickerController :

```
//Swift
self.imagePickerController.sourceType = .Camera // options: .Camera , .PhotoLibrary ,
.SavedPhotosAlbum
self.presentViewController(self.imagePickerController, animated: true, completion: nil)

//Objective-C
imagePickerController.sourceType = UIImagePickerControllerSourceTypeCamera; // options:
UIImagePickerControllerSourceTypeCamera, UIImagePickerControllerSourceTypePhotoLibrary,
UIImagePickerControllerSourceTypeSavedPhotosAlbum
[self presentViewController:imagePickerController animated:YES completion:nil];
```

Paso 3: Implementar los métodos de delegado:

```
//Swift
func imagePickerController(picker: UIImagePickerController, didFinishPickingMediaWithInfo
info: [String: AnyObject]) {
    if let pickedImage = info[UIImagePickerControllerOriginalImage] as? UIImage {
        // You have pickedImage now, do your logic here
    }
    self.dismissViewControllerAnimated(true, completion: nil)
}

func imagePickerControllerDidCancel(picker: UIImagePickerController) {
    self.dismissViewControllerAnimated(true, completion: nil)
}

//Objective-C
- (void)imagePickerController:(UIImagePickerController *)picker
didFinishPickingMediaWithInfo:(NSDictionary *)info {

    UIImage *pickedImage = info[UIImagePickerControllerOriginalImage];

    if (pickedImage) {

        //You have pickedImage now, do your logic here

    }

    [self dismissViewControllerAnimated:YES completion:nil];
}

- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker {

    [self dismissViewControllerAnimated:YES completion:nil];
}
}
```

Lea UIImagePickerController en línea:

<https://riptutorial.com/es/ios/topic/3023/uiimagepickercontroller>

Capítulo 172: UIImageView

Examples

Crear un UIImageView

Para crear un `UIImageView` programáticamente, todo lo que necesita hacer es crear una instancia de `UIImageView` :

```
//Swift
let imageView = UIImageView()

//Objective-C
UIImageView *imageView = [[UIImageView alloc] init];
```

Puede establecer el tamaño y la posición de `UIImageView` con un `CGRect` :

```
//Swift
imageView.frame = CGRect(x: 0, y: 0, width: 200, height: 200)

//Objective-C
imageView.frame = CGRectMake(0,0,200,200);
```

O puede establecer el tamaño durante la inicialización:

```
//Swift
UIImageView(frame: CGRect(x: 0, y: 0, width: 200, height: 200))

//Objective-C
UIImageView *imageView = [[UIImageView alloc] initWithFrame:CGRectMake(0,0,200,200);

//Alternative way of defining frame for UIImageView
UIImageView *imageView = [[UIImageView alloc] init];
CGRect imageViewFrame = imageView.frame;
imageViewFrame.size.width = 200;
imageViewFrame.size.height = 200;
imageViewFrame.origin.x = 0;
imageViewFrame.origin.y = 0;
imageView.frame = imageViewFrame;
```

Nota: debe importar `UIKit` para utilizar un `UIImageView` .

Asignando una imagen a un UIImageView

Puede asignar una imagen a un `UIImageView` durante la inicialización, o más tarde, usando la propiedad de `image` :

```
//Swift
UIImageView(image: UIImage(named: "image1"))
```

```
UIImageView(image: UIImage(named: "image1"), highlightedImage: UIImage(named: "image2"))

imageView.image = UIImage(named: "image1")

//Objective-C
[[UIImageView alloc] initWithImage:[UIImage imageNamed:@"image1"];

[[UIImageView alloc] initWithImage:[UIImage imageNamed:@"image1"] highlightedImage:[UIImage
imageNamed:@"image2"]];

imageView.image = [UIImage imageNamed:@"image1"];
```

Animando un UIImageView

Puede animar un `UIImageView` mostrando rápidamente imágenes en él en una secuencia usando las propiedades de animación de `UIImageView` :

```
imageView.animationImages = [UIImage(named: "image1")!,
                             UIImage(named: "image2")!,
                             UIImage(named: "image3")!,
                             UIImage(named: "image4")!,
                             UIImage(named: "image5")!,
                             UIImage(named: "image6")!,
                             UIImage(named: "image7")!,
                             UIImage(named: "image8")!]

imageView.animationDuration = 0.3
imageView.animationRepeatCount = 1
```

La propiedad `animationImages` es una `Array` de `UIImage`s que se ejecuta de arriba a abajo cuando se activa la animación.

La propiedad `animationDuration` es un `Double` dice cuántos segundos durará la animación.

La propiedad `animationRepeatCount` es un `Int` que dice cuántas veces se ejecutará la animación.

Para iniciar y detener la animación, puede llamar a los métodos apropiados para hacerlo:

```
imageView.startAnimating()
imageView.stopAnimating()
```

Hay un método `isAnimating()` que devuelve un valor `Boolean` que indica si la animación se está ejecutando en un momento o no.

Tenga en cuenta que esta no es una forma muy eficiente de crear animaciones: es bastante lenta y consume muchos recursos. Considera usar `Capas` o `Sprites` para obtener mejores resultados

Haciendo una imagen en un círculo o redondeado.

Este ejemplo muestra, cómo hacer un `UIView` o `UIImageView` , redondeado con un radio como este:



C objetivo

```
someImageView.layer.cornerRadius = CGRectGetHeight(someImageView.frame) / 2;  
someImageView.clipsToBounds = YES;
```

Rápido

```
someImageView.layer.cornerRadius = someImageView.frame.height/2  
// this should alleviate the performance hit that adding transparency may cause - see  
// http://stackoverflow.com/a/6254531/189804  
// Be sure to check scrolling performance with Instruments if you take this approach.  
someImageView.layer.shouldRasterize = true  
someImageView.clipsToBounds = true // All parts of the image that are outside its bounds (the  
// frame) are cut out (makes the rounded corners visible)
```

Se sugiere que si se utiliza el diseño automático que se pone el `someImageView.layer.cornerRadius` código en `viewDidLayoutSubviews` . Esto permitirá que el `cornerRadius` de la imagen se actualice si la imagen cambia de tamaño.

```
override func viewDidLayoutSubviews() {  
    super.viewDidLayoutSubviews()  
}
```

```
someImageView.layer.cornerRadius = someImageView.frame.size.width/2
someImageView.layer.masksToBounds = true
}
```

UIImageView enmascarado con etiqueta

Esto hace que la imagen enmascarada a la forma de las letras de la etiqueta:

C objetivo

```
self.maskImage.layer.mask = self.maskLabel.layer;
self.maskImage.layer.masksToBounds = YES;
```

Swift 3

```
maskImageView.mask = maskLabel
maskImageView.masksToBounds = true
```

Aquí está el resultado:



Cambiar color de una imagen.

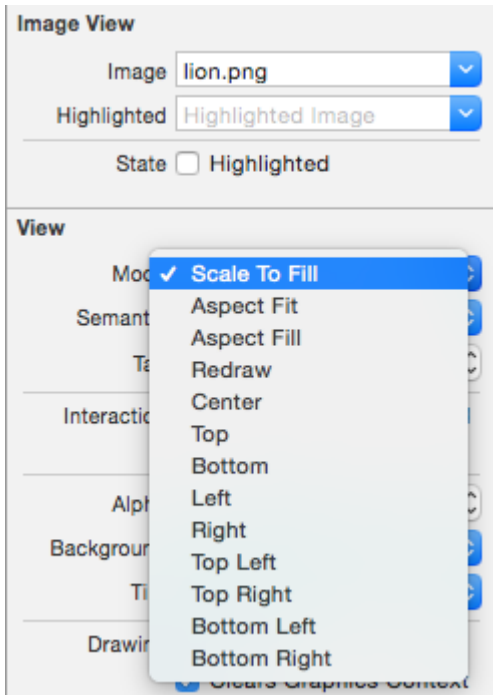
```
//Swift
imageView.tintColor = UIColor.redColor()
imageView.image = imageView.image?.imageWithRenderingMode(.AlwaysTemplate)

//Swift 3
imageView.tintColor = UIColor.red
imageView.image = imageView.image?.withRenderingMode(.alwaysTemplate)

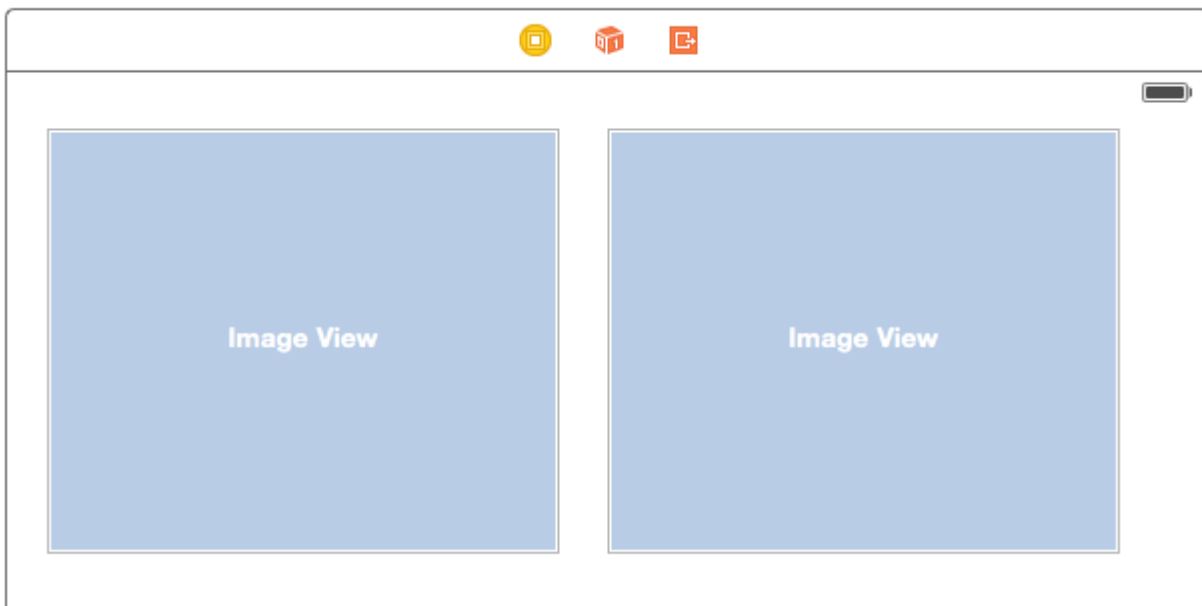
//Objective-C
imageView.tintColor = [UIColor redColor];
imageView.image = [imageView.image imageWithRenderingMode:UIImageRenderingModeAlwaysTemplate]
```

Cómo afecta la propiedad Modo a una imagen

La [propiedad de modo de contenido](#) de una vista dice cómo se debe distribuir su contenido. En el Interface Builder, los distintos modos se pueden seleccionar en el inspector de atributos.



Usemos dos vistas de imagen para ver cómo funcionan los distintos modos.

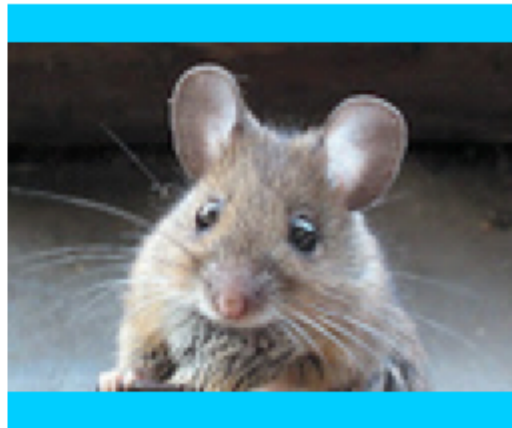


Escala para llenar



Las alturas y los anchos de la imagen se estiran para coincidir con el tamaño de `UIImageView` .

Ajuste de aspecto



El lado más largo (alto o ancho) de la imagen se estira para que coincida con la vista. Esto hace que la imagen sea lo más grande posible mientras se muestra la imagen completa y no distorsiona la altura o el ancho. (Puse el fondo de `UIImageView` en azul para que su tamaño sea claro.)

Relleno de aspecto



El lado más corto (alto o ancho) de la imagen se estira para que coincida con la vista. Al igual que "Ajuste de aspecto", las proporciones de la imagen no se distorsionan de su relación de aspecto original.

Redibujar

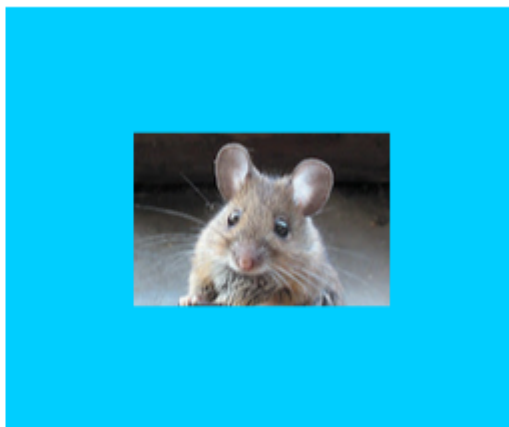


Redibujar es solo para vistas personalizadas que necesitan hacer su propia escala y cambio de tamaño. No estamos usando una vista personalizada, por lo que no deberíamos usar Redraw. Tenga en cuenta que aquí `UIImageView` solo nos da el mismo resultado que Scale to Fill, pero está haciendo más trabajo detrás de escena.

Sobre Redraw, la [documentación de Apple](#) dice:

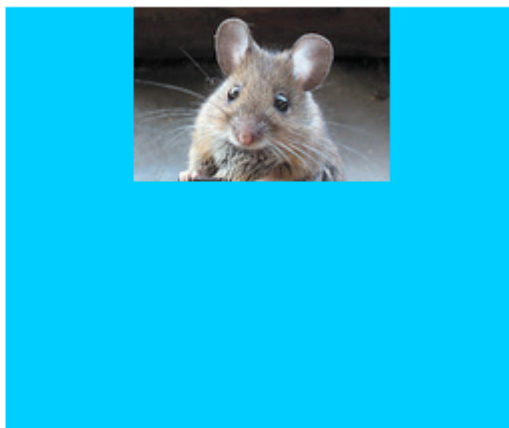
Los modos de contenido son buenos para reciclar el contenido de su vista, pero también puede establecer el modo de contenido en el valor `UIViewContentModeRedraw` cuando desee específicamente que sus vistas personalizadas se `UIViewContentModeRedraw` a dibujar durante la escala y las operaciones de cambio de tamaño. Establecer el modo de contenido de su vista en este valor obliga al sistema a llamar al método `drawRect:` su vista en respuesta a los cambios de geometría. En general, debe evitar usar este valor siempre que sea posible, y ciertamente no debe usarlo con las vistas estándar del sistema.

Centrar



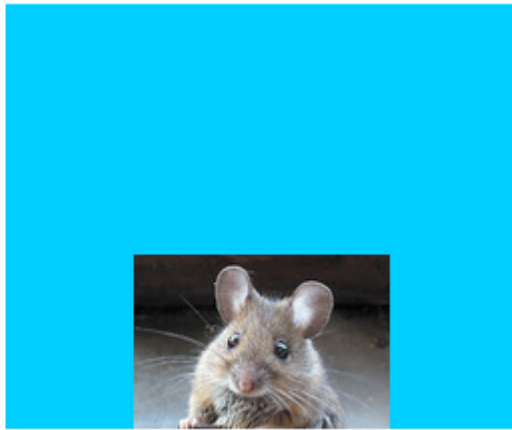
La imagen se centra en la vista, pero la longitud y el ancho de la imagen no se estiran.

Parte superior



El borde superior de la imagen está centrado horizontalmente en la parte superior de la vista, y la longitud y el ancho de la imagen no se estiran.

Fondo



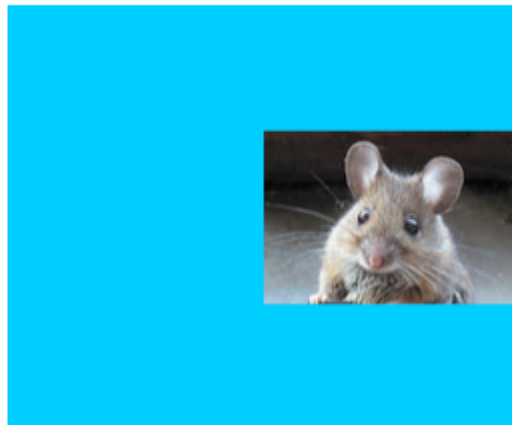
El borde inferior de la imagen se centra horizontalmente en la parte inferior de la vista, y la longitud y el ancho de la imagen no se estiran.

Izquierda



El borde izquierdo de la imagen se centra verticalmente a la izquierda de la vista, y la longitud y el ancho de la imagen no se estiran.

Derecha



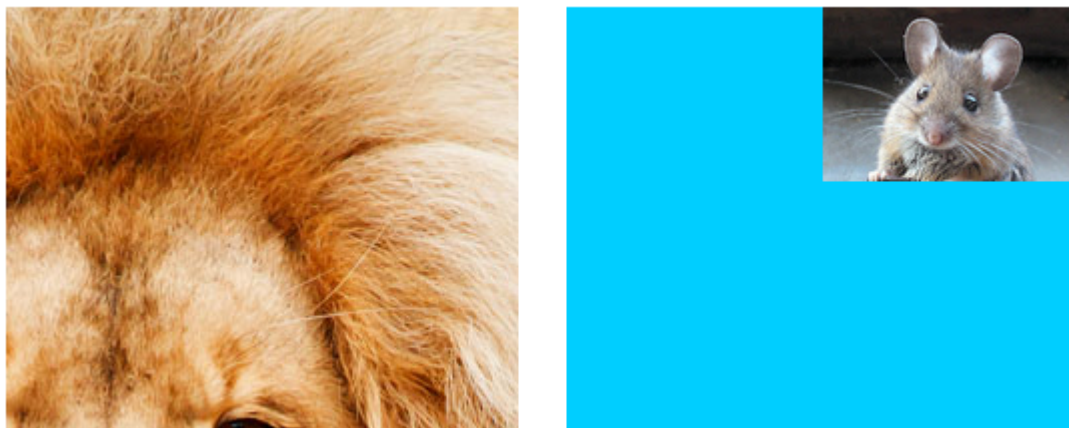
El borde derecho de la imagen se centra verticalmente a la derecha de la vista, y la longitud y el ancho de la imagen no se estiran.

Arriba a la izquierda



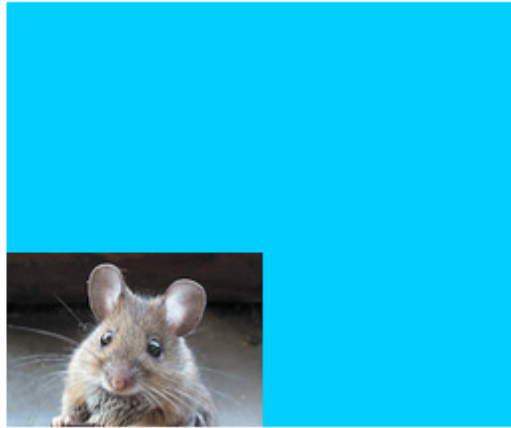
La esquina superior izquierda de la imagen se coloca en la esquina superior izquierda de la vista. La longitud y el ancho de la imagen no se estiran.

Parte superior derecha



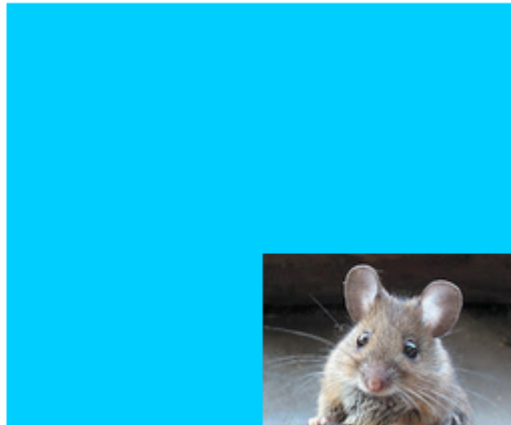
La esquina superior derecha de la imagen se coloca en la esquina superior derecha de la vista. La longitud y el ancho de la imagen no se estiran.

Abajo a la izquierda



La esquina inferior izquierda de la imagen se coloca en la esquina inferior izquierda de la vista. La longitud y el ancho de la imagen no se estiran.

Abajo a la derecha



La esquina inferior derecha de la imagen se coloca en la esquina inferior derecha de la vista. La longitud y el ancho de la imagen no se estiran.

Notas

- Este ejemplo viene originalmente de [aquí](#) .
- Si el contenido (en nuestro caso, la imagen) tiene el mismo tamaño que la vista (en nuestro caso, el `UIImageView`), cambiar el modo de contenido no hará una diferencia notable.
- Vea [esto](#) y [esta](#) pregunta para una discusión sobre los modos de contenido para vistas que no sean `UIImageView` .
- En Swift, para configurar el modo de contenido mediante programación, haga lo siguiente:

```
imageView.contentMode = UIViewContentMode.scaleToFill  
imageView.contentMode = UIViewContentMode.scaleAspectFit
```

```
imageView.contentMode = UIViewContentMode.scaleAspectFill
imageView.contentMode = UIViewContentMode.redraw
imageView.contentMode = UIViewContentMode.center
imageView.contentMode = UIViewContentMode.top
imageView.contentMode = UIViewContentMode.bottom
imageView.contentMode = UIViewContentMode.left
imageView.contentMode = UIViewContentMode.right
imageView.contentMode = UIViewContentMode.topLeft
imageView.contentMode = UIViewContentMode.topRight
imageView.contentMode = UIViewContentMode.bottomLeft
imageView.contentMode = UIViewContentMode.bottomRight
```

Lea UIImageView en línea: <https://riptutorial.com/es/ios/topic/695/uiimageView>

Capítulo 173: UIKit Dynamics

Introducción

UIKit Dynamics es un motor completo de física del mundo real integrado en UIKit. Te permite crear interfaces que se sienten reales al agregar comportamientos como la gravedad, los accesorios, la colisión y las fuerzas. Usted define los rasgos físicos que desea que adopten los elementos de su interfaz, y el motor de dinámica se encarga del resto.

Observaciones

Una cosa importante a tener en cuenta cuando se utiliza UIKit Dynamics es que las vistas posicionadas por el animador no pueden ser posicionadas fácilmente por otros métodos comunes de diseño de iOS.

Los recién llegados a UIKit Dynamics a menudo luchan con esta importante advertencia. Poner restricciones en una vista que también es un elemento de un `UIDynamicBehavior` probablemente causará confusión, ya que tanto el motor de diseño automático como el motor dinámico de animador luchan por la posición apropiada. De manera similar, intentar establecer el marco directamente de una vista controlada por el animador normalmente dará como resultado una animación inestable y una colocación inesperada. Agregar una vista como un elemento a un `UIDynamicBehavior` significa que el animador asumirá la responsabilidad de posicionar una vista y, como tal, los cambios de las posiciones de vista deben implementarse a través del animador.

Se puede configurar el marco de una vista que se está actualizando mediante un animador dinámico, pero se debe seguir inmediatamente enviando un mensaje al animador para actualizar el modelo interno de la jerarquía de vistas del animador. Por ejemplo, si tengo `UILabel`, la `label` que es un elemento de `UIGravityBehavior` puedo moverla a la parte superior de la pantalla para verla caer nuevamente diciendo:

Rápido

```
label.frame = CGRect(x: 0.0, y: 0.0, width: label.intrinsicContentSize.width, height:
label.intrinsicContentSize.height)
dynamicAnimator.updateItem(usingCurrentState: label)
```

C objetivo

```
self.label.frame = CGRectMake(0.0, 0.0, self.label.intrinsicContentSize.width,
self.label.intrinsicContentSize.height);
[self.dynamicAnimator updateItemUsingCurrentState: self.label];
```

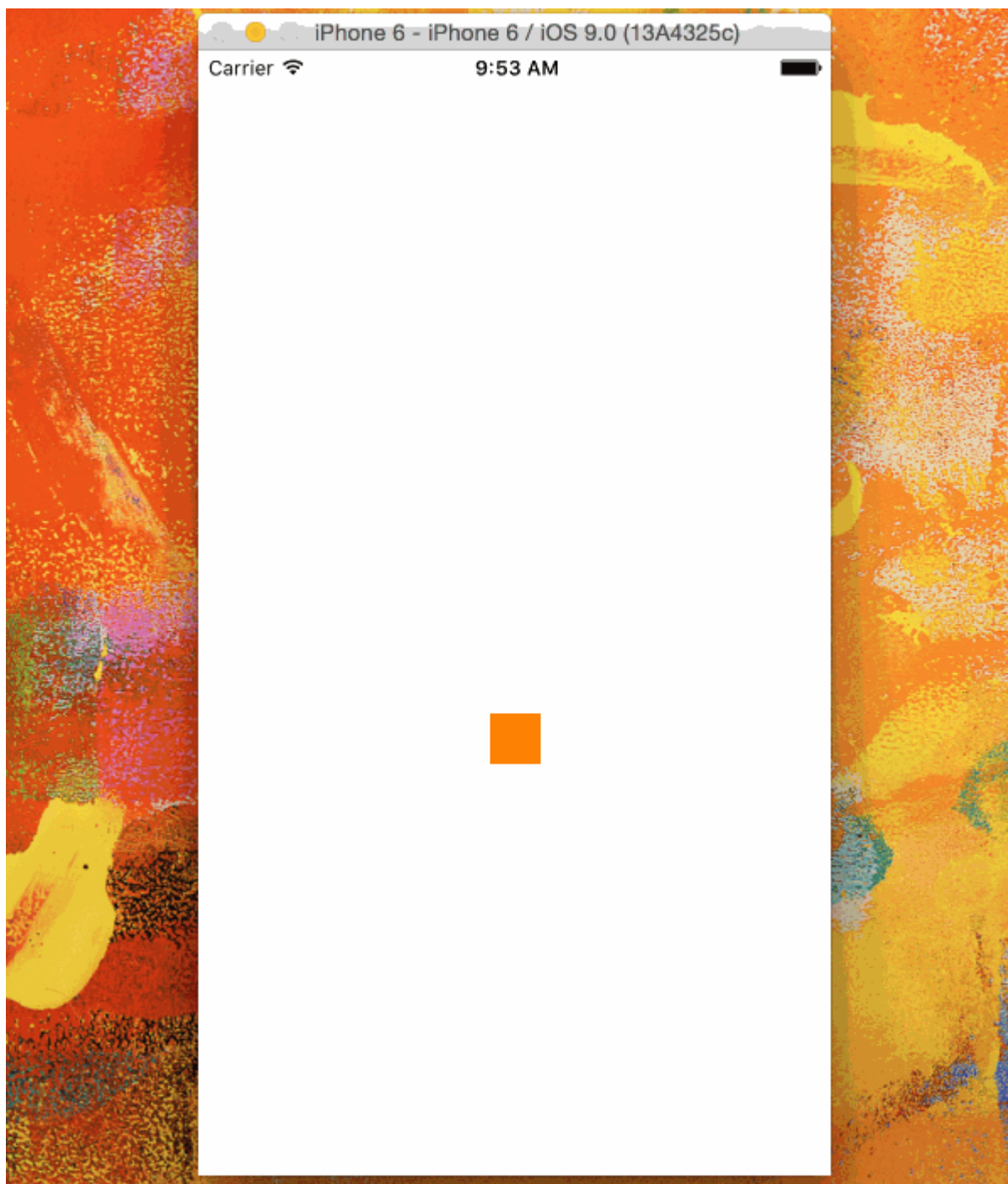
Después de lo cual el animador aplicará el comportamiento de la gravedad desde la nueva ubicación de la etiqueta.

Otra técnica común es usar `UIDynamicBehaviors` para posicionar las vistas. Por ejemplo, si se desea colocar una vista bajo un evento táctil, crear una `UIAttachmentBehavior` y actualizar su punto de `anchorPoint` en una `touchesMoved` o en una acción `UIGestureRecognizer` es una estrategia efectiva.

Examples

La plaza que cae

Permite dibujar un cuadrado en el medio de nuestra vista y hacer que caiga hacia abajo y se detenga en el borde inferior que colisiona con el límite inferior de la pantalla.



```
@IBOutlet var animationView: UIView!
```

```

var squareView:UIView!
var collision: UICollisionBehavior!
var animator: UIDynamicAnimator!
var gravity: UIGravityBehavior!

override func viewDidLoad() {
    super.viewDidLoad()
    let squareSize = CGSize(width: 30.0, height: 30.0)
    let centerPoint = CGPoint(x: self.animationView.bounds.midX - (squareSize.width/2), y:
self.animationView.bounds.midY - (squareSize.height/2))
    let frame = CGRect(origin: centerPoint, size: squareSize)
    squareView = UIView(frame: frame)
    squareView.backgroundColor = UIColor.orangeColor()
    animationView.addSubview(squareView)
    animator = UIDynamicAnimator(referenceView: view)
    gravity = UIGravityBehavior(items: [squareView])
    animator.addBehavior(gravity)
    collision = UICollisionBehavior(items: [square])
    collision.translatesReferenceBoundsIntoBoundary = true
    animator.addBehavior(collision)
}

```

Vista de película basada en la velocidad del gesto

Este ejemplo muestra cómo hacer que una vista haga un seguimiento de un gesto de panorámica y se retire de una manera basada en la física.

Carrier 11:34 AM



Rápido

```

class ViewController: UIViewController
{
    // Adjust to change speed of view from flick
    let magnitudeMultiplier: CGFloat = 0.0008

    lazy var dynamicAnimator: UIDynamicAnimator =
    {
        let dynamicAnimator = UIDynamicAnimator(referenceView: self.view)
        return dynamicAnimator
    }()
}

```



```

lazy var gravity: UIGravityBehavior =
{
    let gravity = UIGravityBehavior(items: [self.orangeView])
    return gravity
}()

lazy var collision: UICollisionBehavior =
{
    let collision = UICollisionBehavior(items: [self.orangeView])
    collision.translatesReferenceBoundsIntoBoundary = true
    return collision
}()

lazy var orangeView: UIView =
{
    let widthHeight: CGFloat = 40.0
    let orangeView = UIView(frame: CGRect(x: 0.0, y: 0.0, width: widthHeight, height:
widthHeight))
    orangeView.backgroundColor = UIColor.orange
    self.view.addSubview(orangeView)
    return orangeView
}()

lazy var panGesture: UIPanGestureRecognizer =
{
    let panGesture = UIPanGestureRecognizer(target: self, action:
#selector(self.handlePan(sender:)))
    return panGesture
}()

lazy var attachment: UIAttachmentBehavior =
{
    let attachment = UIAttachmentBehavior(item: self.orangeView, attachedToAnchor: .zero)
    return attachment
}()

override func viewDidLoad()
{
    super.viewDidLoad()
    dynamicAnimator.addBehavior(gravity)
    dynamicAnimator.addBehavior(collision)
    orangeView.addGestureRecognizer(panGesture)
}

override func viewDidLoadSubviews()
{
    super.viewDidLoadSubviews()
    orangeView.center = view.center
    dynamicAnimator.updateItem(usingCurrentState: orangeView)
}

func handlePan(sender: UIPanGestureRecognizer)
{
    let location = sender.location(in: view)
    let velocity = sender.velocity(in: view)
    let magnitude = sqrt((velocity.x * velocity.x) + (velocity.y * velocity.y))
    switch sender.state
    {
    case .began:
        attachment.anchorPoint = location
    }
}

```

```

        dynamicAnimator.addBehavior(attachment)
    case .changed:
        attachment.anchorPoint = location
    case .cancelled, .ended, .failed, .possible:
        let push = UIPushBehavior(items: [self.orangeView], mode: .instantaneous)
        push.pushDirection = CGVector(dx: velocity.x, dy: velocity.y)
        push.magnitude = magnitude * magnitudeMultiplier
        dynamicAnimator.removeBehavior(attachment)
        dynamicAnimator.addBehavior(push)
    }
}
}

```

C objetivo

```

@interface ViewController ()

@property (nonatomic, assign) CGFloat magnitudeMultiplier;
@property (nonatomic, strong) UIDynamicAnimator *dynamicAnimator;
@property (nonatomic, strong) UIGravityBehavior *gravity;
@property (nonatomic, strong) UICollisionBehavior *collision;
@property (nonatomic, strong) UIView *orangeView;
@property (nonatomic, strong) UIPanGestureRecognizer *panGesture;
@property (nonatomic, strong) UIAttachmentBehavior *attachment;

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.dynamicAnimator addBehavior:self.gravity];
    [self.dynamicAnimator addBehavior:self.collision];
    [self.orangeView addGestureRecognizer:self.panGesture];
    // Adjust to change speed of view from flick
    self.magnitudeMultiplier = 0.0008f;
}

- (void)viewDidLayoutSubviews
{
    [super viewDidLayoutSubviews];
    self.orangeView.center = self.view.center;
    [self.dynamicAnimator updateItemUsingCurrentState:self.orangeView];
}

- (void)handlePan:(UIPanGestureRecognizer *)sender
{
    CGPoint location = [sender locationInView:self.view];
    CGPoint velocity = [sender velocityInView:self.view];
    CGFloat magnitude = sqrt((velocity.x * velocity.x) + (velocity.y * velocity.y));
    if (sender.state == UIGestureRecognizerStateBegan)
    {
        self.attachment.anchorPoint = location;
        [self.dynamicAnimator addBehavior:self.attachment];
    }
    else if (sender.state == UIGestureRecognizerStateChanged)
    {
        self.attachment.anchorPoint = location;
    }
}

```

```

    }
    else if (sender.state == UIGestureRecognizerStateCancelled ||
            sender.state == UIGestureRecognizerStateEnded ||
            sender.state == UIGestureRecognizerStateFailed ||
            sender.state == UIGestureRecognizerStatePossible)
    {
        UIPushBehavior *push = [[UIPushBehavior alloc] initWithItems:@[self.orangeView]
mode:UIPushBehaviorModeInstantaneous];
        push.pushDirection = CGVectorMake(velocity.x, velocity.y);
        push.magnitude = magnitude * self.magnitudeMultiplier;
        [self.dynamicAnimator removeBehavior:self.attachment];
        [self.dynamicAnimator addBehavior:push];
    }
}

#pragma mark - Lazy Init
- (UIDynamicAnimator *)dynamicAnimator
{
    if (!_dynamicAnimator)
    {
        _dynamicAnimator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
    }
    return _dynamicAnimator;
}

- (UIGravityBehavior *)gravity
{
    if (!_gravity)
    {
        _gravity = [[UIGravityBehavior alloc] initWithItems:@[self.orangeView]];
    }
    return _gravity;
}

- (UICollisionBehavior *)collision
{
    if (!_collision)
    {
        _collision = [[UICollisionBehavior alloc] initWithItems:@[self.orangeView]];
        _collision.translatesReferenceBoundsIntoBoundary = YES;
    }
    return _collision;
}

- (UIView *)orangeView
{
    if (!_orangeView)
    {
        CGFloat widthHeight = 40.0f;
        _orangeView = [[UIView alloc] initWithFrame:CGRectMake(0.0, 0.0, widthHeight,
widthHeight)];
        _orangeView.backgroundColor = [UIColor orangeColor];
        [self.view addSubview:_orangeView];
    }
    return _orangeView;
}

- (UIPanGestureRecognizer *)panGesture
{
    if (!_panGesture)
    {

```

```

        _panGesture = [[UIPanGestureRecognizer alloc] initWithTarget:self
action:@selector(handlePan:)];
    }
    return _panGesture;
}

- (UIAttachmentBehavior *)attachment
{
    if (!_attachment)
    {
        _attachment = [[UIAttachmentBehavior alloc] initWithItem:self.orangeView
attachedToAnchor:CGPointZero];
    }
    return _attachment;
}

@end

```

Efecto de las "esquinas pegajosas" usando UITextFieldBehaviors

Este ejemplo muestra cómo lograr un efecto similar al de FaceTime donde se atrae una vista para que apunte una vez que ingresa a una región en particular, en este caso dos regiones, una superior y una inferior.

Carrier 12:59 PM



Rápido

```

class ViewController: UIViewController
{
    lazy var dynamicAnimator: UIDynamicAnimator =
    {
        let dynamicAnimator = UIDynamicAnimator(referenceView: self.view)
        return dynamicAnimator
    }()

    lazy var collision: UICollisionBehavior =
    {
        let collision = UICollisionBehavior(items: [self.orangeView])
        collision.translatesReferenceBoundsIntoBoundary = true
    }()
}

```

```

        return collision
    }()

    lazy var fieldBehaviors: [UIFieldBehavior] =
    {
        var fieldBehaviors = [UIFieldBehavior]()
        for _ in 0 ..< 2
        {
            let field = UIFieldBehavior.springField()
            field.addItem(self.orangeView)
            fieldBehaviors.append(field)
        }
        return fieldBehaviors
    }()

    lazy var itemBehavior: UIDynamicItemBehavior =
    {
        let itemBehavior = UIDynamicItemBehavior(items: [self.orangeView])
        // Adjust these values to change the "stickiness" of the view
        itemBehavior.density = 0.01
        itemBehavior.resistance = 10
        itemBehavior.friction = 0.0
        itemBehavior.allowsRotation = false
        return itemBehavior
    }()

    lazy var orangeView: UIView =
    {
        let widthHeight: CGFloat = 40.0
        let orangeView = UIView(frame: CGRect(x: 0.0, y: 0.0, width: widthHeight, height:
widthHeight))
        orangeView.backgroundColor = UIColor.orange
        self.view.addSubview(orangeView)
        return orangeView
    }()

    lazy var panGesture: UIPanGestureRecognizer =
    {
        let panGesture = UIPanGestureRecognizer(target: self, action:
#selector(self.handlePan(sender:)))
        return panGesture
    }()

    lazy var attachment: UIAttachmentBehavior =
    {
        let attachment = UIAttachmentBehavior(item: self.orangeView, attachedToAnchor: .zero)
        return attachment
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        dynamicAnimator.addBehavior(collision)
        dynamicAnimator.addBehavior(itemBehavior)
        for field in fieldBehaviors
        {
            dynamicAnimator.addBehavior(field)
        }

        orangeView.addGestureRecognizer(panGesture)
    }

```

```

override func viewDidLoadSubviews()
{
    super.viewDidLoadSubviews()

    orangeView.center = view.center
    dynamicAnimator.updateItem(usingCurrentState: orangeView)

    for (index, field) in fieldBehaviors.enumerated()
    {
        field.position = CGPoint(x: view.bounds
            .midX, y: view.bounds.height * (0.25 + 0.5 * CGFloat(index)))
        field.region = UIRegion(size: CGSize(width: view.bounds.width, height:
view.bounds.height * 0.5))
    }
}

func handlePan(sender: UIPanGestureRecognizer)
{
    let location = sender.location(in: view)
    let velocity = sender.velocity(in: view)
    switch sender.state
    {
    case .began:
        attachment.anchorPoint = location
        dynamicAnimator.addBehavior(attachment)
    case .changed:
        attachment.anchorPoint = location
    case .cancelled, .ended, .failed, .possible:
        itemBehavior.addLinearVelocity(velocity, for: self.orangeView)
        dynamicAnimator.removeBehavior(attachment)
    }
}
}
}

```

C objetivo

```

@interface ViewController ()

@property (nonatomic, strong) UIDynamicAnimator *dynamicAnimator;
@property (nonatomic, strong) UICollisionBehavior *collision;
@property (nonatomic, strong) UIAttachmentBehavior *attachment;
@property (nonatomic, strong) UIDynamicItemBehavior *itemBehavior;
@property (nonatomic, strong) NSArray <UIFieldBehavior *> *fieldBehaviors;
@property (nonatomic, strong) UIView *orangeView;
@property (nonatomic, strong) UIPanGestureRecognizer *panGesture;

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.dynamicAnimator addBehavior:self.collision];
    [self.dynamicAnimator addBehavior:self.itemBehavior];
    for (UIFieldBehavior *field in self.fieldBehaviors)
    {
        [self.dynamicAnimator addBehavior:field];
    }
}

```

```

    }

    [self.orangeView addGestureRecognizer:self.panGesture];
}

- (void)viewDidLoadSubviews
{
    [super viewDidLoadSubviews];
    self.orangeView.center = self.view.center;
    [self.dynamicAnimator updateItemUsingCurrentState:self.orangeView];

    for (NSInteger i = 0; i < self.fieldBehaviors.count; i++)
    {
        UITextField *field = self.fieldBehaviors[i];
        field.position = CGPointMake(CGRectGetMidX(self.view.bounds),
        CGRectGetHeight(self.view.bounds) * (0.25f + 0.5f * i));
        field.region = [[UIRegion
        alloc] initWithSize:CGSizeMake(CGRectGetWidth(self.view.bounds),
        CGRectGetHeight(self.view.bounds) * 0.5)];
    }
}

- (void)handlePan:(UIPanGestureRecognizer *)sender
{
    CGPoint location = [sender locationInView:self.view];
    CGPoint velocity = [sender velocityInView:self.view];
    if (sender.state == UIGestureRecognizerStateBegan)
    {
        self.attachment.anchorPoint = location;
        [self.dynamicAnimator addBehavior:self.attachment];
    }
    else if (sender.state == UIGestureRecognizerStateChanged)
    {
        self.attachment.anchorPoint = location;
    }
    else if (sender.state == UIGestureRecognizerStateCancelled ||
             sender.state == UIGestureRecognizerStateEnded ||
             sender.state == UIGestureRecognizerStateFailed ||
             sender.state == UIGestureRecognizerStatePossible)
    {
        [self.itemBehavior addLinearVelocity:velocity forItem:self.orangeView];
        [self.dynamicAnimator removeBehavior:self.attachment];
    }
}

#pragma mark - Lazy Init
- (UIDynamicAnimator *)dynamicAnimator
{
    if (!_dynamicAnimator)
    {
        _dynamicAnimator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
    }
    return _dynamicAnimator;
}

- (UICollisionBehavior *)collision
{
    if (!_collision)
    {
        _collision = [[UICollisionBehavior alloc] initWithItems:@[self.orangeView]];
        _collision.translatesReferenceBoundsIntoBoundary = YES;
    }
}

```

```

    }
    return _collision;
}

- (NSArray <UIFieldBehavior *> *)fieldBehaviors
{
    if (!_fieldBehaviors)
    {
        NSMutableArray *fields = [[NSMutableArray alloc] init];
        for (NSInteger i = 0; i < 2; i++)
        {
            UIFieldBehavior *field = [UIFieldBehavior springField];
            [field addItem:self.orangeView];
            [fields addObject:field];
        }
        _fieldBehaviors = fields;
    }
    return _fieldBehaviors;
}

- (UIDynamicItemBehavior *)itemBehavior
{
    if (!_itemBehavior)
    {
        _itemBehavior = [[UIDynamicItemBehavior alloc] initWithItems:@[self.orangeView]];
        // Adjust these values to change the "stickiness" of the view
        _itemBehavior.density = 0.01;
        _itemBehavior.resistance = 10;
        _itemBehavior.friction = 0.0;
        _itemBehavior.allowsRotation = NO;
    }
    return _itemBehavior;
}

- (UIView *)orangeView
{
    if (!_orangeView)
    {
        CGFloat widthHeight = 40.0f;
        _orangeView = [[UIView alloc] initWithFrame:CGRectMake(0.0, 0.0, widthHeight,
widthHeight)];
        _orangeView.backgroundColor = [UIColor orangeColor];
        [self.view addSubview:_orangeView];
    }
    return _orangeView;
}

- (UIPanGestureRecognizer *)panGesture
{
    if (!_panGesture)
    {
        _panGesture = [[UIPanGestureRecognizer alloc] initWithTarget:self
action:@selector(handlePan:)];
    }
    return _panGesture;
}

- (UIAttachmentBehavior *)attachment
{
    if (!_attachment)
    {

```



```
        _attachment = [[UIAttachmentBehavior alloc] initWithItem:self.orangeView
attachedToAnchor:CGPointZero];
    }
    return _attachment;
}

@end
```

Para obtener más información sobre `UIFieldBehaviors`, puede ver la [Sesión WWDC 2015 "Novedades en UIKit Dynamics and Visual Effects"](#) y el [código de muestra que lo acompaña](#).

UIDynamicBehavior Driven Custom Transition



Este ejemplo muestra cómo crear una transición de presentación personalizada que está controlada por un `UIDynamicBehavior` compuesto. Podemos comenzar por crear un controlador de vista de presentación que presentará un modal.

Rápido

```
class PresentingViewController: UIViewController
{
    lazy var button: UIButton =
    {
        let button = UIButton()
        button.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(button)
        button.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
        button.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        button.setTitle("Present", for: .normal)
        button.setTextColor(UIColor.blue, for: .normal)

        return button
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
    }
}
```

```

        button.addTarget(self, action: #selector(self.didPressPresent), for: .touchUpInside)
    }

    func didPressPresent()
    {
        let modal = ModalViewController()
        modal.view.frame = CGRect(x: 0.0, y: 0.0, width: 200.0, height: 200.0)
        modal.modalPresentationStyle = .custom
        modal.transitioningDelegate = modal
        self.present(modal, animated: true)
    }
}

```

C objetivo

```

@interface PresentingViewController ()
@property (nonatomic, strong) UIButton *button;
@end

@implementation PresentingViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.button addTarget:self action:@selector(didPressPresent)
    forControlEvents:UIControlEventTouchUpInside];
}

- (void)didPressPresent
{
    ModalViewController *modal = [[ModalViewController alloc] init];
    modal.view.frame = CGRectMake(0.0, 0.0, 200.0, 200.0);
    modal.modalPresentationStyle = UIModalPresentationCustom;
    modal.transitioningDelegate = modal;
    [self presentViewController:modal animated:YES completion:nil];
}

- (UIButton *)button
{
    if (!_button)
    {
        _button = [[UIButton alloc] init];
        _button.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_button];
        [_button.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_button.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
        [_button setTitle:@"Present" forState:UIControlStateNormal];
        [_button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal];
    }
    return _button;
}

@end

```

Cuando se pulsa el botón actual, creamos un `ModalViewController` y configuramos su estilo de presentación en `.custom` y configuramos su `transitioningDelegate` a sí mismo. Esto nos permitirá vender un animador que impulsará su transición modal. También configuramos el marco

de la vista `modal` para que sea más pequeño que la pantalla completa.

Veamos ahora `ModalViewController` :

Rápido

```
class ModalViewController: UIViewController
{
    lazy var button: UIButton =
    {
        let button = UIButton()
        button.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(button)
        button.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
        button.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        button.setTitle("Dismiss", for: .normal)
        button.setTitleColor(.white, for: .normal)

        return button
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        button.addTarget(self, action: #selector(self.didPressDismiss), for: .touchUpInside)
        view.backgroundColor = .red
        view.layer.cornerRadius = 15.0
    }

    func didPressDismiss()
    {
        dismiss(animated: true)
    }
}

extension ModalViewController: UIViewControllerTransitioningDelegate
{
    func animationController(forPresented presented: UIViewController, presenting: UIViewController, source: UIViewController) -> UIViewControllerAnimatedTransitioning?
    {
        return DropOutAnimator(duration: 1.5, isAppearing: true)
    }

    func animationController(forDismissed dismissed: UIViewController) -> UIViewControllerAnimatedTransitioning?
    {
        return DropOutAnimator(duration: 4.0, isAppearing: false)
    }
}
```

C objetivo

```
@interface ModalViewController () <UIViewControllerTransitioningDelegate>
@property (nonatomic, strong) UIButton *button;
@end
```

```

@implementation ModalViewController

- (void) viewDidLoad
{
    [super viewDidLoad];
    [self.button addTarget:self action:@selector(didPressPresent)
forControlEvents:UIControlEventTouchUpInside];
    self.view.backgroundColor = [UIColor redColor];
    self.view.layer.cornerRadius = 15.0f;
}

- (void) didPressPresent
{
    [self dismissViewControllerAnimated:YES completion:nil];
}

- (UIButton *) button
{
    if (!_button)
    {
        _button = [[UIButton alloc] init];
        _button.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_button];
        [_button.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
        [_button.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
        [_button setTitle:@"Dismiss" forState:UIControlStateNormal];
        [_button setTitleColor:[UIColor blueColor] forState:UIControlStateNormal];
    }
    return _button;
}

-
(id<UIViewControllerAnimatedTransitioning>) animationControllerForPresentedController:(UIViewController
*) presented presentingController:(UIViewController *) presenting
sourceController:(UIViewController *) source
{
    return [[DropOutAnimator alloc] initWithDuration:1.5 appearing:YES];
}

-
(id<UIViewControllerAnimatedTransitioning>) animationControllerForDismissedController:(UIViewController
*) dismissed
{
    return [[DropOutAnimator alloc] initWithDuration:4.0 appearing:NO];
}

@end

```

Aquí creamos el controlador de vista que se presenta. Además, debido a que `ModalViewController` es su propio `transitioningDelegate`, también es responsable de vender un objeto que administrará su animación de transición. Para nosotros eso significa pasar una instancia de nuestra subclase `UIDynamicBehavior` compuesta.

Nuestro animador tendrá dos transiciones diferentes: una para presentar y otra para despedir. Para la presentación, la vista del controlador de vista de presentación bajará desde arriba y para descartar, la vista parecerá oscilar desde una cuerda y luego se retirará. Debido a que `DropOutAnimator` ajusta a `UIViewControllerAnimatedTransitioning` mayor parte de este trabajo se

realizará en su implementación de la func `animateTransition(using transitionContext: UIViewControllerContextTransitioning)` .

Rápido

```
class DropOutAnimator: UIDynamicBehavior
{
    let duration: TimeInterval
    let isAppearing: Bool

    var transitionContext: UIViewControllerContextTransitioning?
    var hasElapsedTimeExceededDuration = false
    var finishTime: TimeInterval = 0.0
    var collisionBehavior: UICollisionBehavior?
    var attachmentBehavior: UIAttachmentBehavior?
    var animator: UIDynamicAnimator?

    init(duration: TimeInterval = 1.0, isAppearing: Bool)
    {
        self.duration = duration
        self.isAppearing = isAppearing
        super.init()
    }
}

extension DropOutAnimator: UIViewControllerAnimatedTransitioning
{
    func animateTransition(using transitionContext: UIViewControllerContextTransitioning)
    {
        // Get relevant views and view controllers from transitionContext
        guard let fromVC = transitionContext.viewController(forKey: .from),
            let toVC = transitionContext.viewController(forKey: .to),
            let fromView = fromVC.view,
            let toView = toVC.view else { return }

        let containerView = transitionContext.containerView
        let duration = self.transitionDuration(using: transitionContext)

        // Hold reference to transitionContext to notify it of completion
        self.transitionContext = transitionContext

        // Create dynamic animator
        let animator = UIDynamicAnimator(referenceView: containerView)
        animator.delegate = self
        self.animator = animator

        // Presenting Animation
        if self.isAppearing
        {
            fromView.isUserInteractionEnabled = false

            // Position toView just off-screen
            let fromViewInitialFrame = transitionContext.initialFrame(for: fromVC)
            var toViewInitialFrame = toView.frame
            toViewInitialFrame.origin.y -= toViewInitialFrame.height
            toViewInitialFrame.origin.x = fromViewInitialFrame.width * 0.5 -
toViewInitialFrame.width * 0.5
            toView.frame = toViewInitialFrame
```

```

containerView.addSubview(toView)

// Prevent rotation and adjust bounce
let bodyBehavior = UIDynamicItemBehavior(items: [toView])
bodyBehavior.elasticity = 0.7
bodyBehavior.allowsRotation = false

// Add gravity at exaggerated magnitude so animation doesn't seem slow
let gravityBehavior = UIGravityBehavior(items: [toView])
gravityBehavior.magnitude = 10.0

// Set collision bounds to include off-screen view and have collision in center
// where our final view should come to rest
let collisionBehavior = UICollisionBehavior(items: [toView])
let insets = UIEdgeInsets(top: toViewInitialFrame.minY, left: 0.0, bottom:
fromViewInitialFrame.height * 0.5 - toViewInitialFrame.height * 0.5, right: 0.0)
collisionBehavior.setTranslatesReferenceBoundsIntoBoundary(with: insets)
self.collisionBehavior = collisionBehavior

// Keep track of finish time in case we need to end the animator before the
animator pauses
self.finishTime = duration + (self.animator?.elapsedTime ?? 0.0)

// Closure that is called after every "tick" of the animator
// Check if we exceed duration
self.action =
{ [weak self] in
    guard let strongSelf = self,
        (strongSelf.animator?.elapsedTime ?? 0.0) >= strongSelf.finishTime else {
return }

    strongSelf.hasElapsedTimeExceededDuration = true
    strongSelf.animator?.removeBehavior(strongSelf)
}

// `DropOutAnimator` is a composit behavior, so add child behaviors to self
self.addChildBehavior(collisionBehavior)
self.addChildBehavior(bodyBehavior)
self.addChildBehavior(gravityBehavior)

// Add self to dynamic animator
self.animator?.addBehavior(self)
}
// Dismissing Animation
else
{
    // Create allow rotation and have a elastic item
    let bodyBehavior = UIDynamicItemBehavior(items: [fromView])
    bodyBehavior.elasticity = 0.8
    bodyBehavior.angularResistance = 5.0
    bodyBehavior.allowsRotation = true

    // Create gravity with exaggerated magnitude
    let gravityBehavior = UIGravityBehavior(items: [fromView])
    gravityBehavior.magnitude = 10.0

    // Collision boundary is set to have a floor just below the bottom of the screen
    let collisionBehavior = UICollisionBehavior(items: [fromView])
    let insets = UIEdgeInsets(top: 0.0, left: -1000, bottom: -225, right: -1000)
    collisionBehavior.setTranslatesReferenceBoundsIntoBoundary(with: insets)
    self.collisionBehavior = collisionBehavior
}

```

```

    // Attachment behavior so view will have effect of hanging from a rope
    let offset = UIOffset(horizontal: 70.0, vertical: fromView.bounds.height * 0.5)
    var anchorPoint = CGPoint(x: fromView.bounds.maxX - 40.0, y: fromView.bounds.minY)
    anchorPoint = containerView.convert(anchorPoint, from: fromView)
    let attachmentBehavior = UIAttachmentBehavior(item: fromView, offsetFromCenter:
offset, attachedToAnchor: anchorPoint)
    attachmentBehavior.frequency = 3.0
    attachmentBehavior.damping = 3.0
    self.attachmentBehavior = attachmentBehavior

    // `DropOutAnimator` is a composit behavior, so add child behaviors to self
    self.addChildBehavior(collisionBehavior)
    self.addChildBehavior(bodyBehavior)
    self.addChildBehavior(gravityBehavior)
    self.addChildBehavior(attachmentBehavior)

    // Add self to dynamic animator
    self.animator?.addBehavior(self)

    // Animation has two parts part one is hanging from rope.
    // Part two is bouncing off-screen
    // Divide duration in two
    self.finishTime = (2.0 / 3.0) * duration + (self.animator?.elapsedTime ?? 0.0)

    // After every "tick" of animator check if past time limit
    self.action =
    { [weak self] in
        guard let strongSelf = self,
            (strongSelf.animator?.elapsedTime ?? 0.0) >= strongSelf.finishTime else {
return }
            strongSelf.hasElapsedTimeExceededDuration = true
            strongSelf.animator?.removeBehavior(strongSelf)
        }
    }

}

func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval
{
    // Return the duration of the animation
    return self.duration
}
}

extension DropOutAnimator: UIDynamicAnimatorDelegate
{
    func dynamicAnimatorDidPause(_ animator: UIDynamicAnimator)
    {
        // Animator has reached stasis
        if self.isAppearing
        {
            // Check if we are out of time
            if self.hasElapsedTimeExceededDuration
            {
                // Move to final positions
                let toView = self.transitionContext?.viewController(forKey: .to)?.view
                let containerView = self.transitionContext?.containerView
                toView?.center = containerView?.center ?? .zero
                self.hasElapsedTimeExceededDuration = false
            }
        }
    }
}

```



```

    {
        _duration = duration;
        _appearing = appearing;
    }
    return self;
}

- (void) animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    // Get relevant views and view controllers from transitionContext
    UIViewController *fromVC = [transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *fromView = fromVC.view;
    UIView *toView = toVC.view;

    UIView *containerView = transitionContext.containerView;
    NSTimeInterval duration = [self transitionDuration:transitionContext];

    // Hold reference to transitionContext to notify it of completion
    self.transitionContext = transitionContext;

    // Create dynamic animator
    UIDynamicAnimator *animator = [[UIDynamicAnimator
alloc] initWithReferenceView:containerView];
    animator.delegate = self;
    self.animator = animator;

    // Presenting Animation
    if (self.isAppearing)
    {
        fromView.userInteractionEnabled = NO;

        // Position toView just above screen
        CGRect fromViewInitialFrame = [transitionContext
initialFrameForViewController:fromVC];
        CGRect toViewInitialFrame = toView.frame;
        toViewInitialFrame.origin.y -= CGRectGetHeight(toViewInitialFrame);
        toViewInitialFrame.origin.x = CGRectGetWidth(fromViewInitialFrame) * 0.5 -
CGRectGetWidth(toViewInitialFrame) * 0.5;
        toView.frame = toViewInitialFrame;

        [containerView addSubview:toView];

        // Prevent rotation and adjust bounce
        UIDynamicItemBehavior *bodyBehavior = [[UIDynamicItemBehavior
alloc] initWithItems:@[toView]];
        bodyBehavior.elasticity = 0.7;
        bodyBehavior.allowsRotation = NO;

        // Add gravity at exaggerated magnitude so animation doesn't seem slow
        UIGravityBehavior *gravityBehavior = [[UIGravityBehavior
alloc] initWithItems:@[toView]];
        gravityBehavior.magnitude = 10.0f;

        // Set collision bounds to include off-screen view and have collision floor in center
        // where our final view should come to rest
        UICollisionBehavior *collisionBehavior = [[UICollisionBehavior
alloc] initWithItems:@[toView]];
        UIEdgeInsets insets = UIEdgeInsetsMake(CGRectGetMinY(toViewInitialFrame), 0.0,

```

```

CGRectGetHeight(fromViewInitialFrame) * 0.5 - CGRectGetHeight(toViewInitialFrame) * 0.5, 0.0);
[collisionBehavior setTranslatesReferenceBoundsIntoBoundaryWithInsets:insets];
self.collisionBehavior = collisionBehavior;

// Keep track of finish time in case we need to end the animator before the animator
pauses
self.finishTime = duration + self.animator.elapsedTime;

// Closure that is called after every "tick" of the animator
// Check if we exceed duration
__weak ObjcDropOutAnimator *weakSelf = self;
self.action = ^{
    __strong ObjcDropOutAnimator *strongSelf = weakSelf;
    if (strongSelf)
    {
        if (strongSelf.animator.elapsedTime >= strongSelf.finishTime)
        {
            strongSelf.elapsedTimeExceededDuration = YES;
            [strongSelf.animator removeBehavior:strongSelf];
        }
    }
};

// `DropOutAnimator` is a composit behavior, so add child behaviors to self
[self addChildBehavior:collisionBehavior];
[self addChildBehavior:bodyBehavior];
[self addChildBehavior:gravityBehavior];

// Add self to dynamic animator
[self.animator addBehavior:self];
}
// Dismissing Animation
else
{
    // Allow rotation and have a elastic item
    UIDynamicItemBehavior *bodyBehavior = [[UIDynamicItemBehavior alloc]
initWithItems:@[fromView]];
bodyBehavior.elasticity = 0.8;
bodyBehavior.angularResistance = 5.0;
bodyBehavior.allowsRotation = YES;

    // Create gravity with exaggerated magnitude
    UIGravityBehavior *gravityBehavior = [[UIGravityBehavior alloc]
initWithItems:@[fromView]];
gravityBehavior.magnitude = 10.0f;

    // Collision boundary is set to have a floor just below the bottom of the screen
    UICollisionBehavior *collisionBehavior = [[UICollisionBehavior alloc]
initWithItems:@[fromView]];
UIEdgeInsets insets = UIEdgeInsetsMake(0, -1000, -225, -1000);
[collisionBehavior setTranslatesReferenceBoundsIntoBoundaryWithInsets:insets];
self.collisionBehavior = collisionBehavior;

    // Attachment behavior so view will have effect of hanging from a rope
    UIOffset offset = UIOffsetMake(70, -(CGRectGetHeight(fromView.bounds) / 2.0));

    CGPoint anchorPoint = CGPointMake(CGRectGetMaxX(fromView.bounds) - 40,
CGRectGetMinY(fromView.bounds));
    anchorPoint = [containerView convertPoint:anchorPoint fromView:fromView];
    UIAttachmentBehavior *attachBehavior = [[UIAttachmentBehavior alloc]
initWithItem:fromView offsetFromCenter:offset attachedToAnchor:anchorPoint];

```

```

attachBehavior.frequency = 3.0;
attachBehavior.damping = 0.3;
attachBehavior.length = 40;
self.attachBehavior = attachBehavior;

// `DropOutAnimator` is a composit behavior, so add child behaviors to self
[self addChildBehavior:collisionBehavior];
[self addChildBehavior:bodyBehavior];
[self addChildBehavior:gravityBehavior];
[self addChildBehavior:attachBehavior];

// Add self to dynamic animator
[self.Animator addBehavior:self];

// Animation has two parts part one is hanging from rope.
// Part two is bouncing off-screen
// Divide duration in two
self.finishTime = (2./3.) * duration + [self.Animator elapsedTime];

// After every "tick" of animator check if past time limit
__weak ObjcDropOutAnimator *weakSelf = self;
self.action = ^{
    __strong ObjcDropOutAnimator *strongSelf = weakSelf;
    if (strongSelf)
    {
        if ([strongSelf.Animator elapsedTime] >= strongSelf.finishTime)
        {
            strongSelf.elapsedTimeExceededDuration = YES;
            [strongSelf.Animator removeBehavior:strongSelf];
        }
    }
};
}

-
(NSTimeInterval)transitionDuration:(id<UIViewControllerContextTransitioning>)transitionContext
{
    return self.duration;
}

- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    // Animator has reached stasis
    if (self.isAppearing)
    {
        // Check if we are out of time
        if (self.elapsedTimeExceededDuration)
        {
            // Move to final positions
            UIView *toView = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey].view;
            UIView *containerView = [self.transitionContext containerView];
            toView.center = containerView.center;
            self.elapsedTimeExceededDuration = NO;
        }

        // Clean up and call completion
        [self.transitionContext completeTransition:[self.transitionContext
transitionWasCancelled]];
        for (UIDynamicBehavior *behavior in self.childBehaviors)

```

```

    {
        [self removeChildBehavior:behavior];
    }
    [animator removeAllBehaviors];
    self.transitionContext = nil;
}
// Dismissing
else
{
    if (self.attachBehavior)
    {
        // If we have an attachment, we are at the end of part one and start part two.
        [self removeChildBehavior:self.attachBehavior];
        self.attachBehavior = nil;
        [animator addBehavior:self];
        NSTimeInterval duration = [self transitionDuration:self.transitionContext];
        self.finishTime = 1./3. * duration + [animator elapsedTime];
    }
    else
    {
        // Clean up and call completion
        UIView *fromView = [self.transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey].view;
        UIView *toView = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey].view;
        [fromView removeFromSuperview];
        toView.userInteractionEnabled = YES;

        [self.transitionContext completeTransition:![self.transitionContext
transitionWasCancelled]];
        for (UIDynamicBehavior *behavior in self.childBehaviors)
        {
            [self removeChildBehavior:behavior];
        }
        [animator removeAllBehaviors];
        self.transitionContext = nil;
    }
}
}
}
}

```

Como comportamiento compuesto, `DropOutAnimator`, puede combinar una serie de comportamientos diferentes para realizar sus animaciones de presentación y descarte.

`DropOutAnimator` también demuestra cómo usar el bloque de `action` de un comportamiento para inspeccionar las ubicaciones de sus elementos, así como el tiempo transcurrido, una técnica que se puede usar para eliminar vistas que se mueven fuera de la pantalla o truncar animaciones que aún no han alcanzado la estasis.

Para obtener más información, [sesión de la WWDC 2013 "Técnicas avanzadas con dinámica de UIKit"](#) y [SOLPresentingFun](#)

Transición de la sombra con la física del mundo real utilizando comportamientos `UIDynamic`

Este ejemplo muestra cómo realizar una transición de presentación interactiva con física del "mundo real" similar a la pantalla de notificaciones de iOS.

Swipe Down From Top

Para empezar, necesitamos un controlador de vista de presentación en el que aparecerá la sombra. Este controlador de vista también actuará como nuestro

`UINavigationControllerTransitioningDelegate` para nuestro controlador de vista presentado y venderá animadores para nuestra transición. Así que crearemos instancias de nuestros animadores interactivos (uno para presentar, otro para despedir). También crearemos una instancia del controlador de vista de sombra, que, en este ejemplo, es solo un controlador de vista con una etiqueta. Debido a que queremos que el mismo gesto panorámico conduzca toda la interacción, pasamos las referencias al controlador de vista de presentación y la pantalla a nuestros animadores interactivos.

Rápido

```
class ViewController: UINavigationController
{
    var presentingAnimator: ShadeAnimator!
    var dismissingAnimator: ShadeAnimator!
    let shadeVC = ShadeViewController()

    lazy var label: UILabel =
    {
        let label = UILabel()
        label.textColor = .blue
        label.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(label)
        label.centerXAnchor.constraint(equalTo: self.view.centerXAnchor).isActive = true
        label.centerYAnchor.constraint(equalTo: self.view.centerYAnchor).isActive = true
        return label
    }()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        label.text = "Swipe Down From Top"
    }
}
```

```

        presentingAnimator = ShadeAnimator(isAppearing: true, presentingVC: self, presentedVC:
shadeVC, transitionDelegate: self)
        dismissingAnimator = ShadeAnimator(isAppearing: false, presentingVC: self,
presentedVC: shadeVC, transitionDelegate: self)
    }
}
extension ViewController: UIViewControllerTransitioningDelegate
{
    func animationController(forPresented presented: UIViewController, presenting:
UIViewController, source: UIViewController) -> UIViewControllerAnimatedTransitioning?
    {
        return EmptyAnimator()
    }

    func animationController(forDismissed dismissed: UIViewController) ->
UIViewControllerAnimatedTransitioning?
    {
        return EmptyAnimator()
    }

    func interactionControllerForPresentation(using animator:
UIViewControllerAnimatedTransitioning) -> UIViewControllerInteractiveTransitioning?
    {
        return presentingAnimator
    }

    func interactionControllerForDismissal(using animator:
UIViewControllerAnimatedTransitioning) -> UIViewControllerInteractiveTransitioning?
    {
        return dismissingAnimator
    }
}
}

```

C objetivo

```

@interface ObjCViewController () <UIViewControllerTransitioningDelegate>
@property (nonatomic, strong) ShadeAnimator *presentingAnimator;
@property (nonatomic, strong) ShadeAnimator *dismissingAnimator;
@property (nonatomic, strong) UILabel *label;
@property (nonatomic, strong) ShadeViewController *shadeVC;
@end

@implementation ObjCViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.label.text = @"Swipe Down From Top";
    self.shadeVC = [[ShadeViewController alloc] init];
    self.presentingAnimator = [[ShadeAnimator alloc] initWithIsAppearing:YES presentingVC:self
presentedVC:self.shadeVC transitionDelegate:self];
    self.dismissingAnimator = [[ShadeAnimator alloc] initWithIsAppearing:NO presentingVC:self
presentedVC:self.shadeVC transitionDelegate:self];
}

- (UILabel *)label
{
    if (!_label)
    {

```

```

    _label = [[UILabel alloc] init];
    _label.textColor = [UIColor blueColor];
    _label.translatesAutoresizingMaskIntoConstraints = NO;
    [self.view addSubview:_label];
    [_label.centerXAnchor constraintEqualToAnchor:self.view.centerXAnchor].active = YES;
    [_label.centerYAnchor constraintEqualToAnchor:self.view.centerYAnchor].active = YES;
}
return _label;
}

#pragma mark - UIViewControllerTransitioningDelegate

-
(id<UIViewControllerAnimatedTransitioning>)animationControllerForPresentedController:(UIViewController
*)presented presentingController:(UIViewController *)presenting
sourceController:(UIViewController *)source
{
    return [[EmptyAnimator alloc] init];
}

-
(id<UIViewControllerAnimatedTransitioning>)animationControllerForDismissedController:(UIViewController
*)dismissed
{
    return [[EmptyAnimator alloc] init];
}

-
(id<UIViewControllerInteractiveTransitioning>)interactionControllerForPresentation:(id<UIViewControllerAn
*)presented presentingController:(UIViewController *)presenting
sourceController:(UIViewController *)source
{
    return self.presentingAnimator;
}

-
(id<UIViewControllerInteractiveTransitioning>)interactionControllerForDismissal:(id<UIViewControllerAn
*)dismissed presentingController:(UIViewController *)presenting
sourceController:(UIViewController *)source
{
    return self.dismissingAnimator;
}

@end

```

Queremos realmente solo querer presentar nuestra sombra a través de una transición interactiva, pero debido a cómo funciona `UIViewControllerTransitioningDelegate` si no devolvemos un controlador de animación normal, nuestro controlador interactivo nunca se utilizará. Por eso creamos una clase `EmptyAnimator` que se ajusta a `UIViewControllerAnimatedTransitioning`.

Rápido

```

class EmptyAnimator: NSObject
{
}

extension EmptyAnimator: UIViewControllerAnimatedTransitioning
{
}

```

```

func animateTransition(using transitionContext: UIViewControllerContextTransitioning)
{

}

func transitionDuration(using transitionContext: UIViewControllerContextTransitioning?) ->
TimeInterval
{
    return 0.0
}
}

```

C objetivo

```

@implementation EmptyAnimator

- (void)animateTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{

}

- (NSTimeInterval)transitionDuration:(id<UIViewControllerContextTransitioning>)transitionContext
{
    return 0.0;
}

@end

```

Finalmente, necesitamos crear realmente `ShadeAnimator` que es una subclase de `UIDynamicBehavior` que se ajusta a `UIViewControllerInteractiveTransitioning`.

Rápido

```

class ShadeAnimator: UIDynamicBehavior
{
    // Whether we are presenting or dismissing
    let isAppearing: Bool

    // The view controller that is not the shade
    weak var presentingVC: UIViewController?

    // The view controller that is the shade
    weak var presentedVC: UIViewController?

    // The delegate will vend the animator
    weak var transitionDelegate: UIViewControllerTransitioningDelegate?

    // Feedback generator for haptics on collisions
    let impactFeedbackGenerator = UIImpactFeedbackGenerator(style: .light)

    // The context given to the animator at the start of the transition
    var transitionContext: UIViewControllerContextTransitioning?

    // Time limit of the dynamic part of the animation
    var finishTime: TimeInterval = 4.0
}

```



```

// The Pan Gesture that drives the transition. Not using EdgePan because triggers
Notifications screen
lazy var pan: UIPanGestureRecognizer =
{
    let pan = UIPanGestureRecognizer(target: self, action:
#selector(self.handlePan(sender:)))
    return pan
}()

// The dynamic animator that we add `ShadeAnimator` to
lazy var animator: UIDynamicAnimator! =
{
    let animator = UIDynamicAnimator(referenceView: self.transitionContext!.containerView)
    return animator
}()

// init with all of our dependencies
init(isAppearing: Bool, presentingVC: UIViewController, presentedVC: UIViewController,
transitionDelegate: UIViewControllerTransitioningDelegate)
{
    self.isAppearing = isAppearing
    self.presentingVC = presentingVC
    self.presentedVC = presentedVC
    self.transitionDelegate = transitionDelegate
    super.init()
    self.impactFeedbackGenerator.prepare()

    if isAppearing
    {
        self.presentingVC?.view.addGestureRecognizer(pan)
    }
    else
    {
        self.presentedVC?.view.addGestureRecognizer(pan)
    }
}

// Setup and moves shade view controller to just above screen if appearing
func setupViewsForTransition(with transitionContext: UIViewControllerContextTransitioning)
{
    // Get relevant views and view controllers from transitionContext
    guard let fromVC = transitionContext.viewController(forKey: .from),
        let toVC = transitionContext.viewController(forKey: .to),
        let toView = toVC.view else { return }

    let containerView = transitionContext.containerView

    // Hold refrence to transitionContext to notify it of completion
    self.transitionContext = transitionContext
    if isAppearing
    {
        // Position toView just off-screen
        let fromViewInitialFrame = transitionContext.initialFrame(for: fromVC)
        var toViewInitialFrame = toView.frame
        toViewInitialFrame.origin.y -= toViewInitialFrame.height
        toViewInitialFrame.origin.x = fromViewInitialFrame.width * 0.5 -
toViewInitialFrame.width * 0.5
        toView.frame = toViewInitialFrame
    }
}

```

```

        containerView.addSubview(toView)
    }
    else
    {
        fromVC.view.addGestureRecognizer(pan)
    }
}

// Handles the entire interaction from presenting/dismissing to completion
func handlePan(sender: UIPanGestureRecognizer)
{
    let location = sender.location(in: transitionContext?.containerView)
    let velocity = sender.velocity(in: transitionContext?.containerView)
    let fromVC = transitionContext?.viewController(forKey: .from)
    let toVC = transitionContext?.viewController(forKey: .to)

    let touchStartHeight: CGFloat = 90.0
    let touchLocationFromBottom: CGFloat = 20.0

    switch sender.state
    {
    case .began:
        let beginLocation = sender.location(in: sender.view)
        if isAppearing
        {
            guard beginLocation.y <= touchStartHeight,
                let presentedVC = self.presentedVC else { break }
            presentedVC.modalPresentationStyle = .custom
            presentedVC.transitioningDelegate = transitionDelegate
            presentingVC?.present(presentedVC, animated: true)
        }
        else
        {
            guard beginLocation.y >= (sender.view?.frame.height ?? 0.0) - touchStartHeight
            else { break }
            presentedVC?.dismiss(animated: true)
        }
    case .changed:
        guard let view = isAppearing ? toVC?.view : fromVC?.view else { return }
        UIView.animate(withDuration: 0.2)
        {
            view.frame.origin.y = location.y - view.bounds.height +
touchLocationFromBottom
        }

        transitionContext?.updateInteractiveTransition(view.frame.maxY / view.frame.height
        )
    case .ended, .cancelled:
        guard let view = isAppearing ? toVC?.view : fromVC?.view else { return }
        let isCancelled = isAppearing ? (velocity.y < 0.5 || view.center.y < 0.0) :
(velocity.y > 0.5 || view.center.y > 0.0)
        addAttachmentBehavior(with: view, isCancelled: isCancelled)
        addCollisionBehavior(with: view)
        addItemBehavior(with: view)

        animator.addBehavior(self)
        animator.delegate = self

        self.action =
        { [weak self] in
            guard let strongSelf = self else { return }

```

```

        if strongSelf.Animator.elapsedTime > strongSelf.finishTime
        {
            strongSelf.Animator.removeAllBehaviors()
        }
        else
        {
            strongSelf.transitionContext?.updateInteractiveTransition(view.frame.maxY
/ view.frame.height
            )
        }
    }
    default:
        break
    }
}

// Add collision behavior that causes bounce when finished
func addCollisionBehavior(with view: UIView)
{
    let collisionBehavior = UICollisionBehavior(items: [view])
    let insets = UIEdgeInsets(top: -view.bounds.height, left: 0.0, bottom: 0.0, right:
0.0)
    collisionBehavior.setTranslatesReferenceBoundsIntoBoundary(with: insets)
    collisionBehavior.collisionDelegate = self
    self.addChildBehavior(collisionBehavior)
}

// Add attachment behavior that pulls shade either to top or bottom
func addAttachmentBehavior(with view: UIView, isCancelled: Bool)
{
    let anchor: CGPoint
    switch (isAppearing, isCancelled)
    {
        case (true, true), (false, false):
            anchor = CGPoint(x: view.center.x, y: -view.frame.height)
        case (true, false), (false, true):
            anchor = CGPoint(x: view.center.x, y: view.frame.height)
    }
    let attachmentBehavior = UIAttachmentBehavior(item: view, attachedToAnchor: anchor)
    attachmentBehavior.damping = 0.1
    attachmentBehavior.frequency = 3.0
    attachmentBehavior.length = 0.5 * view.frame.height
    self.addChildBehavior(attachmentBehavior)
}

// Makes view more bouncy
func addItemBehavior(with view: UIView)
{
    let itemBehavior = UIDynamicItemBehavior(items: [view])
    itemBehavior.allowsRotation = false
    itemBehavior.elasticity = 0.6
    self.addChildBehavior(itemBehavior)
}
}
extension ShadeAnimator: UIDynamicAnimatorDelegate
{
    // Determines transition has ended
    func dynamicAnimatorDidPause(_ animator: UIDynamicAnimator)
    {
        guard let transitionContext = self.transitionContext else { return }

```

```

let fromVC = transitionContext.viewController(forKey: .from)
let toVC = transitionContext.viewController(forKey: .to)
guard let view = isAppearing ? toVC?.view : fromVC?.view else { return }
switch (view.center.y < 0.0, isAppearing)
{
case (true, true), (true, false):
    view.removeFromSuperview()
    transitionContext.finishInteractiveTransition()
    transitionContext.completeTransition(!isAppearing)
case (false, true):
    toVC?.view.frame = transitionContext.finalFrame(for: toVC!)
    transitionContext.finishInteractiveTransition()
    transitionContext.completeTransition(true)
case (false, false):
    fromVC?.view.frame = transitionContext.initialFrame(for: fromVC!)
    transitionContext.cancelInteractiveTransition()
    transitionContext.completeTransition(false)
}
childBehaviors.forEach { removeChildBehavior($0) }
animator.removeAllBehaviors()
self.animator = nil
self.transitionContext = nil
}
}
extension ShadeAnimator: UICollisionBehaviorDelegate
{
    // Triggers haptics
    func collisionBehavior(_ behavior: UICollisionBehavior, beganContactFor item:
UIDynamicItem, withBoundaryIdentifier identifier: NSCopying?, at p: CGPoint)
    {
        guard p.y > 0.0 else { return }
        impactFeedbackGenerator.impactOccurred()
    }
}
extension ShadeAnimator: UIViewControllerInteractiveTransitioning
{
    // Starts transition
    func startInteractiveTransition(_ transitionContext: UIViewControllerContextTransitioning)
    {
        setupViewsForTransition(with: transitionContext)
    }
}
}

```

C objetivo

```

@interface ShadeAnimator() <UIDynamicAnimatorDelegate, UICollisionBehaviorDelegate>
@property (nonatomic, assign) BOOL isAppearing;
@property (nonatomic, weak) UIViewController *presentingVC;
@property (nonatomic, weak) UIViewController *presentedVC;
@property (nonatomic, weak) NSObject<UIViewControllerTransitioningDelegate>
*transitionDelegate;
@property (nonatomic, strong) UIImpactFeedbackGenerator *impactFeedbackGenerator;
@property (nonatomic, strong) id<UIViewControllerContextTransitioning> transitionContext;
@property (nonatomic, assign) NSTimeInterval finishTime;
@property (nonatomic, strong) UIPanGestureRecognizer *pan;
@property (nonatomic, strong) UIDynamicAnimator *animator;
@end

@implementation ShadeAnimator

```

```

- (instancetype)initWithIsAppearing:(BOOL)isAppearing presentingVC:(UIViewController
*)presentingVC presentedVC:(UIViewController *)presentedVC
transitionDelegate:(id<UIViewControllerTransitioningDelegate>)transitionDelegate
{
    self = [super init];
    if (self)
    {
        _isAppearing = isAppearing;
        _presentingVC = presentingVC;
        _presentedVC = presentedVC;
        _transitionDelegate = transitionDelegate;
        _impactFeedbackGenerator = [[UIImpactFeedbackGenerator
alloc]initWithStyle:UIImpactFeedbackStyleLight];
        [_impactFeedbackGenerator prepare];
        if (_isAppearing)
        {
            [_presentingVC.view addGestureRecognizer:self.pan];
        }
        else
        {
            [_presentedVC.view addGestureRecognizer:self.pan];
        }
    }
    return self;
}

#pragma mark - Lazy Init
- (UIPanGestureRecognizer *)pan
{
    if (!_pan)
    {
        _pan = [[UIPanGestureRecognizer alloc]initWithTarget:self
action:@selector(handlePan:)];
    }
    return _pan;
}

- (UIDynamicAnimator *)animator
{
    if (!_animator)
    {
        _animator = [[UIDynamicAnimator
alloc]initWithReferenceView:self.transitionContext.containerView];
    }
    return _animator;
}

#pragma mark - Setup
-
(void)setupViewForTransitionWithContext:(id<UIViewControllerContextTransitioning>)transitionContext
{
    UIViewController *fromVC = [transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *toView = toVC.view;
    UIView *containerView = transitionContext.containerView;
    self.transitionContext = transitionContext;
    if (self.isAppearing)

```

```

    {
        CGRect fromViewInitialFrame = [transitionContext
initialFrameForViewController:fromVC];
        CGRect toViewInitialFrame = toView.frame;
        toViewInitialFrame.origin.y -= CGRectGetHeight(toViewInitialFrame);
        toViewInitialFrame.origin.x = CGRectGetWidth(fromViewInitialFrame) * 0.5 -
CGRectGetWidth(toViewInitialFrame) * 0.5;

        [containerView addSubview:toView];
    }
else
    {
        [fromVC.view addGestureRecognizer:self.pan];
    }
}

#pragma mark - Gesture
- (void)handlePan:(UIPanGestureRecognizer *)sender
{
    CGPoint location = [sender locationInView:self.transitionContext.containerView];
    CGPoint velocity = [sender velocityInView:self.transitionContext.containerView];
    UIViewController *fromVC = [self.transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];

    CGFloat touchStartHeight = 90.0;
    CGFloat touchLocationFromBottom = 20.0;

    if (sender.state == UIGestureRecognizerStateBegan)
    {
        CGPoint beginLocation = [sender locationInView:sender.view];
        if (self.isAppearing)
        {
            if (beginLocation.y <= touchStartHeight)
            {
                self.presentedVC.modalPresentationStyle = UIModalPresentationCustom;
                self.presentedVC.transitioningDelegate = self.transitionDelegate;
                [self.presentingVC presentViewController:self.presentedVC animated:YES
completion:nil];
            }
        }
        else
        {
            if (beginLocation.y >= [sender locationInView:sender.view].y - touchStartHeight)
            {
                [self.presentedVC dismissViewControllerAnimated:true completion:nil];
            }
        }
    }
    else if (sender.state == UIGestureRecognizerStateChanged)
    {
        UIView *view = self.isAppearing ? toVC.view : fromVC.view;
        [UIView animateWithDuration:0.2 animations:^(
            CGRect frame = view.frame;
            frame.origin.y = location.y - CGRectGetHeight(view.bounds) +
touchLocationFromBottom;
            view.frame = frame;
        )];
        [self.transitionContext updateInteractiveTransition:CGRectGetMaxY(view.frame) /
CGRectGetHeight(view.frame)];
    }
}

```

```

    }
    else if (sender.state == UIGestureRecognizerStateEnded || sender.state ==
UIGestureRecognizerStateCancelled)
    {
        UIView *view = self.isAppearing ? toVC.view : fromVC.view;
        BOOL isCancelled = self.isAppearing ? (velocity.y < 0.5 || view.center.y < 0.0) :
(velocity.y > 0.5 || view.center.y > 0.0);
        [self addAttachmentBehaviorWithView:view isCancelled:isCancelled];
        [self addCollisionBehaviorWithView:view];
        [self addItemBehaviorWithView:view];

        [self.animator addBehavior:self];
        self.animator.delegate = self;

        __weak ShadeAnimator *weakSelf = self;
        self.action =
        ^{
            if (weakSelf.animator.elapsedTime > weakSelf.finishTime)
            {
                [weakSelf.animator removeAllBehaviors];
            }
            else
            {
                [weakSelf.transitionContext
updateInteractiveTransition:CGRectGetMaxY(view.frame) / CGRectGetHeight(view.frame)];
            }
        };
    }
}

#pragma mark - UIViewControllerInteractiveTransitioning
- (void)startInteractiveTransition:(id<UIViewControllerContextTransitioning>)transitionContext
{
    [self setupViewForTransitionWithContext:transitionContext];
}

#pragma mark - Behaviors
- (void)addCollisionBehaviorWithView:(UIView *)view
{
    UICollisionBehavior *collisionBehavior = [[UICollisionBehavior
alloc] initWithItems:@[view]];
    UIEdgeInsets insets = UIEdgeInsetsMake(-CGRectGetHeight(view.bounds), 0.0, 0.0, 0.0);
    [collisionBehavior setTranslatesReferenceBoundsIntoBoundaryWithInsets:insets];
    collisionBehavior.collisionDelegate = self;
    [self addChildBehavior:collisionBehavior];
}

- (void)addItemBehaviorWithView:(UIView *)view
{
    UIDynamicItemBehavior *itemBehavior = [[UIDynamicItemBehavior
alloc] initWithItems:@[view]];
    itemBehavior.allowsRotation = NO;
    itemBehavior.elasticity = 0.6;
    [self addChildBehavior:itemBehavior];
}

- (void)addAttachmentBehaviorWithView:(UIView *)view isCancelled:(BOOL)isCancelled
{
    CGPoint anchor;
    if ((self.isAppearing && isCancelled) || (!self.isAppearing && isCancelled))
    {

```

```

        anchor = CGPointMake(view.center.x, -CGRectGetHeight(view.frame));
    }
    else
    {
        anchor = CGPointMake(view.center.x, -CGRectGetHeight(view.frame));
    }
    UIAttachmentBehavior *attachmentBehavior = [[UIAttachmentBehavior alloc] initWithItem:view
attachedToAnchor:anchor];
    attachmentBehavior.damping = 0.1;
    attachmentBehavior.frequency = 3.0;
    attachmentBehavior.length = 0.5 * CGRectGetHeight(view.frame);
    [self addChildBehavior:attachmentBehavior];
}

#pragma mark - UICollisionBehaviorDelegate
- (void)collisionBehavior:(UICollisionBehavior *)behavior
beganContactForItem:(id<UIDynamicItem>)item withBoundaryIdentifier:(id<NSCopying>)identifier
atPoint:(CGPoint)p
{
    if (p.y > 0.0)
    {
        [self.impactFeedbackGenerator impactOccurred];
    }
}

#pragma mark - UIDynamicAnimatorDelegate
- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    UIViewController *fromVC = [self.transitionContext
viewControllerForKey:UITransitionContextFromViewControllerKey];
    UIViewController *toVC = [self.transitionContext
viewControllerForKey:UITransitionContextToViewControllerKey];
    UIView *view = self.isAppearing ? toVC.view : fromVC.view;
    if (view.center.y < 0.0 && (self.isAppearing || !self.isAppearing))
    {
        [view removeFromSuperview];
        [self.transitionContext finishInteractiveTransition];
        [self.transitionContext completeTransition:!self.isAppearing];
    }
    else if (view.center.y >= 0.0 && self.isAppearing)
    {
        toVC.view.frame = [self.transitionContext finalFrameForViewController:toVC];
        [self.transitionContext finishInteractiveTransition];
        [self.transitionContext completeTransition:YES];
    }
    else
    {
        fromVC.view.frame = [self.transitionContext initialFrameForViewController:fromVC];
        [self.transitionContext cancelInteractiveTransition];
        [self.transitionContext completeTransition:NO];
    }
    for (UIDynamicBehavior *behavior in self.childBehaviors)
    {
        [self removeChildBehavior:behavior];
    }
    [animator removeAllBehaviors];
    self.animator = nil;
    self.transitionContext = nil;
}

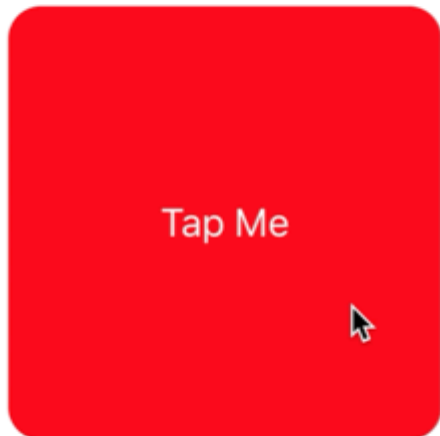
@end

```


El animador activa el inicio de la transición cuando comienza el gesto de desplazamiento. Y simplemente mueve la vista a medida que cambia el gesto. Pero cuando el gesto termina, es cuando `UIDynamicBehaviors` determina si la transición debe completarse o cancelarse. Para ello utiliza un comportamiento adjunto y colisión. Para obtener más información, consulte la [Sesión WWDC 2013 "Técnicas avanzadas con UIKit Dynamics"](#).

Animación dinámica del mapa Cambios de posición a límites

Este ejemplo muestra cómo personalizar el protocolo `UIDynamicItem` para asignar cambios de posición de una vista que se anima dinámicamente a cambios de límites para crear un `UIButton` que se expande y se contrae de manera elástica.



Para comenzar, necesitamos crear un nuevo protocolo que implemente `UIDynamicItem` pero que también tenga una propiedad de `bounds` configurables y obtenibles.

Rápido

```
protocol ResizableDynamicItem: UIDynamicItem
{
    var bounds: CGRect { set get }
}
extension UIView: ResizableDynamicItem {}
```

C objetivo

```
@protocol ResizableDynamicItem <UIDynamicItem>
@property (nonatomic, readwrite) CGRect bounds;
@end
```

Luego, crearemos un objeto envoltorio que envolverá un `UIDynamicItem` pero asignará los cambios del centro a la anchura y la altura del elemento. También proporcionaremos pasajes para los `bounds` y la `transform` del elemento subyacente. Esto provocará que los cambios que el animador dinámico realice en los valores `x` e `y` del centro del elemento subyacente se apliquen al ancho y alto del elemento.

Rápido

```
final class PositionToBoundsMapping: NSObject, UIDynamicItem
{
    var target: ResizableDynamicItem

    init(target: ResizableDynamicItem)
    {
        self.target = target
        super.init()
    }

    var bounds: CGRect
    {
        get
        {
            return self.target.bounds
        }
    }

    var center: CGPoint
    {
        get
        {
            return CGPoint(x: self.target.bounds.width, y: self.target.bounds.height)
        }
        set
        {

```

```

        self.target.bounds = CGRect(x: 0.0, y: 0.0, width: newValue.x, height: newValue.y)
    }
}

var transform: CGAffineTransform
{
    get
    {
        return self.target.transform
    }

    set
    {
        self.target.transform = newValue
    }
}
}

```

C objetivo

```

@interface PositionToBoundsMapping ()
@property (nonatomic, strong) id<ResizableDynamicItem> target;
@end

@implementation PositionToBoundsMapping

- (instancetype)initWithTarget:(id<ResizableDynamicItem>)target
{
    self = [super init];
    if (self)
    {
        _target = target;
    }
    return self;
}

- (CGRect)bounds
{
    return self.target.bounds;
}

- (CGPoint)center
{
    return CGPointMake(self.target.bounds.size.width, self.target.bounds.size.height);
}

- (void)setCenter:(CGPoint)center
{
    self.target.bounds = CGRectMake(0, 0, center.x, center.y);
}

- (CGAffineTransform)transform
{
    return self.target.transform;
}

- (void)setTransform:(CGAffineTransform)transform
{
    self.target.transform = transform;
}

```

```
}  
  
@end
```

Finalmente, crearemos un `UIViewController` que tendrá un botón. Cuando se presiona el botón, crearemos `PositionToBoundsMapping` con el botón como elemento dinámico envuelto. Creamos un `UIAttachmentBehavior` en su posición actual y luego le agregamos un `UIPushBehavior` instantáneo. Sin embargo, debido a que hemos mapeado los cambios de sus límites, el botón no se mueve, sino que crece y se reduce.

Rápido

```
final class ViewController: UIViewController  
{  
    lazy var button: UIButton =  
    {  
        let button = UIButton(frame: CGRect(x: 0.0, y: 0.0, width: 300.0, height: 200.0))  
        button.backgroundColor = .red  
        button.layer.cornerRadius = 15.0  
        button.setTitle("Tap Me", for: .normal)  
        self.view.addSubview(button)  
        return button  
    }()  
  
    var buttonBounds = CGRect.zero  
    var animator: UIDynamicAnimator?  
  
    override func viewDidLoad()  
    {  
        super.viewDidLoad()  
        view.backgroundColor = .white  
        button.addTarget(self, action: #selector(self.didPressButton(sender:)), for:  
.touchUpInside)  
        buttonBounds = button.bounds  
    }  
  
    override func viewDidLoadSubviews()  
    {  
        super.viewDidLoadSubviews()  
        button.center = view.center  
    }  
  
    func didPressButton(sender: UIButton)  
    {  
        // Reset bounds so if button is press twice in a row, previous changes don't propogate  
        button.bounds = buttonBounds  
        let animator = UIDynamicAnimator(referenceView: view)  
  
        // Create mapping  
        let buttonBoundsDynamicItem = PositionToBoundsMapping(target: button)  
  
        // Add Attachment behavior  
        let attachmentBehavior = UIAttachmentBehavior(item: buttonBoundsDynamicItem,  
attachedToAnchor: buttonBoundsDynamicItem.center)  
  
        // Higher frequency faster oscillation  
        attachmentBehavior.frequency = 2.0
```

```

        // Lower damping longer oscillation lasts
        attachmentBehavior.damping = 0.1
        animator.addBehavior(attachmentBehavior)

        let pushBehavior = UIPushBehavior(items: [buttonBoundsDynamicItem], mode:
.instantaneous)

        // Change angle to determine how much height/ width should change 45° means
height:width is 1:1
        pushBehavior.angle = .pi / 4.0

        // Larger magnitude means bigger change
        pushBehavior.magnitude = 30.0
        animator.addBehavior(pushBehavior)
        pushBehavior.active = true

        // Hold refrence so animator is not released
        self.animator = animator
    }
}

```

C objetivo

```

@interface ViewController ()
@property (nonatomic, strong) UIButton *button;
@property (nonatomic, assign) CGRect buttonBounds;
@property (nonatomic, strong) UIDynamicAnimator *animator;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.view.backgroundColor = [UIColor whiteColor];
    [self.button addTarget:self action:@selector(didTapButton:)
forControlEvents:UIControlEventTouchUpInside];
    self.buttonBounds = self.button.bounds;
}

- (void)viewDidLayoutSubviews
{
    [super viewDidLayoutSubviews];
    self.button.center = self.view.center;
}

- (UIButton *)button
{
    if (!_button)
    {
        _button = [[UIButton alloc] initWithFrame:CGRectMake(0.0, 0.0, 200.0, 200.0)];
        _button.backgroundColor = [UIColor redColor];
        _button.layer.cornerRadius = 15.0;
        [_button setTitle:@"Tap Me" forState:UIControlStateNormal];
        [self.view addSubview:_button];
    }
    return _button;
}

```

```
- (void)didTapButton:(id)sender
{
    self.button.bounds = self.buttonBounds;
    UIDynamicAnimator *animator = [[UIDynamicAnimator alloc] initWithReferenceView:self.view];
    PositionToBoundsMapping *buttonBoundsDynamicItem = [[PositionToBoundsMapping
alloc] initWithTarget:sender];
    UIAttachmentBehavior *attachmentBehavior = [[UIAttachmentBehavior
alloc] initWithItem:buttonBoundsDynamicItem attachedToAnchor:buttonBoundsDynamicItem.center];
    [attachmentBehavior setFrequency:2.0];
    [attachmentBehavior setDamping:0.3];
    [animator addBehavior:attachmentBehavior];

    UIPushBehavior *pushBehavior = [[UIPushBehavior alloc]
initWithItems:@[buttonBoundsDynamicItem] mode:UIPushBehaviorModeInstantaneous];
    pushBehavior.angle = M_PI_4;
    pushBehavior.magnitude = 2.0;
    [animator addBehavior:pushBehavior];

    [pushBehavior setActive:TRUE];

    self.animator = animator;
}

@end
```

Para más información vea el [catálogo de UIKit Dynamics](#).

Lea UIKit Dynamics en línea: <https://riptutorial.com/es/ios/topic/9479/uikit-dynamics>

Capítulo 174: UIKit Dynamics con UICollectionView

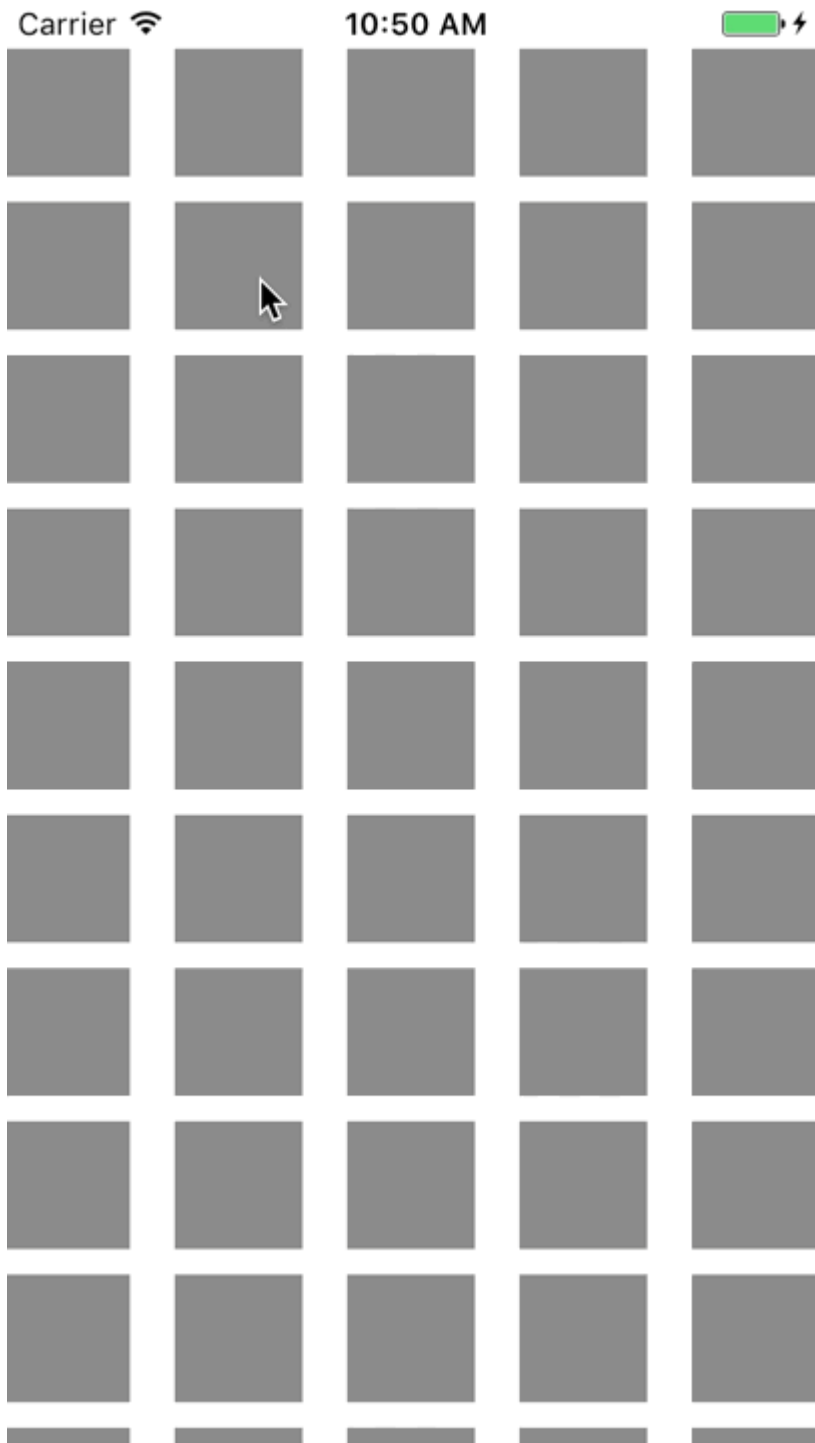
Introducción

UIKit Dynamics es un motor de física integrado en UIKit. UIKit Dynamics ofrece un conjunto de API que ofrece interoperabilidad con `UICollectionView` y `UICollectionViewLayout`

Examples

Creación de un comportamiento de arrastre personalizado con UIDynamicAnimator

Este ejemplo muestra cómo crear un comportamiento de arrastre personalizado mediante la subclase `UIDynamicBehavior` y la subclase `UICollectionViewFlowLayout`. En el ejemplo, tenemos `UICollectionView` que permite la selección de varios elementos. Luego, con un gesto de pulsación prolongada, esos elementos se pueden arrastrar en una animación elástica y "elástica" dirigida por un `UIDynamicAnimator`.



El comportamiento de arrastre se produce combinando un comportamiento de bajo nivel que agrega un comportamiento `UIAttachmentBehavior` a las esquinas de un `UIDynamicItem` y un comportamiento de alto nivel que administra el comportamiento de bajo nivel para varios `UIDynamicItems`.

Podemos comenzar creando este comportamiento de bajo nivel, lo llamaremos `RectangleAttachmentBehavior`

Rápido

```
final class RectangleAttachmentBehavior: UIDynamicBehavior
{
```



```

init(item: UIDynamicItem, point: CGPoint)
{
    // Higher frequency more "ridged" formation
    let frequency: CGFloat = 8.0

    // Lower damping longer animation takes to come to rest
    let damping: CGFloat = 0.6

    super.init()

    // Attachment points are four corners of item
    let points = self.attachmentPoints(for: point)

    let attachmentBehaviors: [UIAttachmentBehavior] = points.map
    {
        let attachmentBehavior = UIAttachmentBehavior(item: item, attachedToAnchor: $0)
        attachmentBehavior.frequency = frequency
        attachmentBehavior.damping = damping
        return attachmentBehavior
    }

    attachmentBehaviors.forEach
    {
        addChildBehavior($0)
    }
}

func updateAttachmentLocation(with point: CGPoint)
{
    // Update anchor points to new attachment points
    let points = self.attachmentPoints(for: point)
    let attachments = self.childBehaviors.flatMap { $0 as? UIAttachmentBehavior }
    let pairs = zip(points, attachments)
    pairs.forEach { $0.1.anchorPoint = $0.0 }
}

func attachmentPoints(for point: CGPoint) -> [CGPoint]
{
    // Width and height should be close to the width and height of the item
    let width: CGFloat = 40.0
    let height: CGFloat = 40.0

    let topLeft = CGPoint(x: point.x - width * 0.5, y: point.y - height * 0.5)
    let topRight = CGPoint(x: point.x + width * 0.5, y: point.y - height * 0.5)
    let bottomLeft = CGPoint(x: point.x - width * 0.5, y: point.y + height * 0.5)
    let bottomRight = CGPoint(x: point.x + width * 0.5, y: point.y + height * 0.5)
    let points = [topLeft, topRight, bottomLeft, bottomRight]
    return points
}
}

```

C objetivo

```

@implementation RectangleAttachmentBehavior

- (instancetype) initWithItem: (id<UIDynamicItem>) item point: (CGPoint) point
{
    CGFloat frequency = 8.0f;
    CGFloat damping = 0.6f;
}

```

```

self = [super init];
if (self)
{
    NSArray <NSValue *> *pointValues = [self attachmentPointValuesForPoint:point];
    for (NSValue *value in pointValues)
    {
        UIAttachmentBehavior *attachment = [[UIAttachmentBehavior alloc] initWithItem:item
attachedToAnchor:[value CGPointValue]];
        attachment.frequency = frequency;
        attachment.damping = damping;
        [self addChildBehavior:attachment];
    }
}
return self;
}

- (void)updateAttachmentLocationWithPoint:(CGPoint)point
{
    NSArray <NSValue *> *pointValues = [self attachmentPointValuesForPoint:point];
    for (NSInteger i = 0; i < pointValues.count; i++)
    {
        NSValue *pointValue = pointValues[i];
        UIAttachmentBehavior *attachment = self.childBehaviors[i];
        attachment.anchorPoint = [pointValue CGPointValue];
    }
}

- (NSArray <NSValue *> *)attachmentPointValuesForPoint:(CGPoint)point
{
    CGFloat width = 40.0f;
    CGFloat height = 40.0f;

    CGPoint topLeft = CGPointMake(point.x - width * 0.5, point.y - height * 0.5);
    CGPoint topRight = CGPointMake(point.x + width * 0.5, point.y - height * 0.5);
    CGPoint bottomLeft = CGPointMake(point.x - width * 0.5, point.y + height * 0.5);
    CGPoint bottomRight = CGPointMake(point.x + width * 0.5, point.y + height * 0.5);

    NSArray <NSValue *> *pointValues = @[ [NSValue valueWithCGPoint:topLeft], [NSValue
valueWithCGPoint:topRight], [NSValue valueWithCGPoint:bottomLeft], [NSValue
valueWithCGPoint:bottomRight]];
    return pointValues;
}

@end

```

A continuación, podemos crear el comportamiento de alto nivel que combinará una cantidad de `RectangleAttachmentBehavior`.

Rápido

```

final class DragBehavior: UIDynamicBehavior
{
    init(items: [UIDynamicItem], point: CGPoint)
    {
        super.init()
        items.forEach
        {
            let rectAttachment = RectangleAttachmentBehavior(item: $0, point: point)

```

```

        self.addChildBehavior(rectAttachment)
    }
}

func updateDragLocation(with point: CGPoint)
{
    // Tell low-level behaviors location has changed
    self.childBehaviors.flatMap { $0 as? RectangleAttachmentBehavior }.forEach {
        $0.updateAttachmentLocation(with: point) }
    }
}

```

C objetivo

```

@implementation DragBehavior

- (instancetype)initWithItems:(NSArray <id<UIDynamicItem>> *)items point: (CGPoint)point
{
    self = [super init];
    if (self)
    {
        for (id<UIDynamicItem> item in items)
        {
            RectangleAttachmentBehavior *rectAttachment = [[RectangleAttachmentBehavior
                alloc] initWithItem:item point:point];
            [self addChildBehavior:rectAttachment];
        }
    }
    return self;
}

- (void)updateDragLocationWithPoint:(CGPoint)point
{
    for (RectangleAttachmentBehavior *rectAttachment in self.childBehaviors)
    {
        [rectAttachment updateAttachmentLocationWithPoint:point];
    }
}

@end

```

Ahora con nuestros comportamientos en su lugar, el siguiente paso es agregarlos a nuestra vista de colección cuando. Como normalmente queremos un diseño de cuadrícula estándar, podemos subclase `UICollectionViewFlowLayout` y solo cambiar los atributos al arrastrar. Lo hacemos principalmente mediante la anulación de `layoutAttributesForElementsInRect` y el uso de los `UIDynamicAnimator`'s método de conveniencia `itemsInRect`.

Rápido

```

final class DraggableLayout: UICollectionViewFlowLayout
{
    // Array that holds dragged index paths
    var indexPathsForDraggingElements: [IndexPath]?

    // The dynamic animator that will animate drag behavior

```

```

var animator: UIDynamicAnimator?

// Custom high-level behavior that dictates drag animation
var dragBehavior: DragBehavior?

// Where dragging starts so can return there once dragging ends
var startDragPoint = CGPoint.zero

// Bool to keep track if dragging has ended
var isFinishedDragging = false

// Method to inform layout that dragging has started
func startDragging(indexPaths selectedIndexPaths: [IndexPath], from point: CGPoint)
{
    indexPathsForDraggingElements = selectedIndexPaths
    animator = UIDynamicAnimator(collectionViewLayout: self)
    animator?.delegate = self

    // Get all of the draggable attributes but change zIndex so above other cells
    let draggableAttributes: [UICollectionViewLayoutAttributes] =
selectedIndexPaths.flatMap {
        let attribute = super.layoutAttributesForItem(at: $0)
        attribute?.zIndex = 1
        return attribute
    }

    startDragPoint = point

    // Add them to high-level behavior
    dragBehavior = DragBehavior(items: draggableAttributes, point: point)

    // Add high-level behavior to animator
    animator?.addBehavior(dragBehavior!)
}

func updateDragLocation(_ point: CGPoint)
{
    // Tell high-level behavior that point has updated
    dragBehavior?.updateDragLocation(with: point)
}

func endDragging()
{
    isFinishedDragging = true

    // Return high-level behavior to starting point
    dragBehavior?.updateDragLocation(with: startDragPoint)
}

func clearDraggedIndexPaths()
{
    // Reset state for next drag event
    animator = nil
    indexPathsForDraggingElements = nil
    isFinishedDragging = false
}

override func layoutAttributesForElements(in rect: CGRect) ->
[UICollectionViewLayoutAttributes]?
{

```

```

        let existingAttributes: [UICollectionViewLayoutAttributes] =
super.layoutAttributesForElements(in: rect) ?? []
        var allAttributes = [UICollectionViewLayoutAttributes]()

        // Get normal flow layout attributes for non-drag items
        for attributes in existingAttributes
        {
            if (indexPathsForDraggingElements?.contains(attributes.indexPath) ?? false) ==
false
                {
                    allAttributes.append(attributes)
                }
        }

        // Add dragged item attributes by asking animator for them
        if let animator = self.animator
        {
            let animatorAttributes: [UICollectionViewLayoutAttributes] = animator.items(in:
rect).flatMap { $0 as? UICollectionViewLayoutAttributes }
            allAttributes.append(contentsOf: animatorAttributes)
        }
        return allAttributes
    }
}
extension DraggableLayout: UIDynamicAnimatorDelegate
{
    func dynamicAnimatorDidPause(_ animator: UIDynamicAnimator)
    {
        // Animator has paused and done dragging; reset state
        guard isFinishedDragging else { return }
        clearDraggedIndexPaths()
    }
}
}

```

C objetivo

```

@interface DraggableLayout () <UIDynamicAnimatorDelegate>
@property (nonatomic, strong) NSArray <NSIndexPath *> *indexPathsForDraggingElements;
@property (nonatomic, strong) UIDynamicAnimator *animator;
@property (nonatomic, assign) CGPoint startDragPoint;
@property (nonatomic, assign) BOOL finishedDragging;
@property (nonatomic, strong) DragBehavior *dragBehavior;
@end

@implementation DraggableLayout

- (void)startDraggingWithIndexPaths:(NSArray <NSIndexPath *> *)selectedIndexPaths
fromPoint:(CGPoint)point
{
    self.indexPathsForDraggingElements = selectedIndexPaths;
    self.animator = [[UIDynamicAnimator alloc] initWithCollectionViewLayout:self];
    self.animator.delegate = self;
    NSMutableArray *draggableAttributes = [[NSMutableArray
alloc] initWithCapacity:selectedIndexPaths.count];
    for (NSIndexPath *indexPath in selectedIndexPaths)
    {
        UICollectionViewLayoutAttributes *attributes = [super
layoutAttributesForItemAtIndexPath:indexPath];
        attributes.zIndex = 1;
    }
}

```

```

        [draggableAttributes addObject:attributes];
    }
    self.startDragPoint = point;
    self.dragBehavior = [[DragBehavior alloc] initWithItems:draggableAttributes point:point];
    [self.animator addBehavior:self.dragBehavior];
}

- (void)updateDragLoactionWithPoint:(CGPoint)point
{
    [self.dragBehavior updateDragLocationWithPoint:point];
}

- (void)endDragging
{
    self.finishedDragging = YES;
    [self.dragBehavior updateDragLocationWithPoint:self.startDragPoint];
}

- (void)clearDraggedIndexPath
{
    self.animator = nil;
    self.indexPathsForDraggingElements = nil;
    self.finishedDragging = NO;
}

- (void)dynamicAnimatorDidPause:(UIDynamicAnimator *)animator
{
    if (self.finishedDragging)
    {
        [self clearDraggedIndexPath];
    }
}

- (NSArray<UICollectionViewLayoutAttributes *>
*)layoutAttributesForElementsInRect:(CGRect)rect
{
    NSArray *existingAttributes = [super layoutAttributesForElementsInRect:rect];
    NSMutableArray *allAttributes = [[NSMutableArray
alloc] initWithCapacity:existingAttributes.count];
    for (UICollectionViewLayoutAttributes *attributes in existingAttributes)
    {
        if (![self.indexPathsForDraggingElements containsObject:attributes.indexPath])
        {
            [allAttributes addObject:attributes];
        }
    }
    [allAttributes addObjectsFromArray:[self.animator itemsInRect:rect]];
    return allAttributes;
}

@end

```

Finalmente, crearemos un controlador de vista que creará nuestra vista `UICollectionView` y manejará nuestro gesto de pulsación prolongada.

Rápido

```
final class ViewController: UIViewController
```

```

{
    // Collection view that displays cells
    lazy var collectionView: UICollectionView =
    {
        let collectionView = UICollectionView(frame: .zero, collectionViewLayout:
DraggableLayout())
        collectionView.backgroundColor = .white
        collectionView.translatesAutoresizingMaskIntoConstraints = false
        self.view.addSubview(collectionView)
        collectionView.topAnchor.constraint(equalTo:
self.topAnchor).isActive = true
        collectionView.leadingAnchor.constraint(equalTo: self.view.leadingAnchor).isActive =
true
        collectionView.trailingAnchor.constraint(equalTo: self.view.trailingAnchor).isActive =
true
        collectionView.bottomAnchor.constraint(equalTo:
self.bottomAnchor).isActive = true

        return collectionView
    }()

    // Gesture that drives dragging
    lazy var longPress: UILongPressGestureRecognizer =
    {
        let longPress = UILongPressGestureRecognizer(target: self, action:
#selector(self.handleLongPress(sender:)))
        return longPress
    }()

    // Array that holds selected index paths
    var selectedIndexPaths = [IndexPath]()

    override func viewDidLoad()
    {
        super.viewDidLoad()
        collectionView.delegate = self
        collectionView.dataSource = self
        collectionView.register(UICollectionViewCell.self, forCellWithReuseIdentifier: "Cell")
        collectionView.addGestureRecognizer(longPress)
    }

    func handleLongPress(sender: UILongPressGestureRecognizer)
    {
        guard let draggableLayout = collectionView.collectionViewLayout as? DraggableLayout
else { return }
        let location = sender.location(in: collectionView)
        switch sender.state
        {
        case .began:
            draggableLayout.startDragging(indexPaths: selectedIndexPaths, from: location)
        case .changed:
            draggableLayout.updateDragLocation(location)
        case .ended, .failed, .cancelled:
            draggableLayout.endDragging()
        case .possible:
            break
        }
    }
}
extension ViewController: UICollectionViewDelegate, UICollectionViewDataSource
{

```

```

func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section:
Int) -> Int
{
    return 1000
}

func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath:
IndexPath) -> UICollectionViewCell
{
    let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "Cell", for:
indexPath)
    cell.backgroundColor = .gray
    if selectedIndexPaths.contains(indexPath) == true
    {
        cell.backgroundColor = .red
    }
    return cell
}

func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath:
IndexPath)
{
    // Bool that determines if cell is being selected or unselected
    let isSelected = !selectedIndexPaths.contains(indexPath)
    let cell = collectionView.cellForItem(at: indexPath)
    cell?.backgroundColor = isSelected ? .red : .gray
    if isSelected
    {
        selectedIndexPaths.append(indexPath)
    }
    else
    {
        selectedIndexPaths.remove(at: selectedIndexPaths.index(of: indexPath)!)
    }
}
}

```

C objetivo

```

@interface ViewController () <UICollectionViewDelegate, UICollectionViewDataSource>
@property (nonatomic, strong) UICollectionView *collectionView;
@property (nonatomic, strong) UILongPressGestureRecognizer *longPress;
@property (nonatomic, strong) NSMutableArray <NSIndexPath *> *selectedIndexPaths;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    self.collectionView.delegate = self;
    self.collectionView.dataSource = self;
    [self.collectionView registerClass:[UICollectionViewCell class]
forCellWithReuseIdentifier:@"Cell"];
    [self.collectionView addGestureRecognizer:self.longPress];
    self.selectedIndexPaths = [[NSMutableArray alloc] init];
}

- (UICollectionView *)collectionView

```



```

{
    if (!_collectionView)
    {
        _collectionView = [[UICollectionView alloc] initWithFrame:CGRectZero
collectionViewLayout:[[DraggableLayout alloc]init]];
        _collectionView.backgroundColor = [UIColor whiteColor];
        _collectionView.translatesAutoresizingMaskIntoConstraints = NO;
        [self.view addSubview:_collectionView];
        [_collectionView.topAnchor
constraintEqualToAnchor:self.topLayoutGuide.bottomAnchor].active = YES;
        [_collectionView.leadingAnchor constraintEqualToAnchor:self.view.leadingAnchor].active
= YES;
        [_collectionView.trailingAnchor
constraintEqualToAnchor:self.view.trailingAnchor].active = YES;
        [_collectionView.bottomAnchor
constraintEqualToAnchor:self.bottomLayoutGuide.topAnchor].active = YES;
    }
    return _collectionView;
}

- (UILongPressGestureRecognizer *)longPress
{
    if (!_longPress)
    {
        _longPress = [[UILongPressGestureRecognizer alloc] initWithTarget:self
action:@selector(handleLongPress:)];
    }
    return _longPress;
}

- (void)handleLongPress:(UILongPressGestureRecognizer *)sender
{
    DraggableLayout *draggableLayout = (DraggableLayout
*)self.collectionView.collectionViewLayout;
    CGPoint location = [sender locationInView:self.collectionView];
    if (sender.state == UIGestureRecognizerStateBegan)
    {
        [draggableLayout startDraggingWithIndexPaths:self.selectedIndexPaths
fromPoint:location];
    }
    else if (sender.state == UIGestureRecognizerStateChanged)
    {
        [draggableLayout updateDragLoactionWithPoint:location];
    }
    else if (sender.state == UIGestureRecognizerStateEnded || sender.state ==
UIGestureRecognizerStateCancelled || sender.state == UIGestureRecognizerStateFailed)
    {
        [draggableLayout endDragging];
    }
}

- (NSInteger)collectionView:(UICollectionView *)collectionView
numberOfItemsInSection:(NSInteger)section
{
    return 1000;
}

- (UICollectionViewCell *)collectionView:(UICollectionView *)collectionView
cellForItemAtIndexPath:(NSIndexPath *)indexPath
{
    UICollectionViewCell *cell = [collectionView

```

```

dequeueReusableCellWithIdentifier:@"Cell" forIndexPath:indexPath];
cell.backgroundColor = [UIColor grayColor];
if ([self.selectedIndexPaths containsObject:indexPath])
{
    cell.backgroundColor = [UIColor redColor];
}
return cell;
}

- (void)collectionView:(UICollectionView *)collectionView
didSelectItemAtIndexPath:(NSIndexPath *)indexPath
{
    BOOL isSelected = ![self.selectedIndexPaths containsObject:indexPath];
    UICollectionViewCell *cell = [collectionView cellForItemAtIndexPath:indexPath];
    if (isSelected)
    {
        cell.backgroundColor = [UIColor redColor];
        [self.selectedIndexPaths addObject:indexPath];
    }
    else
    {
        cell.backgroundColor = [UIColor grayColor];
        [self.selectedIndexPaths removeObject:indexPath];
    }
}

@end

```

Para obtener más información, [sesión de la WWDC 2013 "Técnicas avanzadas con dinámica de UIKit"](#)

Lea [UIKit Dynamics con UICollectionView en línea](#): <https://riptutorial.com/es/ios/topic/10079/uikit-dynamics-con-uicollectionview>

Capítulo 175: UILabel

Introducción

La clase UILabel implementa una vista de texto de solo lectura. Puede usar esta clase para dibujar una o varias líneas de texto estático, como las que podría usar para identificar otras partes de su interfaz de usuario. La clase UILabel base proporciona soporte para el estilo simple y complejo del texto de la etiqueta. También puede controlar aspectos de la apariencia, como si la etiqueta usa una sombra o dibuja con un resaltado. Si es necesario, puede personalizar aún más la apariencia de su texto por subclasificación.

Sintaxis

- UILabel.numberOfLines: Int // obtiene o establece el número máximo de líneas que puede tener la etiqueta. 0 es ilimitado
- UILabel.text: String? // obtener o establecer el texto que muestra la etiqueta
- UILabel.textColor: UIColor! // obtener o establecer el color del texto en la etiqueta
- UILabel.tintColor: UIColor! // obtener o establecer el color del tinte de la etiqueta
- UILabel.attributedString: NSAttributedString? // obtener o establecer el texto atribuido de la etiqueta
- UILabel.font: UIFont! // obtener o establecer la fuente del texto en la etiqueta
- UILabel.textAlignment: NSTextAlignment // obtiene o establece la alineación del texto

Observaciones

UILabels son vistas que pueden usarse para mostrar una o varias líneas de texto. Contiene múltiples formas de estilizar el texto, como sombras, colores de texto y fuentes.

UILabels también puede mostrar Cadenas Atribuidas, que es texto + marcado en línea para aplicar estilos a partes del texto.

UILabel no cumple con el protocolo UICollectionAppearance, por lo que no puede usar los métodos de proxy UICollectionAppearance para personalizar la apariencia de UILabels. Vea esta [discusión](#) para más.

Referencia de desarrollador de Apple [aquí](#)

Examples

Cambio de texto en una etiqueta existente

Se puede cambiar el texto de una UILabel existente accediendo y modificando la propiedad de `text` de la UILabel. Esto se puede hacer directamente usando literales de `String` o indirectamente usando variables.

Configuración del texto con literales de `String`

Rápido

```
label.text = "the new text"
```

C objetivo

```
// Dot Notation
label.text = @"the new text";

// Message Pattern
[label setText:@"the new text"];
```

Configurando el texto con una variable

Rápido

```
let stringVar = "basic String var"
label.text = stringVar
```

C objetivo

```
NSString * stringVar = @"basic String var";

// Dot Notation
label.text = stringVar;

// Message Pattern
[label setText: stringVar];
```

Color de texto

Puede utilizar la propiedad `textColor` la etiqueta para aplicar un color de texto a todo el texto de la etiqueta.

Rápido

```
label.textColor = UIColor.redColor()
label.textColor = UIColor(red: 64.0/255.0, green: 88.0/255.0, blue: 41.0/225.0, alpha: 1)
```

Swift 3

```
label.textColor = UIColor.red
label.textColor = UIColor(red: 64.0/255.0, green: 88.0/255.0, blue: 41.0/225.0, alpha: 1)
```

C objetivo

```
label.textColor = [UIColor redColor];
label.textColor = [UIColor colorWithRed:64.0f/255.0f green:88.0f/255.0f blue:41.0f/255.0f
alpha:1.0f];
```

Aplicando color de texto a una porción del texto.

También puede variar el color del texto (u otros atributos) de partes del texto usando

[NSAttributedString](#) :

C objetivo

```
attributedString = [[NSMutableAttributedString alloc] initWithString:@"The grass is green; the
sky is blue."];
[attributedString addAttribute:NSForegroundColorAttributeName value:[UIColor greenColor]
range:NSMakeRange(13, 5)];
[attributedString addAttribute:NSForegroundColorAttributeName value:[UIColor blueColor]
range:NSMakeRange(31, 4)];
label.attributedString = attributedString;
```

Rápido

```
let attributedString = NSMutableAttributedString(string: "The grass is green; the sky is
blue.")
attributedString.addAttribute(NSForegroundColorAttributeName, value: UIColor.green(), range:
NSRange(location: 13, length: 5))
attributedString.addAttribute(NSForegroundColorAttributeName, value: UIColor.blue(), range:
NSRange(location: 31, length: 4))
label.attributedString = attributedString
```

Alineación del texto

Rápido

```
label.textAlignment = NSTextAlignment.left
//or the shorter
label.textAlignment = .left
```

Cualquier valor en el [NSTextAlignment](#) enumeración es válido: `.left` , `.center` , `.right` , `.justified` , `.natural`

C objetivo

```
label.textAlignment = NSTextAlignmentLeft;
```

Cualquier valor en la enumeración [NSTextAlignment](#) es válido: `NSTextAlignmentLeft` , `NSTextAlignmentCenter` , `NSTextAlignmentRight` , `NSTextAlignmentJustified` , `NSTextAlignmentNatural`

La alineación vertical en `UILabel` no se admite fuera de la caja: [alinear verticalmente el texto con la](#)

parte superior dentro de una UILabel

Crea una UILabel

Con un marco

Cuando conoce las dimensiones exactas que desea establecer para su etiqueta, puede inicializar un UILabel con un marco CGRect .

Rápido

```
let frame = CGRect(x: 0, y: 0, width: 200, height: 21)
let label = UILabel(frame: frame)
view.addSubview(label)
```

C objetivo

```
CGRect frame = CGRectMake(0, 0, 200, 21);
UILabel *label = [[UILabel alloc] initWithFrame:frame];
[view addSubview:label];
```

Con diseño automático

Puede agregar restricciones en una UILabel cuando desee que iOS calcule dinámicamente su marco en tiempo de ejecución.

Rápido

```
let label = UILabel()
label.backgroundColor = .red
label.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(label)

NSLayoutConstraint.activate([
    //stick the top of the label to the top of its superview:
    label.topAnchor.constraint(equalTo: view.topAnchor)

    //stick the left of the label to the left of its superview
    //if the alphabet is left-to-right, or to the right of its
    //superview if the alphabet is right-to-left:
    label.leadingAnchor.constraint(equalTo: view.leadingAnchor)

    //stick the label's bottom to the bottom of its superview:
    label.bottomAnchor.constraint(equalTo: view.bottomAnchor)

    //the label's width should be equal to 100 points:
    label.widthAnchor.constraint(equalToConstant: 100)
```

```
] )
```

C objetivo

```
UILabel *label = [[UILabel alloc] init];
```

Con Objective-c + Visual Format Language (VFL)

```
UILabel *label = [UILabel new];
label.translatesAutoresizingMaskIntoConstraints = NO;
[self.view addSubview label];
// add horizontal constraints with 5 left and right padding from the leading and trailing

[self.view addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"V:|-5-
[labelName]-5-|"
                                                                    options:0
                                                                    metrics:nil

views:@{@"labelName":label}}];
// vertical constraints that will use the height of the superView with no padding on top and
bottom
[self.view addConstraints:[NSLayoutConstraint constraintsWithVisualFormat:@"H:|[labelName]|"
                                                                    options:0
                                                                    metrics:nil

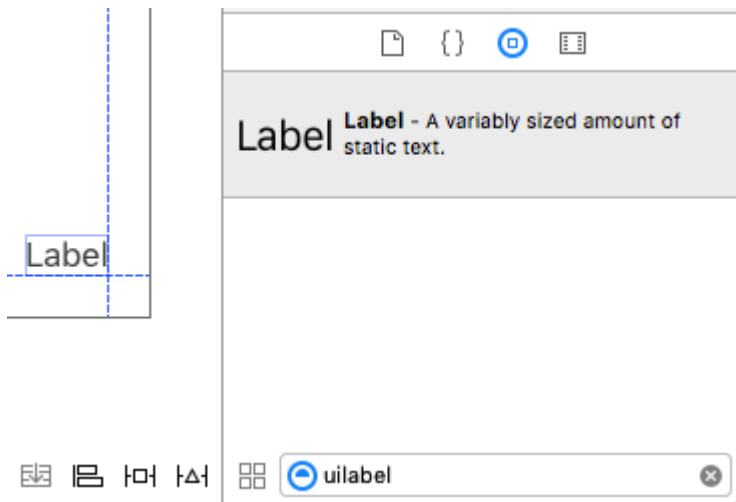
views:@{@"labelName":label}}];
```

La documentación de VFL se puede encontrar [aquí](#)

Una vez creada la etiqueta, asegúrese de establecer las dimensiones a través del diseño automático. Xcode mostrará errores si se hace incorrectamente.

Con Interface Builder

También utiliza Interface Builder para agregar una `UILabel` a su archivo `Storyboard O .xib` arrastrando una `Label` desde el panel Biblioteca de objetos y soltándola en una vista en el lienzo:

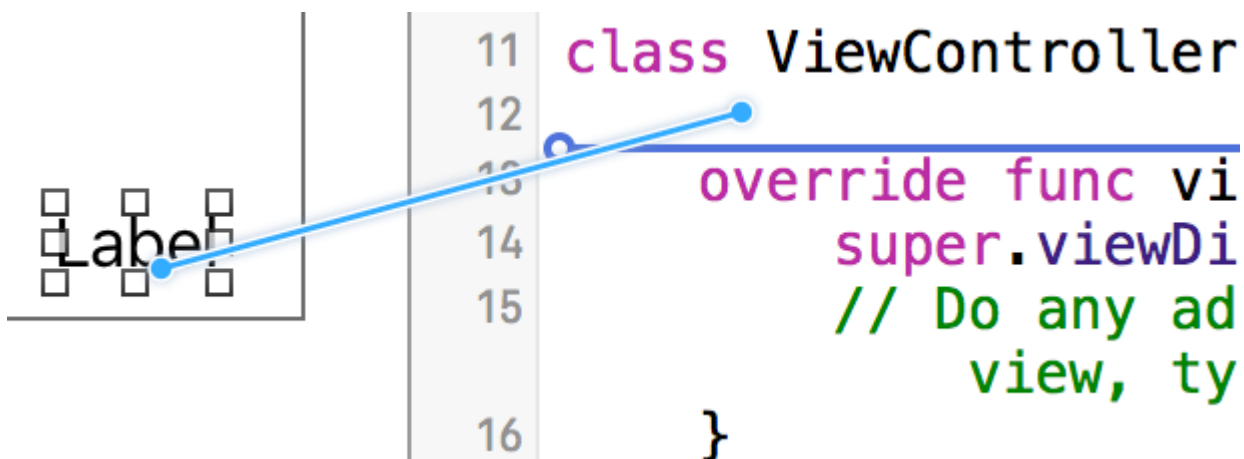


En lugar de especificar un marco (posición y tamaño) para un UILabel programación, un Storyboard o un .xib permite usar [Diseño automático](#) para agregar restricciones al control.

Para acceder a esta etiqueta creada desde el storyboard o xib cree un IBOutlet de esta etiqueta.

Enlace entre Interface Builder y View Controller

Una vez que haya agregado una UILabel a su Storyboard o .xib el archivo, puede vincularlo a su código presionando `Control ^` y luego arrastrando el mouse entre la UILabel a su ViewController, o puede arrastrar el código mientras hace clic derecho para tienen el mismo efecto



En el cuadro de diálogo de propiedades, puede establecer el nombre de UILabel y establecerlo como `strong` o `weak`. Para más información sobre `strong` y `weak`, vea [esto](#),

La otra forma es hacer la salida programáticamente de la siguiente manera:

Rápido

```
@IBOutlet weak var nameLabel : UILabel!
```


C objetivo

```
@property (nonatomic, weak) IBOutlet UILabel *nameLabel;
```

Establecer fuente

Rápido

```
let label = UILabel()
```

C objetivo

```
UILabel *label = [[UILabel alloc] init];  
or  
UILabel *label = [UILabel new]; // convenience method for calling alloc-init
```

Cambiar el tamaño de la fuente por defecto

Rápido

```
label.font = UIFont.systemFontSize(17)
```

Swift 3

```
label.font = UIFont.systemFont(ofSize: 17)
```

C objetivo

```
label.font = [UIFont systemFontOfSize:17];
```

Use un peso de fuente específico

iOS 8.2

Rápido

```
label.font = UIFont.systemFontSize(17, weight: UIFontWeightBold)
```

Swift3

```
label.font = UIFont.systemFont(ofSize: 17, weight: UIFontWeightBold)
```

C objetivo

```
label.font = [UIFont systemFontOfSize:17 weight:UIFontWeightBold];
```

iOS 8.2

Rápido

```
label.font = UIFont.boldSystemFont(ofSize: 17)
```

Swift3

```
label.font = UIFont.boldSystemFont(ofSize: 17)
```

C objetivo

```
label.font = [UIFont boldSystemFontOfSize:17];
```

Utilice un estilo de texto de tipo dinámico.

La fuente y el tamaño en puntos se basarán en el tamaño de lectura preferido del usuario.

Rápido

```
label.font = UIFont.preferredFontForTextStyle(UIFontTextStyleBody)
```

Swift 3

```
label.font = UIFont.preferredFont(forTextStyle: .body)
```

C objetivo

```
label.font = [UIFont preferredFontForTextStyle:UIFontTextStyleBody];
```

Use una fuente diferente por completo

Rápido

```
label.font = UIFont(name: "Avenir", size: 15)
```

C objetivo

```
label.font = [UIFont fontWithName:@"Avenir" size:15];
```

Anular tamaño de fuente

Una forma de establecer el tamaño de fuente sin saber la familia de fuentes es utilizar la propiedad de **fuente** de `UILabel`.

Rápido

```
label.font = label.font.fontWithSize(15)
```

Swift 3

```
label.font = label.font.withSize(15)
```

C objetivo

```
label.font = [label.font fontWithSize:15];
```

Usar fuente personalizada Swift

[Consulte este enlace](#)

Número de líneas

Cuando haga una etiqueta y configure su texto para que sea más de una línea que pueda mostrar, se truncará y verá solo una línea de texto que termina con tres puntos (...). Esto se debe a que una propiedad llamada `numberOfLines` se establece en 1 y, por lo tanto, solo se mostrará una línea. Es un error común en el manejo de `UILabel`s, y muchas personas lo consideran como un error, o pueden usar más de una etiqueta para mostrar más de una línea de texto, pero solo editando esta propiedad, podemos decirle a `UILabel` que Acepta hasta el número especificado de

líneas. Por ejemplo, si esta propiedad se establece en 5, la etiqueta puede mostrar 1, 2, 3, 4 o 5 líneas de datos.

Configurando el valor programáticamente

Para establecer esta propiedad, simplemente asigne un nuevo número entero:

Rápido

```
label.numberOfLines = 2
```

C objetivo

```
label.numberOfLines = 2;
```

Nota

Es posible establecer esta propiedad en 0. Sin embargo, esto no significa que no acepte ninguna línea, sino que la etiqueta puede tener tantas líneas como sea necesario (también conocido como "Infinito"):

Rápido

```
label.numberOfLines = 0
```

C objetivo

```
label.numberOfLines = 0;
```

Nota

Si la etiqueta tiene una restricción de altura, la restricción se respetará. En este caso, `label.numberOfLines = 0` puede no funcionar como se esperaba.

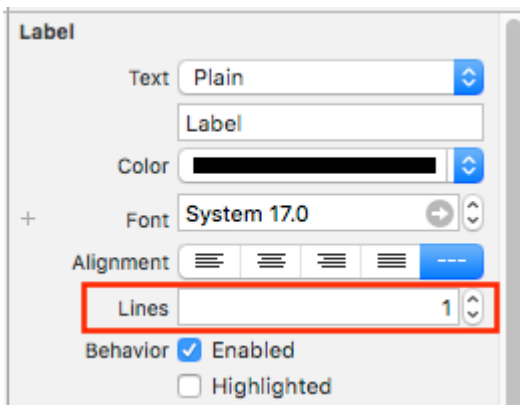
Nota

Para un texto de líneas múltiples más complejo, [UITextView](#) puede ser un mejor ajuste. *

Estableciendo el valor en el Interface Builder

En lugar de configurar `numberOfLines` programación, puede usar un `Storyboard` o un `.xib` y establecer la propiedad `numberOfLines`. De esa manera, logramos los mismos resultados que el código anterior.

Al igual que a continuación:



Tamaño para caber

Supongamos que tiene una `UILabel` en su `storyboard` y que ha creado un `IBOutlet` para él en `ViewController.swift / ViewController.m` y lo ha llamado `labelOne`.

Para hacer que los cambios sean fácilmente visibles, cambie el `backgroundColor` y el `textColor` de `labelOne` en el método `viewDidLoad`:

La función `sizeToFit` se utiliza cuando desea cambiar automáticamente el tamaño de una etiqueta en función del contenido almacenado en ella.

Rápido

```
labelOne.backgroundColor = UIColor.blueColor()
labelOne.textColor = UIColor.whiteColor()
labelOne.text = "Hello, World!"
labelOne.sizeToFit()
```

Swift 3

```
labelOne.backgroundColor = UIColor.blue
labelOne.textColor = UIColor.white
labelOne.text = "Hello, World!"
labelOne.sizeToFit()
```

C objetivo

```
labelOne.backgroundColor = [UIColor blueColor];
labelOne.textColor = [UIColor whiteColor];
labelOne.text = @"Hello, World!";
```

```
[labelOne sizeToFit];
```

La salida para el código anterior es:



Como puede ver, no hay cambios, ya que el texto se ajusta perfectamente a labelOne. `sizeToFit` solo cambia el marco de la etiqueta.

Cambemos el texto a uno un poco más largo:

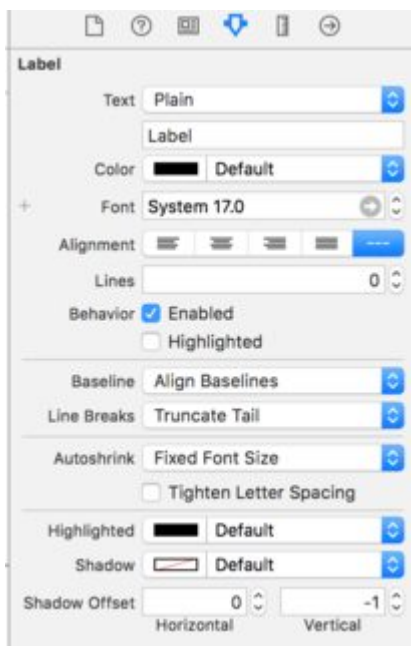
```
labelOne.text = "Hello, World! I'm glad to be alive!"
```

Ahora, labelOne se ve así:



Incluso llamando a `sizeToFit` no cambia nada. Esto se debe a que, de forma predeterminada, el número de líneas mostradas por UILabel se establece en 1. Vamos a cambiarlo a cero en el guión

gráfico:



Esta vez, cuando ejecutamos la aplicación, labelOne aparece correctamente:



La propiedad `numberOfLines` también se puede cambiar en el archivo `ViewController` :

```
// Objective-C
labelOne.numberOfLines = 0;

// Swift
labelOne.numberOfLines = 0
```

Color de fondo

Rápido

```
label.backgroundColor = UIColor.redColor()

label.backgroundColor = .redColor()
```

Swift 3

```
label.backgroundColor = UIColor.red
```

C objetivo

```
label.backgroundColor = [UIColor redColor];
```

Añadir sombras al texto

Rápido

```
label1.layer.shadowOffset = CGSize(width: 3, height: 3)
label1.layer.shadowOpacity = 0.7
label1.layer.shadowRadius = 2
```

Swift 3

```
label1.layer.shadowOffset = CGSize(width: 3, height: 3)
label1.layer.shadowOpacity = 0.7
label1.layer.shadowRadius = 2
```

C objetivo

```
label1.layer.shadowOffset = CGSizeMake(3, 3);
label1.layer.shadowOpacity = 0.7;
label1.layer.shadowRadius = 2;
```

I Like My Cat

Altura variable utilizando restricciones.

Puedes hacer una `UILabel` con una altura dinámica usando el diseño automático.

`numberOfLines` establecer `numberOfLines` en cero (0) y agregar una altura mínima configurando una restricción con una relación de tipo `.GreaterThanOrEqualTo` en el atributo `.Height`

ios 6

Rápido


```
label.numberOfLines = 0

let heightConstraint = NSLayoutConstraint(
    item: label,
    attribute: .Height,
    relatedBy: .GreaterThanOrEqual,
    toItem: nil,
    attribute: .NotAnAttribute,
    multiplier: 0,
    constant: 20
)

label.addConstraint(heightConstraint)
```

iOS 9

Rápido

```
label.numberOfLines = 0
label.translatesAutoresizingMaskIntoConstraints = false
label.heightAnchor.constraintGreaterThanOrEqualToConstant(20).active = true
```

LineBreakMode

Usando código

```
UILabel.lineBreakMode: NSLineBreakMode
```

Rápido

```
label.lineBreakMode = .ByTruncatingTail
```

- .ByWordWrapping
- .ByCharWrapping
- .ByClipping
- .ByTruncatingHead
- .ByTruncatingTail
- .ByTruncatingMiddle

Swift 3

```
label.lineBreakMode = .byTruncatingTail
```

- .byWordWrapping
- .byCharWrapping
- .byClipping

- .byTruncatingHead
- .byTruncatingTail
- .byTruncatingMiddle

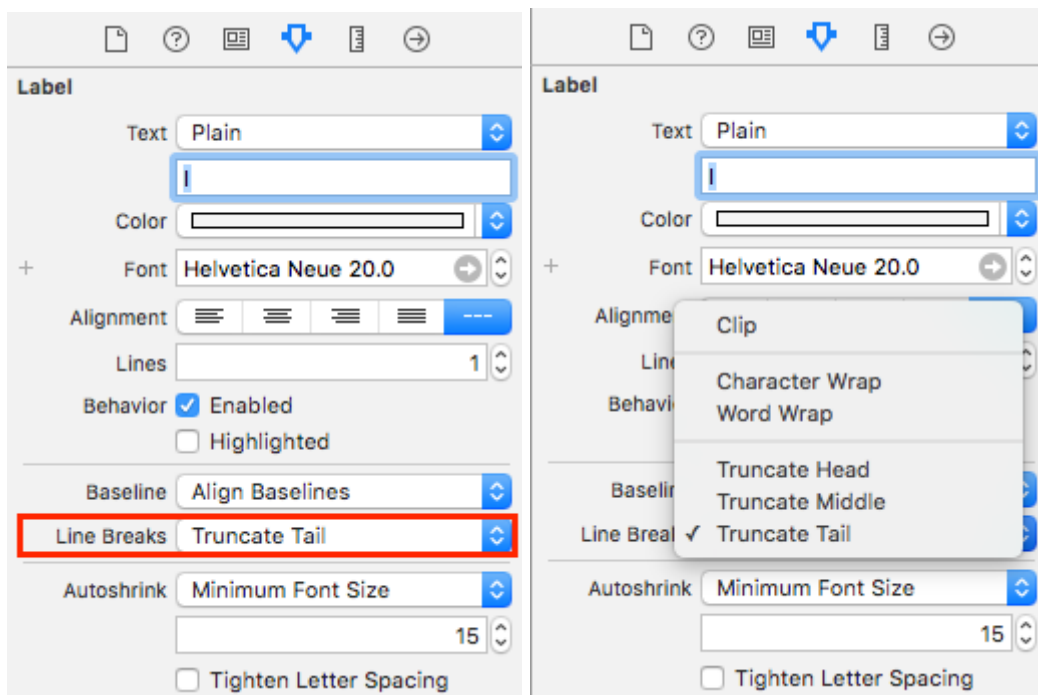
C objetivo

```
[label setLineBreakMode:NSLineBreakByTruncatingTail];
```

- NSLineBreakByWordWrapping
- NSLineBreakByCharWrapping
- NSLineBreakByClipping
- NSLineBreakByTruncatingHead
- NSLineBreakByTruncatingTail
- NSLineBreakByTruncatingMiddle

Usando el gui3n gr3fico

Esto tambi3n se puede configurar en el inspector de atributos de un UILabel:



Constantes

- Ajuste de palabra: el ajuste se produce en los l3mites de la palabra, a menos que la palabra en s3 no encaje en una sola l3nea
- Char Wrapping: la envoltura se produce antes del primer car3cter que no encaja
- Recorte: las l3neas simplemente no se dibujan m3s all3 del borde del contenedor de texto
- Cabeza truncada: la l3nea se muestra de manera que el extremo se ajusta al contenedor y el texto que falta al principio de la l3nea se indica con un glifo de elipsis

- **Truncating Tail:** la línea se muestra de manera que el comienzo se ajusta al contenedor y el texto que falta al final de la línea se indica con un glifo de elipsis
- **Medio truncado:** la línea se muestra de manera que el inicio y el final se ajustan en el contenedor y el texto que falta en el medio se indica con un glifo de elipsis

Calcular límites de contenido (por ejemplo, alturas de celda dinámicas)

Un caso de uso común para querer calcular el marco que ocupará una etiqueta es dimensionar adecuadamente las celdas de la vista de tabla. La forma recomendada de hacer esto es usar el método `NSString boundingRectWithSize:options:attributes:context:`

`options` toma `options` dibujo de cadena:

- `NSStringDrawingUsesLineFragmentOrigin` debe usarse para etiquetas con múltiples líneas
- `NSStringDrawingTruncatesLastVisibleLine` debe agregarse utilizando el `|` operador si hay un número máximo de líneas

`attributes` es un `NSDictionary` de atributos que afectan a las cadenas atribuidas (lista completa: [Apple Docs](#)) pero los factores que afectan a la altura incluyen:

- **NSFontAttributeName** : muy importante, el tamaño y la familia de fuentes es una parte crítica del tamaño mostrado de la etiqueta.
- **NSParagraphStyleAttributeName** : para personalizar cómo se muestra el texto. Esto incluye interlineado, alineación de texto, estilo de truncamiento y algunas otras opciones. Si no cambió explícitamente ninguno de estos valores, no debería preocuparse por esto, pero puede ser importante si ha cambiado algunos valores en IB.

`context` debería ser `nil` ya que el caso de uso principal de `NSStringDrawingContext` es permitir que la fuente `NSStringDrawingContext` de tamaño para que se ajuste a un `rect` especificado, lo que no debería ser el caso si calculamos una altura dinámica.

C objetivo

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView cellForRowAtIndexPath:indexPath];

    NSString *labelContent = cell.textLabel.text;
    // you may choose to get the content directly from the data source if you have done
    minimal customizations to the font or are comfortable with hardcoding a few values
    // NSString *labelContent = [self.dataSource objectAtIndex:indexPath:indexPath];

    // value may be hardcoded if retrieved from data source
    UIFont *labelFont = [cell.textLabel font];

    // The NSParagraphStyle, even if you did not code any changes these values may have been
    altered in IB
    NSMutableParagraphStyle *paragraphStyle = [NSMutableParagraphStyle new];
    paragraphStyle.lineBreakMode = NSLineBreakByWordWrapping;
    paragraphStyle.alignment = NSTextAlignmentCenter;
}
```

```

NSDictionary *attributes = @{@"NSFontAttributeName: labelFont,
                             NSParagraphStyleAttributeName: paragraphStyle};

// The width is also important to the height
CGFloat labelWidth = CGRectGetWidth(cell.theLabel.frame);
// If you have been hardcoding up to this point you will be able to get this value by
subtracting the padding on left and right from tableView.bounds.size.width
//    CGFloat labelWidth = CGRectGetWidth(tableView.frame) - 20.0f - 20.0f;

CGRect bodyBounds = [labelContent boundingRectWithSize:CGSizeMake(width, CGFLOAT_MAX)
options:NSStringDrawingUsesLineFragmentOrigin attributes:attributes context:nil];

return CGRectGetHeight(bodyBounds) + heightForObjectsOnTopOfLabel +
heightForObjectBelowLabel;
}

```

Swift 3

```

override func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) ->
CGFloat {
    var cell = tableView.cellForRow(atIndexPath: indexPath)!
    var labelContent = cell.theLabel.text
    var labelFont = cell.theLabel.font
    var paragraphStyle = NSMutableParagraphStyle()

    paragraphStyle.lineBreakMode = .byWordWrapping
    paragraphStyle.alignment = .center

    var attributes = [NSFontAttributeName: labelFont, NSParagraphStyleAttributeName:
paragraphStyle]

    var labelWidth: CGFloat = cell.theLabel.frame.width

    var bodyBounds = labelContent.boundingRect(withSize: CGSize(width: width, height:
CGFLOAT_MAX), options: .usesLineFragmentOrigin, attributes: attributes, context: nil)

    return bodyBounds.height + heightForObjectsOnTopOfLabel + heightForObjectBelowLabel
}

```

A la inversa, si tiene un número máximo de líneas establecido, primero deberá calcular la altura de una sola línea para asegurarse de que no obtengamos un valor más alto que el tamaño permitido:

```

// We calculate the height of a line by omitting the NSStringDrawingUsesLineFragmentOrigin
option, which will assume an infinitely wide label
CGRect singleLineRect = [labelContent boundingRectWithSize:CGSizeMake(CGFLOAT_MAX,
CGFLOAT_MAX)

options:NSStringDrawingTruncatesLastVisibleLine
context:nil];

CGFloat lineHeight = CGRectGetHeight(singleLineRect);
CGFloat maxHeight = lineHeight * cell.theLabel.numberOfLines;

// Now you can call the method appropriately
CGRect bodyBounds = [labelContent boundingRectWithSize:CGSizeMake(width, maxHeight)
options:(NSStringDrawingUsesLineFragmentOrigin|NSStringDrawingTruncatesLastVisibleLine)
attributes:attributes context:nil];

```

```
return CGRectGetHeight(bodyBounds) + heightOfObjectsOnTopOfLabel +
heightOfObjectBelowLabel;
```

Etiqueta cliqueable

NOTA: En la mayoría de los casos, es mejor usar un `UIButton` lugar de hacer un `UILabel` que puede tocar. Solo use este ejemplo, si está seguro, que no quiere usar un `UIButton` por alguna razón.

1. Crear etiqueta
2. Habilitar la interacción del usuario
3. Añadir `UITapGestureRecognizer`

La clave para crear una `UILabel` es habilitar la interacción del usuario.

Rápido

```
let label = UILabel()
label.userInteractionEnabled = true

let gesture = UITapGestureRecognizer(target: self, action: #selector(labelClicked(_:)))
label.addGestureRecognizer(gesture)
```

C objetivo

```
UILabel *label = [[UILabel alloc] init];
[label setUserInteractionEnabled:YES];

UITapGestureRecognizer* gesture = [[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(labelClicked:)];
[label addGestureRecognizer:gesture];
```

Configuración de "userInteractionEnabled" en el inspector de atributos del guión gráfico

En lugar de usar código, puede seleccionar la `UILabel` dentro del guión gráfico y marcar la opción:



Marco de etiqueta dinámico de longitud de texto desconocido

A veces tenemos que cambiar el tamaño de una UILabel según el contenido dinámico en el que se desconoce la longitud del texto. En este ejemplo, el ancho de la UILabel se fija en 280 puntos y la altura es infinita, digamos 9999. Estimación del marco con respecto al estilo de texto y `maximumLabelSize`.

C objetivo

```
UILabel * label = [[UILabel alloc] init];

NSString *message = @"Some dynamic text for label";

//set the text and style if any.
label.text = message;

label.numberOfLines = 0;

CGSize maximumLabelSize = CGSizeMake(280, 9999); //280:max width of label and 9999-max height
of label.

// use font information from the UILabel to calculate the size
CGSize expectedLabelSize = [label sizeThatFits:maximumLabelSize];

//Deprecated in iOS 7.0
//CGSize expectedLabelSize = [message sizeWithFont:label.font
constrainedToSize:maximumLabelSize lineBreakMode:NSLineBreakByWordWrapping];

// create a frame that is filled with the UILabel frame data
CGRect newFrame = label.frame;

// resizing the frame to calculated size
newFrame.size.height = expectedLabelSize.height;

// put calculated frame into UILabel frame
label.frame = newFrame;
```

Rápido

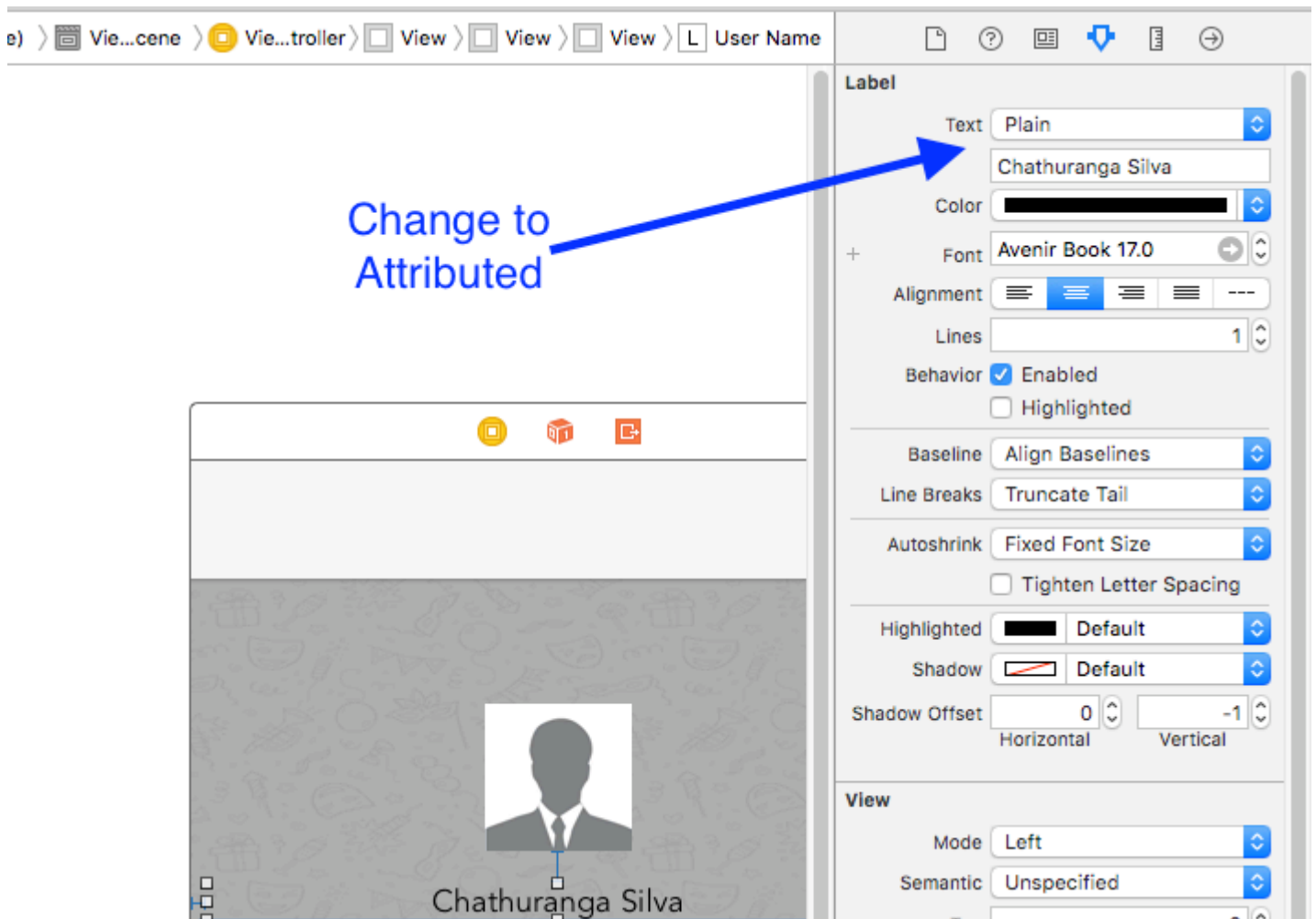
```
var message: String = "Some dynamic text for label"
//set the text and style if any.
label.text = message
label.numberOfLines = 0
var maximumLabelSize: CGSize = CGSize(width: 280, height: 9999)
var expectedLabelSize: CGSize = label.sizeThatFits(maximumLabelSize)
// create a frame that is filled with the UILabel frame data
var newFrame: CGRect = label.frame
// resizing the frame to calculated size
newFrame.size.height = expectedLabelSize.height
// put calculated frame into UILabel frame
label.frame = newFrame
```

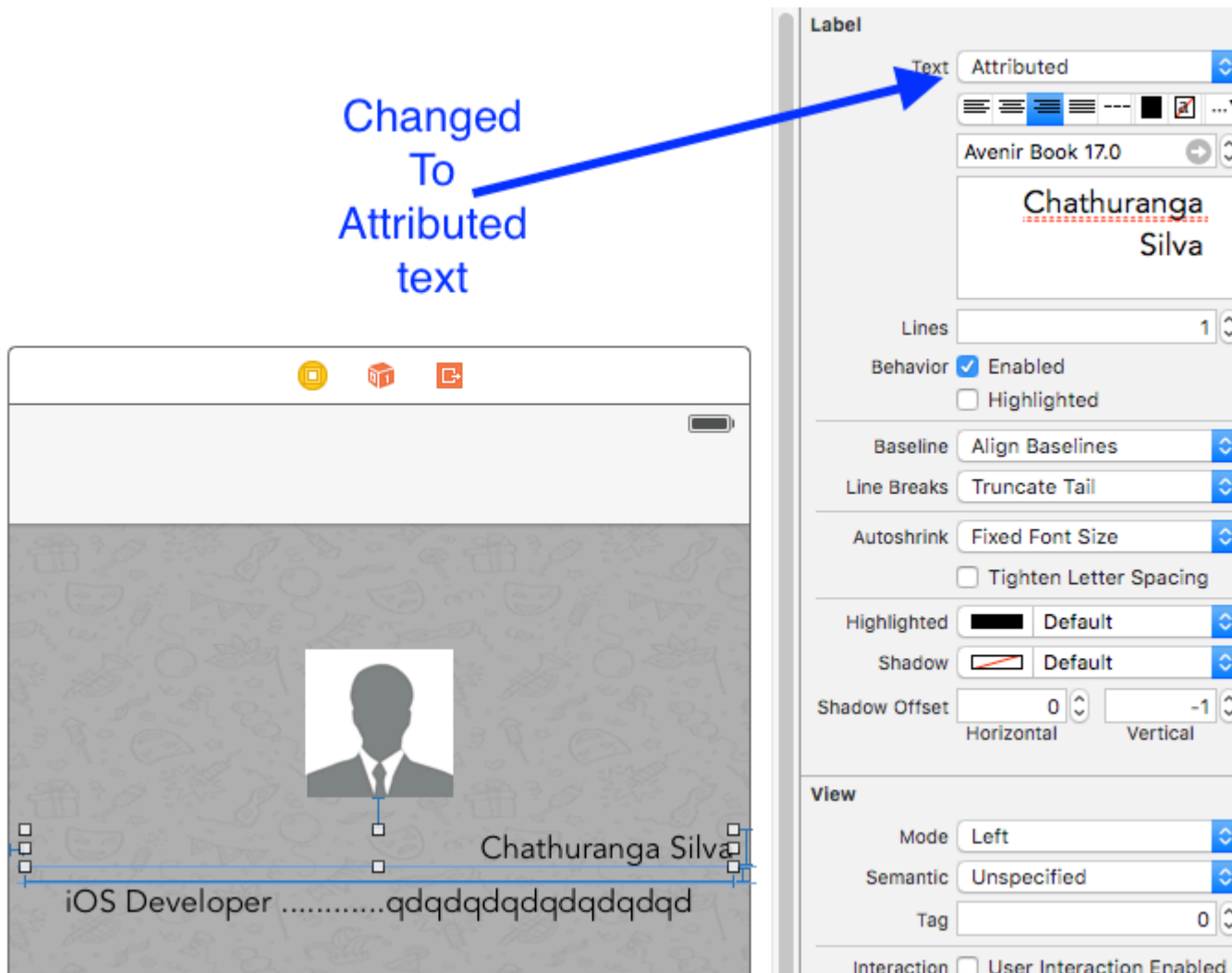
Etiqueta de texto atribuido

01. Texto subrayado: - Línea simple / doble, atravesar: - Línea simple / doble

Paso 1

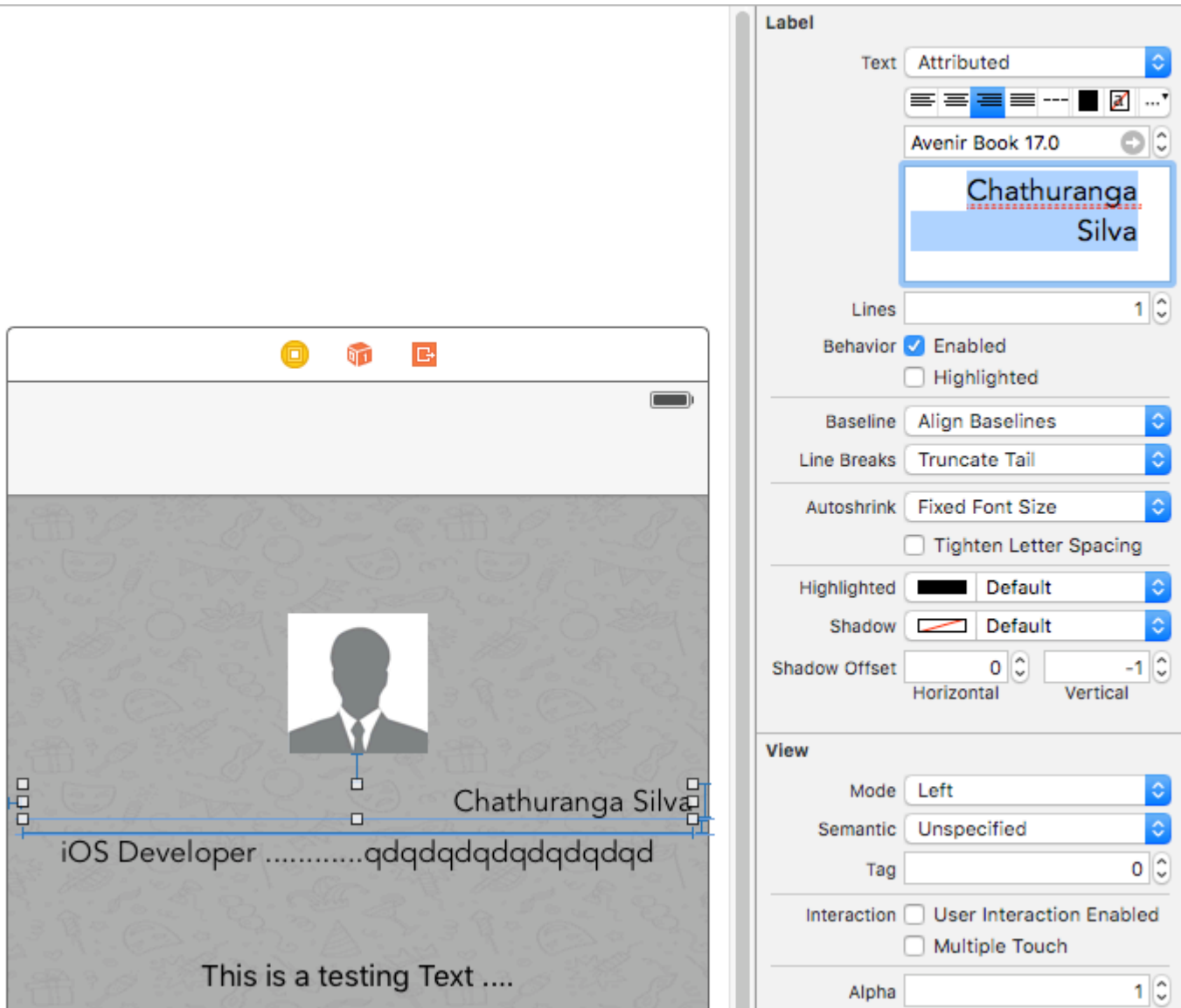
Seleccione la etiqueta y cambie el tipo de etiqueta Plain to Attributed





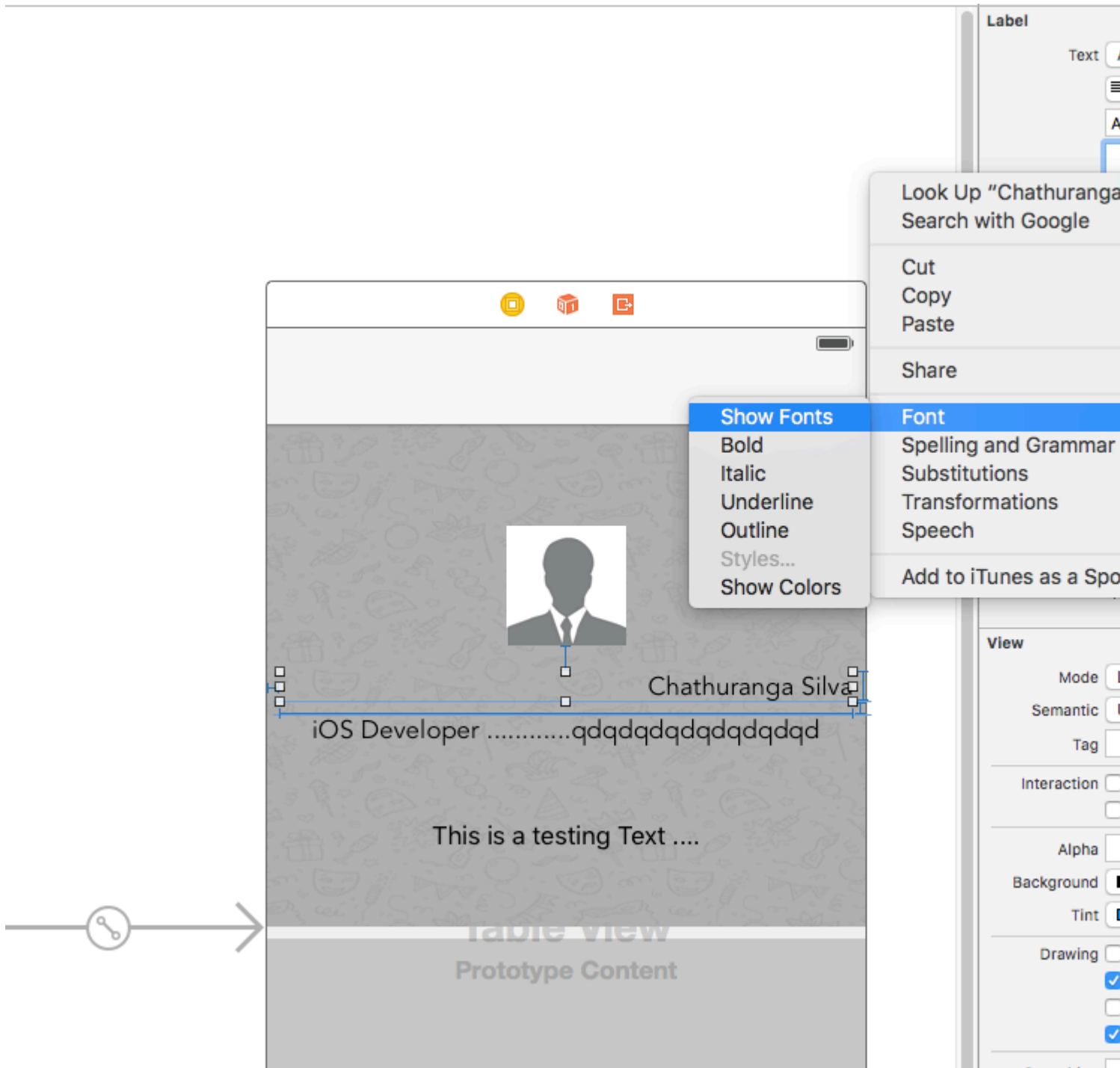
Paso 2

Haga clic en el texto de la etiqueta y haga clic derecho



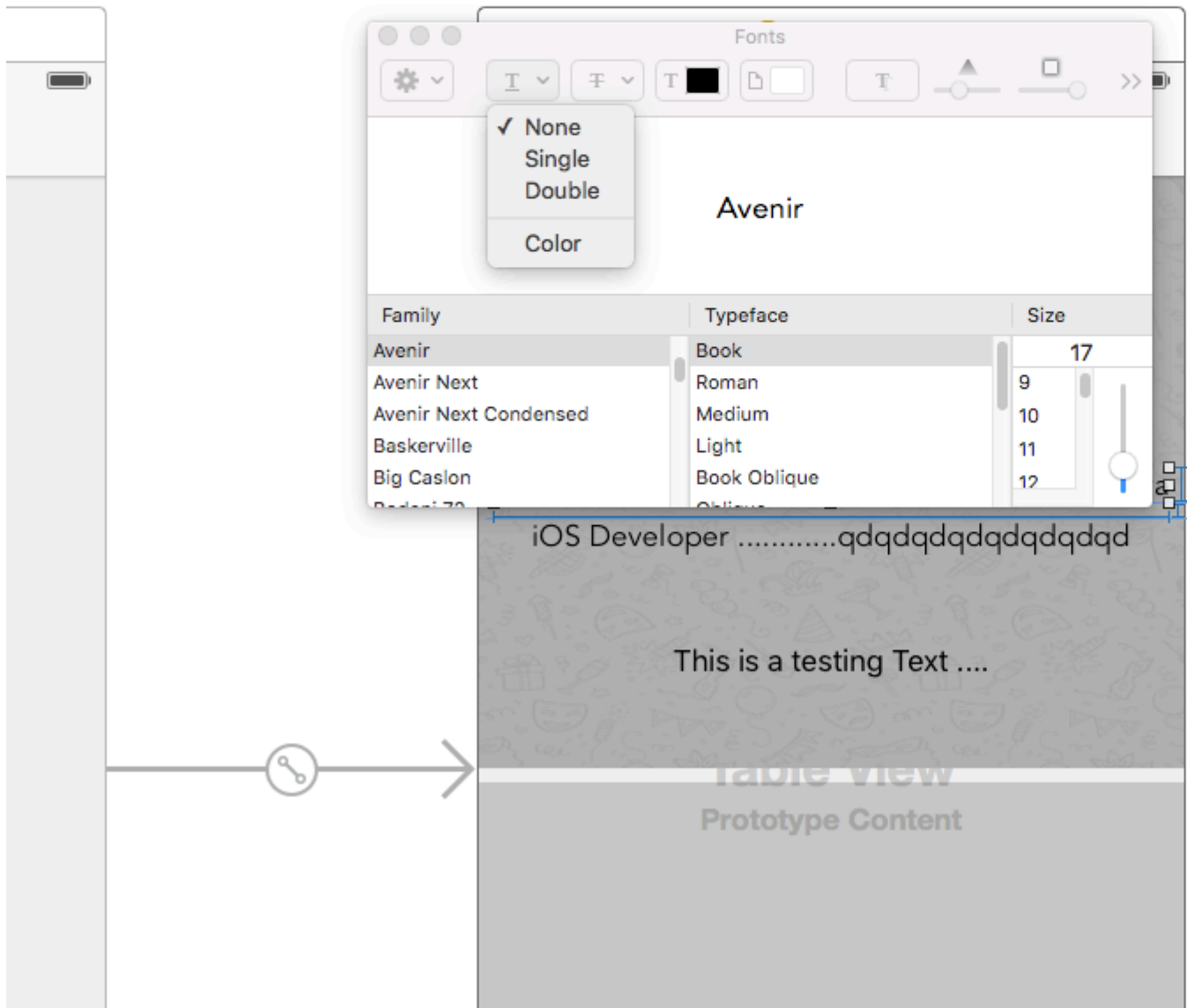
Paso 3

Luego haga clic en Fuente -> Mostrar fuentes

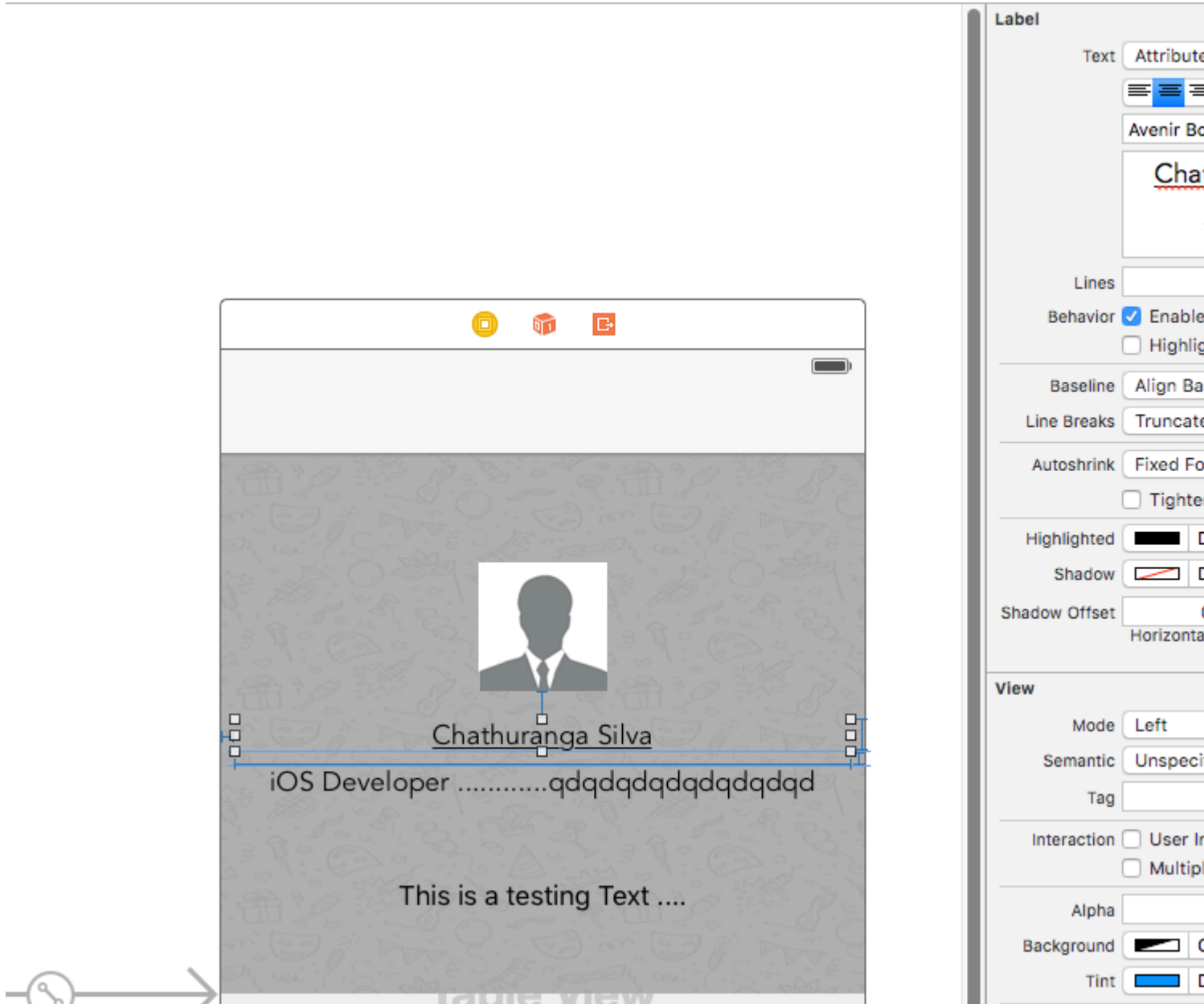


Etapa 4

Luego se mostrará la vista de fuente y haga clic en el botón de subrayado para hacer que el texto esté subrayado o haga clic en el botón de tachado para hacer que el texto aparezca tachado. Y seleccione una línea o una línea doble.

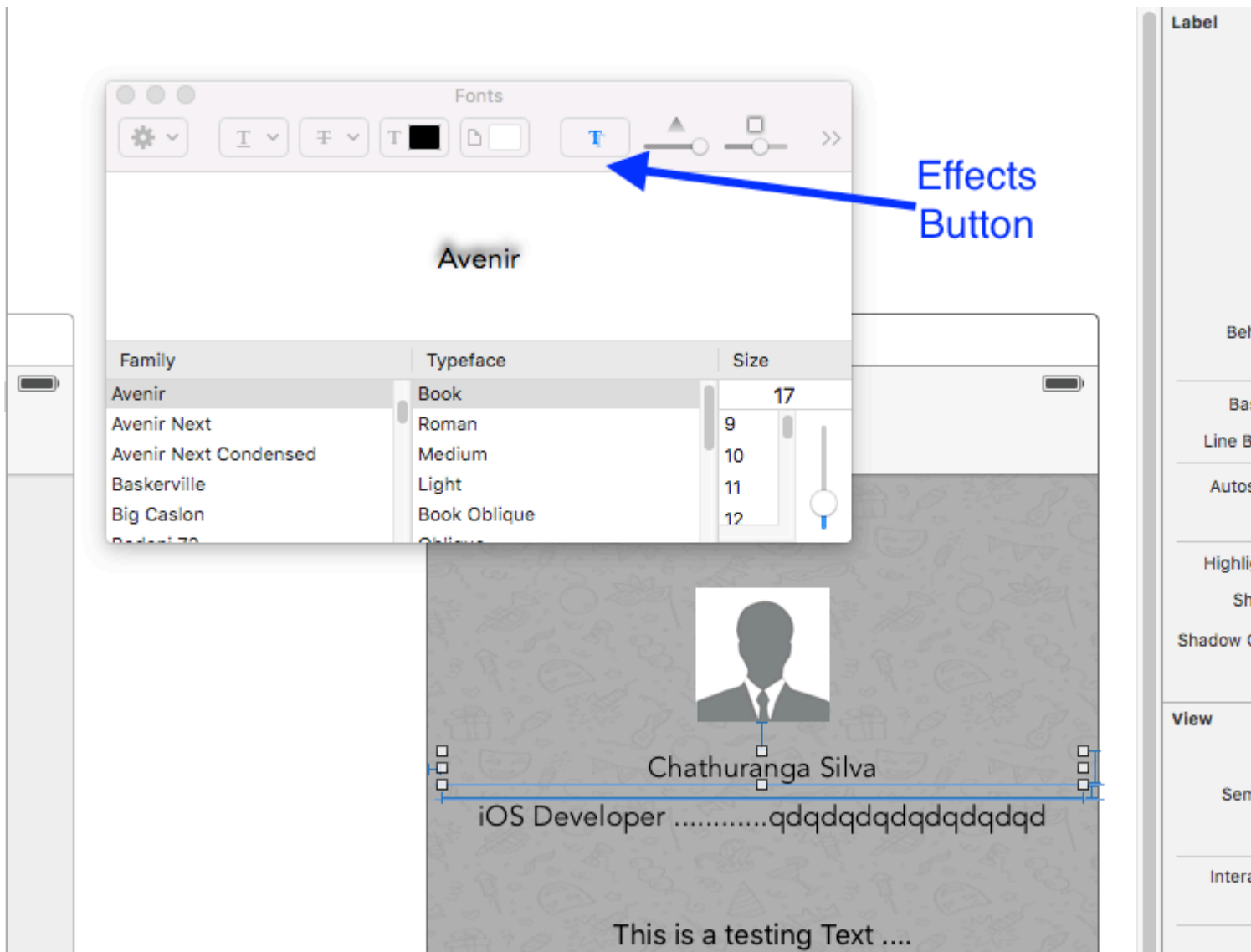


Finalmente, haga clic en Intro y la etiqueta se mostrará subrayada o tachada según su selección.

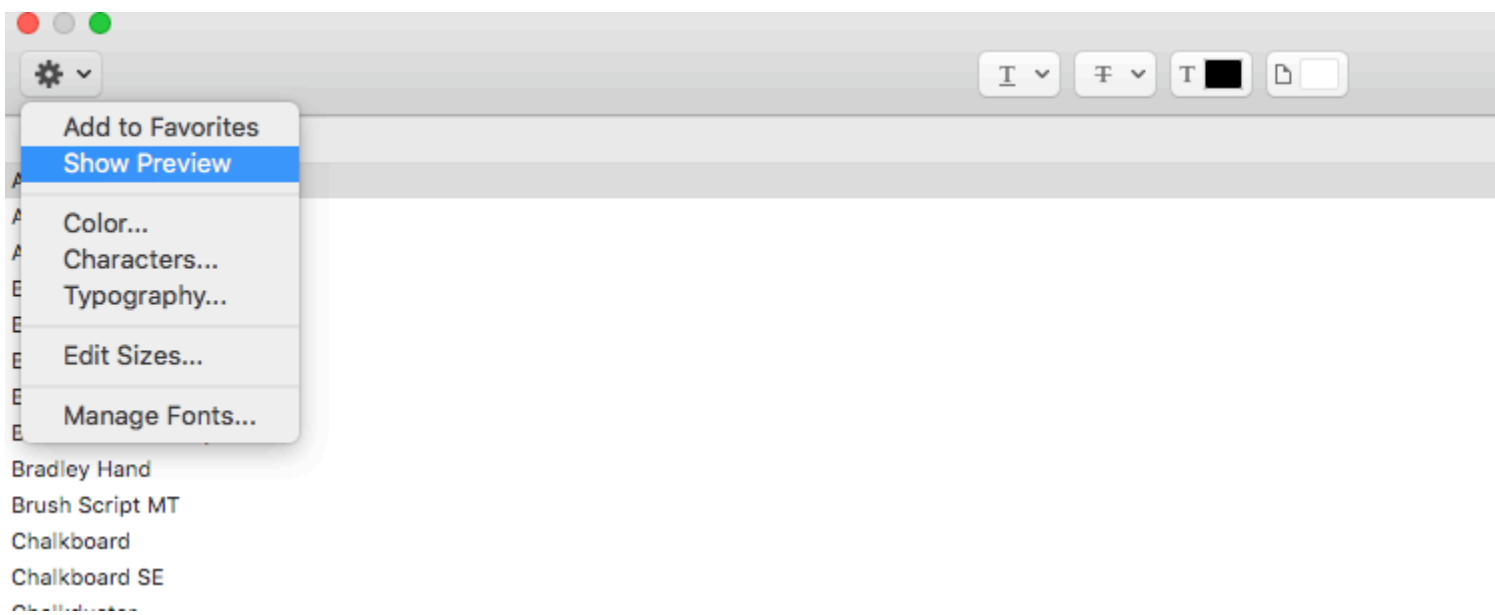


02. Añadir texto sombreado / efectos de fondo borroso

Obtenga la vista de fuente como se describe anteriormente y haga clic en el botón de efectos.



Si no ve la vista previa, haga clic en Mostrar imagen en la configuración.



Finalmente cambie la sombra y el desplazamiento de acuerdo a sus preferencias.



Avenir

Justifica el texto

Rápido

```
let sampleText = "Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint
occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est
laborum."
```

```
// Create label
let label = UILabel(frame: CGRectMake(0, 0, view.frame.size.width, 400))
label.numberOfLines = 0
label.lineBreakMode = NSLineBreakMode.ByWordWrapping
```

```
// Justify text through paragraph style
let paragraphStyle = NSMutableParagraphStyle()
paragraphStyle.alignment = NSTextAlignment.Justified
let attributes = [NSParagraphStyleAttributeName: paragraphStyle,
NSBaselineOffsetAttributeName: NSNumber(float: 0)]
let attributedString = NSAttributedString(string: sampleText, attributes: attributes)
label.attributedString = attributedString
view.addSubview(label)
```

C objetivo

```
NSString *sampleText = @"Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis
nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure
dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim
id est laborum.";
```

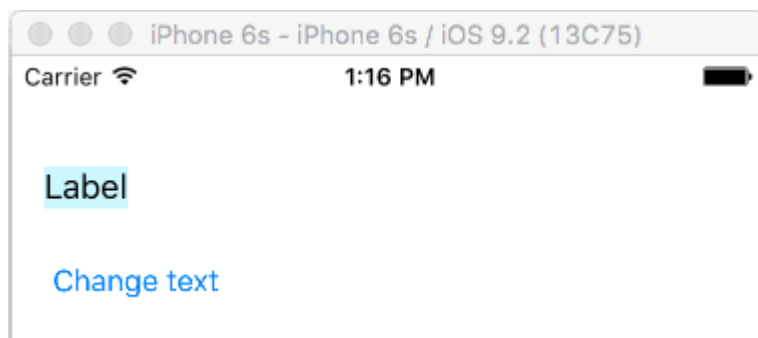
```
// Create label
UILabel *label = [[UILabel alloc] initWithFrame:CGRectMake(0, 0,
self.view.frame.size.width, 400)];
label.numberOfLines = 0;
label.lineBreakMode = NSLineBreakByWordWrapping;
```

```
// Justify text through paragraph style
NSMutableParagraphStyle *paragraphStyle = [[NSMutableParagraphStyle alloc] init];
paragraphStyle.alignment = NSTextAlignmentJustified;
NSAttributedString *attributedString = [[NSAttributedString alloc]
initWithString:sampleText attributes:@{
    NSParagraphStyleAttributeName : paragraphStyle,
    NSBaselineOffsetAttributeName : [NSNumber numberWithInt:0]}
```

```
    }];  
    label.attributedString = attributedString;  
    [self.view addSubview:label];
```

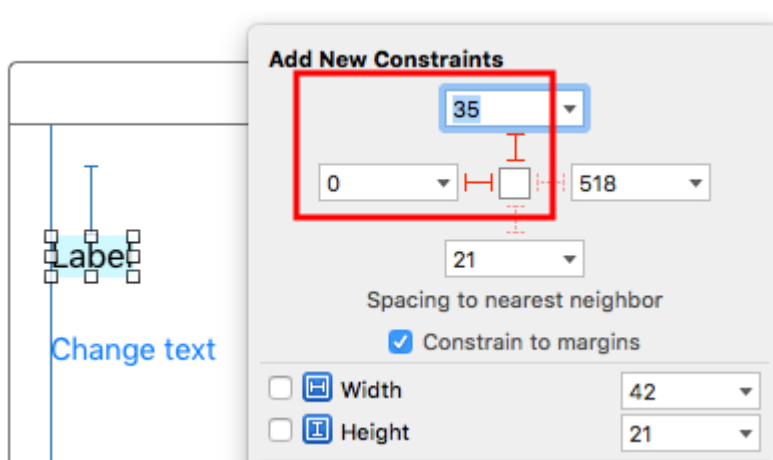
Etiqueta de tamaño automático para ajustar el texto

Este ejemplo muestra cómo el ancho de una etiqueta puede redimensionarse automáticamente cuando cambia el contenido del texto.



Pin los bordes izquierdo y superior

Simplemente use el diseño automático para agregar restricciones para fijar los lados izquierdo y superior de la etiqueta.



Después de eso cambiará automáticamente de tamaño.

Notas

- Este ejemplo proviene de [esta respuesta de desbordamiento de pila](#).
- No agregue restricciones para el ancho y la altura. Las etiquetas tienen un tamaño *intrínseco* basado en su contenido de texto.
- No es necesario establecer `sizeToFit` cuando se utiliza el diseño automático. El código completo para el proyecto de ejemplo está aquí:

```
import UIKit
class ViewController: UIViewController {

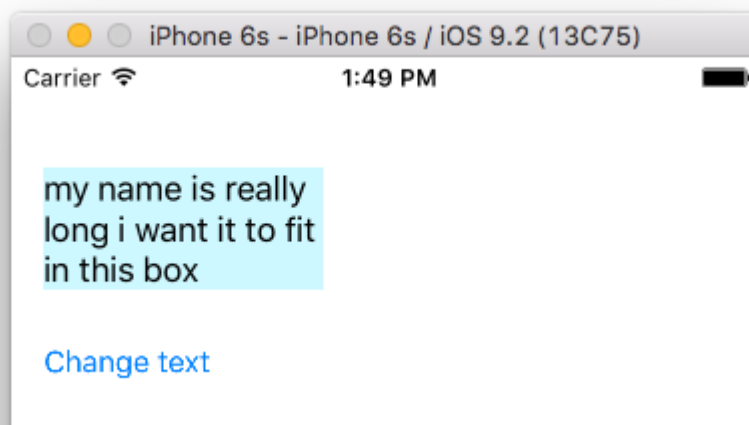
    @IBOutlet weak var myLabel: UILabel!

    @IBAction func changeTextButtonTapped(sender: UIButton) {
        myLabel.text = "my name is really long i want it to fit in this box"
    }
}
```

- Este método también se puede usar para espaciar correctamente varias etiquetas horizontalmente como en [este ejemplo](#) .



- Si desea que su etiqueta se ajuste de línea, establezca el número de líneas en 0 en IB y agregue `myLabel.preferredMaxLayoutWidth = 150 // or whatever` en el código. (El botón también está sujeto a la parte inferior de la etiqueta para que se mueva hacia abajo cuando la altura de la etiqueta aumenta).



Obtenga el tamaño de UILabel basado estrictamente en su texto y fuente

`NSString` proporciona el método `boundingRectWithSize` que se puede usar para predecir el `CGSize` resultante de un `UILabel` basado en su texto y fuente sin la necesidad de crear un `UILabel`

C objetivo

```
[[text boundingRectWithSize:maxSize options:(NSStringDrawingTruncatesLastVisibleLine |
NSStringDrawingUsesLineFragmentOrigin) attributes:@{NSFontAttributeName: fontName}
context:nil] size];
```


Rápido

```
let nsText = text as NSString?
nsText?.boundingRectWithSize(maxSize, options: [.TruncatesLastVisibleLine,
.UsesLineFragmentOrigin], attributes: [NSFontAttributeName: fontName], context: nil).size
```

Rápido

Crear etiqueta y etiquetar salida de restricción de altura. Agregue el siguiente código donde asignará el texto a la etiqueta.

```
@IBOutlet var lblDescriptionHeightConstration: NSLayoutConstraint!
@IBOutlet weak var lblDescription: UILabel!

let maxWidth = UIScreen.mainScreen().bounds.size.width - 40
let sizeOfLabel = self.lblDesc.sizeThatFits(CGSize(width: maxWidth, height: CGFloat.max))
self.lblDescriptionHeightConstration.constant = sizeOfLabel.height
```

Nota: "40" es el espacio del lado izquierdo y derecho de la pantalla.

Color de texto resaltado y resaltado

C objetivo

```
UILabel *label = [[UILabel alloc] init];
label.highlighted = YES;
label.highlightedTextColor = [UIColor redColor];
```

Rápido

```
let label = UILabel()
label.highlighted = true
label.highlightedTextColor = UIColor.redColor()
```

Swift 3

```
let label = UILabel()
label.isHighlighted = true
label.highlightedTextColor = UIColor.red
```

Lea UILabel en línea: <https://riptutorial.com/es/ios/topic/246/UILabel>

Capítulo 176: UILabel texto subrayado

Examples

Subrayando un texto en un UILabel usando Objective C

```
UILabel *label=[[UILabel alloc]initWithFrame:CGRectMake(0, 0, 320, 480)];
label.backgroundColor=[UIColor lightGrayColor];
NSMutableAttributedString *attributedString;
attributedString = [[NSMutableAttributedString alloc] initWithString:@"Apply Underlining"];
[attributedString addAttribute:NSUnderlineStyleAttributeName value:@1 range:NSMakeRange(0,
[attributedString length])];
[label setAttributedText:attributedString];
```

Subrayando un texto en UILabel usando Swift

```
let label = UILabel.init(frame: CGRect(x: 0, y:0, width: 100, height: 40))
label.backgroundColor = .lightGray
let attributedString = NSMutableAttributedString.init(string: "Apply UnderLining")
attributedString.addAttribute(NSUnderlineStyleAttributeName, value: 1, range:
NSRange.init(location: 0, length: attributedString.length))
label.attributedText = attributedString
```

Lea UILabel texto subrayado en línea: <https://riptutorial.com/es/ios/topic/7219/ui-label-texto-subrayado>

Capítulo 177: UILocalNotification

Introducción

Las notificaciones locales permiten que su aplicación notifique al usuario sobre el contenido que no requiere el uso de un servidor.

A diferencia de las notificaciones remotas que se activan desde un servidor, las notificaciones locales se programan y activan dentro de una aplicación. Las notificaciones en general están dirigidas a aumentar la interacción del usuario con la aplicación, invitando o tentando al usuario a abrir e interactuar con ella.

UILocalNotification estaba en desuso en iOS 10. En su lugar, use el marco UserNotifications.

Observaciones

No confunda UILocalNotification con notificaciones push. Su dispositivo desencadena UILocalNotification y, cuando está programado, se copia en el sistema.

Campo de golf:

- [Referencia de la clase UILocalNotification](#)
- [UILocalNotification on Stack Overflow](#)

Examples

Programación de una notificación local.

Asegúrate de ver [Registro para notificaciones locales](#) para que esto funcione:

Rápido

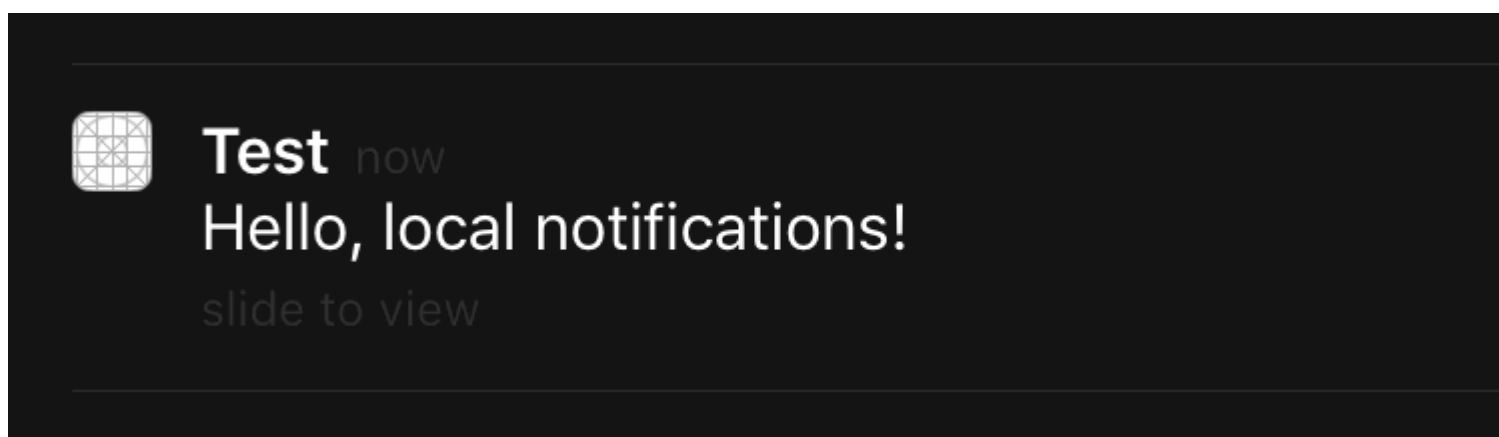
```
let notification = UILocalNotification()
notification.alertBody = "Hello, local notifications!"
notification.fireDate = NSDate().dateByAddingTimeInterval(10) // 10 seconds after now
UIApplication.sharedApplication().scheduleLocalNotification(notification)
```

C objetivo

```
UILocalNotification *notification = [[UILocalNotification alloc] init];
notification.alertBody = @"Hello, local notifications!";
notification.fireDate = [NSDate dateWithTimeIntervalSinceNow:10]; // 10 seconds after now
[[UIApplication sharedApplication] scheduleLocalNotification:notification];
```

Para ver la notificación en el simulador de iOS, escriba `⌘H` (control-command-H) para ir a casa y luego escriba `⌘L` (command-L) para bloquear el dispositivo. Espere unos segundos, y la

notificación debería aparecer (esta apariencia variará según el tipo de notificación que se describe en "Registro para notificaciones locales"):



Deslice el dedo en la notificación para volver a la aplicación (tenga en cuenta que si lo llamó en la primera vista del controlador `viewDidLoad`, `viewWillAppear`, `viewDidAppear`, etc., la notificación se programará nuevamente).

Registro para notificaciones locales

iOS 8

Para presentar notificaciones locales al usuario, debe registrar su aplicación en el dispositivo:

Rápido

```
let settings = UIUserNotificationSettings(forTypes: [.Badge, .Sound, .Alert], categories: nil)
UIApplication.sharedApplication().registerUserNotificationSettings(settings)
```

C objetivo

```
UIUserNotificationSettings *settings = [UIUserNotificationSettings
settingsForTypes:(UIUserNotificationTypeBadge | UIUserNotificationTypeSound |
UIUserNotificationTypeAlert) categories:nil];
[[UIApplication sharedApplication] registerUserNotificationSettings:settings];
```

Esto presentará una alerta la primera vez que se llame:

"Test" Would Like to Send You Notifications

Notifications may include alerts, sounds, and icon badges. These can be configured in Settings.

Don't Allow

OK

Independientemente de lo que elija el usuario, la alerta no volverá a aparecer y los cambios deberán ser iniciados por el usuario en Configuración.

Respondiendo a la notificación local recibida

IMPORTANTE: este método de delegado solo se llama en primer plano.

Rápido

```
func application(application: UIApplication, didReceiveLocalNotification notification:
UILocalNotification) {
}
}
```

C objetivo

```
- (void)application:(UIApplication *)application
didReceiveLocalNotification:(UILocalNotification *)notification {
}
}
```

Este método generalmente se reemplaza en AppDelegate, que cumple con el protocolo UIApplicationDelegate.

Gestionando notificaciones locales utilizando UUID

Muchas veces tendrá que ser capaz de administrar sus notificaciones, al poder realizar un seguimiento de ellos y cancelarlos.

Rastrear una notificación

Puede asignar un UUID (identificador único universal) a una notificación, por lo que puede rastrearlo:

Rápido

```
let notification = UILocalNotification()
let uuid = NSUUID().uuidString
notification.userInfo = ["UUID": uuid]
UIApplication.shared.scheduleLocalNotification(notification)
```

C objetivo

```
UILocalNotification *notification = [[UILocalNotification alloc] init];
NSString *uuid = [[NSUUID UUID] UUIDString];
notification.userInfo = @{@"UUID": uuid };
[[UIApplication sharedApplication] scheduleLocalNotification:notification];
```

Cancelar una notificación

Para cancelar una notificación, primero obtenemos una lista de todas las notificaciones y luego encontramos la que tiene un UUID correspondiente. Finalmente, lo cancelamos.

Rápido

```
let scheduledNotifications = UIApplication.shared.scheduledLocalNotifications

guard let scheduledNotifications = scheduledNotifications else {
    return
}

for notification in scheduledNotifications where "\(notification.userInfo!["UUID"]!)" ==
UUID_TO_CANCEL {
    UIApplication.sharedApplication().cancelLocalNotification(notification)
}
```

C objetivo

```
NSArray *scheduledNotifications = [[UIApplication sharedApplication]
scheduledLocalNotifications];

for (UILocalNotification *notification in scheduledNotifications) {
    if ([[notification.userInfo objectForKey:@"UUID"] compare: UUID_TO_CANCEL]) {
        [[UIApplication sharedApplication] cancelLocalNotification:notification];
        break;
    }
}
```

Probablemente querrá almacenar todos estos UUID en Core Data o Realm.

Presentando una notificación local de inmediato.

Si desea mostrar una notificación local de inmediato, debe llamar a:

Swift 3

```
UIApplication.shared.presentLocalNotificationNow(notification)
```

Swift 2

```
UIApplication.sharedApplication().presentLocalNotificationNow(notification)
```

C objetivo

```
[[UIApplication sharedApplication] presentLocalNotificationNow:notification];
```

Una ventaja de usar esto es que no tendrá que configurar las propiedades `fireDate` y `timeZone` de su objeto `UILocalNotification`.

Sonido de notificación

Se pueden proporcionar sonidos personalizados para las notificaciones generadas por su aplicación. Cuando el sistema muestra una alerta para una notificación local o asigna un icono a una aplicación, reproduce este sonido (siempre que el usuario no haya desactivado los sonidos de notificación).

El valor predeterminado es `nil`, lo que significa que no se reproduce ningún sonido para su notificación.

Para proporcionar un sonido personalizado, agregue un `.caf`, `.wav` o `.aiff` a su paquete de aplicaciones. No se admiten los sonidos que duran más de 30 segundos. El suministro de un sonido que no cumpla con esos requisitos hará que se reproduzca el sonido predeterminado (`UILocalNotificationDefaultSoundName`).

C objetivo

```
UILocalNotification *notification = [UILocalNotification new];  
notification.soundName = @"nameOfSoundInBundle.wav"; // Use  
UILocalNotificationDefaultSoundName for the default alert sound
```

Rápido

```
let notification = UILocalNotification()  
notification.soundName = "nameOfSoundInBundle.wav"
```

Regístrese y programe notificaciones locales en Swift 3.0 (iOS 10)

Registro

en AppDelegate

```
import UserNotifications
```

en el método **didFinishLaunchingWithOptions** ,

```
UNUserNotificationCenter.current().requestAuthorization(options: [.alert, .sound, .badge]) {
    (granted, error) in

    // Here you can check Request is Granted or not.

}
```

Crear y programar la notificación.

```
let content = UNMutableNotificationContent()
content.title = "10 Second Notification Demo"
content.subtitle = "From Wolverine"
content.body = "Notification after 10 seconds - Your pizza is Ready!!"
content.categoryIdentifier = "myNotificationCategory"

let trigger = UNTimeIntervalNotificationTrigger(
    timeInterval: 10.0,
    repeats: false)

let request = UNNotificationRequest(
    identifier: "10.second.message",
    content: content,
    trigger: trigger
)
UNUserNotificationCenter.current().add(request, withCompletionHandler: nil)
```

Donde sea que se active esta parte del código, si ha permitido el Permiso de Notificación, recibirá una notificación.

Para probarlo correctamente, asegúrese de que su aplicación en modo de fondo.

Novedades en UILocalNotification con iOS10

Puede usar `UILocalNotification` , las API antiguas también funcionan bien con iOS10, pero es mejor que utilicemos las API en el marco de notificaciones de usuarios. También hay algunas características nuevas que solo puede usar con el marco de notificaciones de usuario de iOS10.

Esto también le sucede a la Notificación Remota, para más información: [Aquí](#) .

Nuevas características:

1. Ahora puede presentar un distintivo de alerta, sonido o aumento mientras la aplicación también está en primer plano con iOS 10
2. Ahora puede manejar todos los eventos en un solo lugar cuando el usuario pulsó (o deslizó) el botón de acción, incluso cuando la aplicación ya se ha eliminado.
3. Soporta toque 3D en lugar de gesto deslizante.
4. Ahora puede eliminar notificaciones locales específicas solo por un código de fila.
5. Admite notificaciones enriquecidas con una interfaz de usuario personalizada.

Es realmente fácil para nosotros convertir `UILocalNotification` API de `UILocalNotification` en las API del marco de notificaciones de usuario de iOS10, son realmente similares.

Escribo una demostración aquí para mostrar cómo usar las API nuevas y antiguas al mismo tiempo: [iOS10AdaptationTips](#) .

Por ejemplo,

Con la implementación Swift:

1. importar notificaciones de usuarios

```
/// Notification become independent from UIKit
import UserNotifications
```

2. autorización de solicitud para la notificación local

```
let center = UNUserNotificationCenter.current()
center.requestAuthorization(options: [.alert, .sound]) { (granted, error) in
    // Enable or disable features based on authorization.
}
```

3. horario localNotificación

4. icono de la aplicación de actualización número de placa

```
@IBAction func triggerNotification(){
    let content = UNMutableNotificationContent()
    content.title = NSLocalizedString(forKey: "Elon said:",
arguments: nil)
    content.body = NSLocalizedString(forKey: "Hello Tom Get up,
let's play with Jerry!", arguments: nil)
    content.sound = UNNotificationSound.default()
    content.badge = UIApplication.shared().applicationIconBadgeNumber + 1;
    content.categoryIdentifier = "com.elonchan.localNotification"
    // Deliver the notification in five seconds.
    let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 60.0, repeats:
true)
    let request = UNNotificationRequest.init(identifier: "FiveSecond", content: content,
trigger: trigger)

    // Schedule the notification.
    let center = UNUserNotificationCenter.current()
    center.add(request)
}

@IBAction func stopNotification(_ sender: AnyObject) {
    let center = UNUserNotificationCenter.current()
    center.removeAllPendingNotificationRequests()
    // or you can remove specifical notification:
    // center.removePendingNotificationRequests(withIdentifiers: ["FiveSecond"])
}
```

Implementación de Objective-C:

1. importar notificaciones de usuarios

```
// Notifications are independent from UIKit
```

```
#import <UserNotifications/UserNotifications.h>
```

2. autorización de solicitud para la notificación local

```
UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];  
[center requestAuthorizationWithOptions:(UNAuthorizationOptionBadge |  
UNAuthorizationOptionSound | UNAuthorizationOptionAlert)  
completionHandler:^(BOOL granted, NSError * _Nullable error) {  
    if (!error) {  
        NSLog(@"request authorization succeeded!");  
        [self showAlert];  
    }  
}];
```

3. horario localNotificación

4. icono de la aplicación de actualización número de placa

```
UNMutableNotificationContent *content = [[UNMutableNotificationContent alloc] init];  
content.title = [NSString localizedUserNotificationStringForKey:@"Elon said:"  
                arguments:nil];  
content.body = [NSString localizedUserNotificationStringForKey:@"Hello Tom Get up, let's  
play with Jerry!"  
                arguments:nil];  
content.sound = [UNNotificationSound defaultSound];  
  
// 4. update application icon badge number  
content.badge = [NSNumber numberWithInt:([UIApplication  
sharedApplication].applicationIconBadgeNumber + 1)];  
// Deliver the notification in five seconds.  
UNTimeIntervalNotificationTrigger *trigger = [UNTimeIntervalNotificationTrigger  
triggerWithTimeInterval:5.f  
repeats:NO];  
UNNotificationRequest *request = [UNNotificationRequest  
requestWithIdentifier:@"FiveSecond"  
                             content:content  
                             trigger:trigger];  
  
/// 3. schedule localNotification  
UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];  
[center addNotificationRequest:request withCompletionHandler:^(NSError * _Nullable error)  
{  
    if (!error) {  
        NSLog(@"add NotificationRequest succeeded!");  
    }  
}];
```

Vaya a aquí para obtener más información: [iOS10AdaptationTips](#) .

#actualizado

Aplicación de terminación debido a la excepción no detectada
'NSInternallnconsistencyException', razón: 'el intervalo de tiempo debe ser de al
menos 60 si se repite'

```
let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 60, repeats: true)
```

Lea UILocalNotification en línea: <https://riptutorial.com/es/ios/topic/635/uilocalnotification>

Capítulo 178: UINavigationController

Observaciones

De la [documentación](#) :

La clase UINavigationController implementa un controlador de vista especializado que administra la navegación del contenido jerárquico. Esta interfaz de navegación hace posible presentar sus datos de manera eficiente y facilita al usuario navegar por ese contenido. Por lo general, usa esta clase tal como está, pero también puede crear una subclase para personalizar el comportamiento de la clase.

Examples

Apareciendo en un controlador de navegación

Al controlador de vista anterior

Para volver a la página anterior, puedes hacer esto:

Rápido

```
navigationController?.popViewControllerAnimated(true)
```

C objetivo

```
[self.navigationController popViewControllerAnimated:YES];
```

Al controlador de vista raíz

Para aparecer en la raíz de la pila de navegación, puede hacer esto:

Rápido

```
navigationController?.popToRootViewControllerAnimated(true)
```

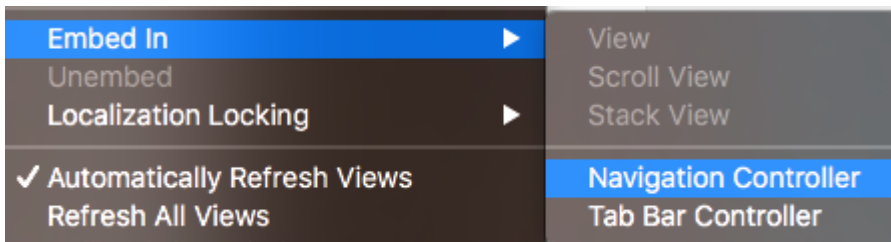
C objetivo

```
[self.navigationController popToRootViewControllerAnimated:YES];
```

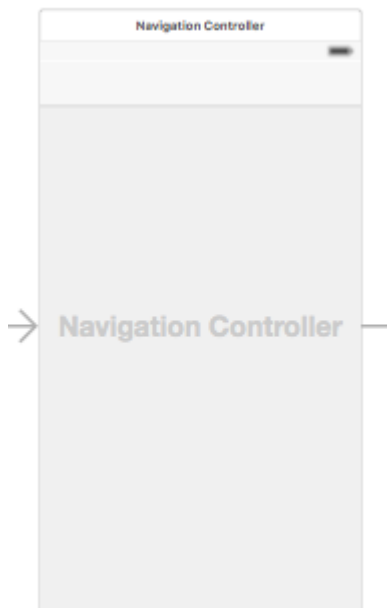
Creación de un controlador de navegación

En su guión gráfico, seleccione el controlador de vista que desea incrustar en un controlador de navegación.

Luego navegue hasta Editor > Incrustar en > Controlador de navegación



Y eso creará tu controlador de navegación.



Incrustar un controlador de vista en un controlador de navegación mediante programación

Rápido

```
//Swift
let viewController = UIViewController()
let navigationController = UINavigationController(rootViewController: viewController)

//Objective-C
UIViewController *viewController = [[UIViewController alloc] init];
UINavigationController *navigationController = [[UINavigationController alloc]
initWithRootViewController:viewController];
```

Presionando un controlador de vista en la pila de navegación

```
//Swift
let fooViewController = UIViewController()
navigationController?.pushViewController(fooViewController, animated: true)
```

```
//Objective-C
UIViewController *fooViewController = [[UIViewController alloc] init];
[navigationController pushViewController:fooViewController animated:YES];
```

Propósito

`UINavigationController` se utiliza para formar una jerarquía en forma de árbol de controladores de vista, que se conoce como una `navigation stack`.

Desde la perspectiva de los desarrolladores:

Puede conectar un controlador de fabricación independiente y obtener todos los beneficios de un administrador de jerarquía gratuito y un presentador de interfaz de usuario común de forma gratuita. `UINavigationController` anima la transición a nuevos controladores y proporciona la funcionalidad de respaldo automáticamente. `UINavigationController` también da acceso a todos los demás controladores en la `navigation stack` que pueden ayudar a acceder a algunas funciones o datos.

Desde la perspectiva del usuario:

`UINavigationController` ayuda a recordar dónde se encuentra el usuario en ese momento (título de la barra de navegación) y cómo puede regresar (botón de nuevo incrustado) a una de las pantallas anteriores.

Lea `UINavigationController` en línea: <https://riptutorial.com/es/ios/topic/1079/uinavigationcontroller>

Capítulo 179: UIPageViewController

Introducción

UIPageViewController brinda a los usuarios la capacidad de realizar fácilmente la transición entre varias vistas mediante un gesto de deslizamiento. Para crear un UIPageViewController, debe implementar los métodos `UIPageViewControllerDataSource`. Estos incluyen métodos para devolver tanto el `UIPageViewController` antes y después del `UIPageViewController` actual junto con los métodos `presentationCount` y `presentationIndex`.

Sintaxis

1. `UIPageViewControllerTransitionStyle`
2. `UIPageViewControllerNavigationOrientation`
3. `UIPageViewControllerSpineLocation`
4. `UIPageViewControllerNavigationDirection`

Observaciones

Referencia de desarrollador de Apple [aquí](#)

Examples

Crear una paginación horizontal UIPageViewController programáticamente

1. Arreglo inicial de controladores de vista que serán administrados por `UIPageViewController`. Agregue una clase de controlador de vista base que tenga un `identifier` propiedad que se usará para identificar controladores de vista cuando trabaje con métodos de fuente de datos `UIPageViewController`. Deje que los controladores de vista hereden de esa clase base.

```
UIViewController *firstVC = [[UIViewController alloc] init];
firstVC.identifier = 0
UIViewController *secondVC = [[UIViewController alloc] init];
secondVC.identifier = 1
NSArray *viewControllers = [[NSArray alloc] initWithObjects: firstVC, secondVC, nil];
```

2. Crear instancia de `UIPageViewController`.

```
UIPageViewController *pageViewController = [[UIPageViewController alloc]
initWithTransitionStyle:UIPageViewControllerTransitionStyleScroll

navigationOrientation:UIPageViewControllerNavigationOrientationHorizontal

options:nil];
```

3. La fuente de datos es la clase actual que debe implementar el protocolo

```
UIPageViewControllerDataSource .
```

```
pageViewController.dataSource = self;
```

4. `setViewControllers` agregará solo el controlador de la primera vista, luego se agregará a la pila usando métodos de fuente de datos

```
if (viewControllers.count) {  
    [pageViewController setViewControllers:@[[viewControllers objectAtIndex:0]]  
                        direction:UIPageViewControllerNavigationDirectionForward  
                        animated:NO  
                        completion:nil];  
}
```

5. Añadir `UIPageViewController` como un controlador de vista del niño por lo que va a recibir de su padre vista del controlador de `appearance` y la `rotation` los acontecimientos.

```
[self addChildViewController:pageViewController];  
pageViewController.view.frame = self.view.frame;  
[self.view addSubview:pageViewController.view];  
[pageViewController didMoveToParentViewController:self];
```

6. Implementando los métodos `UIPageViewControllerDataSource`

```
- (UIViewController *)pageViewController:(UIPageViewController *)pageViewController  
viewControllerBeforeViewController:(UIViewController *)viewController  
{  
    index = [(Your View Controller Base Class *)viewController identifier];  
    index--;  
    return [self childViewControllerAtIndex:index];  
}  
  
- (UIViewController *)pageViewController:(UIPageViewController *)pageViewController  
viewControllerAfterViewController:(UIViewController *)viewController  
{  
    index = [(Your View Controller Base Class *)viewController identifier];  
    index++;  
    return [self childViewControllerAtIndex:index];  
}  
  
- (NSInteger)presentationCountForPageViewController:(UIPageViewController *)pageViewController  
{  
    return [viewControllers count];  
}  
  
- (NSInteger)presentationIndexForPageViewController:(UIPageViewController *)pageViewController  
{  
    return index;  
}
```

7. Método de utilidad que devuelve un controlador de vista utilizando un índice, si el índice está fuera de los límites, devuelve cero.

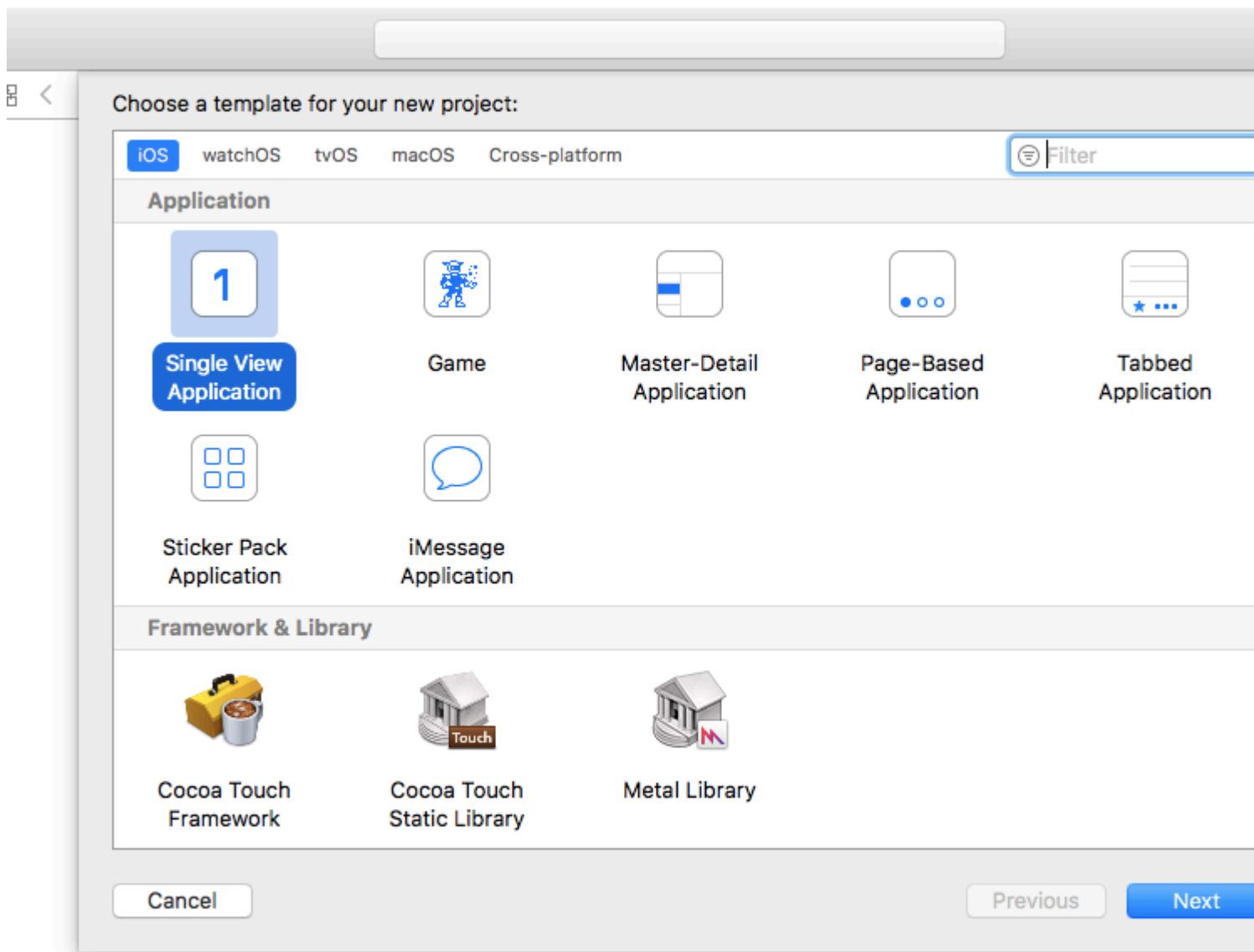
```
- (UIViewController *)childViewControllerAtIndex:(NSInteger) index
```



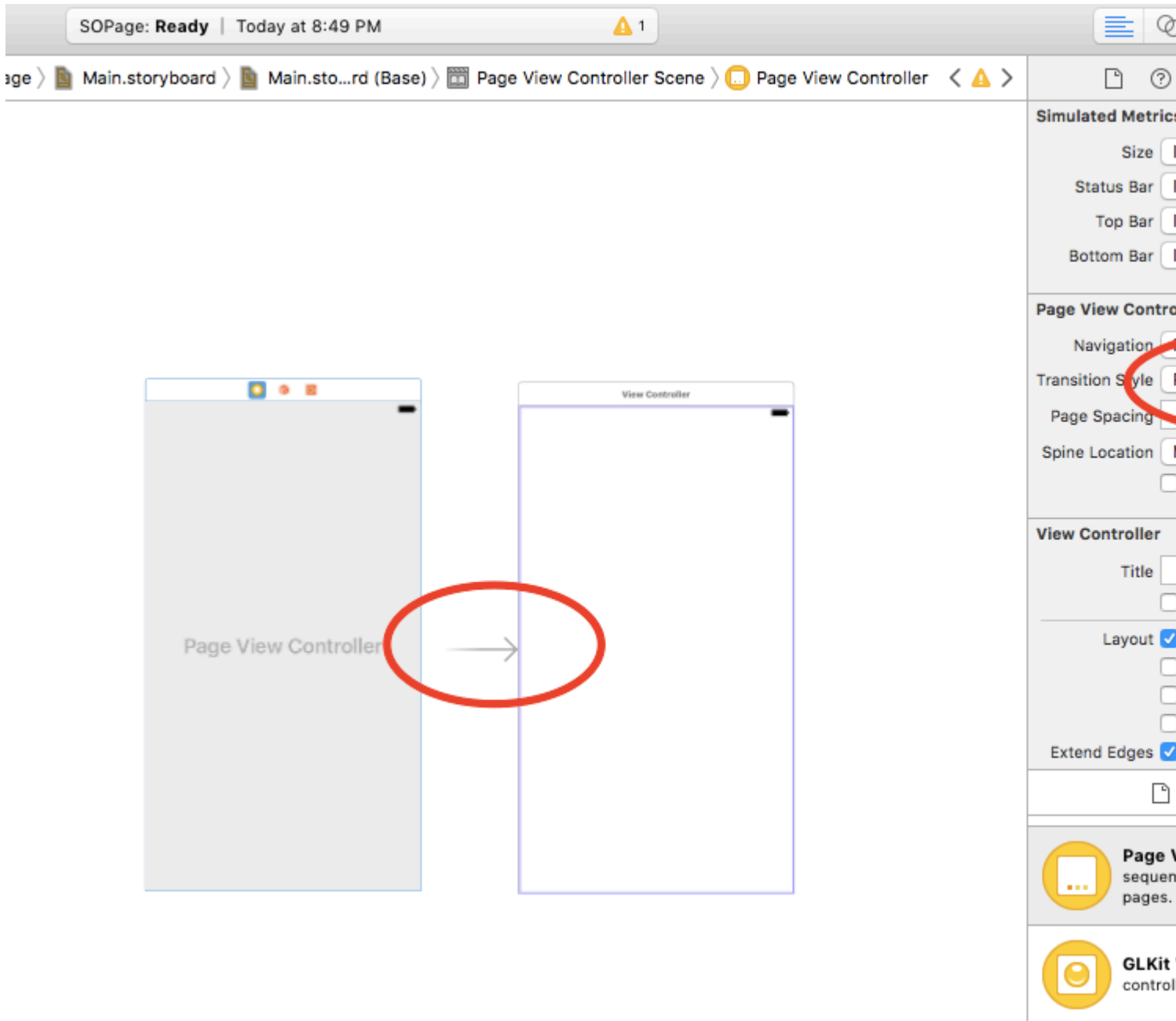
```
{
  if (index <= ([viewControllers count] - 1)) {
    return [viewControllers objectAtIndex:index];
  } else {
    return nil;
  }
}
```

Una forma sencilla de crear controladores de vista de página horizontales (páginas infinitas)

1. Vamos a crear un nuevo proyecto, estoy eligiendo la aplicación de vista única para una mejor demostración



2. Arrastre un controlador de vista de página al guión gráfico, hay dos cosas que debe cambiar después de eso:
 1. Establecer el controlador de vista de página como controlador de vista inicial
 2. Cambia el estilo de transición para desplazarte



3. Y necesita crear una clase `UIPageViewController`, luego configurarlo como clase personalizada del controlador de vista de página en el guión gráfico
4. Pegue este código en su clase `UIPageViewController`, debería obtener una aplicación paginada infinita y colorida :)

```
class PageViewController: UIPageViewController, UIPageViewControllerDataSource {  
  
    override func viewDidLoad() {  
        self.dataSource = self  
        let controller = createViewController()  
        self.setViewControllers([controller], direction: .forward, animated: false,  
completion: nil)  
    }  
  
    func pageViewController(_ pageViewController: UIPageViewController,  
viewControllerBefore viewController: UIViewController) -> UIViewController? {  
        let controller = createViewController()  
    }  
}
```

```

        return controller
    }

    func pageViewController(_ pageViewController: UIPageViewController,
viewControllerAfter viewController: UIViewController) -> UIViewController? {
        let controller = createViewController()
        return controller
    }

    func createViewController() -> UIViewController {
        var randomColor: UIColor {
            return UIColor(hue: CGFloat(arc4random_uniform(360))/360, saturation: 0.5,
brightness: 0.8, alpha: 1)
        }
        let storyboard = UIStoryboard(name: "Main", bundle: nil)
        let controller = storyboard.instantiateViewController(withIdentifier: "View
Controller")
        controller.view.backgroundColor = randomColor
        return controller
    }
}

```

Así es como se ve el proyecto final, obtienes un controlador de vista con un color diferente con cada desplazamiento:

Capítulo 180: UIPheonix: marco de IU fácil, flexible, dinámico y altamente escalable

Introducción

Inspirado en el desarrollo de juegos, UIPheonix es un concepto de UI framework súper fácil, flexible, dinámico y altamente escalable para crear componentes / aplicaciones reutilizables basadas en control para macOS, iOS y tvOS. La misma API se aplica para el desarrollo multiplataforma! Piensa que es como usar bloques de Lego, puedes usar bloques similares y moverlos fácilmente como un pastel.

<https://github.com/MKGitHub/UIPheonix>

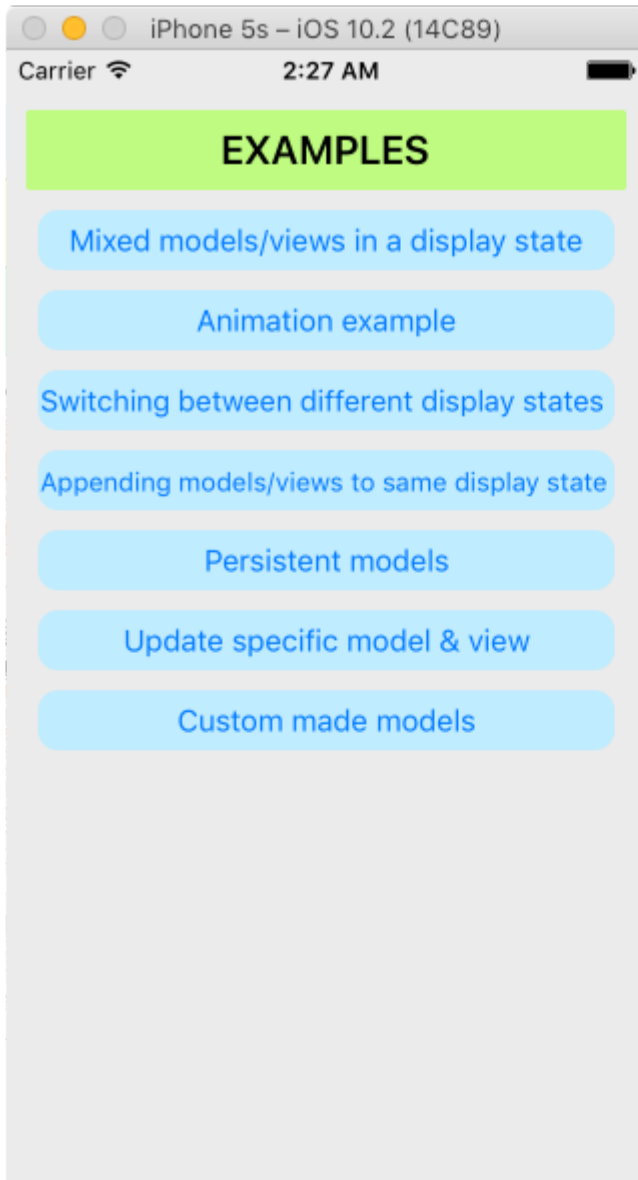
Observaciones

- Olvídense de diseños estáticos, problemas de restricción y explosiones de advertencia en la consola.
- Olvídense de todo el código de pegamento, todo el código de la placa de la caldera y toda la pila innecesaria de código de basura excesivamente diseñada en sus aplicaciones.
- Construye y realiza cambios en tu IU rápidamente en un instante.
- Haga su interfaz de usuario reutilizable.
- Concéntrese en crear su aplicación, no en combatir los problemas de diseño.
- Configuración mínima, impacto mínimo en su aplicación, peso ligero, sin dependencias, sin dolor, ¡y mucho beneficio!
- Se basa en vistas de colección y vistas de tabla, para que pueda mezclar y combinar fácilmente.
- No reemplaza las tecnologías de Apple con implementaciones personalizadas, por lo que siempre estará seguro y actualizado, y podrá revertirlo fácilmente en cualquier momento.
- Aplicaciones de demostración proporcionadas para macOS, iOS y tvOS (Kung Fu!)

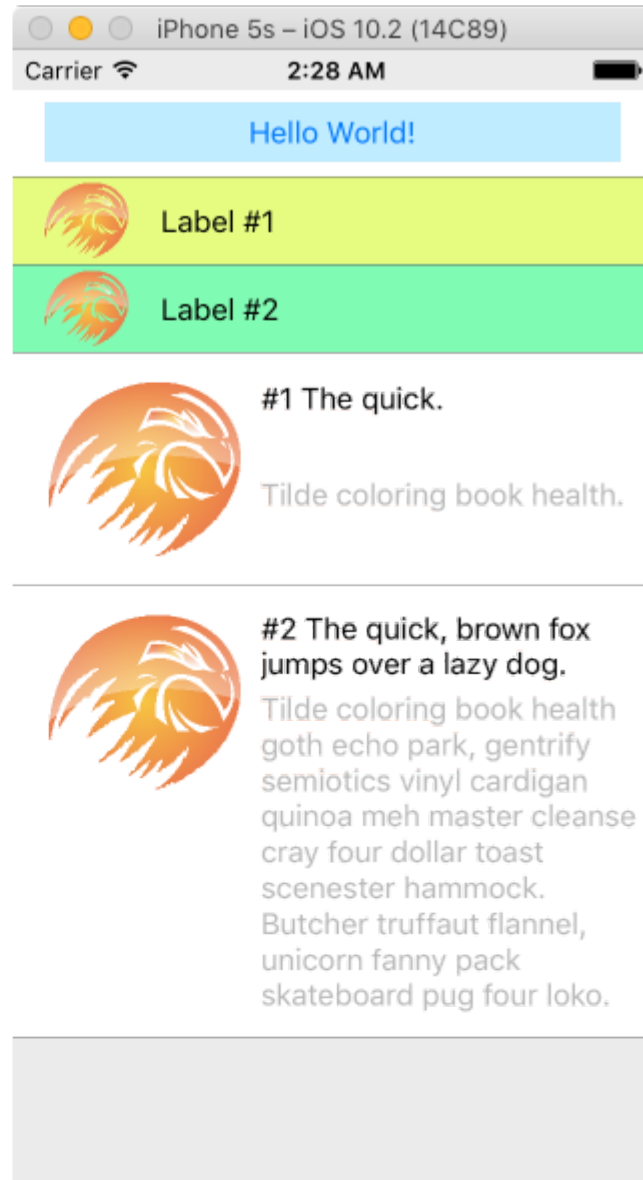
Examples

Ejemplo de componentes de interfaz de usuario

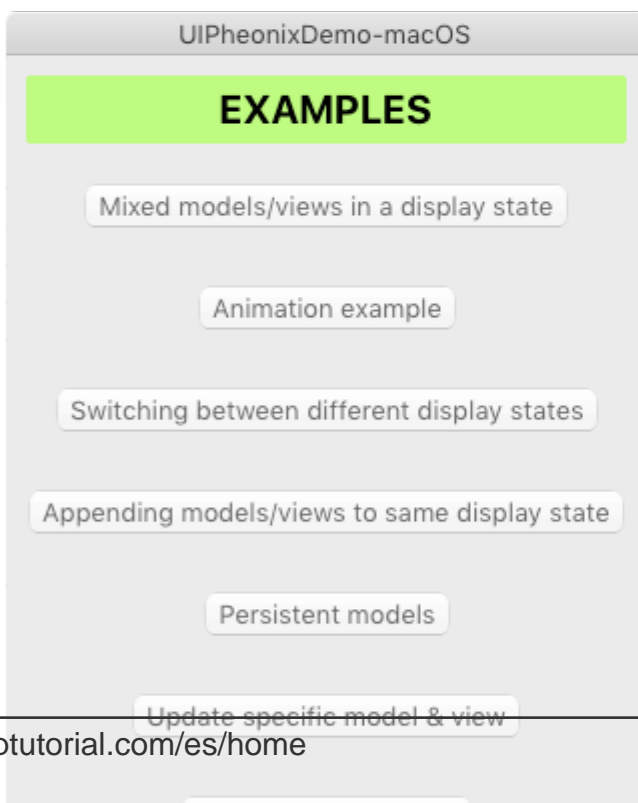
iOS - Collection View



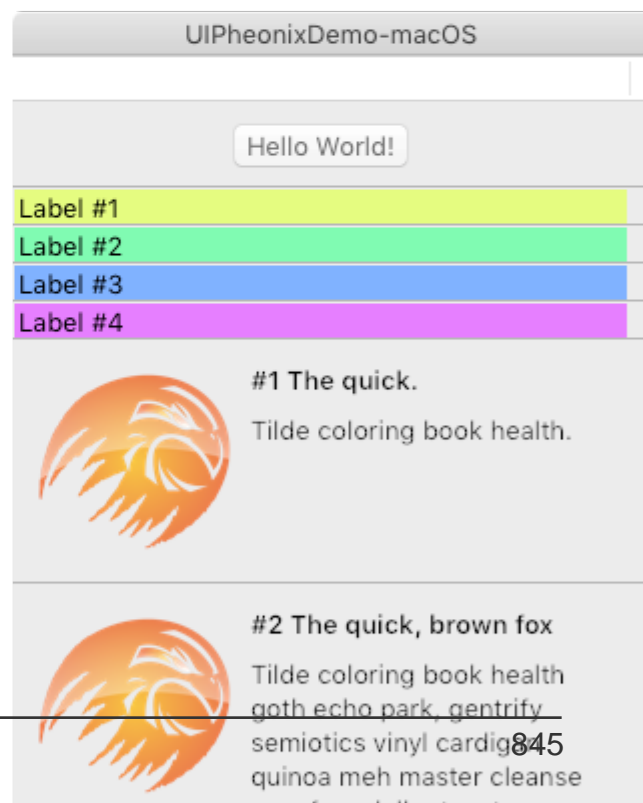
iOS - Table View



macOS - Collection View



macOS - Table View



<https://riptutorial.com/es/ios/topic/9120/uipeonix--marco-de-iu-facil--flexible--dinamico-y-altamente-escalable>

Capítulo 181: UIPickerViewView

Examples

Ejemplo basico

Rápido

```
class PickerViewExampleViewController : UIViewController, UIPickerViewDelegate,
UIPickerViewDataSource {
    @IBOutlet weak var btnFolder: UIButton!
    let pickerView = UIPickerView()
    let pickerViewRows = ["First row,", "Secound row,", "Third row,", "Fourth row"]

    override func viewDidLoad() {
        super.viewDidLoad()
        self.btnFolder.addTarget(self, action: #selector(CreateListVC.btnFolderPress),
forControlEvents: UIControlEvents.TouchUpInside)
    }

    @objc private func btnFolderPress() {
        self.pickerView.delegate = self
        self.pickerView.dataSource = self
        self.view.addSubview(self.pickerView)
    }

    //MARK: UIPickerViewDelegate

    func pickerView(pickerView: UIPickerView, titleForRow row: Int, forComponent component:
Int) -> String? {
        return self.pickerViewRows[row]
    }

    //MARK: UIPickerViewDataSource

    func numberOfComponentsInPickerView(pickerView: UIPickerView) -> Int {
        return 1
    }

    func pickerView(pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
        return self.pickerViewRows.count
    }
}
```

C objetivo

```
@property (nonatomic, strong) UIPickerView *countryPicker;
@property (nonatomic, strong) NSArray *countryNames;

- (void)viewDidLoad {
```



```

    [super viewDidLoad];
    _countryNames = @[@"Australia (AUD)", @"China (CNY)",
                    @"France (EUR)", @"Great Britain (GBP)", @"Japan (JPY)", @"INDIA
(IN)", @"AUSTRALIA (AUS)", @"NEW YORK (NW)"];

    [self pickcountry];
}

-(void)pickcountry {
    _countryPicker = [[UIPickerView alloc] init];

    _countryPicker.delegate = self;
    _countryPicker.dataSource = self;

    [[UIPickerView appearance] setBackgroundColor:[UIColor colorWithRed:21/255.0
green:17/255.0 blue:50/255.0 alpha:1.0]];
}

#pragma mark- pickerView Delegates And datasource

- (NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerView {
    return 1;
}

- (NSInteger)pickerView:(UIPickerView *)pickerView
numberOfRowsInComponent:(NSInteger)component {
    return _countryNames.count;
}

- (NSString *)pickerView:(UIPickerView *)pickerView
titleForRow:(NSInteger)row
forComponent:(NSInteger)component {
    return _countryNames[row];
}

- (void)pickerView:(UIPickerView *)pickerView didSelectRow:(NSInteger)row
inComponent:(NSInteger)component {
    NSString *pickedCountryName = _countryNames[row];
}

```

Cambiando pickerView Color de fondo y color de texto

C objetivo

```

//Displays the country pickerView with black background and white text
[self.countryPicker setValue:[UIColor whiteColor] forKey:@"textColor"];
[self.countryPicker setValue:[UIColor blackColor] forKey:@"backgroundColor"];

```

Rápido

```

let color1 = UIColor(colorLiteralRed: 1, green: 1, blue: 1, alpha: 1)
let color2 = UIColor(colorLiteralRed: 0, green: 0, blue: 0, alpha: 1)
pickerView2.setValue(color1, forKey: "textColor")
pickerView2.setValue(color2, forKey: "backgroundColor")

```

Lea UIPickerView en línea: <https://riptutorial.com/es/ios/topic/4242/uipickerview>

Capítulo 182: UIRefreshControl TableView

Introducción

Un objeto `UIRefreshControl` proporciona un control estándar que puede utilizarse para iniciar la actualización de los contenidos de una vista de tabla. Usted vincula un control de actualización a una tabla a través de un objeto controlador de vista de tabla asociado. El controlador de vista de tabla maneja el trabajo de agregar el control a la apariencia visual de la tabla y administrar la visualización de ese control en respuesta a los gestos apropiados del usuario.

Examples

Objective-C Ejemplo

Primero declara una propiedad como esta en el `ViewController`

```
@property (nonatomic) UIRefreshControl *refreshControl;
```

Más adelante, en `viewDidLoad()` configure `refreshControl` como se indica a continuación:

```
self.refreshControl = [[UIRefreshControl alloc] init];
[self.tableView addSubview:self.refreshControl];
[self.refreshControl addTarget:self action:@selector(refreshTable)
forControlEvents:UIControlEventValueChanged];
//Setting the tint Color of the Activity Animation
self.refreshControl.tintColor = [UIColor redColor];
//Setting the attributed String to the text
NSMutableAttributedString * string = [[NSMutableAttributedString alloc]
initWithString:@"firstsecondthird"];
[string addAttribute:NSForegroundColorAttributeName value:[UIColor redColor]
range:NSMakeRange(0, 5)];
[string addAttribute:NSForegroundColorAttributeName value:[UIColor greenColor]
range:NSMakeRange(5, 6)];
[string addAttribute:NSForegroundColorAttributeName value:[UIColor blueColor]
range:NSMakeRange(11, 5)];
self.refreshControl.attributedTitle = string;
```

Ahora la función `refreshTable` se define como:

```
- (void)refreshTable {
    //TODO: refresh your data
    [self.refreshControl endRefreshing];
    [self.refreshControl beginRefreshing];
    [self.tableView reloadData];
    [self.refreshControl endRefreshing];
}
```



Configurar refreshControl en tableView:

```
UIRefreshControl *refreshControl = [[UIRefreshControl alloc] init];
[refreshControl addTarget:self action:@selector(pullToRefresh:)
forControlEvents:UIControlEventValueChanged];
self.scrollView.alwaysBounceVertical = YES;
[self.scrollView addSubview:refreshControl];

- (void)pullToRefresh:(UIRefreshControl*) sender{
//Do work off the main thread
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
// Simulate network traffic (sleep for 2 seconds)
[NSThread sleepForTimeInterval:2];
//Update data
//Call complete on the main thread
dispatch_sync(dispatch_get_main_queue(), ^{
//Update network activity UI
NSLog(@"COMPLETE");
[sender endRefreshing];
});
});
}
```

Lea UIRefreshControl TableView en línea: <https://riptutorial.com/es/ios/topic/8278/uirefreshcontrol-tableview>

Capítulo 183: UIScrollView

Examples

Crear un UIScrollView

Cree una instancia de `UIScrollView` con un `CGRect` como marco.

Rápido

```
let scrollView = UIScrollView.init(frame: CGRect(x: 0, y: 0, width: 320, height: 400))
```

C objetivo

```
UIScrollView *scrollView = [[UIScrollView alloc] initWithFrame:CGRectMake(0, 0, 320, 400)];
```

Tamaño de contenido de la vista de desplazamiento

La propiedad `contentSize` debe establecer en el tamaño del contenido desplazable. Esto especifica el tamaño del área desplazable. El desplazamiento es visible cuando el área desplazable, es decir, `contentSize` es más grande que el tamaño del marco `UIScrollView`.

Con Autolayout:

Cuando el contenido de la vista de desplazamiento se configura mediante la reproducción automática, debe tener un tamaño explícito tanto vertical como horizontalmente y tener los 4 bordes fijados a la vista de desplazamiento que contiene. De esa manera, `contentSize` se calcula automáticamente en función del contenido de la vista de desplazamiento y también se actualiza cuando se modifica el diseño del contenido.

A mano:

Rápido

```
scrollView.contentSize = CGSize(width: 640, height: 800)
```

C objetivo

```
scrollView.contentSize = CGSizeMake(640, 800);
```

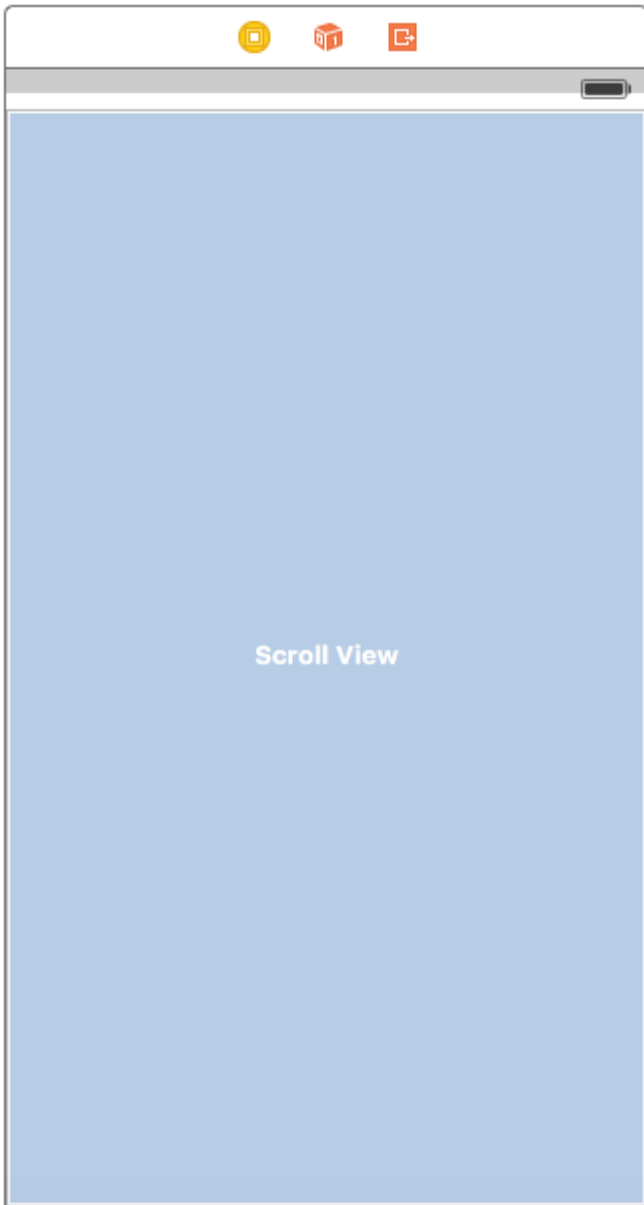
ScrollView con AutoLayout

Pasos simples para usar `scrollView` con `autolayout`.

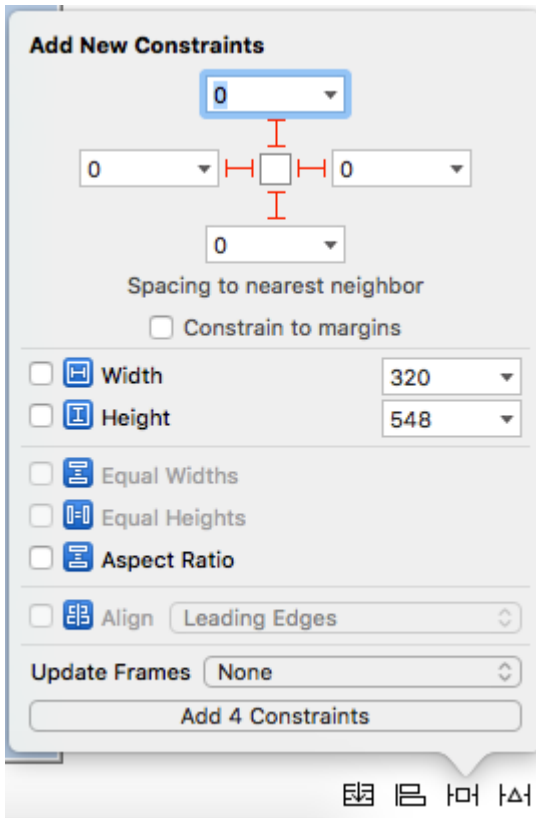
- Crear un nuevo proyecto con aplicación de vista única.
- Seleccione el controlador de vista predeterminado y cambie su tamaño de pantalla a

iPhone-4 pulgadas desde el inspector de atributos.

- Agregue una vista de desplazamiento a la vista del controlador de vista de la siguiente manera y establezca el color de fondo en azul



- Añadir restricciones en él como se muestra en la imagen de abajo



Lo que esto hará es, simplemente pegar cada borde de la vista de desplazamiento a la vista del controlador de vista

Escenario 1:

Ahora digamos que nuestro contenido es enorme, y queremos que se desplace horizontal y verticalmente.

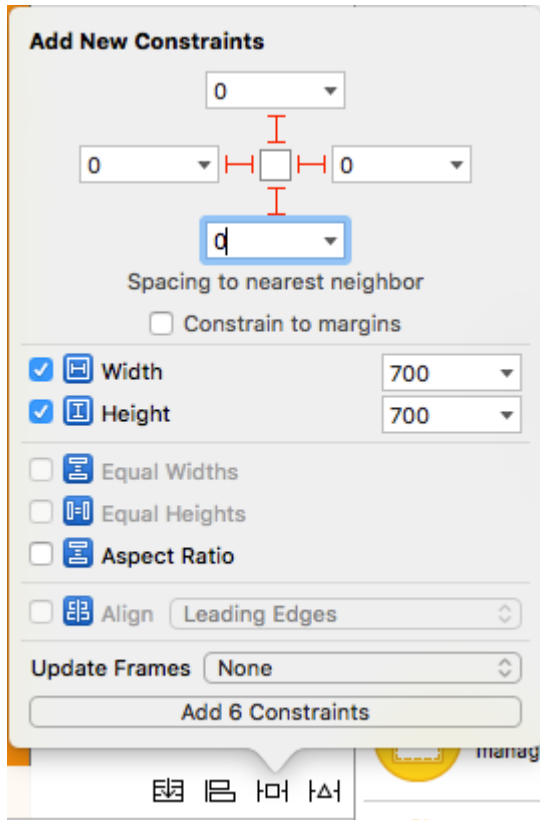
Para esto,

- Agregue un UIView a la vista de desplazamiento del marco (0,0,700,700). Le permite darle un color de fondo naranja para identificarlo de manera diferente.



Luego viene la parte importante, la necesitamos para desplazarnos horizontal y verticalmente.

- Seleccione la vista naranja y agregue las siguientes restricciones



Déjame explicarte lo que hicimos en el paso anterior.

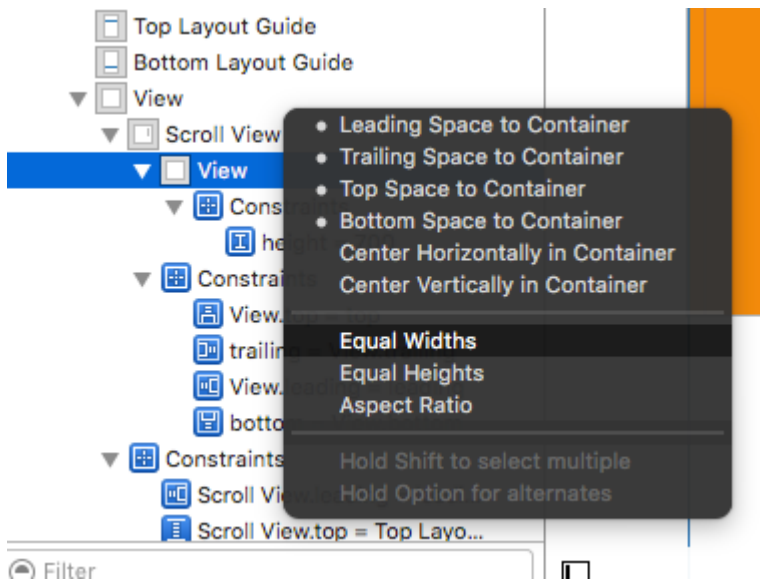
- Fijamos la altura y el ancho a 700.
- Establecemos un espacio al final para scrollbar = 0 que le dice a scrollbar que el contenido se puede desplazar horizontalmente.
- Configuramos el espacio inferior para scrollbar = 0, que le dice a scrollbar que el contenido se puede desplazar verticalmente.

Ahora ejecuta el proyecto y comprueba.

Escenario 2: Consideremos un escenario en el que sabemos que el ancho del contenido será igual al ancho del desplazamiento, pero la altura es mayor que la vista del desplazamiento.

Siga los pasos para desplazar el contenido verticalmente.

- Eliminar la restricción de ancho en el caso anterior.
- Cambie el ancho de la vista naranja para que coincida con el ancho de la vista de desplazamiento.
- Presione la tecla Ctrl y arrastre desde la vista naranja a la vista de desplazamiento y agregue restricciones de **ancho igual** .



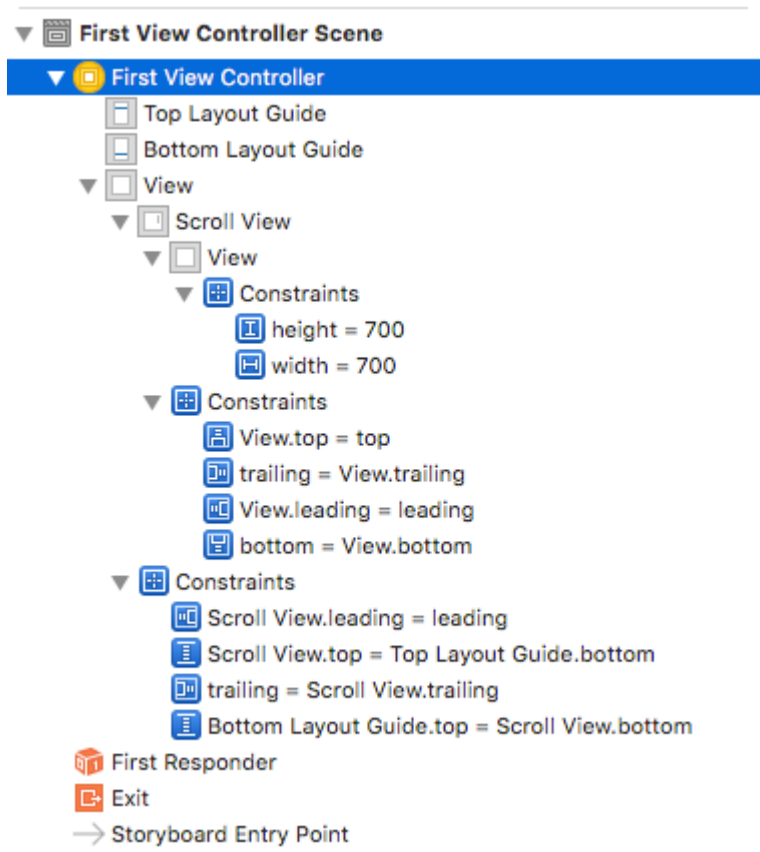
- ¡¡¡Y hecho!!! Simplemente ejecute y verifique si se desplaza verticalmente

Escenario 3:

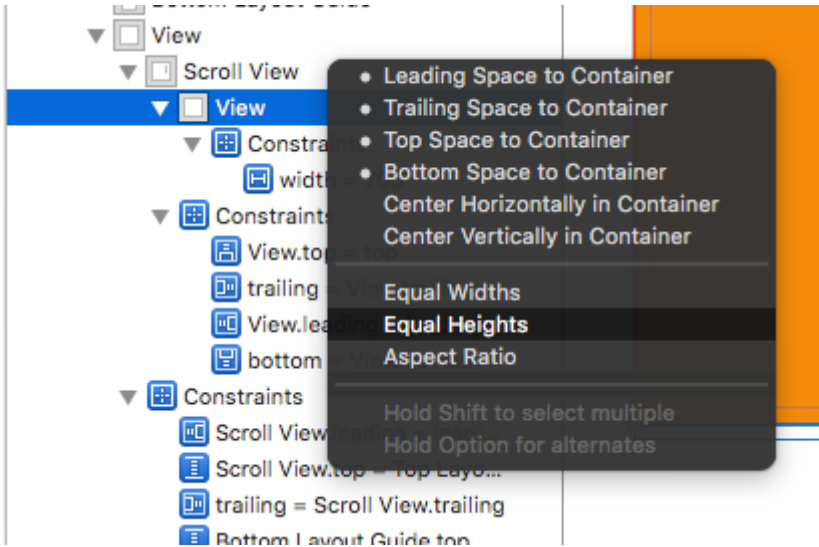
Ahora queremos desplazarnos solo horizontalmente y no verticalmente.

Sigue los pasos para desplazarte horizontalmente por el contenido.

- Deshaga todos los cambios para lograr restricciones como se indica a continuación (es decir, **restaurar las restricciones originales que lograron el desplazamiento vertical y horizontal**)



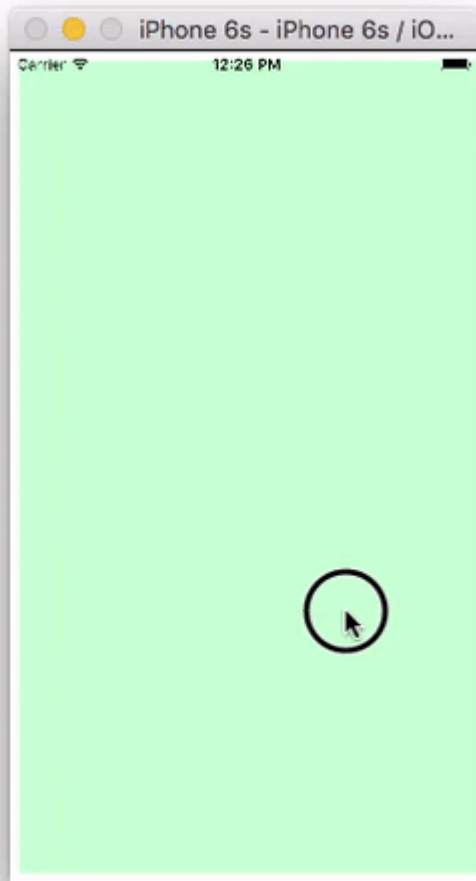
- Verifique el marco de la vista naranja, que debería ser (0,0,700,700)
- Eliminar la restricción de altura de la vista naranja.
- Cambie la altura de la vista naranja para que coincida con la altura de la vista de desplazamiento.
- Presione la tecla Ctrl y arrastre desde la vista naranja a la vista de desplazamiento y agregue **una** restricción de **alturas iguales** .



- ¡¡¡Y hecho!!! Simplemente ejecute y verifique si se desplaza verticalmente

Desplazar contenido con diseño automático habilitado

Este proyecto es un ejemplo autocontenido hecho completamente en el Interface Builder. Debes poder trabajar en 10 minutos o menos. Luego puedes aplicar los conceptos que aprendiste a tu propio proyecto.



Aquí solo uso `UIView` s pero pueden representar cualquier vista que te guste (es decir, botón, etiqueta, etc.). También elegí el desplazamiento horizontal porque las capturas de pantalla del guión gráfico son más compactas para este formato. Sin embargo, los principios son los mismos para el desplazamiento vertical.

Conceptos clave

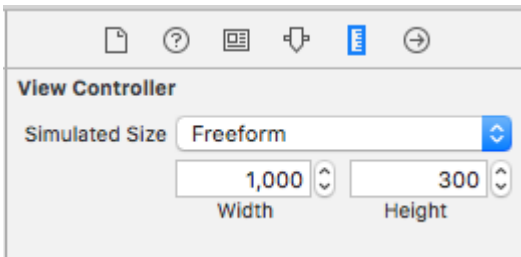
- El `UIScrollView` solo debe usar una subvista. Esta es una 'Vista UIV' que sirve como vista de contenido para contener todo lo que deseas desplazar.
- Haga que la vista de contenido y el *padre* de la vista de desplazamiento tengan alturas iguales para el desplazamiento horizontal. (Anchos iguales para desplazamiento vertical)
- Asegúrese de que todo el contenido desplazable tenga un ancho definido y esté anclado en todos los lados.

Iniciar un nuevo proyecto

Puede ser solo una aplicación de vista única.

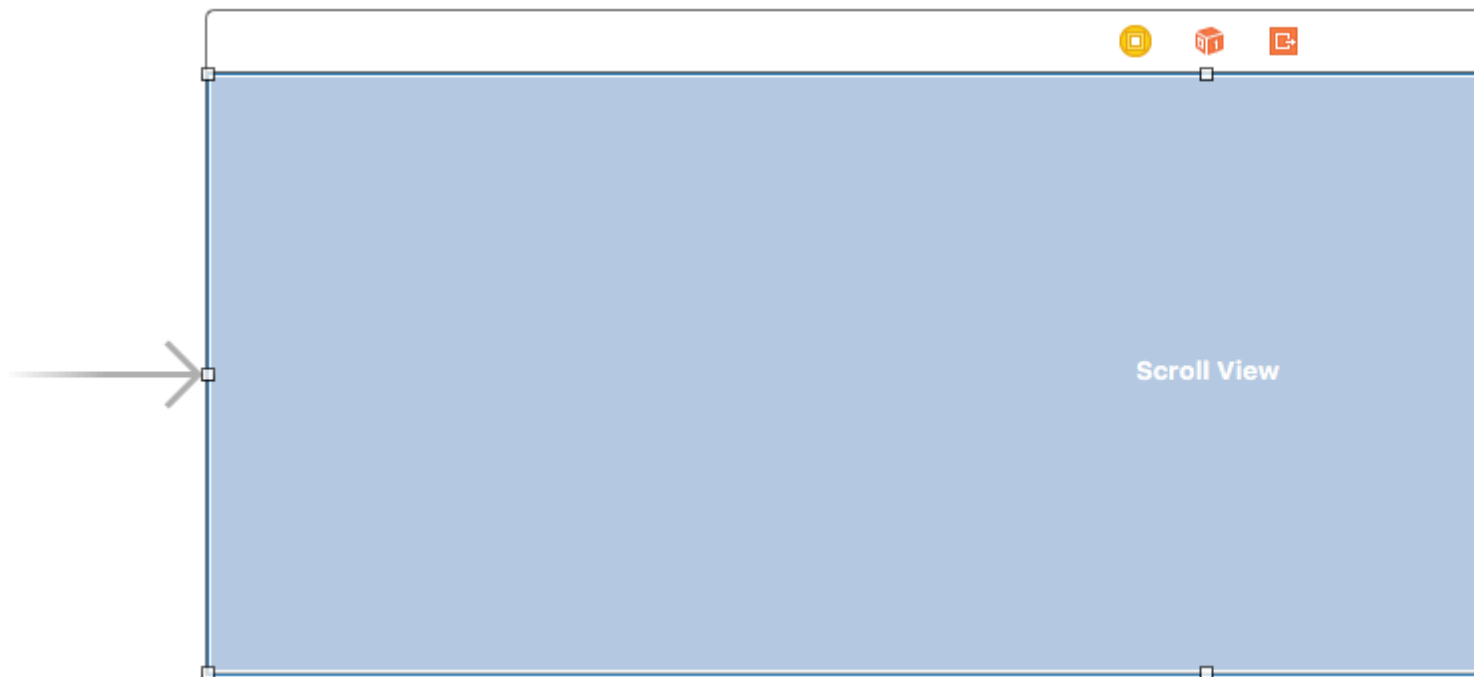
Guión gráfico

En este ejemplo, haremos una vista de desplazamiento horizontal. Seleccione el Controlador de vista y luego elija Forma libre en el Inspector de tamaño. Hacer el ancho 1,000 y la altura 300 . Esto solo nos da el espacio en el guión gráfico para agregar contenido que se desplazará.



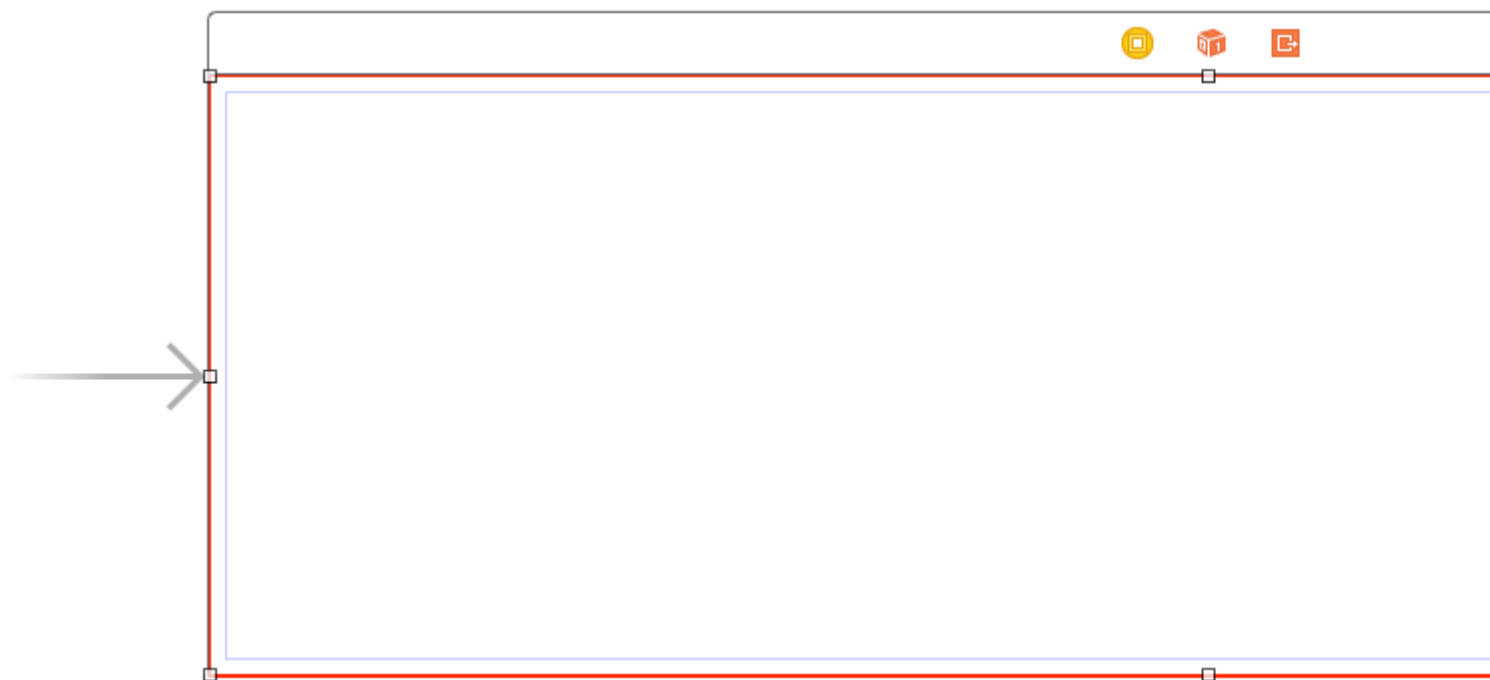
Añadir una vista de desplazamiento

Agregue un `UIScrollView` y fije los cuatro lados a la vista raíz del controlador de vista.



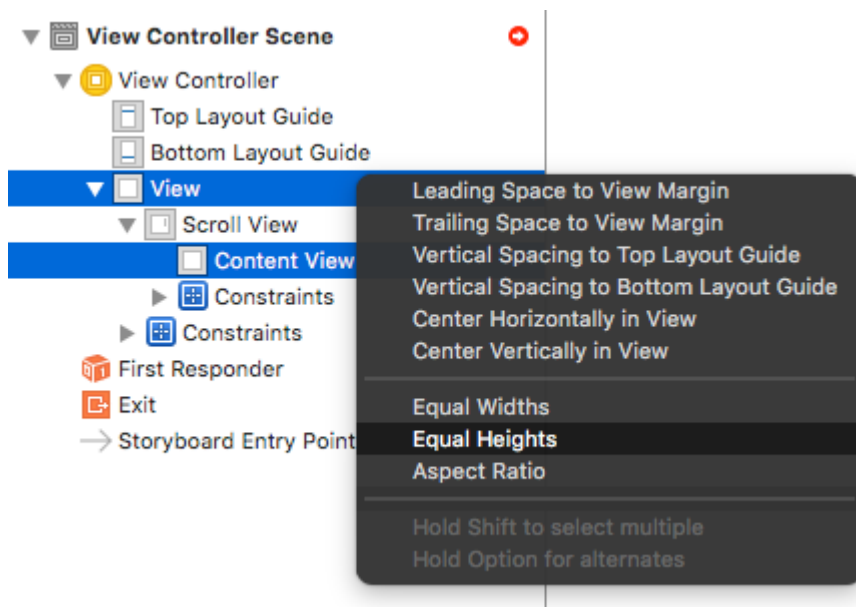
Añadir una vista de contenido

Agregue un `UIView` como una subvista a la vista de desplazamiento. *Esta es la clave.* No intente agregar muchas subvistas a la vista de desplazamiento. Sólo agregue un solo `UIView` . Esta será su vista de contenido para las otras vistas que desea desplazar. Fije la vista de contenido a la vista de desplazamiento en los cuatro lados.



Alturas iguales

Ahora en el Esquema del documento, **Comando** haga clic en la vista de contenido y en la vista *principal* de la vista de desplazamiento para seleccionarlos. Luego, establezca las alturas para que sean iguales (**Control** </ kbd arrastre desde la Vista de contenido a la Vista de desplazamiento>). *Esto también es clave*. Debido a que nos estamos desplazando horizontalmente, la vista de contenido de la vista de desplazamiento no sabrá qué tan alta debe ser a menos que la configuremos de esta manera.

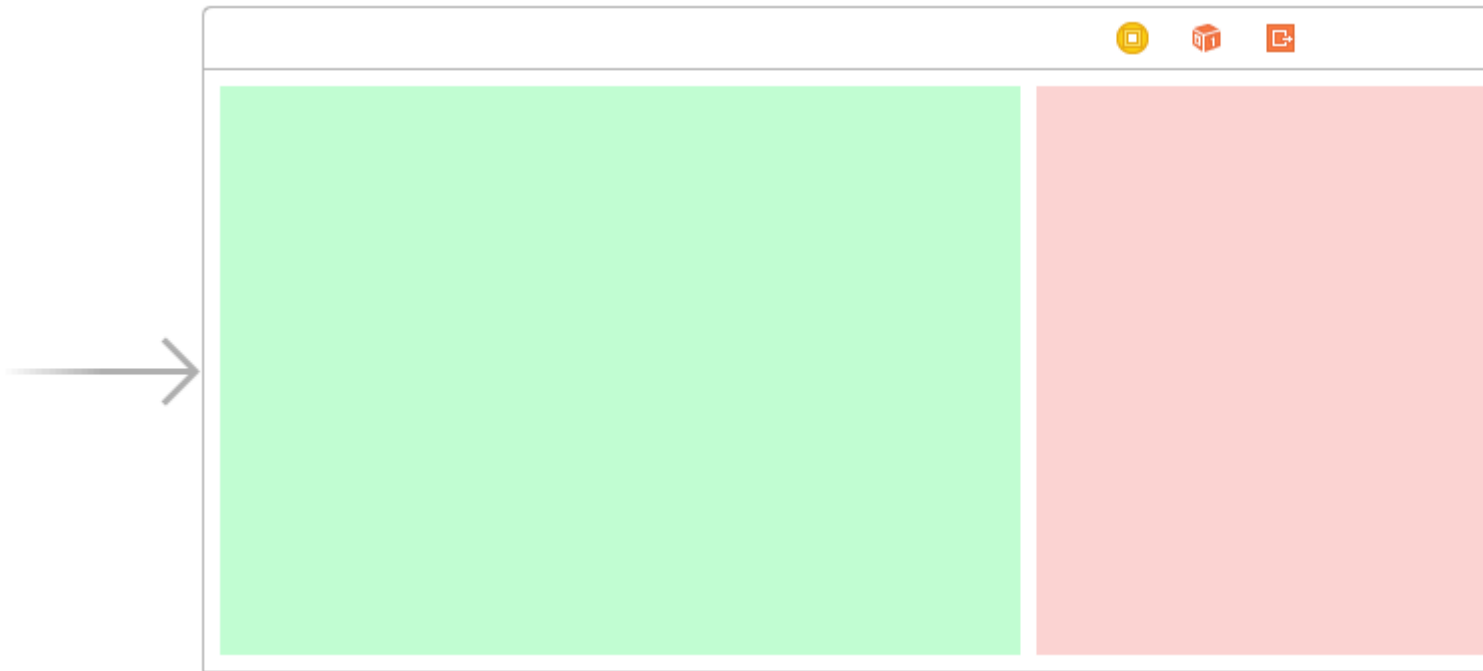


Nota:

- Si estuviéramos haciendo que el contenido se desplace verticalmente, entonces estableceríamos el ancho de la vista de contenido para que sea igual al ancho del elemento primario de la vista de desplazamiento.

Agregar contenido

Agrega tres `UIView`s y dales todas las restricciones. Utilicé márgenes de 8 puntos para todo.



Restricciones:

- Vista verde: fija los bordes superior, izquierdo e inferior. Hacer el ancho 400.
- Vista roja: fija los bordes superior, izquierdo e inferior. Hacer el ancho 300.
- Vista morada: pin los cuatro bordes. Haz el ancho cualquiera que sea el espacio restante (268 en este caso).

Establecer las restricciones de ancho también es clave para que la vista de desplazamiento sepa cuán amplia será la vista de contenido.

Terminado

Eso es todo. Puede ejecutar su proyecto ahora. Debe comportarse como la imagen de desplazamiento en la parte superior de esta respuesta.

Estudio adicional

- [iOS: cómo hacer que AutoLayout funcione en un UIScrollView](#)
- [Cómo configurar un UIScrollView con diseño automático en Interface Builder](#)
- Video tutorial de YouTube: [UIScrollView - Cómo mantener sus vistas en pantalla](#)

Activar / Desactivar desplazamiento

La propiedad `scrollEnabled` almacena un valor `Boolean` que determina si el desplazamiento está habilitado o no.

Si el valor de esta propiedad es verdadero / Sí, el desplazamiento está habilitado, de lo contrario no. El valor predeterminado es `true`

Rápido

```
scrollview.isScrollEnabled = true
```

C objetivo

```
scrollview.scrollEnabled = YES;
```

Zoom In / Out UImageVIew

Crear instancia de UIScrollView

```
let scrollview = UIScrollView.init(frame: self.view.bounds)
```

Y luego establecer estas propiedades:

```
scrollView.minimumZoomScale = 0.1
scrollView.maximumZoomScale = 4.0
scrollView.zoomScale = 1.0
scrollview.delegate = self as? UIScrollViewDelegate
```

Para acercar y alejar la imagen, debemos especificar la cantidad que el usuario puede acercar y alejar. Hacemos esto configurando los valores de las propiedades `minimumZoomScale` y `maximumZoomScale` de la vista de desplazamiento. Ambos están configurados en 1.0 por defecto.

Y `zoomScale` a 1.0, que especifica el factor de zoom para el zoom mínimo y máximo.

Para admitir el zoom, debemos establecer un delegado para su vista de desplazamiento. El objeto delegado debe cumplir con el protocolo `UIScrollViewDelegate`. Esa clase delegada debe implementar el método `viewForZoomingInScrollView()` y devolver la vista a zoom.

Modifique su `ViewController` como se muestra

```
class ViewController: UIViewController, UIScrollViewDelegate
```

Luego agregue la siguiente función de delegado a la clase.

```
func viewForZoomingInScrollView(scrollView: UIScrollView) -> UIView? {
    return imageView
}
```

Ahora crea la instancia de UImageVIew

Hacer esta variable como variable de clase

```
var imageView:UIImageView = UIImageView.init(image: UIImage.init(named: "someImage.jpg"))
```

Y luego agregarlo a scrollView

```
scrollView?.addSubview(imageView)
```

Referencia

- [Guía de programación Scroll View para iOS](#)
- [Tutorial de UIScrollView](#)

Detectar cuándo UIScrollView terminó de desplazarse con métodos delegados

scrollViewDidEndDecelerating: esto le dice al delegado que la vista de desplazamiento ha terminado de desacelerar el movimiento de desplazamiento.

C objetivo:

```
- (void)scrollViewDidEndDecelerating:(UIScrollView *)scrollView {
    [self stoppedScrolling];
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate {
    if (!decelerate) {
        [self stoppedScrolling];
    }
}

- (void)stoppedScrolling {
    // done, do whatever
}
```

Rápido:

```
func scrollViewDidEndDragging(scrollView: UIScrollView, willDecelerate decelerate: Bool) {
    if !decelerate {
        stoppedScrolling()
    }
}

func scrollViewDidEndDecelerating(scrollView: UIScrollView) {
    stoppedScrolling()
}

func stoppedScrolling() {
    // done, do whatever
}
```


Restringir la dirección de desplazamiento

Puede restringir las direcciones a las que el usuario puede desplazarse utilizando el siguiente código:

```
func scrollViewDidScroll(_ scrollView: UIScrollView) {
    if scrollView.contentOffset.x != 0 {
        scrollView.contentOffset.x = 0
    }
}
```

Cada vez que el usuario se desplaza en el eje x, el desplazamiento del contenido de scrollView se establece en 0.

Obviamente, puede cambiar de x s a y s y, por lo tanto, bloquear la dirección para que sea solo horizontal.

También debe asegurarse de colocar este código en el método delegado `scrollViewDidScroll(_ scrollView: UIScrollView)`. De lo contrario, no conseguirás que funcione.

Además, asegúrese de haber importado el `UIScrollViewDelegate` en su declaración de clase, así:

```
class ViewController: UIViewController, UIScrollViewDelegate
```

... y configura el delegado de scrollView a self en algún método como `viewDidLoad(_:)`

```
scrollView.delegate = self
```

Lea `UIScrollView` en línea: <https://riptutorial.com/es/ios/topic/1575/uiscrollview>

Capítulo 184: UIScrollView AutoLayout

Examples

Controlador de desplazamiento

Cuando se utiliza Autolayout con un `UIScrollView`, NO se redimensiona correctamente dependiendo del tamaño de su contenido o subvistas.

Para que un `UIScrollView` se desplace automáticamente cuando su contenido sea demasiado grande para ajustarse al área visible, debemos agregar un `ContentView` y algunas restricciones que permitan al `UIScrollView` determinar el tamaño de su contenido Y su ancho y alto en su padre. ver.

```
import Foundation
import UIKit

class ScrollableController : UIViewController {

    private var scrollView: UIScrollView!
    private var contentView: UIView!

    override func viewDidLoad() {
        super.viewDidLoad()

        //Setup
        self.initControls()
        self.setTheme()
        self.layoutScrollView()
        self.layoutContentView()

        //Add child views
        self.addChildViews()
    }

    func initControls() {
        self.scrollView = UIScrollView()
        self.contentView = UIView()
    }

    func setTheme() {
        self.scrollView.backgroundColor = UIColor.blue()
        self.contentView.backgroundColor = UIColor.orange()
    }

    func layoutScrollView() {
        self.view.addSubview(self.scrollView)

        let views: NSDictionary = ["scrollView": self.scrollView]
        var constraints = Array<String>()

        //Constrain the scrollView to our controller's self.view.
        constraints.append("H:|-0-[scrollView]-0-|")
        constraints.append("V:|-0-[scrollView]-0-|")

        for constraint in constraints {
```

```

        self.view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutConstraintOptions(rawValue: 0), metrics: nil, views: views as!
[String : AnyObject]))
    }

    self.scrollView.translatesAutoresizingMaskIntoConstraints = false
}

func layoutContentView() {
    self.scrollView.addSubview(self.contentView)

    let views: NSDictionary = ["contentView": self.contentView, "view": self.view]
    var constraints = Array<String>()

    //Constrain the contentView to the scrollView.
    constraints.append("H:|-0-[contentView]-0-|")
    constraints.append("V:|-0-[contentView]-0-|")

    for constraint in constraints {
        self.scrollView.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutConstraintOptions(rawValue: 0), metrics: nil, views: views as!
[String : AnyObject]))
    }

    //Disable Horizontal Scrolling by making the contentView EqualWidth with our
controller's self.view (ScrollView's parentView).
    self.view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
"H:[contentView(==view)]", options: NSLayoutConstraintOptions(rawValue: 0), metrics: nil, views:
views as! [String : AnyObject]))

    self.contentView.translatesAutoresizingMaskIntoConstraints = false
}

func addChildViews() {
    //Init
    let greenView = UIView()
    let whiteView = UIView()

    //Theme
    greenView.backgroundColor = UIColor.green()
    whiteView.backgroundColor = UIColor.orange()

    //Layout -- Child views are added to the 'ContentView'
    self.contentView.addSubview(greenView)
    self.contentView.addSubview(whiteView)

    let views: NSDictionary = ["greenView": greenView, "whiteView": whiteView];
    var constraints = Array<String>()

    //Constrain the greenView to the contentView with a height of 400 and 15 spacing all
around.
    constraints.append("H:|-15-[greenView]-15-|")
    constraints.append("V:|-15-[greenView(400)]")

    //Constrain the whiteView below the greenView with 15 spacing all around and a height
of 500.
    constraints.append("H:|-15-[whiteView]-15-|")
    constraints.append("V:[greenView]-15-[whiteView(500)]-15-|")

    for constraint in constraints {
        self.contentView.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:

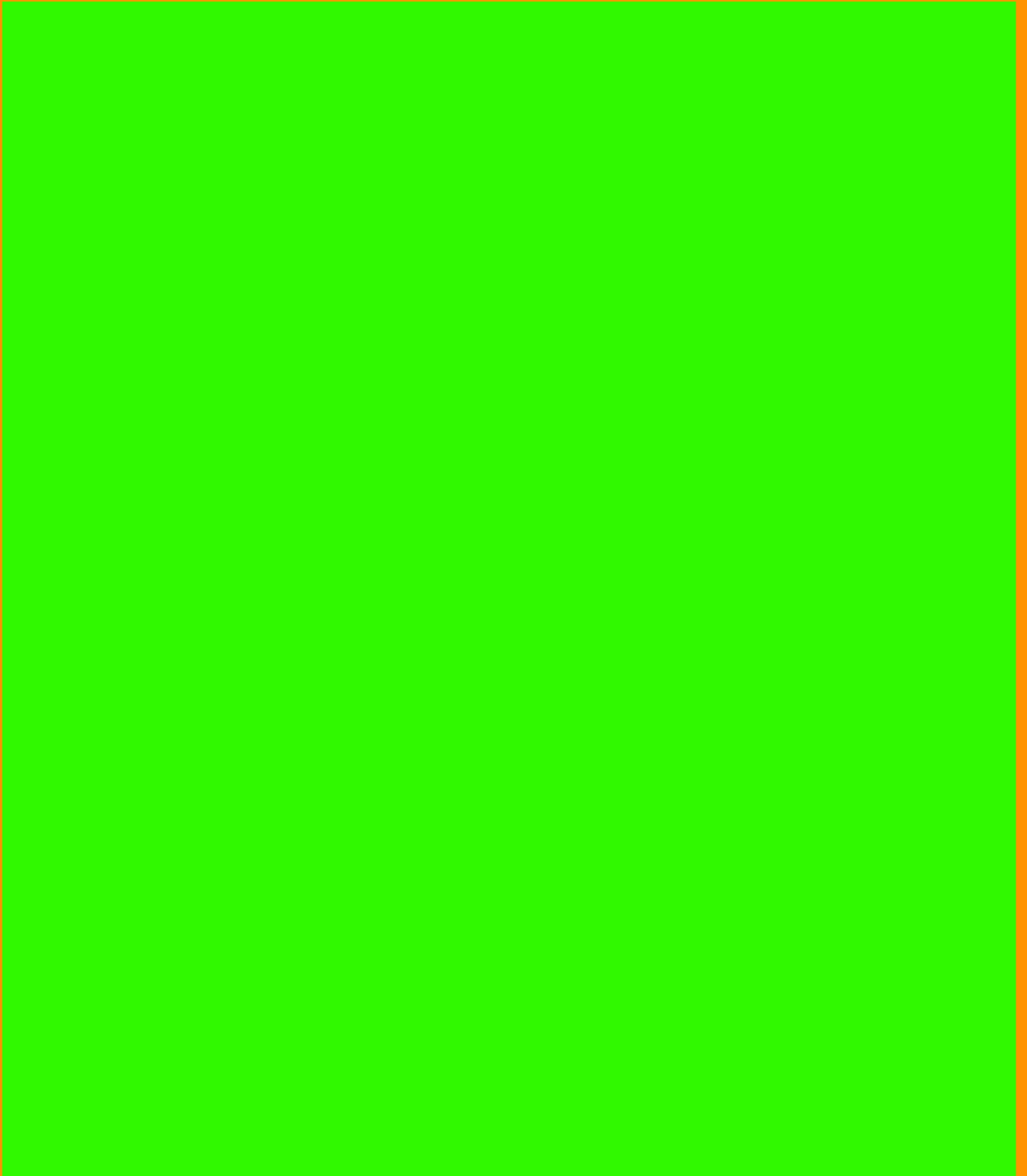
```

```
constraint, options: NSLayoutFormatOptions(rawValue: 0), metrics: nil, views: views as!
[String : AnyObject]))
    }

    greenView.translatesAutoresizingMaskIntoConstraints = false
    whiteView.translatesAutoresizingMaskIntoConstraints = false
}
}
```

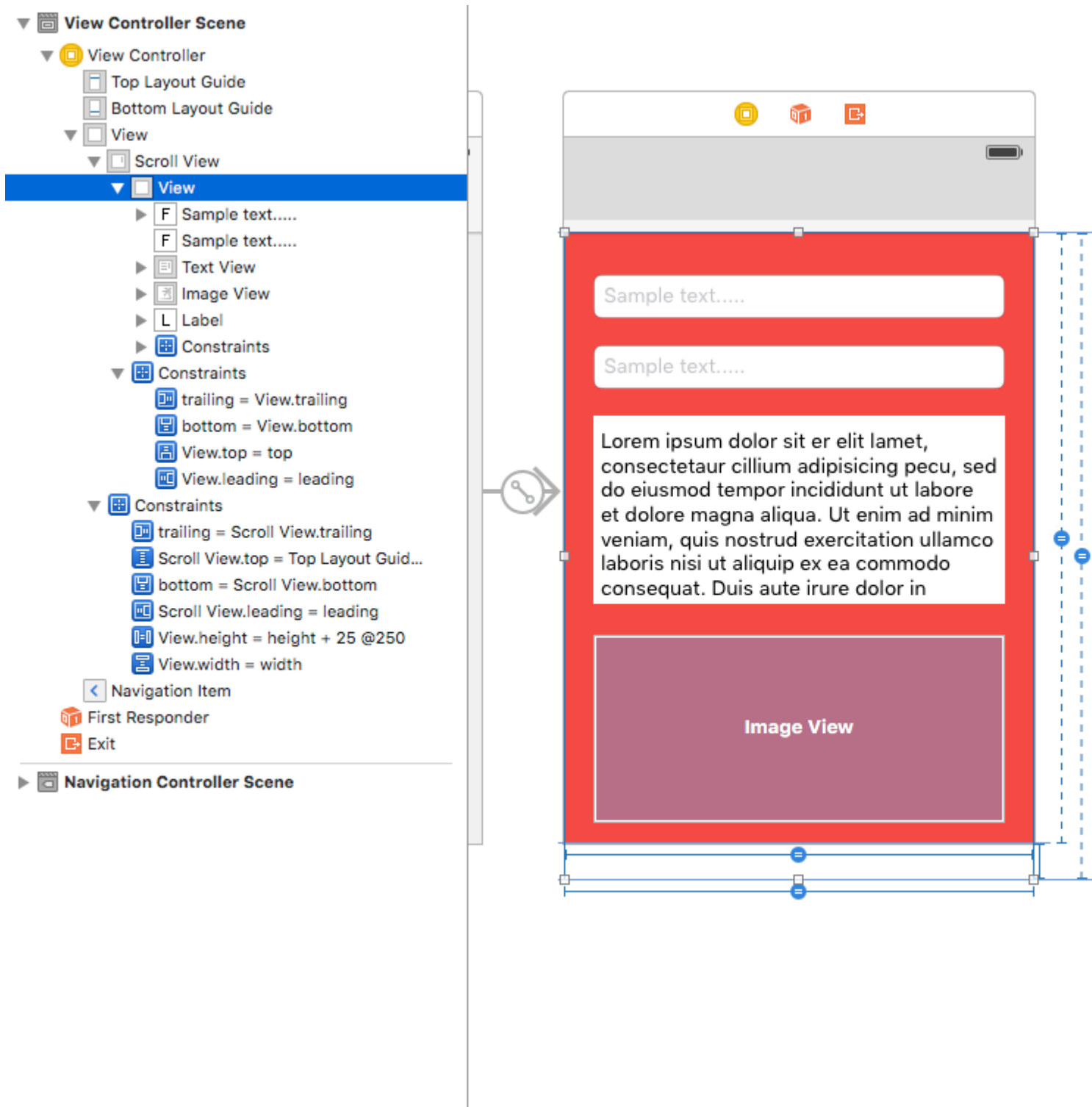
Ahora podemos ver que el `greenView` (altura 400) + el `whiteView` (altura 500) es más grande que nuestra pantalla. Esto hará que el tamaño del contenido de `ScrollView` crezca para adaptarse a AMBAS vistas, lo que le permite desplazarse verticalmente.

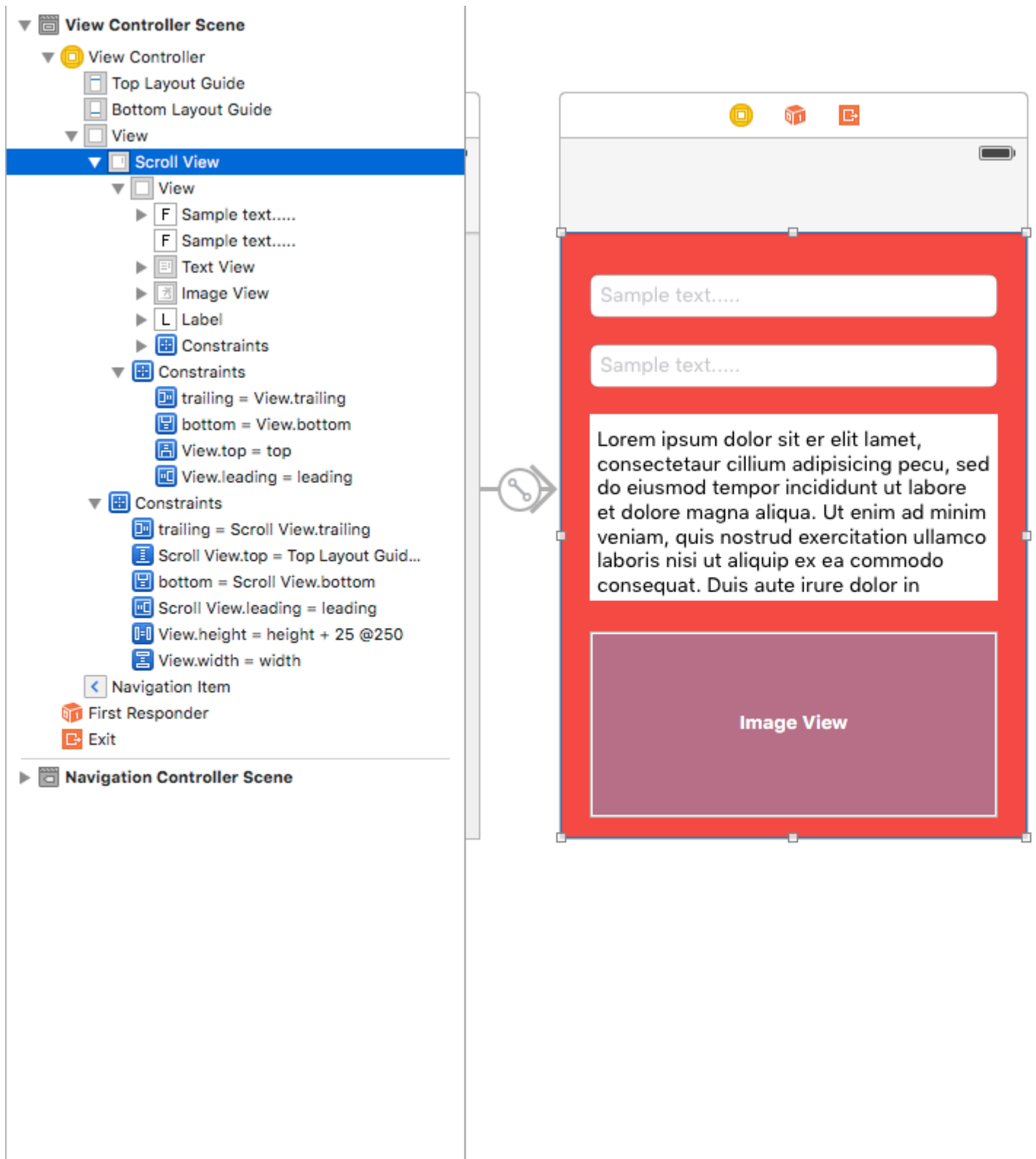
Deshabilitamos el desplazamiento horizontal utilizando la restricción `EqualWidth` en `contentView` y `self.view`



2. Agregue la misma altura y el mismo ancho a la vista principal (es decir, que contiene scrollView). Para una altura igual, establezca la prioridad en baja. (Este es el paso importante para configurar el tamaño del contenido).
3. La altura de esta vista de contenido será de acuerdo con el número de vistas agregadas a la vista. Digamos que si agregó la última vista es una etiqueta y su posición Y es 420 y la altura es 20, entonces su vista de contenido será 440.

Paso 3: agregue restricciones a todas las vistas que agregó dentro de la vista de contenido según sus requisitos.





Lea UIScrollView AutoLayout en línea: <https://riptutorial.com/es/ios/topic/4671/uiscrollview-autolayout>

Capítulo 185: UIScrollView con niño StackView

Examples

Un ejemplo de StackView complejo dentro de Scrollview

A continuación, se muestra un ejemplo de lo que se puede hacer con las StackViews anidadas, dando al usuario la impresión de una experiencia de desplazamiento continuo utilizando elementos de alineación o elementos complejos de la interfaz de usuario.



Prevenir el diseño ambiguo.

Una pregunta frecuente sobre StackViews dentro de Scrollviews proviene de las alertas ambiguas con / heigh en el constructor de interfaces. Como se explica en [esta respuesta](#) , es necesario:

1. Agregue en el UIScrollView una UIView (el contentScrollView);
2. En este contentScrollView, establezca los márgenes superior, inferior, izquierdo y derecho en 0
3. El conjunto también alinea el centro horizontalmente y verticalmente;

Desplazarse hasta el contenido dentro de StackViews anidados

El gran problema del desplazamiento es determinar el desplazamiento necesario para presentar (por ejemplo) un campo de **texto dentro de un StackView dentro de ScrollView** .

Si intentas obtener la **posición de Textfield.frame.minY puede ser 0** , porque el marco minY solo está considerando la distancia entre el elemento y la parte superior de StackView. Así que tienes que **considerar todas las demás vistas / vistas de pila**.

Una buena solución para esto es:

1 - Implementar la extensión ScrollView

```
extension UIScrollView {  
  
    func scrollToShowView(view: UIView){  
        var offset = view.frame.minY  
        var superview = view.superview  
        while((superview != nil)){  
            offset += (superview?.frame.minY)!  
            superview = superview?.superview  
        }  
  
        offset -= 100 //optional margin added on offset  
  
        self.contentOffset = CGPoint.init(x: 0, y: offset)  
    }  
  
}
```

Esto considerará todas las vistas principales y sumará el desplazamiento necesario para que la vista de desplazamiento presente la vista necesaria en la pantalla (por ejemplo, un campo de texto que no pueda quedar detrás del teclado del usuario)

Ejemplo de uso:

```
func textViewDidBeginEditing(_ textView: UITextView) {  
    self.contentOffset.scrollToShowView(view: textView)  
}
```

Lea UIScrollView con niño StackView en línea:

<https://riptutorial.com/es/ios/topic/9404/uiscrollview-con-nino-stackview>

Capítulo 186: UISearchController

Sintaxis

- `UISearchController (searchResultsController: UIViewController?)` // Pase nil como parámetro si el controlador de actualización de búsqueda también muestra el contenido que se puede buscar.
- `func updateSearchResults (para searchController: UISearchController)` // Método requerido para implementar al adoptar el protocolo `UISearchResultsUpdating`

Parámetros

Parámetro	Detalles
<code>UISearchController.searchBar</code>	La barra de búsqueda para instalar en su interfaz. <i>(solo lectura)</i>
<code>UISearchController.searchResultsUpdater</code>	El objeto responsable de actualizar los contenidos del controlador de resultados de búsqueda.
<code>UISearchController.isActive</code>	El estado presentado de la interfaz de búsqueda.
<code>UISearchController.obscuresBackgroundDuringPresentation</code>	Un valor booleano que indica si el contenido subyacente se oculta durante una búsqueda.
<code>UISearchController.dimsBackgroundDuringPresentation</code>	Un valor booleano que indica si el contenido subyacente se atenúa durante una búsqueda.
<code>UISearchController.hidesNavigationBarDuringPresentation</code>	Un valor booleano que indica si la barra de navegación debe estar oculta durante la búsqueda.
<code>UIViewController.definesPresentationContext</code>	Un valor booleano que indica si la vista de este controlador de vista se cubre cuando el controlador de vista o uno de sus descendientes presenta un controlador de vista.
<code>UIViewController.navigationItem.titleView</code>	Una vista personalizada que se muestra en el centro de la barra de navegación cuando el receptor es el elemento superior en el que se

Parámetro	Detalles
	puede colocar una barra de búsqueda.
<code>UITableViewController.tableView.tableHeaderView</code>	Devuelve una vista de accesorios que se muestra sobre la tabla en la que se puede colocar una barra de búsqueda.

Observaciones

Referencia de UIKit Framework:

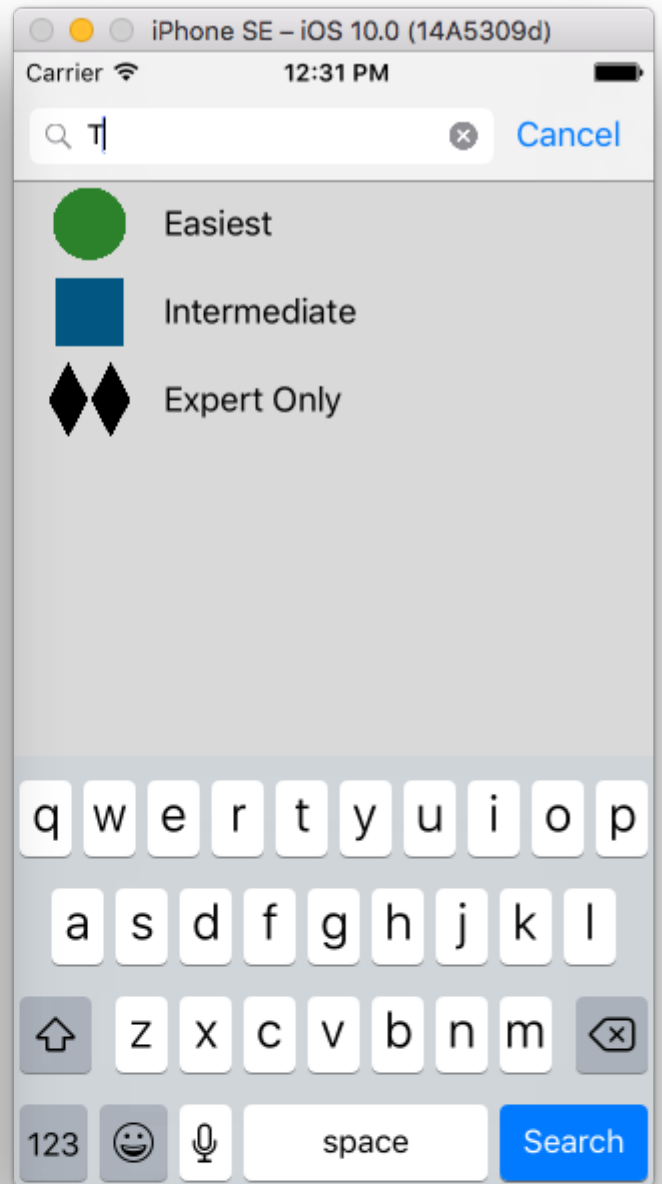
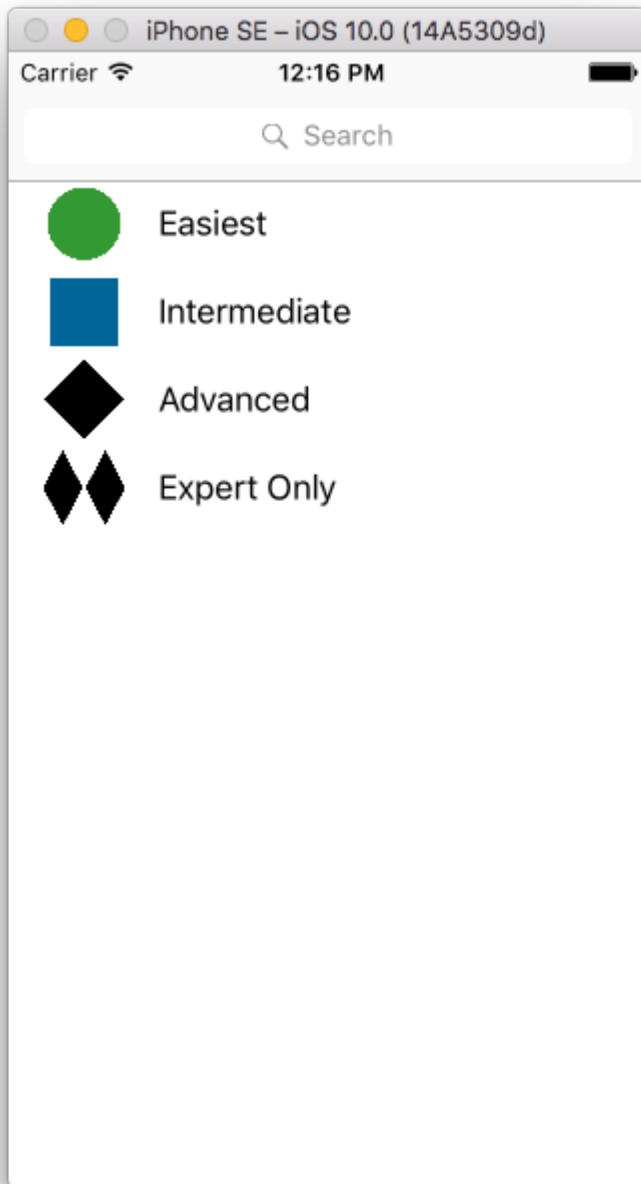
[UISearchController](#)

[UISearchResultsUpdating](#)

Examples

Barra de búsqueda en el título de la barra de navegación

Este ejemplo utiliza un controlador de búsqueda para filtrar los datos dentro de un controlador de vista de tabla. La barra de búsqueda se coloca dentro de la barra de navegación en la que está incrustada la vista de tabla.



UINavigationController (que contiene la barra de navegación). A continuación, configure nuestra clase personalizada ViewController para heredar de UITableViewController y adopte el protocolo UISearchResultsUpdating .

```
class ViewController: UITableViewController, UISearchResultsUpdating {

    let entries = [(title: "Easiest", image: "green_circle"),
                  (title: "Intermediate", image: "blue_square"),
                  (title: "Advanced", image: "black_diamond"),
                  (title: "Expert Only", image: "double_black_diamond")]

    // An empty tuple that will be updated with search results.
    var searchResults : [(title: String, image: String)] = []

    let searchController = UISearchController(searchResultsController: nil)

    override func viewDidLoad() {
        super.viewDidLoad()

        searchController.searchResultsUpdater = self
        self.definesPresentationContext = true

        // Place the search bar in the navigation item's title view.
        self.navigationItem.titleView = searchController.searchBar

        // Don't hide the navigation bar because the search bar is in it.
        searchController.hidesNavigationBarDuringPresentation = false
    }

    func filterContent(for searchText: String) {
        // Update the searchResults array with matches
        // in our entries based on the title value.
        searchResults = entries.filter({ (title: String, image: String) -> Bool in
            let match = title.range(of: searchText, options: .caseInsensitive)
            // Return the tuple if the range contains a match.
            return match != nil
        })
    }

    // MARK: - UISearchResultsUpdating method

    func updateSearchResults(for searchController: UISearchController) {
        // If the search bar contains text, filter our data with the string
        if let searchText = searchController.searchBar.text {
            filterContent(for: searchText)
            // Reload the table view with the search result data.
            tableView.reloadData()
        }
    }

    // MARK: - UITableViewController methods

    override func numberOfSections(in tableView: UITableView) -> Int { return 1 }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        // If the search bar is active, use the searchResults data.
        return searchController.isActive ? searchResults.count : entries.count
    }
}
```

```

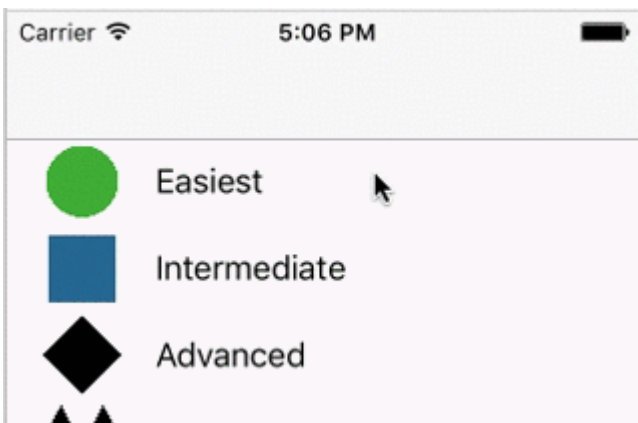
    override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
    UITableViewCell {
        // If the search bar is active, use the searchResults data.
        let entry = searchController.isActive ?
            searchResults[indexPath.row] : entries[indexPath.row]

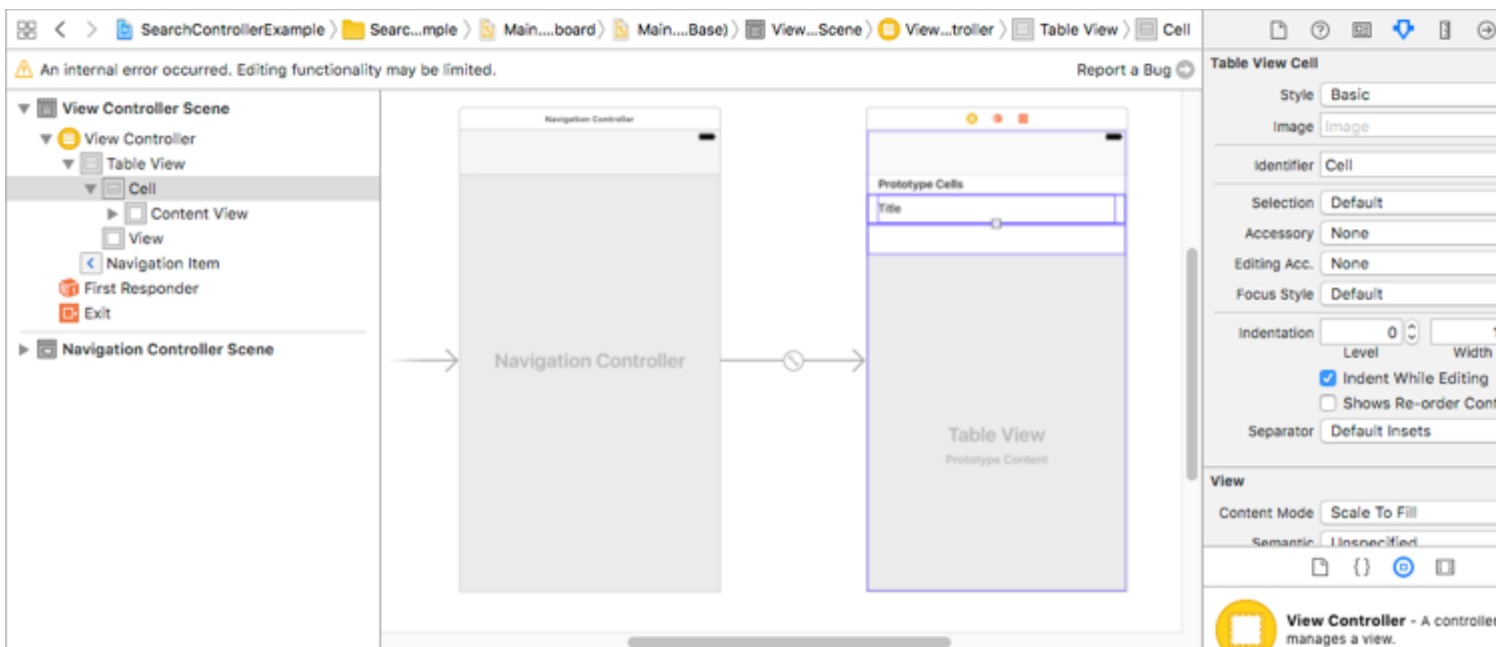
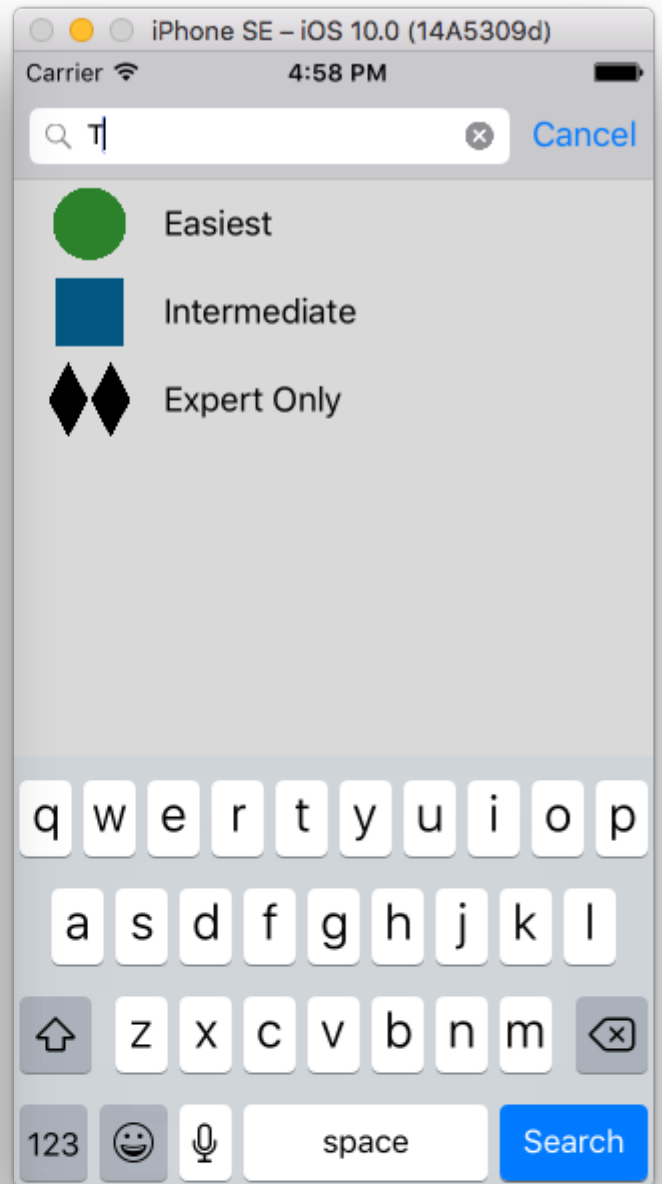
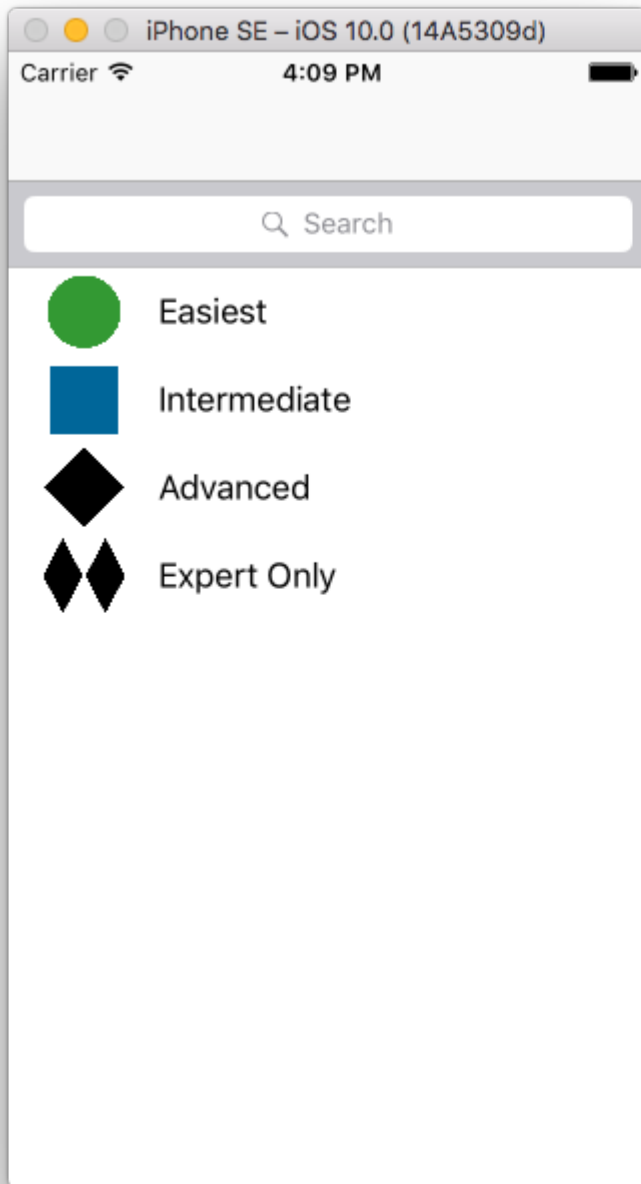
        let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
        cell.textLabel?.text = entry.title
        cell.imageView?.image = UIImage(named: entry.image)
        return cell
    }
}

```

Barra de búsqueda en el encabezado de vista de tabla

Este ejemplo utiliza un controlador de búsqueda para filtrar las celdas en un controlador de vista de tabla. La barra de búsqueda se coloca dentro de la vista de encabezado de la vista de tabla. El contenido de la vista de tabla se desplaza con la misma altura que la barra de búsqueda, de modo que la barra de búsqueda se oculta al principio. Al desplazarse hacia arriba más allá del borde superior de la vista de tabla, se muestra la barra de búsqueda. Luego, cuando la barra de búsqueda se activa, oculta la barra de navegación.





que proviene del protocolo `UISearchResultsUpdating` :

```
func updateSearchResultsForSearchController(searchController: UISearchController) {  
  
}
```

UISerachController en Objective-C

```
Delegate: UISearchBarDelegate, UISearchControllerDelegate, UISearchBarDelegate  
  
@property (strong, nonatomic) UISearchController *searchController;  
  
- (void)searchBarConfiguration  
{  
    self.searchController = [[UISearchController alloc] initWithSearchResultsController:nil];  
    self.searchController.searchBar.delegate = self;  
    self.searchController.hidesNavigationBarDuringPresentation = NO;  
  
    // Hides search bar initially. When the user pulls down on the list, the search bar is  
    revealed.  
    [self.tableView setContentOffset:CGPointMake(0,  
self.searchController.searchBar.frame.size.height)];  
  
    self.searchController.searchBar.backgroundColor = [UIColor DarkBlue];  
    self.searchController.searchBar.tintColor = [UIColor DarkBlue];  
  
    self.tableView.contentOffset = CGPointMake(0,  
CGRectGetHeight(_searchController.searchBar.frame));  
    self.tableView.tableHeaderView = _searchController.searchBar;  
    _searchController.searchBar.delegate = self;  
    _searchController.searchBar.showsCancelButton = YES;  
    self.tapGestureRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self  
action:@selector(resetSearchbarAndTableView)];  
    [self.view addGestureRecognizer:self.tapGestureRecognizer];  
  
}  
  
- (void)resetSearchbarAndTableView{  
    // Reload your tableview and resign keyboard.  
}  
  
- (void)searchBarCancelButtonClicked:(UISearchBar *)searchBar{  
    // Search cancelled  
}  
- (void)searchBarSearchButtonClicked:(UISearchBar *)searchBar{  
    // Implement filtration of your data as per your need using NSPredicate or else.  
    // then reload your data control like Tableview.  
}
```

Lea `UISearchController` en línea: <https://riptutorial.com/es/ios/topic/2813/uisearchcontroller>

Capítulo 187: UISlider

Examples

UISlider

C objetivo

Declare una propiedad de control deslizante en `ViewController.h` o en la interfaz de `ViewController.m`

```
@property (strong, nonatomic)UISlider *slider;

//Define frame of slider and add to view
CGRect frame = CGRectMake(0.0, 100.0, 320.0, 10.0);
UISlider *slider = [[UISlider alloc] initWithFrame:frame];
[slider addTarget:self action:@selector(sliderAction:)
forControlEvents:UIControlEventValueChanged];
[self.slider setBackgroundColor:[UIColor clearColor]];
self.slider.minimumValue = 0.0;
self.slider.maximumValue = 50.0;
//sending a NO/False would update the value of slider only when the user is no longer touching
the screen. Hence sending only the final value
self.slider.continuous = YES;
self.slider.value = 25.0;
[self.view addSubview slider];
```

Manejar el evento de cambio de control deslizante

```
- (IBAction)sliderAction:(id)sender {
    NSLog(@"Slider Value %f", sender.value);
}
```

Ejemplo de SWIFT

```
let frame = CGRect(x: 0, y: 100, width: 320, height: 10)
let slider = UISlider(frame: frame)
slider.addTarget(self, action: #selector(sliderAction), for: .valueChanged)
slider.backgroundColor = .clear
slider.minimumValue = 0.0
slider.maximumValue = 50.0
//sending a NO/False would update the value of slider only when the user is no longer
touching the screen. Hence sending only the final value
slider.isContinuous = true
slider.value = 25.0
view.addSubview(slider)
```

Manejando el evento de cambio deslizante

```
func sliderAction(sender:UISlider!)
{
```

```
print ("value--\ (sender.value) ")
}
```

Añadiendo una imagen de pulgar personalizada

Para agregar una imagen personalizada para el pulgar del control deslizante, simplemente llame al método `setThumbImage` con su imagen personalizada:

Swift 3.1:

```
let slider = UISlider()
let thumbImage = UIImage
slider.setThumbImage(thumbImage, for: .normal)
```

Lea UISlider en línea: <https://riptutorial.com/es/ios/topic/7402/uislider>

Capítulo 188: UISplitViewController

Observaciones

`UISplitViewController` es una clase contenedora como `UITabViewController`, `UINavigationController`. Separa la vista principal en dos Controladores de vista `masterViewController` (`PrimaryViewController`) y `detailViewController` (`SecondaryViewController`). podemos enviar una matriz con dos controladores de vista y Apple recomienda a `UISplitViewController` como un controlador de vista de raíz para su aplicación. Para interactuar entre los controles de vista, uso `NSNotificationCenter`.

Examples

Interacción de la vista maestra y de detalle usando delegados en el objetivo C

`UISplitViewController` debe ser el `rootViewController` de su aplicación.

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    self.window.backgroundColor = [UIColor blackColor];
    [self.window makeKeyAndVisible];
    self.window.clipsToBounds = YES;
    SplitViewController *spView = [[SplitViewController alloc] init];
    self.window.rootViewController = spView;
    [self.window makeKeyAndVisible];
    return YES;
}
```

Simplemente crea un objeto para el `UISplitViewController` y configura ese controlador de vista como el controlador de vista de raíz para tu aplicación.

SplitViewController.h

```
#import <UIKit/UIKit.h>
#import "MasterViewController.h"
#import "DetailViewController.h"
@interface ViewController : UISplitViewController
{
    DetailViewController *detailVC;
    MasterViewController *masterVC;
    NSMutableArray *array;
}
@end
```

`MasterViewController` es siempre en el lado izquierdo del dispositivo se puede configurar el ancho

en `UISplitViewController` métodos delegado y `DetailViewController` está en el lado derecho de la aplicación

SplitViewController.m

```
#import "ViewController.h"
#define ANIMATION_LENGTH 0.3
@interface ViewController ()
@end

@implementation ViewController
- (void) viewDidLoad
{
    [super viewDidLoad];
    masterVC = [[MasterViewController alloc] init];
    detailVC = [[DetailViewController alloc] init];
    [masterVC setDetailDelegate:(id) detailVC];
    NSArray *vcArray = [NSArray arrayWithObjects:masterVC, detailVC, nil];
    self.preferredDisplayMode = UISplitViewControllerDisplayModeAutomatic;
    self.viewControllers = vcArray;
    self.delegate = (id) self;
    self.presentsWithGesture = YES;
}
}
```

Los ViewControllers maestro y de detalle creados se agregan a una matriz que se establece en `self.viewControllers` en `UISplitViewController`. `self.preferredDisplayMode` es el modo establecido para mostrar la [documentación de Apple](#) `master` y `DetailViewController` [para DisplayMode](#). `self.presentsWithGesture` habilita el gesto de barrido para mostrar el `self.presentsWithGesture` `MasterViewcontroller`

MasterViewController.h

```
#import <UIKit/UIKit.h>

@protocol DetailViewDelegate <NSObject>
@required
- (void) sendSelectedNavController:(UIViewController *) viewController;
@end

@interface MasterViewController : UIViewController
{
    UITableView *mainTableView;
    NSMutableArray *viewControllerArray;
}
@property (nonatomic, retain) id<DetailViewDelegate> detailDelegate;
@end
```

Cree un delegado `DetailViewDelegate` con el `sendSelectedNavController:(UIViewController *) viewController` para enviar el `UIViewController` al `DetailViewController`. Luego, en `MasterViewController` `mainTableView` es la vista de tabla en el lado izquierdo. El `viewControllerArray` contiene todos los `UITableViewController` que deben mostrarse en `DetailViewController`

MasterViewController.m

```

#import "MasterViewController.h"

@implementation MasterViewController
@synthesize detailDelegate;

-(void)viewDidLoad
{
[super viewDidLoad];

UIViewController *dashBoardVC = [[UIViewController alloc]init];
[dashBoardVC.view setBackgroundColor:[UIColor redColor]];
UIViewController *inventVC = [[UIViewController alloc]init];
[inventVC.view setBackgroundColor:[UIColor whiteColor]];
UIViewController *alarmVC = [[UIViewController alloc]init];
[alarmVC.view setBackgroundColor: [UIColor purpleColor]];
UIViewController *scanDeviceVC = [[UIViewController alloc]init];
[scanDeviceVC.view setBackgroundColor:[UIColor cyanColor]];
UIViewController *serverDetailVC = [[UIViewController alloc]init];
[serverDetailVC.view setBackgroundColor: [UIColor whiteColor]];
viewControllerArray = [[NSMutableArray
alloc] initWithObjects:dashBoardVC, inventVC, alarmVC, scanDeviceVC, serverDetailVC, nil];
mainTableView = [[UITableView alloc] initWithFrame:CGRectMake(0, 50, self.view.frame.size.width,
self.view.frame.size.height-50) style:UITableViewStylePlain];
[mainTableView setDelegate: (id)self];
[mainTableView setDataSource: (id)self];
[mainTableView setSeparatorStyle:UITableViewCellStyleNone];
[mainTableView setScrollsToTop:NO];
[self.view addSubview:mainTableView];
}

-(CGFloat)tableView:(UITableView *)tableView
heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
return 100;
}

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection: (NSInteger) section
{
return [viewControllerArray count];
}

-(NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
return 1; //count of section
}

-(UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
NSString *cellId = [NSString
stringWithFormat:@"Cell%i%d", (long) indexPath.section, (long) indexPath.row];
UITableViewCell *cell =[tableView dequeueReusableCellWithIdentifier:cellId];

if (cell == nil)
{
cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:cellId];
}
[cell.contentView setBackgroundColor:[UIColor redColor]];
cell.textLabel.text =[NSString stringWithFormat:@"My VC at index %ld", (long) indexPath.row];
return cell;
}

```

```

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    [detailDelegate sendSelectedNavController:[viewControllerArray
objectAtIndex:indexPath.row]];
}

@end

```

Cree algunos `UITableViewController`s y los agregue a una matriz. La vista de la tabla se inicializa luego en el método `didSelectRowAtIndexPath`. Envíe un `UITableViewController` al `DetailViewController` utilizando `detailDelegate` con el `UITableViewController` correspondiente en una matriz como parámetro

DetailViewController.h

```

#import <UIKit/UIKit.h>

@interface DetailViewController : UIViewController<UICollectionViewDelegate>
{
    UITableViewController *tempNav;
}

@end

```

DetailViewController.m

```

#import "DetailViewController.h"

@implementation DetailViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
    [self.view setBackgroundColor:[UIColor whiteColor]];
}
- (void)sendSelectedNavController:(UITableViewController *)navController
{
    NSArray *viewsToRemove = [self.view subviews];
    for (UIView *v in viewsToRemove) {
        [v removeFromSuperview];
    }
    tempNav = navController;
    [self.view addSubview:tempNav.view];
}

@end

```

`sendSelectedNavController` se declara aquí eliminando todas las vistas en `DetailViewController` y agregando el `UITableViewController` pasado del `MasterViewController`

Añadiendo algunas capturas de pantalla de la aplicación.



`preferredDisplayMode` como automático al deslizar la pantalla obtenemos el `MasterViewController` como se adjunta en la imagen de abajo, pero en el modo `DetailViewController` obtenemos tanto `MasterViewController` como `DetailViewController`

Carrier 

8:26 PM

My VC at index 0

My VC at index 1

My VC at index 2

My VC at index 3

My VC at index 4

My VC at index 0

My VC at index 1

My VC at index 2

My VC at index 3

My VC at index 4

Capítulo 189: UISplitViewController

Observaciones

En iOS 8 y versiones posteriores, puede usar la clase `UISplitViewController` en todos los dispositivos iOS. En versiones anteriores de iOS, la clase solo está disponible en iPad.

`UISplitViewController` es una clase contenedora como `UITabViewController`, `UINavigationController`. Separa la vista principal en dos `UITableViewController` `masterViewController` (`PrimaryViewController`) y `detailViewController` (`SecondaryViewController`). podemos enviar un `NSArray` con dos `UITableViewController` y Apple recomienda `UISplitViewController` como controlador de vista de raíz para su aplicación. Para interactuar entre los `UITableViewController` utilizo `NSNotificationCenter`.

Examples

Interactuando entre la vista maestra y la vista detallada usando delegados en el Objetivo C

`UISplitViewController` necesita el controlador de vista raíz de la ventana de su aplicación

AppDelegate.m

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]]
    self.window.backgroundColor = [UIColor blackColor];
    [self.window makeKeyAndVisible];
    self.window.clipsToBounds = YES;
    SplitViewController *spView = [[SplitViewController alloc] init];
    self.window.rootViewController = spView;
    [self.window makeKeyAndVisible];
    return YES;
}
```

Simplemente cree un objeto para su `UISplitViewController` y `UISplitViewController` como `rootViewController` para su aplicación.

SplitViewController.h

```
#import <UIKit/UIKit.h>
#import "MasterViewController.h"
#import "DetailViewController.h"
@interface ViewController : UISplitViewController
{
    DetailViewController *detailVC;
    MasterViewController *masterVC;
    NSMutableArray *array;
}
@end
```

`MasterViewController` es un `UIViewController` que se establece en el lado izquierdo del dispositivo, puede establecer el ancho en `UISplitViewController` usando `maximumPrimaryColumnWidth` y `DetailViewController` está en el lado derecho

SplitViewController.m

```
#import "ViewController.h"
#define ANIMATION_LENGTH 0.3
@interface ViewController ()
@end

@implementation ViewController
- (void)viewDidLoad
{
    [super viewDidLoad];
    masterVC = [[MasterViewController alloc] init];
    detailVC = [[DetailViewController alloc] init];
    [masterVC setDetailDelegate:(id)detailVC];
    NSArray *vcArray = [NSArray arrayWithObjects:masterVC, detailVC, nil];
    self.preferredDisplayMode = UISplitViewControllerDisplayModeAutomatic;
    self.viewControllers = vcArray;
    self.delegate = (id)self;
    self.presentsWithGesture = YES;
}
}
```

El `UIViewController` maestro y los detalles se agregan a una `NSArray` que se establece en `self.viewControllers`. `self.preferredDisplayMode` es el modo establecido para mostrar `MasterViewController` y `DetailViewController`. `self.presentsWithGesture` habilita el gesto de barrido para mostrar `MasterViewController`

MasterViewController.h

```
#import <UIKit/UIKit.h>

@protocol DetailViewDelegate <NSObject>
@required
- (void)sendSelectedNavController:(UIViewController *)viewController;
@end

@interface MasterViewController : UIViewController
{
    UITableView *mainTableView;
    NSMutableArray *viewControllerArray;
}
@property (nonatomic, retain) id<DetailViewDelegate> detailDelegate;
@end
```

Cree un delegado `DetailViewDelegate` con el método `sendSelectedNavController` para enviar los `UITableViewController` al `DetailViewController`. Luego en `MasterViewController` se crea un `UITableView`. El `viewControllerArray` contiene todos los `UITableViewController` que deben mostrarse en `DetailViewController`

MasterViewController.m

```
#import "MasterViewController.h"
```

```

@implementation MasterViewController
@synthesize detailDelegate;

-(void)viewDidLoad
{
[super viewDidLoad];

UIViewController *dashBoardVC = [[UIViewController alloc]init];
[dashBoardVC.view setBackgroundColor:[UIColor redColor]];
UIViewController *inventVC = [[UIViewController alloc]init];
[inventVC.view setBackgroundColor:[UIColor whiteColor]];
UIViewController *alarmVC = [[UIViewController alloc]init];
[alarmVC.view setBackgroundColor: [UIColor purpleColor]];
UIViewController *scanDeviceVC = [[UIViewController alloc]init];
[scanDeviceVC.view setBackgroundColor:[UIColor cyanColor]];
UIViewController *serverDetailVC = [[UIViewController alloc]init];
[serverDetailVC.view setBackgroundColor: [UIColor whiteColor]];
viewControllerArray = [[NSMutableArray
alloc]initWithObjects:dashBoardVC,inventVC,alarmVC,scanDeviceVC,serverDetailVC,nil];
mainTableView = [[UITableView alloc]initWithFrame:CGRectMake(0, 50,self.view.frame.size.width,
self.view.frame.size.height-50) style:UITableViewStylePlain];
[mainTableView setDelegate:(id)self];
[mainTableView setDataSource:(id)self];
[mainTableView setSeparatorStyle:UITableViewCellStyleNone];
[mainTableView setScrollsToTop:NO];
[self.view addSubview:mainTableView];
}

-(CGFloat)tableView:(UITableView *)tableView
heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
return 100;
}

-(NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection: (NSInteger)section
{
return [viewControllerArray count];
}

-(NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
return 1; //count of section
}

-(UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
NSString *cellId = [NSString
stringWithFormat:@"Cell%li%ld", (long)indexPath.section, (long)indexPath.row];
UITableViewCell *cell =[tableView dequeueReusableCellWithIdentifier:cellId];

if (cell == nil)
{
cell = [[UITableViewCell alloc]initWithStyle:UITableViewCellStyleDefault
reuseIdentifier:cellId];
}
[cell.contentView setBackgroundColor:[UIColor redColor]];
cell.textLabel.text =[NSString stringWithFormat:@"My VC at index %ld", (long)indexPath.row];
return cell;
}

```

```

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    [detailDelegate sendSelectedNavController:[viewControllerArray
objectAtIndex:indexPath.row]];
}

@end

```

Creó algo de `UIViewController` y lo agregé a un `NSMutableArray`. El `UITableView` se inicializa luego en el método `didselectrowatindexpath`. Envío un `UIViewController` al `DetailViewController` utilizando el delegado `detailDelegate` con el `UIViewController` correspondiente en el `NSMutableArray` como parámetro

DetailViewController.h

```

#import <UIKit/UIKit.h>

@interface DetailViewController : UIViewController<UICollectionViewDelegate>
{
    UIViewController *tempNav;
}
@end

```

DetailViewController.m

```

#import "DetailViewController.h"

@implementation DetailViewController
-(void)viewDidLoad
{
    [super viewDidLoad];
    [self.view setBackgroundColor:[UIColor whiteColor]];
}
-(void)sendSelectedNavController:(UIViewController *)navController
{
    NSArray *viewsToRemove = [self.view subviews];
    for (UIView *v in viewsToRemove) {
        [v removeFromSuperview];
    }
    tempNav = navController;
    [self.view addSubview:tempNav.view];
}
@end

```

El `sendSelectedNavController` se declara aquí con la eliminación de todas las `UIView` en `DetailViewController` y la adición del `UIViewController` pasado del `MasterViewController`.

Lea `UISplitViewController` en línea: <https://riptutorial.com/es/ios/topic/4844/uisplitviewController>

Capítulo 190: UIStackView

Examples

Crear una vista de pila horizontal programáticamente

Swift 3

```
let stackView = UIStackView()
stackView.axis = .horizontal
stackView.alignment = .fill // .leading .firstBaseline .center .trailing .lastBaseline
stackView.distribution = .fill // .fillEqually .fillProportionally .equalSpacing
    .equalCentering

let label = UILabel()
label.text = "Text"
stackView.addArrangedSubview(label)
// for horizontal stack view, you might want to add width constraint to label or whatever view
you're adding.
```

Rápido

```
let stackView = UIStackView()
stackView.axis = .Horizontal
stackView.alignment = .Fill // .Leading .FirstBaseline .Center .Trailing .LastBaseline
stackView.distribution = .Fill // .FillEqually .FillProportionally .EqualSpacing
    .EqualCentering

let label = UILabel(frame: CGRectZero)
label.text = "Label"
stackView.addArrangedSubview(label)
// for horizontal stack view, you might want to add width constraint to label or whatever view
you're adding.
```

C objetivo

```
UIStackView *stackView = [[UIStackView alloc] init];
stackView.axis = UILayoutConstraintAxisHorizontal;
stackView.alignment = UIStackViewAlignmentFill; //UIStackViewAlignmentLeading,
UIStackViewAlignmentFirstBaseline, UIStackViewAlignmentCenter, UIStackViewAlignmentTrailing,
UIStackViewAlignmentLastBaseline
stackView.distribution = UIStackViewDistributionFill; //UIStackViewDistributionFillEqually,
UIStackViewDistributionFillProportionally, UIStackViewDistributionEqualSpacing,
UIStackViewDistributionEqualCentering

UILabel *label = [[UILabel alloc] initWithFrame:CGRectZero];
label.text = @"Label";
[stackView addArrangedSubview:label];
//For horizontal stack view, you might want to add a width constraint to your label or
whatever view you are adding.
```

Crear una vista de pila vertical programáticamente

Rápido

```
let stackView = UIStackView()
stackView.axis = .Vertical
stackView.alignment = .Fill // .Leading .FirstBaseline .Center .Trailing .LastBaseline
stackView.distribution = .Fill // .FillEqually .FillProportionally .EqualSpacing
.EqualCentering

let label = UILabel(frame: CGRectZero)
label.text = "Label"
stackView.addArrangedSubview(label)
// for vertical stack view, you might want to add height constraint to label or whatever view
you're adding.
```

C objetivo

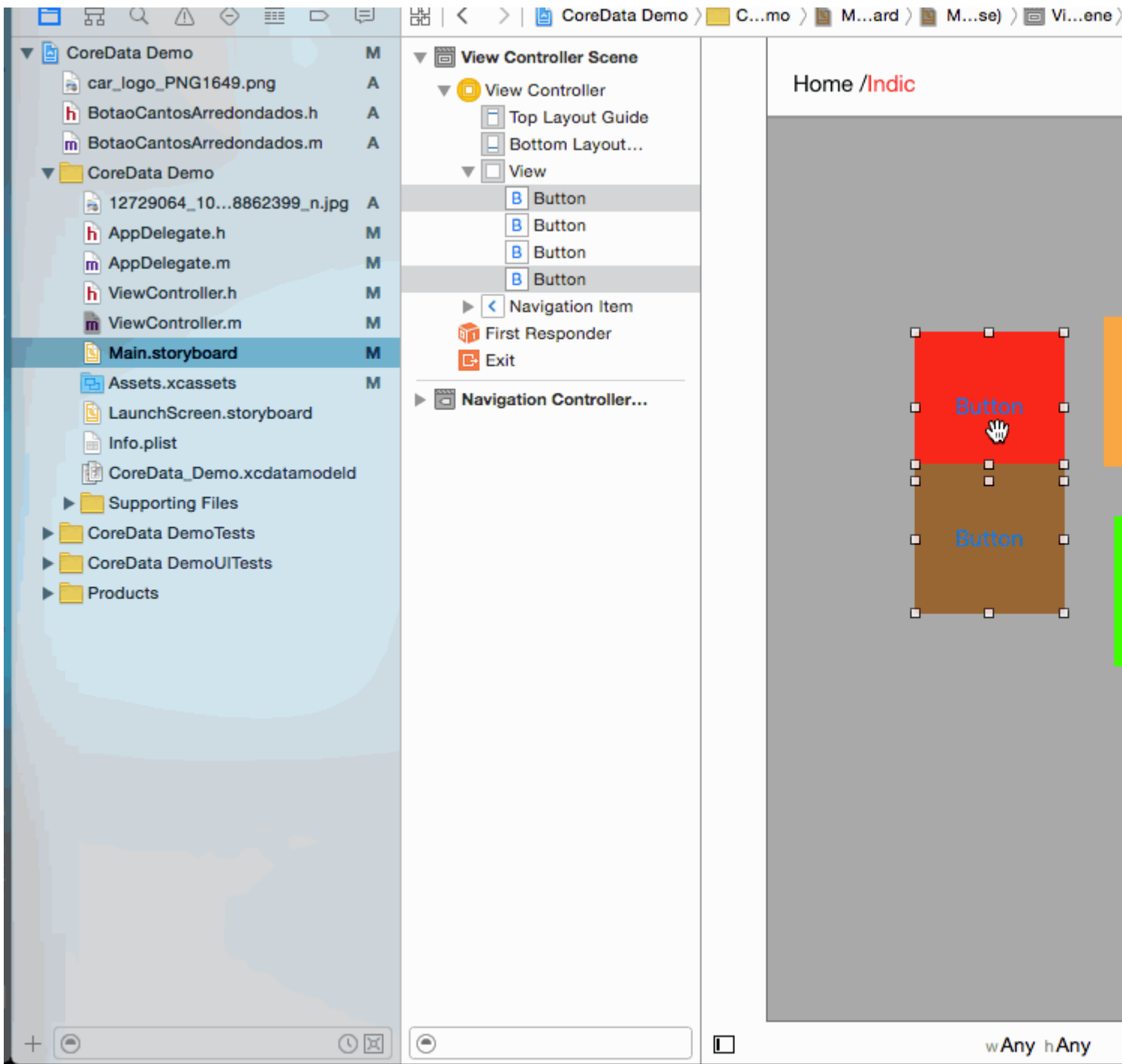
```
UIStackView *stackView = [[UIStackView alloc] init];
stackView.axis = UILayoutConstraintAxisVertical;
stackView.alignment = UIStackViewAlignmentFill; //UIStackViewAlignmentLeading,
UIStackViewAlignmentFirstBaseline, UIStackViewAlignmentCenter, UIStackViewAlignmentTrailing,
UIStackViewAlignmentLastBaseline
stackView.distribution = UIStackViewDistributionFill; //UIStackViewDistributionFillEqually,
UIStackViewDistributionFillProportionally, UIStackViewDistributionEqualSpacing,
UIStackViewDistributionEqualCentering

UILabel *label = [[UILabel alloc] initWithFrame:CGRectZero];
label.text = @"Label";
[stackView addArrangedSubview:label];
//For vertical stack view, you might want to add a height constraint to your label or whatever
view you are adding.
```

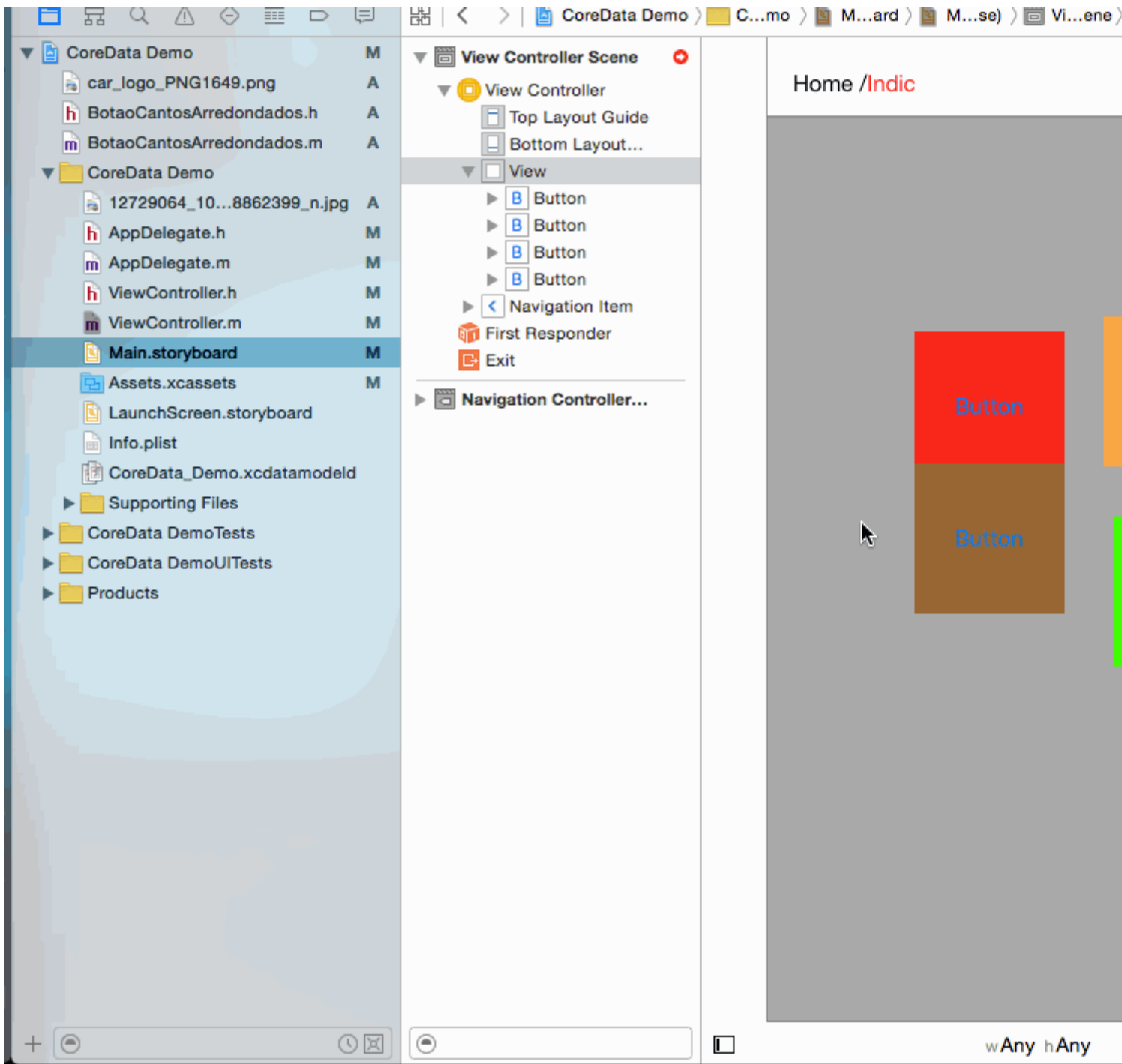
Botones centrales con UIStackview

Paso 1: toma el botón 4 en tu Guión Gráfico. Button1, Button2, Button 3, Button4

Paso 2: - Dar altura y ancho fijos a todos los botones.

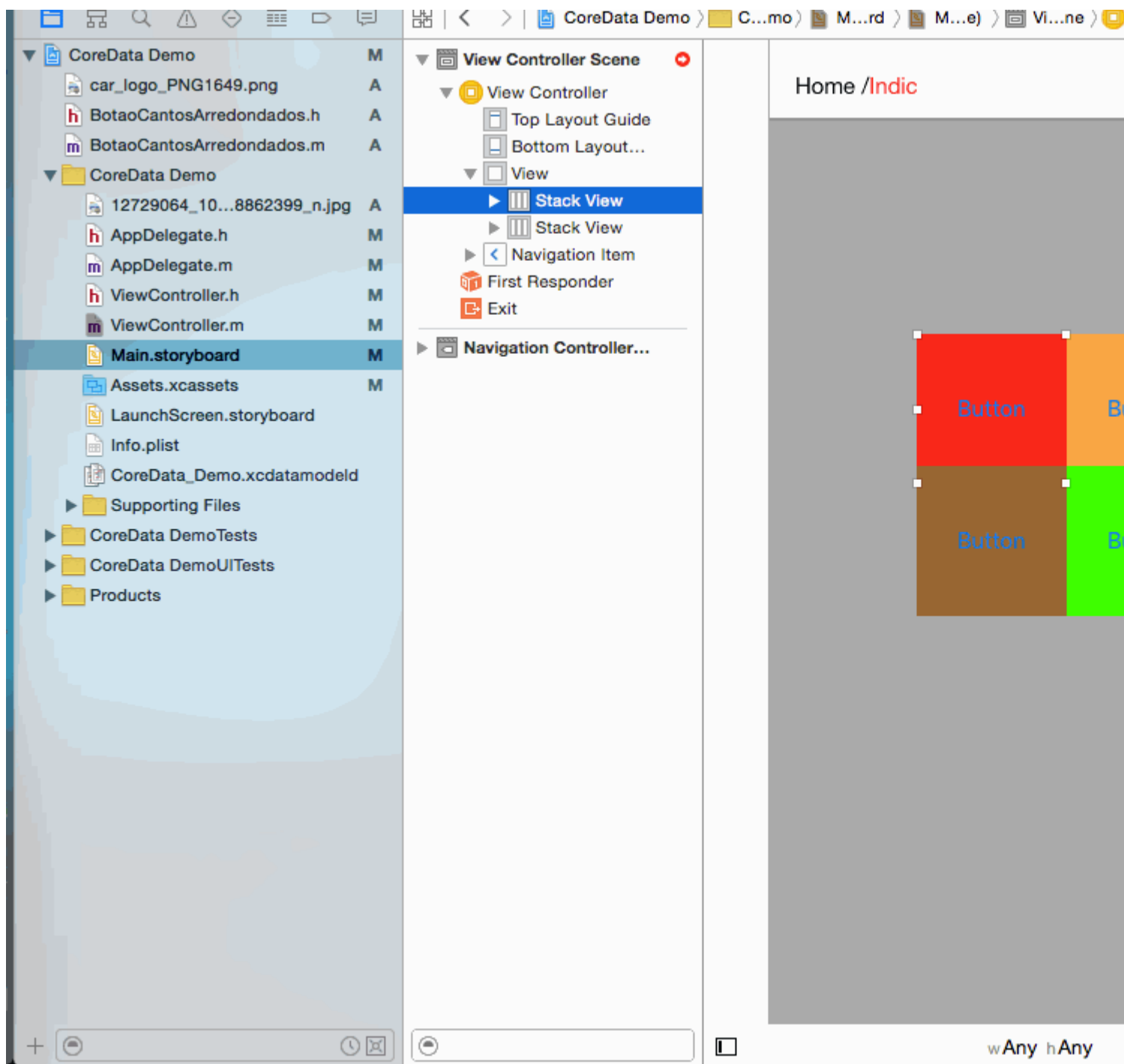
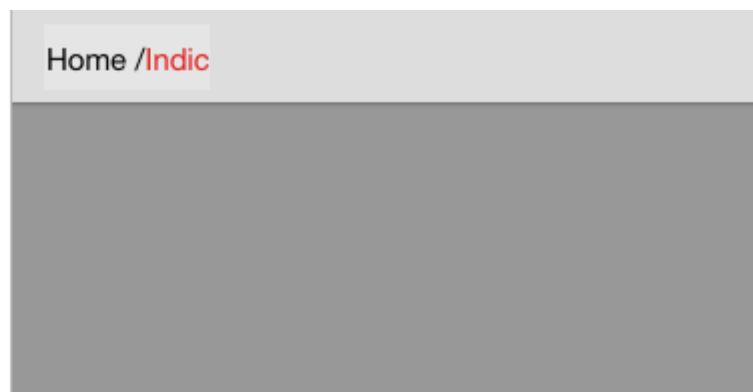
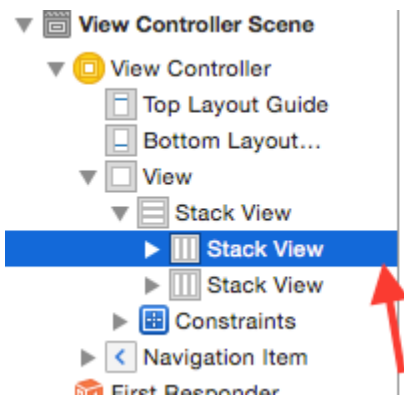


Paso 3: - Todos los 2 - 2 pares de botones en 2 stackview.

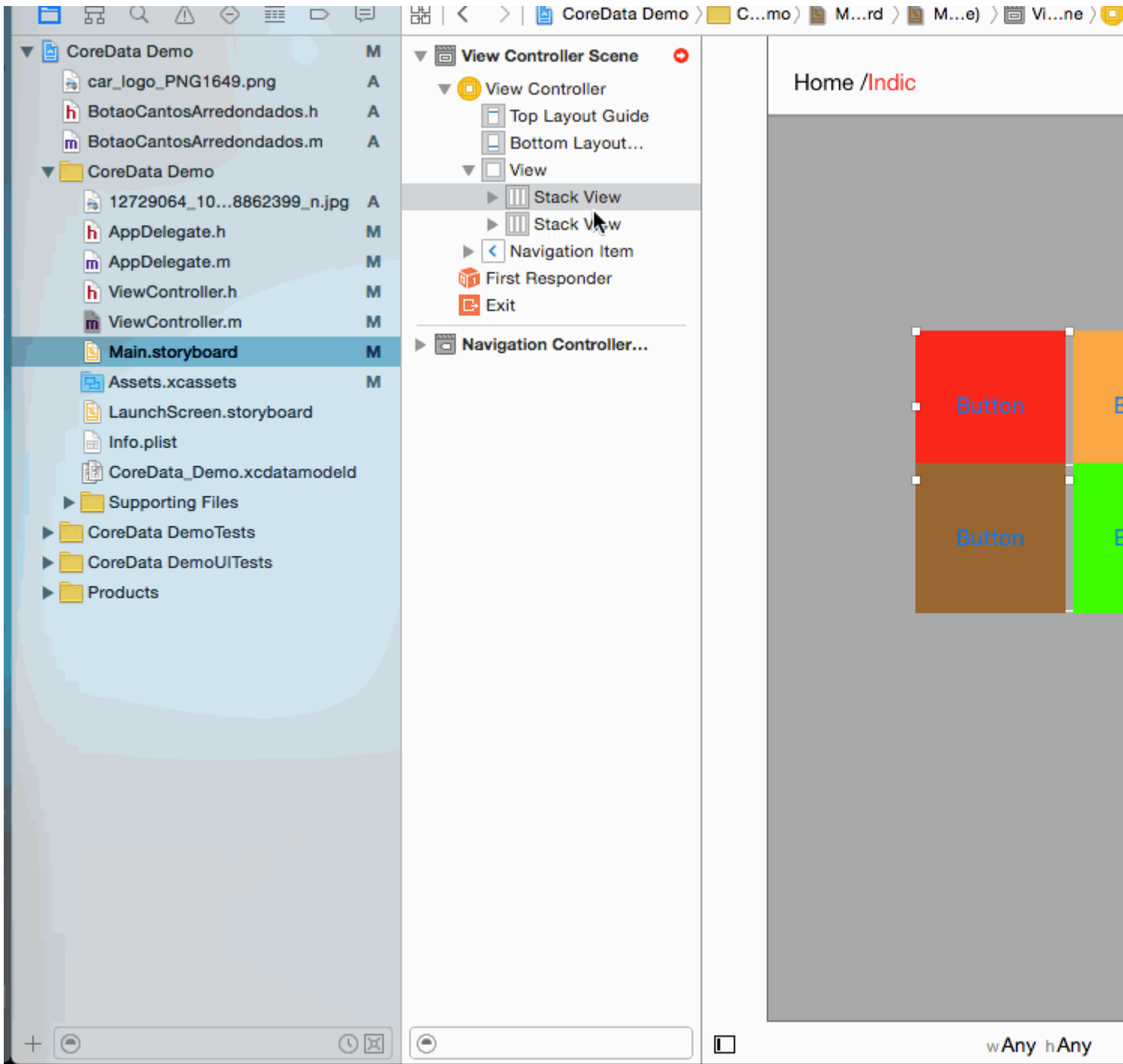


Paso 4: - Establecer la propiedad UIStackview para ambos.

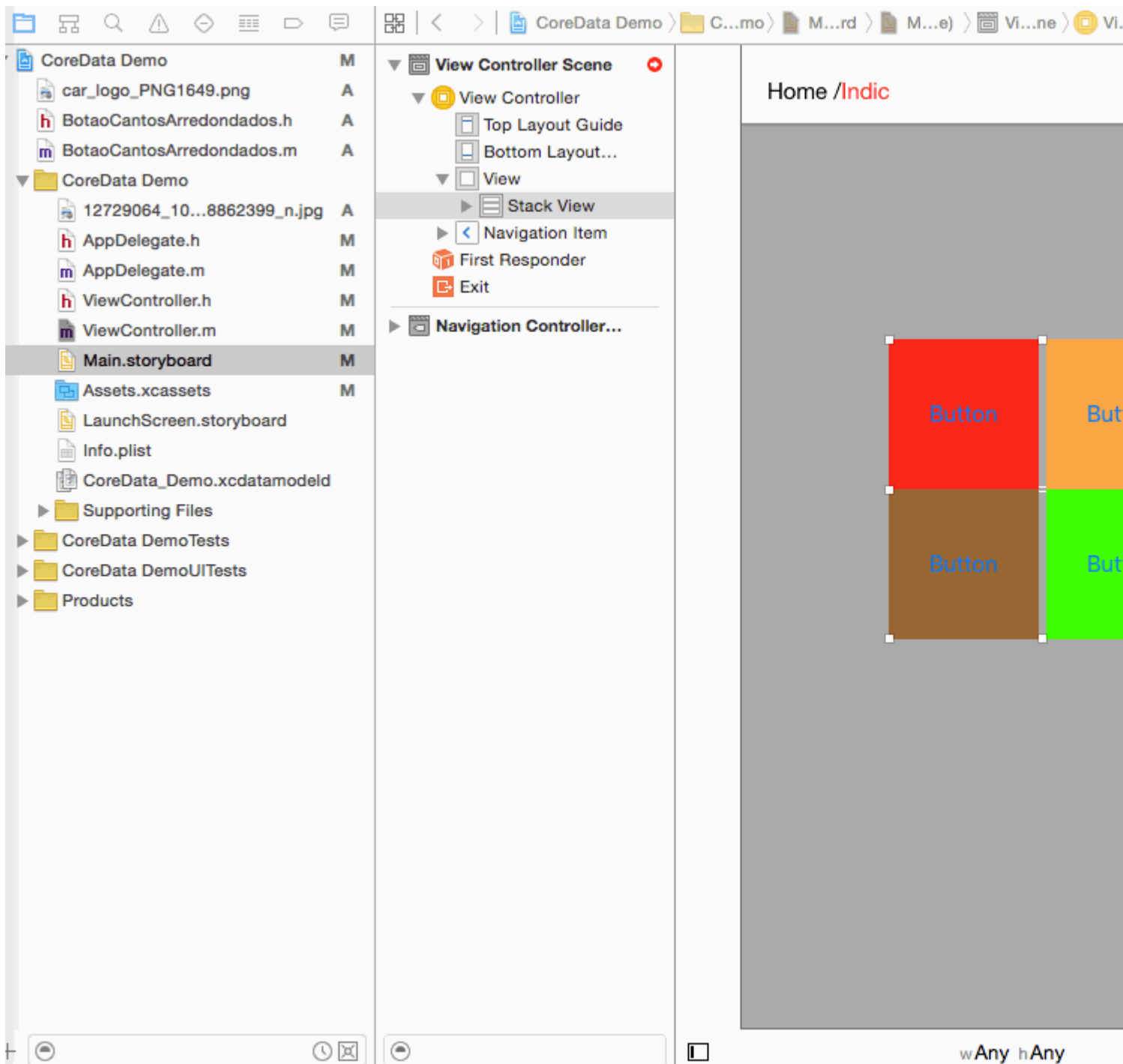
```
Distribution -> Fill Equally  
Spacing -> 5 (as per your requirement)
```



Paso 5: - Agrega ambas Stackview en una Stackview

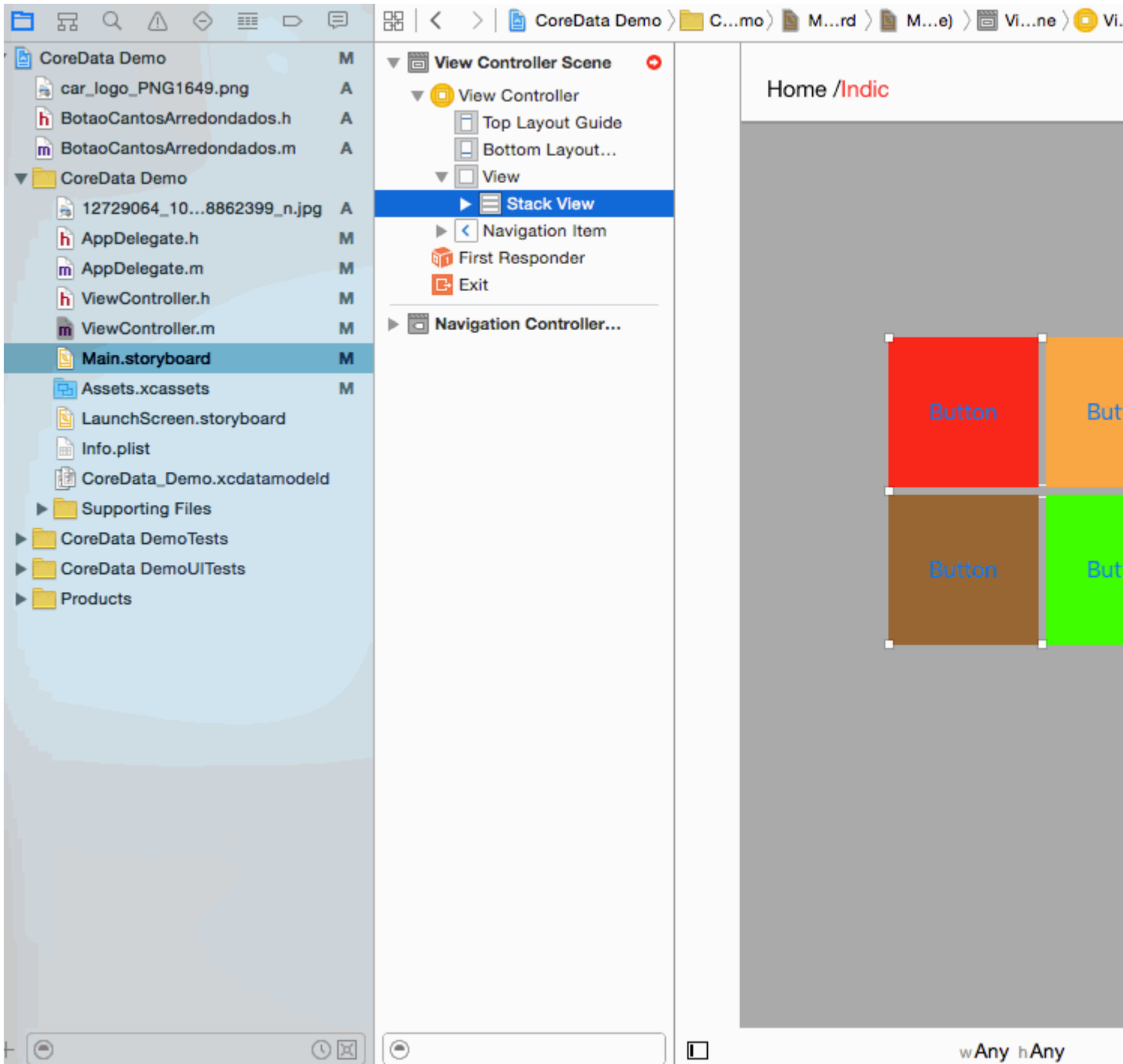


Paso 6: - Establezca `Distribution = Fill equally Spacing =5` en la vista de pila principal (establezca según sus requisitos)

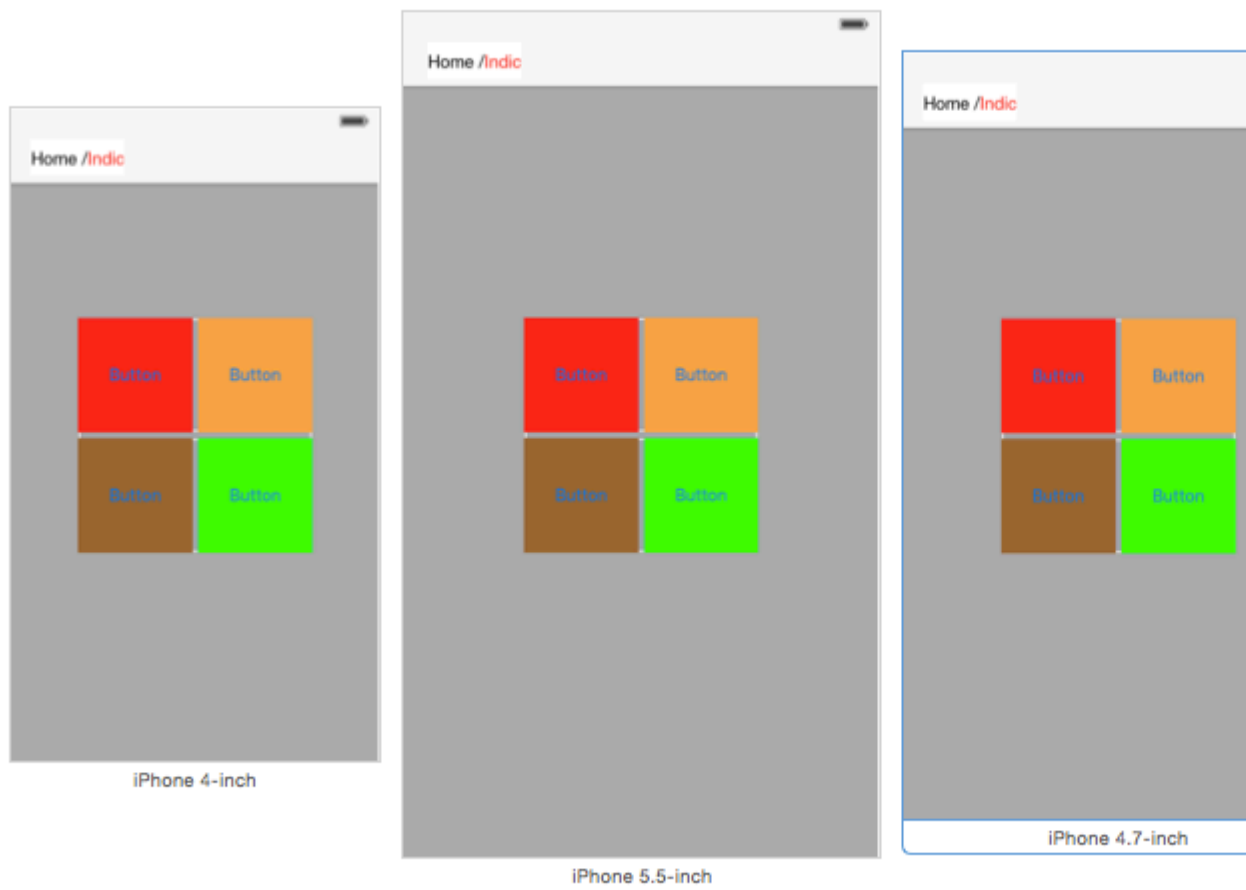


Paso 7: - Ahora establece Restricción a la vista de pila principal

```
center Horizontally in container  
  
center vertically in container  
  
and select Update Frame.
```



Paso 8: - Es hora de salida para todos los dispositivos.



Lea UICollectionViewController en línea: <https://riptutorial.com/es/ios/topic/1390/uistackview>

Capítulo 191: UIStoryboard

Introducción

Un objeto UIStoryboard encapsula el gráfico del controlador de vista almacenado en un archivo de recursos del storyboard de Interface Builder. Este gráfico del controlador de vista representa los controladores de vista para la totalidad o parte de la interfaz de usuario de la aplicación.

Examples

Obteniendo una instancia de UIStoryboard programáticamente

RÁPIDO:

Obtener una instancia de **UIStoryboard** mediante programación se puede hacer de la siguiente manera:

```
let storyboard = UIStoryboard(name: "Main", bundle: nil)
```

dónde:

- **nombre** => el nombre del guión gráfico sin la extensión
- **bundle** => el paquete que contiene el archivo del guión gráfico y sus recursos relacionados. Si especifica nil, este método busca en el paquete principal de la aplicación actual.

Por ejemplo, puede usar la instancia creada anteriormente para acceder a un determinado **UIViewController creado** dentro de ese guión gráfico:

```
let viewController = storyboard.instantiateViewController(withIdentifier: "yourIdentifier")
```

C OBJETIVO:

Obtener una instancia de **UIStoryboard** en Objective-C se puede hacer de la siguiente manera:

```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"MainStoryboard" bundle:nil];
```

Ejemplo de acceso a **UIViewController** instanciado dentro de ese guión gráfico:

```
MyViewController *myViewController = [storyboard  
instantiateViewControllerWithIdentifier:@"MyViewControllerIdentifier"];
```

Abre otro guión gráfico

```
let storyboard = UIStoryboard(name: "StoryboardName", bundle: nil)
let vc = storyboard.instantiateViewController(withIdentifier: "ViewControllerID") as
YourViewController
self.present(vc, animated: true, completion: nil)
```

Lea UIStoryboard en línea: <https://riptutorial.com/es/ios/topic/8795/uistoryboard>

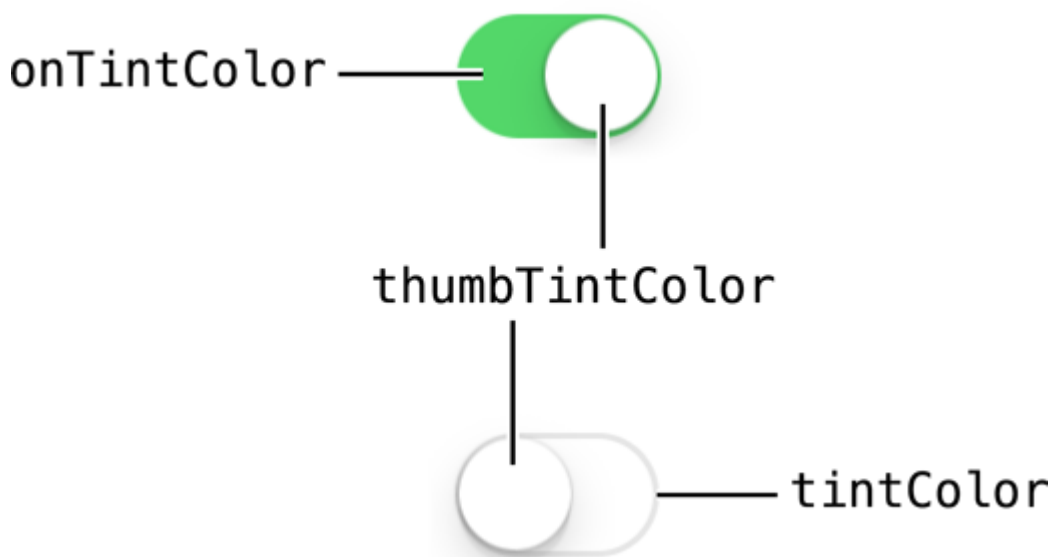
Capítulo 192: UISwitch

Sintaxis

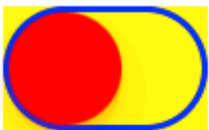
- (instancetype) initWithFrame: (CGRect) frame;
- (void) setOn: (BOOL) en animated: (BOOL) animated;
- (tipo de instancia anulable) initWithCoder: (NSCoder *) aDecoder;

Observaciones

1. Referencia de UISwitch: [Documentación de Apple](#)



2. Otra referencia dada por: [Enoch Huang](#)



Examples

Activar / desactivar

C objetivo

```
[mySwitch setOn:YES];  
//or  
[mySwitch setOn:YES animated:YES];
```

Rápido

```
mySwitch.setOn(false)
//or
mySwitch.setOn(false, animated: false)
```

Establecer color de fondo

C objetivo

```
mySwitch.backgroundColor = [UIColor yellowColor];
[mySwitch setBackgroundColor: [UIColor yellowColor]];
mySwitch.backgroundColor = [UIColor colorWithRed:255/255.0 green:0/255.0 blue:0/255.0
alpha:1.0];
mySwitch.backgroundColor= [UIColor colorWithWhite: 0.5 alpha: 1.0];
mySwitch.backgroundColor=[UIColor colorWithHue: 0.4 saturation: 0.3 brightness:0.7 alpha:
1.0];
```

Rápido

```
mySwitch.backgroundColor = UIColor.yellow
mySwitch.backgroundColor = UIColor(red: 255.0/255, green: 0.0/255, blue: 0.0/255, alpha: 1.0)
mySwitch.backgroundColor = UIColor(white: 0.5, alpha: 1.0)
mySwitch.backgroundColor = UIColor(hue: 0.4,saturation: 0.3,brightness: 0.7,alpha: 1.0)
```

Establecer color de tinte

C objetivo

```
//for off-state
mySwitch.tintColor = [UIColor blueColor];
[mySwitch setTintColor: [UIColor blueColor]];

//for on-state
mySwitch.onTintColor = [UIColor cyanColor];
[mySwitch setOnTintColor: [UIColor cyanColor]];
```

Rápido

```
//for off-state
mySwitch.tintColor = UIColor.blueColor()

//for on-state
mySwitch.onTintColor = UIColor.cyanColor()
```

Establecer imagen para estado activado / desactivado

C objetivo

```
//set off-image
mySwitch.offImage = [UIImage imageNamed:@"off_image"];
```

```
[mySwitch setOffImage:[UIImage imageNamed:@"off_image"]];  
  
//set on-image  
mySwitch.onImage = [UIImage imageNamed:@"on_image"];  
[mySwitch setOnImage:[UIImage imageNamed:@"on_image"]];
```

Rápido

```
//set off-image  
mySwitch.offImage = UIImage(named: "off_image")  
  
//set on-image  
mySwitch.onImage = UIImage(named: "on_image")
```

Lea UISwitch en línea: <https://riptutorial.com/es/ios/topic/2182/uiswitch>

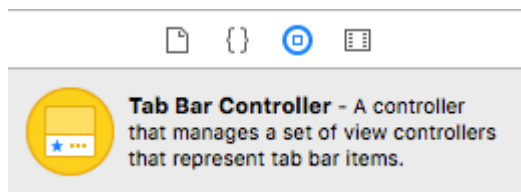
Capítulo 193: UITabBarController

Examples

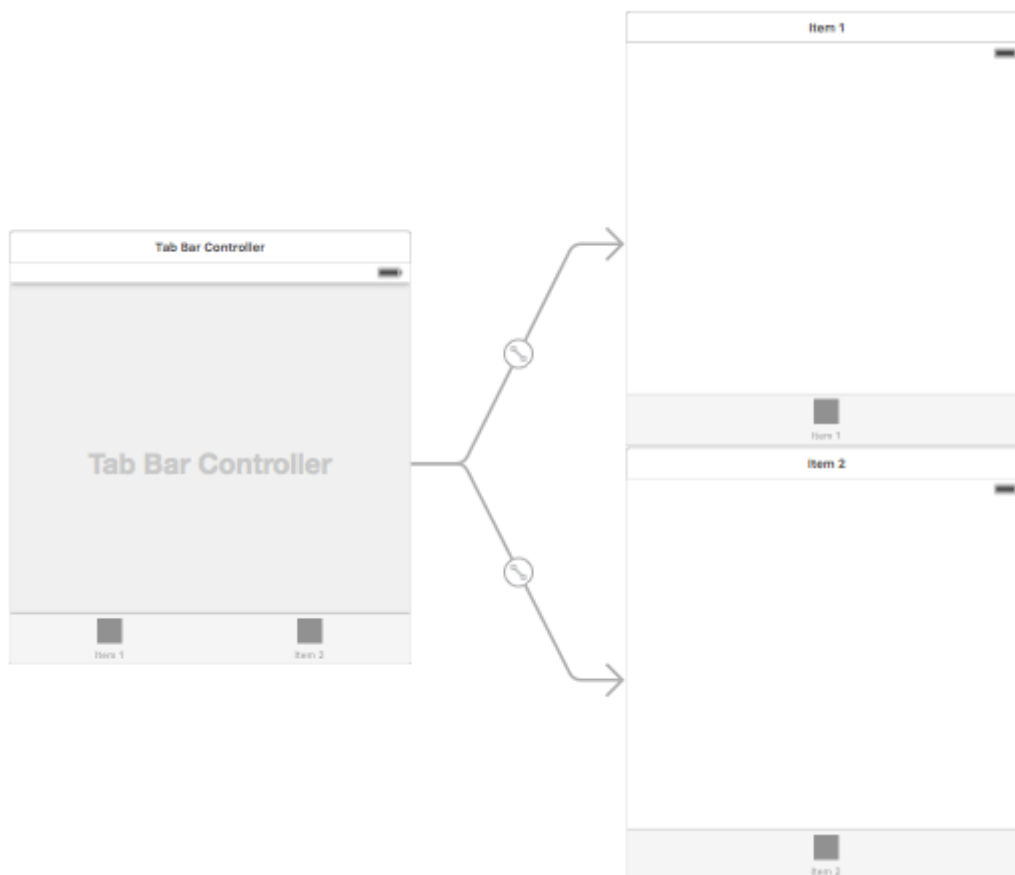
Crear una instancia

Una 'barra de pestañas' se encuentra comúnmente en la mayoría de las aplicaciones de iOS y se usa para presentar distintas vistas en cada pestaña.

Para crear un controlador de la barra de pestañas utilizando el generador de interfaz, arrastre un controlador de la barra de pestañas desde la Biblioteca de objetos al lienzo.



Por defecto, un controlador de barra de pestañas viene con dos vistas. Para agregar vistas adicionales, controle el arrastre desde el controlador de la barra de pestañas a la nueva vista y seleccione 'ver controladores' en el menú desplegable de segue.



Cambio del título de la barra de pestañas y el icono

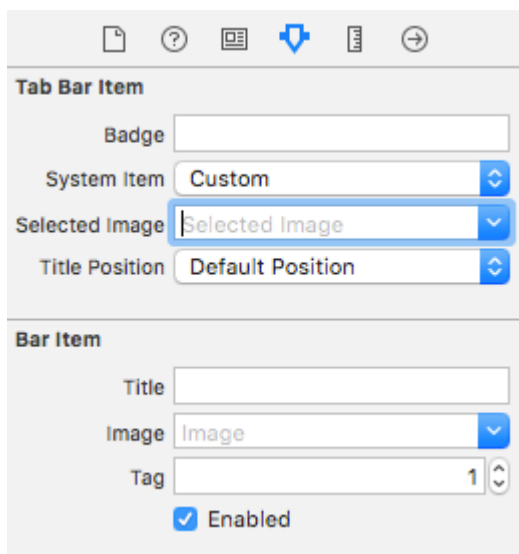
Usando el Story Board:

Seleccione el elemento de la barra de pestañas desde el controlador de vista correspondiente y vaya al inspector de atributos

Si desea un icono y título integrados, configure el 'Elemento del sistema' en el valor correspondiente.

Para un icono personalizado, agregue las imágenes requeridas a la carpeta de activos y configure el 'Elemento del sistema' de antes a 'personalizado'.

Ahora, configure el ícono que se mostrará cuando se seleccione la pestaña del menú desplegable "Imagen seleccionada" y el ícono de pestaña predeterminado del menú desplegable "Imagen". Agregue el título correspondiente en el campo 'título'.



Programmáticamente:

En el método `viewDidLoad()` del controlador de vista, agregue el siguiente código:

C objetivo:

```
self.title = @"item";

self.tabBarItem.image = [UIImage imageNamed:@"item"];
self.tabBarItem.selectedImage = [UIImage imageNamed:@"item_selected"];
```

Rápido:

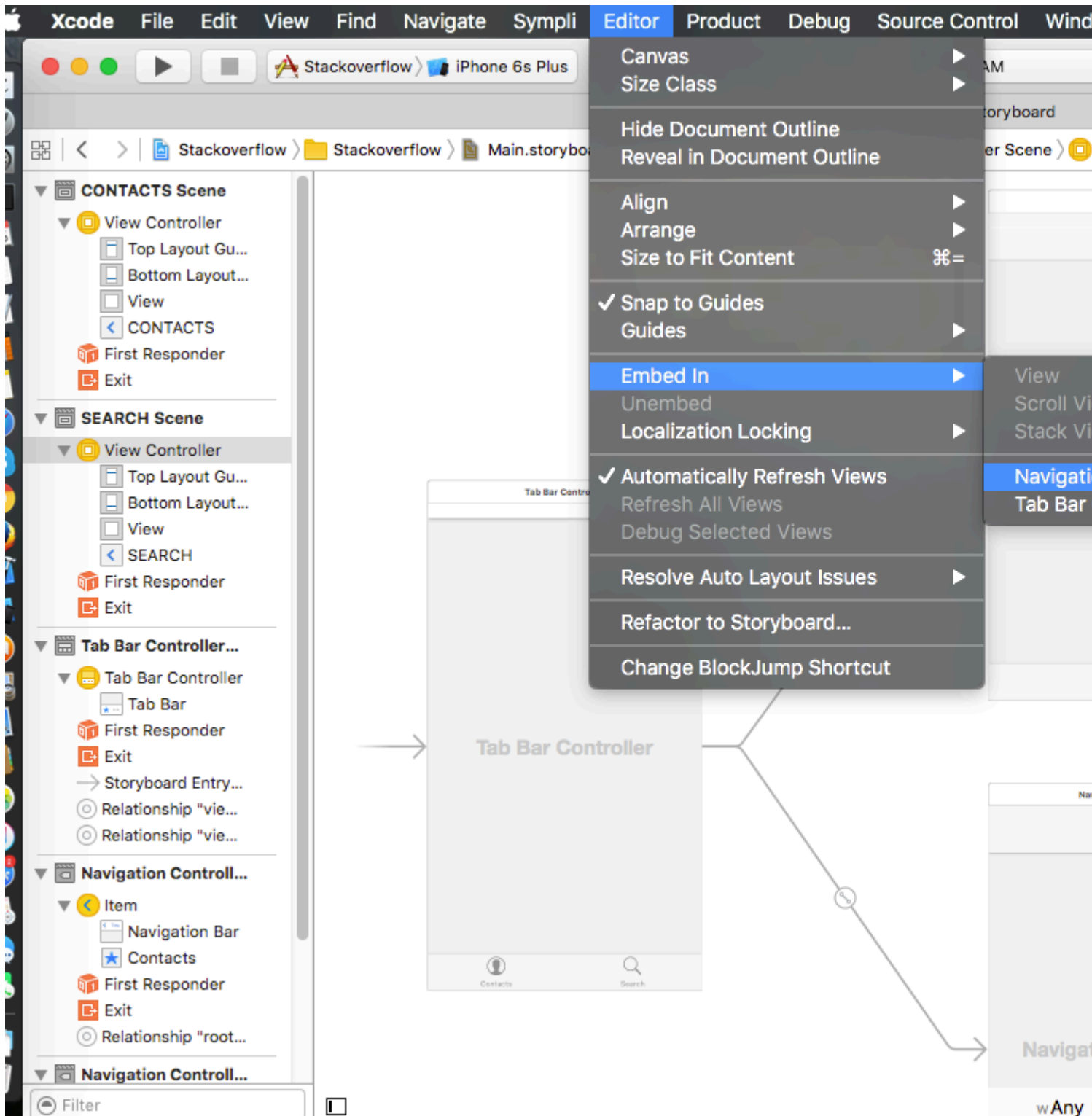
```
self.title = "item"
self.tabBarItem.image = UIImage(named: "item")
self.tabBarItem.selectedImage = UIImage(named: "item_selected")
```

Controlador de navegación con TabBar

El controlador de navegación se puede incrustar en cada pestaña usando el guión gráfico. Puede ser como en la captura de pantalla añadida.

Para agregar un controlador de navegación a un controlador de vista que se conecta desde el controlador de la barra de pestañas, aquí está el flujo

- Seleccione el controlador de vista para el que necesitamos agregar el controlador de navegación. Aquí deje que sea el controlador de vista de búsqueda como la pantalla de selección.
- En el menú **Editor** de Xcode, seleccione **Incrustar en ->** opción **Control de navegación**



Personalización del color de la barra de pestañas

```
[[UITabBar appearance] setTintColor:[UIColor whiteColor]];
[[UITabBar appearance] setBarTintColor:[UIColor tabBarBackgroundColor]];
[[UITabBar appearance] setBackgroundColor:[UIColor tabBarInactiveColor]];
[[UINavigationController appearance] setBarTintColor:[UIColor appBlueColor]];
[[UINavigationController appearance] setTintColor:[UIColor whiteColor]];
[[UINavigationController appearance] setBarStyle:UIBarStyleBlack];
```

UITabBarController con selección de color personalizada

UITabBarController en Swift 3 Cambie el color y el título de la imagen según la selección cambiando el color de la pestaña seleccionada.

```
import UIKit

class TabBarController: UITabBarController {

    override func viewDidLoad() {
        super.viewDidLoad()

        self.navigationController?.isNavigationBarHidden = true

        UITabBar.appearance().tintColor = UIColor.purple

        // set red as selected background color
        let numberOfItems = CGFloat(tabBar.items!.count)
        let tabBarItemSize = CGSize(width: tabBar.frame.width / numberOfItems, height:
tabBar.frame.height)
        tabBar.selectionIndicatorImage =
UIImage.imageWithColor(UIColor.lightText.withAlphaComponent(0.5), size:
tabBarItemSize).resizableImage(withCapInsets: UIEdgeInsets.zero)

        // remove default border
        tabBar.frame.size.width = self.view.frame.width + 4
        tabBar.frame.origin.x = -2
    }

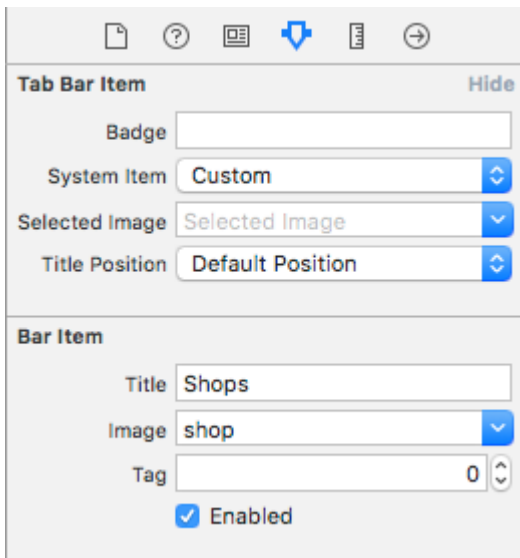
    override func viewWillAppear(_ animated: Bool) {
        // For Images
        let firstViewController:UIViewController = NotificationVC()
        // The following statement is what you need
        let customTabBarItem:UITabBarItem = UITabBarItem(title: nil, image: UIImage(named:
"notification@2x")?.withRenderingMode(UIImageRenderingMode.alwaysOriginal), selectedImage:
UIImage(named: "notification_sel@2x"))
        firstViewController.tabBarItem = customTabBarItem

        for item in self.tabBar.items! {
            let unselectedItem = [NSForegroundColorAttributeName: UIColor.white]
            let selectedItem = [NSForegroundColorAttributeName: UIColor.purple]

            item.setTitleTextAttributes(unselectedItem, for: .normal)
            item.setTitleTextAttributes(selectedItem, for: .selected)
        }
    }
}
```

```
extension UIImage {
    class func colorWithColor(_ color: UIColor, size: CGSize) -> UIImage {
        let rect: CGRect = CGRect(origin: CGPoint(x: 0,y :0), size: CGSize(width: size.width,
height: size.height))
        UIGraphicsBeginImageContextWithOptions(size, false, 0)
        color.setFill()
        UIRectFill(rect)
        let image: UIImage = UIGraphicsGetImageFromCurrentImageContext()!
        UIGraphicsEndImageContext()
        return image
    }
}
```

Elegir imagen para la barra de pestañas y configurar el título de la pestaña aquí





Selección de otra pestaña



Crear controlador de barra de pestañas programáticamente sin guión gráfico

```
class AppDelegate: UIResponder, UIApplicationDelegate {  
  
    var window: UIWindow?  
  
    var firstTabNavigationController : UINavigationController!  
    var secondTabNavigationControoller : UINavigationController!  
    var thirdTabNavigationController : UINavigationController!  
    var fourthTabNavigationControoller : UINavigationController!  
    var fifthTabNavigationController : UINavigationController!
```

```

func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplicationLaunchOptionsKey: Any]?) -> Bool {
    // Override point for customization after application launch.
    Fabric.with([Crashlytics.self])

    window = UIWindow(frame: UIScreen.main.bounds)

    window?.backgroundColor = UIColor.black

    let tabBarController = UITabBarController()

    firstTabNavigationController = UINavigationController.init(rootViewController:
FirstViewController())
    secondTabNavigationControoller = UINavigationController.init(rootViewController:
SecondViewController())
    thirdTabNavigationController = UINavigationController.init(rootViewController:
ThirdViewController())
    fourthTabNavigationControoller = UINavigationController.init(rootViewController:
FourthViewController())
    fifthTabNavigationController = UINavigationController.init(rootViewController:
FifthViewController())

    tabBarController.viewControllers = [firstTabNavigationController,
secondTabNavigationControoller, thirdTabNavigationController, fourthTabNavigationControoller,
fifthTabNavigationController]

    let item1 = UITabBarItem(title: "Home", image: UIImage(named: "ico-home"), tag: 0)
    let item2 = UITabBarItem(title: "Contest", image: UIImage(named: "ico-contest"), tag:
1)
    let item3 = UITabBarItem(title: "Post a Picture", image: UIImage(named: "ico-photo"),
tag: 2)
    let item4 = UITabBarItem(title: "Prizes", image: UIImage(named: "ico-prizes"), tag:
3)
    let item5 = UITabBarItem(title: "Profile", image: UIImage(named: "ico-profile"), tag:
4)

    firstTabNavigationController.tabBarItem = item1
    secondTabNavigationControoller.tabBarItem = item2
    thirdTabNavigationController.tabBarItem = item3
    fourthTabNavigationControoller.tabBarItem = item4
    fifthTabNavigationController.tabBarItem = item5

    UITabBar.appearance().tintColor = UIColor(red: 0/255.0, green: 146/255.0, blue:
248/255.0, alpha: 1.0)

    self.window?.rootViewController = tabBarController

    window?.makeKeyAndVisible()

    return true
}

```

Lea UITabBarController en línea: <https://riptutorial.com/es/ios/topic/2763/uitabBarController>

Capítulo 194: UITableView

Introducción

Una vista simple, ampliamente utilizada, pero muy poderosa que puede presentar datos en forma de lista usando filas y una sola columna. Los usuarios pueden desplazarse verticalmente a través de los elementos en una vista de tabla y, opcionalmente, manipular y seleccionar contenido.

Sintaxis

- - (CGFloat) tableView: (UITableView *) tableView heightForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (CGFloat) tableView: (UITableView *) tableView heightForHeaderInSection: (NSInteger) sección;
- - (CGFloat) tableView: (UITableView *) tableView heightForFooterInSection: (NSInteger) sección;
- - (UIView *) tableView: (UITableView *) tableView viewForHeaderInSection: (NSInteger) sección;
- - (UIView *) tableView: (UITableView *) tableView viewForFooterInSection: (NSInteger) sección;
- - (UITableViewCellAccessoryType) tableView: (UITableView *) tableView accessoryTypeForRowWithIndexPath: (NSIndexPath *) indexPath
- - (void) tableView: (UITableView *) tableView accessoryButtonTappedForRowWithIndexPath: (NSIndexPath *) indexPath;
- - (NSIndexPath *) tableView: (UITableView *) tableView willSelectRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSIndexPath *) tableView: (UITableView *) tableView willDeselectRowAtIndexPath: (NSIndexPath *) indexPath
- - (void) tableView: (UITableView *) tableView didSelectRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView didDeselectRowAtIndexPath: (NSIndexPath *) indexPath
- - (UITableViewCellEditingStyle) tableView: (UITableView *) tableView editingStyleForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSString *) tableView: (UITableView *) tableView

titleForDeleteConfirmationButtonForRowAtIndexPath: (NSIndexPath *) indexPath

- - (BOOL) tableView: (UITableView *) tableView shouldIndentWhileEditingRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView willBeginEditingRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView didEndEditingRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSIndexPath *) tableView: (UITableView *) tableView targetIndexPathForMoveFromRowAtIndexPath: (NSIndexPath *) sourceIndexPath toProposedIndexPath: (NSIndexPath *) proposedDestinationIndexPath;
- - (NSInteger) tableView: (UITableView *) tableView indentationLevelForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSInteger) tableView: (UITableView *) tableView numberOfRowsInSection: (NSInteger) sección;
- - (UITableViewCell *) tableView: (UITableView *) tableView cellForRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSInteger) numberOfSectionsInTableView: (UITableView *) tableView;
- - (NSString *) tableView: (UITableView *) tableView titleForHeaderInSection: (NSInteger) sección; // estilo de fuente fija. Usa vista personalizada (UILabel) si quieres algo diferente
- - (NSString *) tableView: (UITableView *) tableView titleForFooterInSection: (NSInteger) sección;
- - (BOOL) tableView: (UITableView *) tableView canEditRowAtIndexPath: (NSIndexPath *) indexPath;
- - (BOOL) tableView: (UITableView *) tableView canMoveRowAtIndexPath: (NSIndexPath *) indexPath;
- - (NSArray *) sectionIndexTitlesForTableView: (UITableView *) tableView;
- - (NSInteger) tableView: (UITableView *) tableView sectionForSectionIndexTitle: (NSString *) title atIndex: (NSInteger) index;
- - (void) tableView: (UITableView *) tableView commitEditingStyle: (UITableViewCellEditingStyle) editingStyle forRowAtIndexPath: (NSIndexPath *) indexPath;
- - (void) tableView: (UITableView *) tableView moveRowAtIndexPath: (NSIndexPath *) sourceIndexPath toIndexPath: (NSIndexPath *) destinationIndexPath;

Observaciones

`UITableView` es una subclase de `UIScrollView`. Las clases que siguen el protocolo `UITableViewDelegate` también siguen el protocolo `UIScrollViewDelegate`. `UITableView` puede ser útil para mostrar listas largas o indeterminadas a través de sus celdas, mientras que `UIScrollView` es mejor cuando el tamaño de las vistas que se muestran es conocido de antemano.

Examples

Células de auto-tamaño

En iOS 8 Apple introdujo la celda de auto dimensionamiento. Diseñe sus `UITableViewCell`s con `Autolayout` explícitamente y `UITableView` se encarga del resto por usted. Altura de la fila se calcula automáticamente, por defecto `rowHeight` valor es `UITableViewAutomaticDimension`.

`UITableView` propiedad `estimatedRowHeight` se utiliza cuando la auto-dimensionamiento celular está calculando.

Cuando crea una celda de vista de tabla de auto-dimensionamiento, necesita establecer esta propiedad y usar restricciones para definir el tamaño de la celda.

- *Apple, Documentación UITableView*

```
self.tableView.estimatedRowHeight = 44.0
```

Tenga en cuenta que la altura del delegado de `heightForRowAtIndexPath` es *necesaria* si desea tener una altura dinámica para todas las celdas. Simplemente configure la propiedad anterior cuando sea necesario y antes de volver a cargar o cargar la vista de tabla. Sin embargo, puede establecer la altura de celdas específicas mientras otras dinámicas se realizan a través de la siguiente función:

Rápido

```
override func tableView(tableView: UITableView, heightForRowAtIndexPath indexPath:
NSIndexPath) -> CGFloat {
    switch indexPath.section {
    case 1:
        return 60
    default:
        return UITableViewAutomaticDimension
    }
}
```

C objetivo

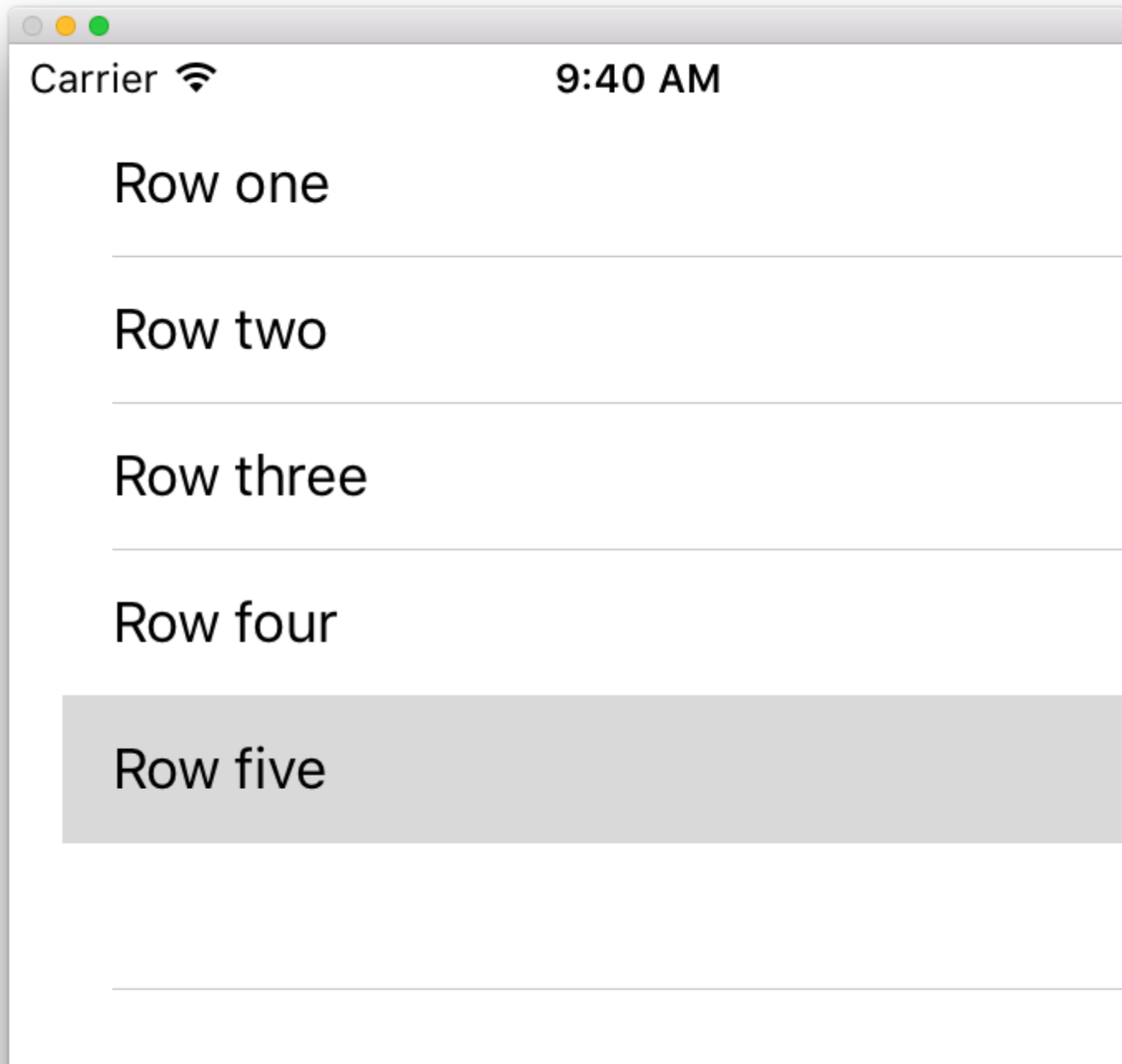
```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    switch (indexPath.section) {
    case 1:
        return 60;
    default:
        return UITableViewAutomaticDimension;
    }
}
```



```
}
```

Creando un UITableView

Una vista de tabla es una lista de filas que se pueden seleccionar. Cada fila se rellena desde un origen de datos. Este ejemplo crea una vista de tabla simple en la que cada fila es una sola línea de texto.



Agrega un UITableView a tu Guión Gráfico

Aunque hay varias maneras de crear un `UITableView`, una de las más fáciles es agregar uno a un Storyboard. Abra su Guión gráfico y arrastre un `UITableView` a su `UIViewController`. Asegúrese de usar el diseño automático para alinear correctamente la tabla (sujete los cuatro lados).

Poblando tu tabla con datos

Para poder visualizar el contenido dinámicamente (es decir, cargarlo desde un origen de datos como una matriz, un modelo de Core Data, un servidor en red, etc.) en su vista de tabla, necesita configurar el origen de datos.

Creando una fuente de datos simple

Una fuente de datos podría, como se indicó anteriormente, ser cualquier cosa con datos. Depende totalmente de usted cómo formatearlo y qué contiene. El único requisito es que debe poder leerlo más tarde para poder rellenar cada fila de su tabla con datos cuando sea necesario.

En este ejemplo, solo estableceremos una matriz con algunas cadenas (texto) como nuestra fuente de datos:

Rápido

```
let mydataArray: [String] = ["Row one", "Row two", "Row three", "Row four", "Row five"]
```

C objetivo

```
// You'll need to define this variable as a global variable (like an @property) so that you  
// can access it later when needed.  
NSArray *mydataArray = @[@"Row one", @"Row two", @"Row three", @"Row four", @"Row five"];
```

Configurando su fuente de datos en su View Controller

Asegúrese de que su controlador de vista cumpla con el protocolo `UITableViewDataSource`.

Rápido

```
class ViewController: UIViewController, UITableViewDataSource {
```

C objetivo

```
@interface ViewController : UIViewController <UITableViewDataSource>
```

Tan pronto como su controlador de vista haya declarado que se **ajustará** a la `UITableViewDataSource` (eso es lo que acabamos de hacer anteriormente), *debe* implementar al menos los siguientes métodos en su clase de controlador de vista:

- `tableView:numberOfRowsInSection`, esto le pregunta cuántas filas debe tener su vista de tabla.

```
// Swift  
  
func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
```

```
return self.mydataArray.count
}
```

- `tableView:cellForRowAtIndexPath` , solicita que cree y devuelva una celda para cada fila que especificó en `tableView:numberOfRowsInSection` . Entonces, si dijiste que necesitabas 10 filas, este método se llamará diez veces para cada fila, y necesitas crear una celda para cada una de esas filas.

```
// Swift

func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
    // Create a new cell here. The reuseIdentifier needs to match the reuse
    // identifier from the cell in your Storyboard
    let cell: UITableViewCell =
    tableView.dequeueReusableCellWithIdentifier(reuseIdentifier) as UITableViewCell!

    // Set the label on your cell to the text from your data array
    cell.textLabel?.text = self.mydataArray[indexPath.row]

    return cell
}
```

ADVERTENCIA : NO puede devolver cero para ninguna celda en

`cellForRowAtIndexPath`: Esto hará que su aplicación se bloquee y verá el siguiente error en la consola:

```
Uncaught exception 'NSInternalInconsistencyException', reason: 'UITableView
dataSource must return a cell from tableView:cellForRowAtIndexPath:'
```

Conectando la fuente de datos de la vista de tabla a su controlador de vista

Puede hacerlo a través del código configurando la propiedad `dataSource` su tabla para que sea `self` en su controlador de vista. O puede seleccionar su vista de tabla en su guión gráfico, abrir el inspector de atributos, seleccionar el panel "Outlets" y arrastrar desde `dataSource` a su controlador de vista (**NOTA** : asegúrese de conectarse a `UIViewCONTROLLER`, **no** a `UIView` u otro objeto en su Controlador `UIView`).

Manejo de selecciones de filas

Cuando un usuario toca una fila en su vista de tabla, generalmente querrá hacer algo: responder. En muchas aplicaciones, cuando toca en una fila, se muestra más información sobre el elemento que tocó. Piense en la aplicación de Mensajes: cuando toca en la fila que muestra uno de sus contactos, la conversación con esa persona se muestra en la pantalla.

Para hacerlo, debe cumplir con el protocolo `UITableViewDelegate` . Hacerlo es similar a cumplir con el protocolo de origen de datos. Esta vez, sin embargo, solo lo agregará junto a

UITableViewDataSource y lo separará con una coma. Entonces debería verse así:

Rápido

```
class ViewController: UIViewController, UITableViewDataSource, UITableViewDelegate {
```

C objetivo

```
@interface ViewController : UIViewController <UITableViewDataSource, UITableViewDelegate>
```

No se requieren métodos para implementar para el delegado de la vista de tabla. Sin embargo, para manejar las selecciones de filas necesitará usar el siguiente método:

- `tableView:didSelectRowAtIndexPath` , esto se llama cada vez que se `tableView:didSelectRowAtIndexPath` una fila, lo que le permite hacer algo en respuesta. Para nuestro ejemplo, simplemente imprimiremos una declaración de confirmación en el registro de Xcode.

```
// Swift

func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
    print("You tapped cell number \(indexPath.row).")
}

// Objective-C

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath {
    NSLog(@"You tapped cell number %ld.", (long)indexPath.row);
}
```

La solución definitiva

Vea a continuación la configuración completa con solo el código, sin explicación.

Rápido

```
import UIKit
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {

    // Data model: These strings will be the data for the table view cells
    let mydataArray: [String] = ["Row one", "Row two", "Row three", "Row four", "Row five"]

    // cell reuse id (cells that scroll out of view can be reused)
    let reuseIdentifier = "cell"

    // don't forget to hook this up from the storyboard
    @IBOutlet var myTableView: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()
    }
}
```

```

        // Register the table view cell class and its reuse id
        myTableView.registerClass(UITableViewCell.self, forCellReuseIdentifier:
cellReuseIdentifier)

        // This view controller itself will provide the delegate methods and row data for the
table view.
        myTableView.delegate = self
        myTableView.dataSource = self
    }

    // number of rows in table view
    func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return self.mydataArray.count
    }

    // create a cell for each table view row
    func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {

        // create a new cell if needed or reuse an old one
        let cell:UITableViewCell =
tableView.dequeueReusableCellWithIdentifier(cellReuseIdentifier) as UITableViewCell!

        // set the text from the data model
        cell.textLabel?.text = self.mydataArray[indexPath.row]

        return cell
    }

    // method to run when table view cell is tapped
    func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
        print("You tapped cell number \(indexPath.row).")
    }
}

```

C objetivo

ViewController.h

```

#import <UIKit/UIKit.h>

@interface ViewController: UIViewController <UITableViewDelegate, UITableViewDataSource> {
    IBOutlet UITableView *myTableView;
    NSArray *mydataArray;
}

@end

```

ViewController.m

```

#import "ViewController.h"

// cell reuse id (cells that scroll out of view can be reused)
NSString * _Nonnull cellReuseIdentifier = @"cell";

@implementation ViewController

```

```

- (void)viewDidLoad {
    [super viewDidLoad];

    // Data model: These strings will be the data for the table view cells
    mydataArray = @[@"Row one", @"Row two", @"Row three", @"Row four", @"Row five"];

    // Register the table view cell class and its reuse id
    [myTableView registerClass:[UITableViewCell class]
    forCellReuseIdentifier:cellReuseIdentifier];

    // This view controller itself will provide the delegate methods and row data for the
    table view.
    myTableView.delegate = self;
    myTableView.dataSource = self;
}

// number of rows in table view
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    return mydataArray.count;
}

// create a cell for each table view row
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath
*)indexPath {
    // create a new cell if needed or reuse an old one
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:cellReuseIdentifier];

    // set the text from the data model
    cell.textLabel.text = mydataArray[indexPath.row];

    return cell;
}

// method to run when table view cell is tapped
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath{
    NSLog(@"You tapped cell number %ld.", (long)indexPath.row);
}

@end

```

Delegado y fuente de datos

El `UITableViewDelegate` se utiliza para controlar cómo se muestra la tabla, y el `UITableViewDataSource` se utiliza para definir los `UITableView` del `UITableView`. Hay dos métodos requeridos y muchos opcionales que se pueden usar para personalizar el tamaño, las secciones, los encabezados y las celdas en el `UITableView`.

UITableViewDataSource

Métodos requeridos

`numberOfRowsInSection`: este método define cuántas celdas se mostrarán en cada sección de la vista de tabla.

C objetivo

```

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    // Return the number of rows for the table view. Usually populated from an array,
    // or can be statically defined.
    return self.myArray.count;
}

```

Swift 3

```

func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // Return the number of rows for the table view. Usually populated from an array,
    // or can be statically defined.
    return self.myArray.count
}

```

`cellForRowAtIndexPath:` este método es donde se crean y configuran las celdas de `UITableView`. Debe devolver un `UITableViewCell` o una subclase personalizada.

Nota: El uso de `dequeueReusableCellWithIdentifier:forIndexPath:` requiere que la clase o punta se haya registrado para ese identificador utilizando el `UITableView`

`registerClass:forCellReuseIdentifier:` o `registerNib:forCellReuseIdentifier:` métodos. Por lo general, esto se hará en el `UIViewController` 's `viewDidLoad` método.

C objetivo

```

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
    MyCustomCell *cell = [tableView dequeueReusableCellWithIdentifier:@"MyCustomCell"
                                                                forIndexPath:indexPath];

    // All additional customization goes here
    cell.titleLabel.text = [NSString stringWithFormat:@"Title Row %lu", indexPath.row];

    return cell;
}

```

Swift 3

```

func tableView(_ tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "MyCustomCell",
forIndexPath:indexPath)

    // All additional customization goes here
    cell.titleLabel.text = String(format: "Title Row %lu", indexPath.row)

    return cell
}

```

Métodos opcionales

`titleForHeaderInSection:` define una cadena como el título para cada encabezado de sección en la vista de tabla. Este método solo permite cambiar el título; se puede hacer una mayor personalización definiendo la vista para el encabezado.

C objetivo

```
- (NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section {
    switch(section) {
        case 0:
            return @"Title 1";
            break;

        case 1:
            return @"Title 2";
            break;

        default:
            return nil;
            break;
    }
}
```

Swift 3

```
func tableView(_ tableView: UITableView, titleForHeaderInSection section: Int) -> String? {
    switch section {
        case 0:
            return "Title 1"
        case 1:
            return "Title 2"
        default:
            return nil
    }
}
```

titleForFooterInSection: define una cadena como el título para cada encabezado de sección en la vista de tabla.

C objetivo

```
- (NSString *)tableView:(UITableView *)tableView titleForFooterInSection:(NSInteger)section {
    return @"Footer text";
}
```

Swift 3

```
func tableView(_ tableView: UITableView, titleForFooterInSection section: Int) -> String? {
    return "Footer text"
}
```

canEditRowAtIndexPath: utiliza para determinar si la IU de edición debe mostrarse para la fila especificada. Debe devolver YES si la fila especificada se puede eliminar o agregar.

C objetivo

```
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndexPath:(NSIndexPath *)indexPath {
    return YES;
}
```



```
}
```

Swift 3

```
func tableView(_ tableView: UITableView, canEditRowAtIndexPath indexPath: NSIndexPath) -> Bool
{
    return true
}
```

`commitEditingStyle:forRowAtIndexPath` Debe realizar el trabajo necesario para controlar la adición o eliminación de la fila especificada. Por ejemplo, elimine la celda del `UITableView` con animación y elimine el objeto asociado del modelo de datos de la tabla.

C objetivo

```
- (void)tableView:(UITableView *)tableView
commitEditingStyle:(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:(NSIndexPath
*)indexPath {
    switch (editingStyle) {
        case UITableViewCellEditingStyleInsert:
            // Insert new data into the backing data model here
            [self insertNewDataIntoDataModel];
            [tableView insertRowsAtIndexPaths:@[indexPath]
withRowAnimation:UITableViewRowAnimationAutomatic];
            break;
        case UITableViewCellEditingStyleDelete:
            [self removeDataFromDataModelAtIndex:indexPath.row];
            [tableView deleteRowsAtIndexPaths:@[indexPath]
withRowAnimation:UITableViewRowAnimationAutomatic];
            break;
        default:
            // Nothing to perform if the editingStyle was neither Insert or Delete
            break;
    }
}
```

Swift 3

```
func tableView(_ tableView: UITableView, commitEditingStyle editingStyle:
UITableViewCellEditingStyle, forRowAtIndexPath indexPath: NSIndexPath) {
    switch editingStyle {
        case .Insert:
            self.insertNewDataIntoDataModel()
            tableView.insertRowsAtIndexPaths([indexPath], withRowAnimation:.Automatic)
        case .Delete:
            self.removeDataFromDataModelAtIndex(indexPath.row)
            tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation:.Automatic)
        default:
            // Nothing to perform if the editingStyle was neither Insert or Delete
    }
}
```

`editActions:forRowAt` Permite la capacidad de agregar acciones o botones adicionales

al modo de edición de una fila dentro de una vista de `UITableView`. Por ejemplo, si desea dos botones, un botón de edición y eliminación cuando el usuario desliza para editar la fila, entonces usaría este método.

Swift 3

```
override func tableView(_ tableView: UITableView, editActionsForRowAt indexPath: IndexPath) -> [UITableViewRowAction]? {
    // In the handler you will get passed the action as well as the indexPath for
    // the row that is being edited
    let editAction = UITableViewRowAction(style: .normal, title: "Edit", handler: { [unowned self] action, indexPath in
        // Do something when edit is tapped
    })

    // Change the color of the edit action
    editAction.backgroundColor = UIColor.blue

    let deleteAction = UITableViewRowAction(style: .destructive, title: "Delete", handler: { [unowned self] action, indexPath in
        // Handel the delete event
    })

    return [deleteAction, editAction]
}
```

UITableViewDelegate

Todos los métodos en `UITableViewDelegate` son opcionales, pero un delegado que los implemente habilitará características adicionales para el `UITableView`.

`numberOfSectionsInTableView`: forma predeterminada, devuelve 1, pero el soporte de múltiples secciones se habilita al devolver un número diferente de secciones.

C objetivo

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return self.numSections;
}
```

Swift 3

```
func numberOfSectionsInTableView(_ tableView: UITableView) -> Int {
    return self.numSections
}
```

`viewForHeaderInSection` Permite la configuración de una vista personalizada como encabezado de la sección.

C objetivo

```

- (UIView *)tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger)section {

    UIView *view = [[UIView alloc] initWithFrame:CGRectMake(0, 0,
CGRectGetWidth(tableView.frame), 22)];
    view.backgroundColor = [UIColor groupTableViewBackgroundColor];

    UILabel *label = [[UILabel alloc] init];
    label.font = [UIFont systemFontOfSize:12];
    label.textColor = [UIColor darkGrayColor];

    switch (section) {
        case 1: {
            label.text = @"Title";
            label.frame = labelFrame;

            UIButton *more = [[UIButton alloc] initWithFrame:btnFrame];
            [more setTitle:@"See more" forState:UIControlStateNormal];
            [more.titleLabel setFont:[UIFont systemFontOfSize:12]];
            [view addSubview:more];
        } break;

        default:
            label.frame = CGRectMake(0, 0, 0, 0);
            break;
    }

    [view addSubview:label];
    return view;
}

```

Swift 3

```

func tableView(_ tableView: UITableView, viewForHeaderInSection section: Int) -> UIView? {
    let view = UIView(frame: CGRect(x: 0, y: 0, width: tableView.frame.size.width, height:
22))
    view.backgroundColor = UIColor.groupTableViewBackgroundColor()

    let label = UILabel()
    label.font = UIFont.systemFont(ofSize: 12)
    label.textColor = UIColor.darkGrayColor()

    switch section {
        case 1:
            label.text = "Title"
            label.frame = labelFrame

            let more = UIButton(frame: btnFrame)
            more.setTitle("See more", forState:.Normal)
            view.addSubview(more)

        default:
            label.frame = CGRect.zero
    }

    view.addSubview(label)
    return view;
}

```

`heightForRowAtIndexPath:` **defina la altura de cada celda en la vista de tabla.**

C objetivo

```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    return 44;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, heightForRowAtIndexPath indexPath: NSIndexPath) ->
CGFloat {
    return 44
}
```

`heightForHeaderInSection:` y `heightForFooterInSection` Defina la altura del encabezado y pie de página de cada sección en la vista de tabla

C objetivo

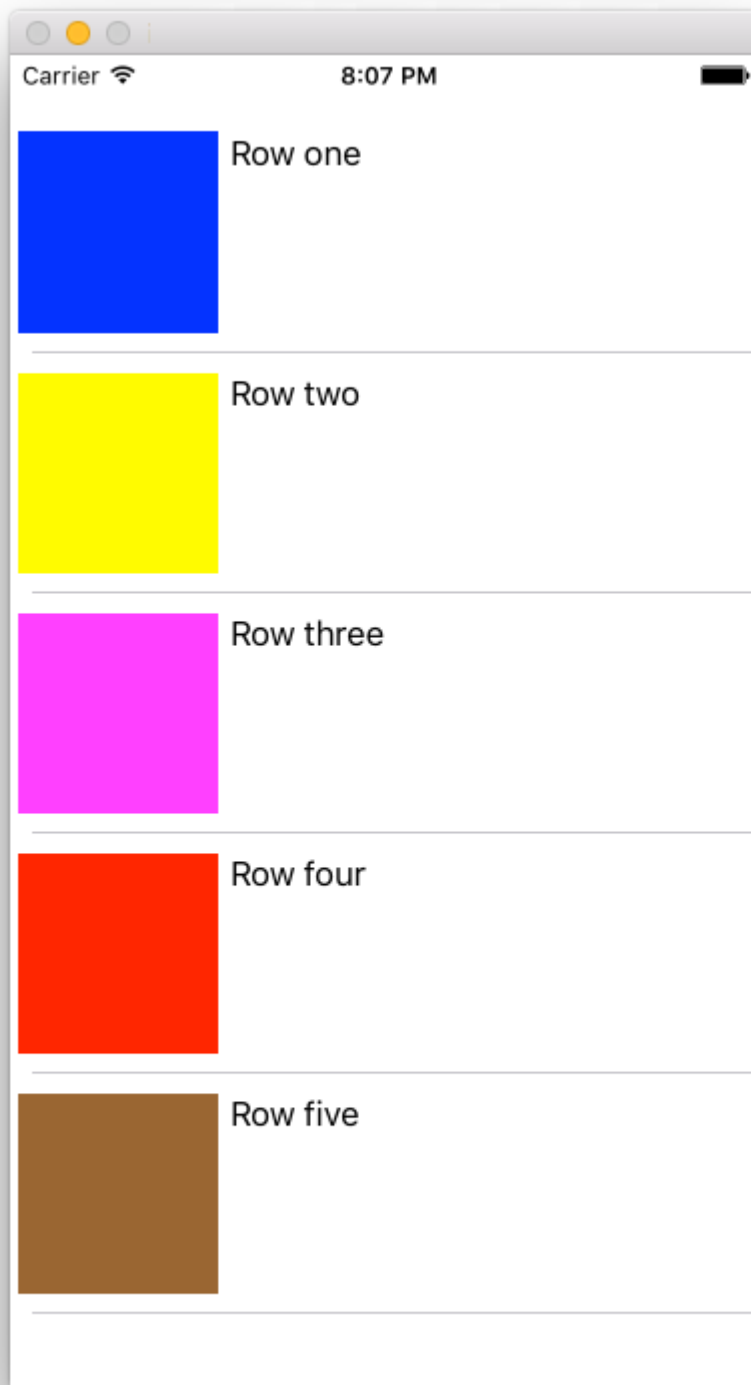
```
- (CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section {
    return 33;
}
```

Swift 3

```
func tableView(_ tableView: UITableView, heightForHeaderInSection section: Int) -> CGFloat {
    return 33
}
```

Celdas personalizadas

La personalización de un `UITableViewCell` puede permitir `UITableViewCell` muy potentes, dinámicas y sensibles. Con una amplia personalización y en combinación con otras técnicas, puede hacer cosas como: actualizar propiedades específicas o elementos de la interfaz a medida que cambian, animar o dibujar cosas en la celda, cargar videos de manera eficiente a medida que el usuario se desplaza, o incluso mostrar imágenes a medida que se descargan de un red. Las posibilidades aquí son casi infinitas. A continuación se muestra un ejemplo simple de cómo puede verse una celda personalizada.



Esta sección cubre los conceptos básicos y, con suerte, se ampliará para detallar procesos más complejos como los descritos anteriormente.

Creación de su celda personalizada

Primero, cree una nueva subclase de `UITableViewCell` (cree una nueva Clase Cocoa Touch en Xcode y establezca `UITableViewCell` como la superclase). A continuación se muestra el aspecto que puede tener su código después de la subclasificación.

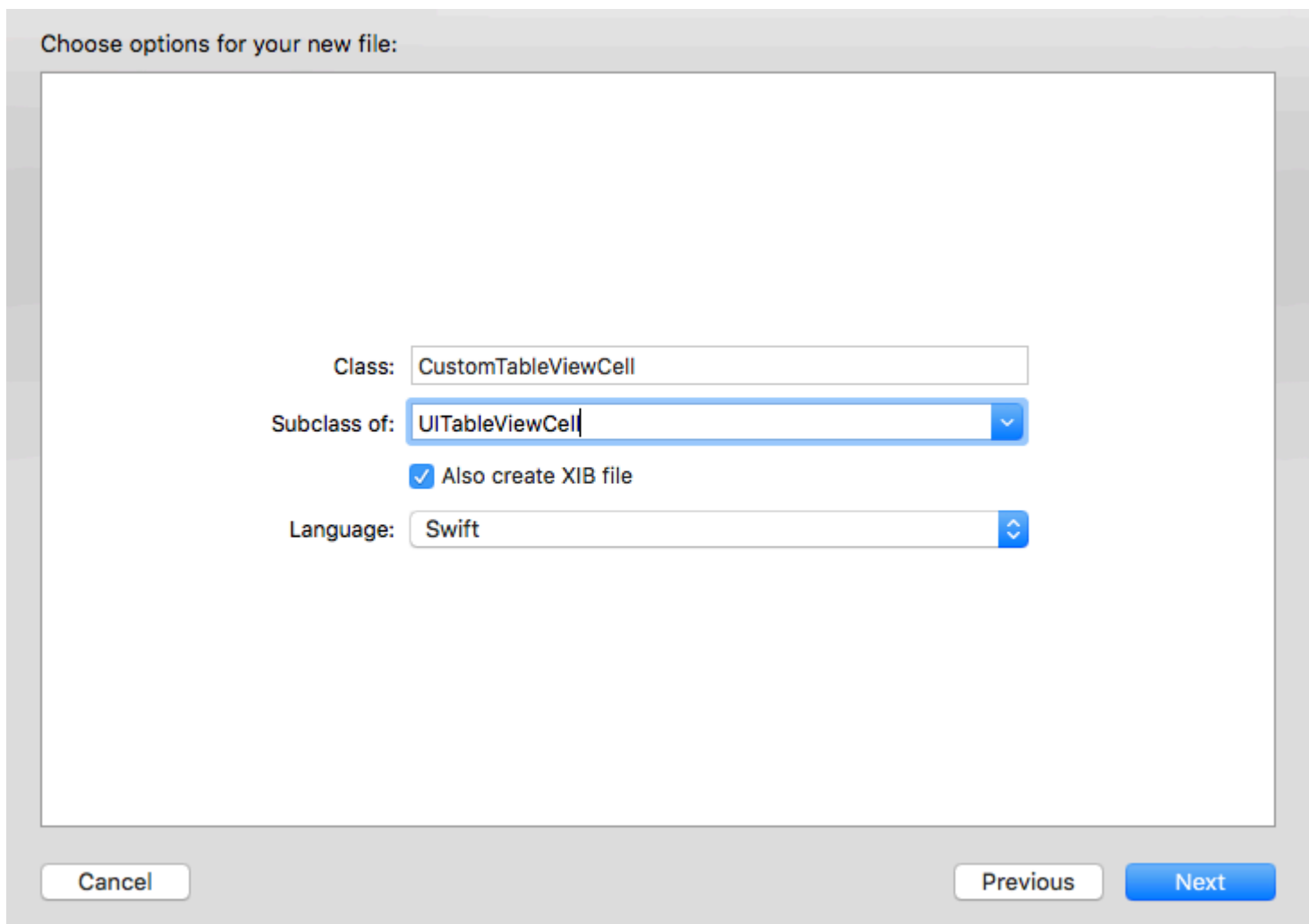
Rápido

```
class CustomTableViewCell: UITableViewCell {
    static var identifier: String {
        return NSStringFromClass(self)
    }

    var customLabel: UILabel!

    override func awakeFromNib() {
        super.awakeFromNib()
        // Initialization code
        customLabel = UILabel(frame: CGRect(x: 0, y: 0, width: contentView.frame.width,
height: contentView.frame.height))
        customLabel.textAlignment = .center
        contentView.addSubview(customLabel)
    }
}
```

Opcionalmente, marque 'Crear también un archivo XIB' al crear su nuevo archivo para personalizarlo mediante el Interface Builder. En el caso de que lo haga, conecte `customLabel` como `@IBOutlet`



En un `UIViewController` contenga `tableView`, registre la nueva clase de celda personalizada (ver más abajo). Tenga en cuenta que esto solo es necesario si no diseña la celda con un Guión gráfico en la interfaz de la vista de tabla.

Rápido

```
override func viewDidLoad() {
    super.viewDidLoad()

    // Register Cell Class
    tableView.register(CustomTableViewCell.self, forCellReuseIdentifier:
CustomTableViewCell.identifier)
}
```

Si elige usar un archivo XIB, `registerNib` lugar:

Rápido

```
// Register Nib
tableView.register(UINib(nibName: CustomTableViewCell.identifier, bundle: nil),
forCellReuseIdentifier: CustomTableViewCell.identifier)
```

Ahora que su `tableView` sabe acerca de su celda personalizada, puede quitarla de la `cellForRowAtIndexPath` en `cellForRowAtIndexPath` :

Rápido

```
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
UITableViewCell {
    // Load the CustomTableViewCell. Make sure the identifier supplied here matches the one
from your cell
    let cell: CustomTableViewCell =
tableView.dequeueReusableCellWithIdentifier(CustomTableViewCell.identifier) as!
CustomTableViewCell

    // This is where the magic happens - setting a custom property on your very own cell
cell.customLabel.text = "My Custom Cell"

    return cell
}
```

Expandiendo y colapsando UITableViewCells

En su Guión gráfico, agregue un objeto `UITableView` en su `UIViewController` y deje que cubra toda la vista. Configure las conexiones `UITableViewDataSource` y `UITableViewDelegate` .

C objetivo

En tu archivo `.h`

```
NSMutableArray *arrayForBool;
NSMutableArray *sectionTitleArray;
```

En su archivo `.m`

```
- (void)viewDidLoad {
```

```

[super viewDidLoad];

arrayForBool = [[NSMutableArray alloc] init];
sectionTitleArray = @[@"Sam",@"Sanju",@"John",@"Staffy"];

for (int i=0; i<[sectionTitleArray count]; i++) {
    [arrayForBool addObject:[NSNumber numberWithInt:NO]];
}

_tableView.dataSource = self;
_tableView.delegate = self;
}

// Declare number of rows in section
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    if ([arrayForBool objectAtIndex:section] boolValue) {
        return section+2;
    } else {
        return 0;
    }
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {

static NSString *cellid=@"hello";
UITableViewCell *cell=[tableView dequeueReusableCellWithIdentifier:cellid];
if (cell==nil) {
    cell=[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleSubtitle
reuseIdentifier:cellid];
}
    BOOL manyCells = [[arrayForBool objectAtIndex:indexPath.section] boolValue];

    /** If the section supposed to be closed*****/
    if(!manyCells){
        cell.backgroundColor=[UIColor clearColor];
        cell.textLabel.text=@"";
    }
    /** If the section supposed to be Opened*****/
    else{
        cell.textLabel.text=[NSString stringWithFormat:@"%d", [sectionTitleArray
objectAtIndex:indexPath.section], indexPath.row+1];
        cell.backgroundColor=[UIColor whiteColor];
        cell.selectionStyle=UITableViewCellSelectionStyleNone ;
    }
cell.textLabel.textColor=[UIColor blackColor];

    /** Add a custom Separator with cell*/
    UIView* separatorLineView = [[UIView alloc] initWithFrame:CGRectMake(15, 40,
_expandableTableView.frame.size.width-15, 1)];
separatorLineView.backgroundColor = [UIColor blackColor];
[cell.contentView addSubview:separatorLineView];
return cell;
}

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
return [sectionTitleArray count];
}

```



```

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
    /***** Close the section, once the data is selected
    *****/
    [arrayForBool replaceObjectAtIndex:indexPath.section withObject:[NSNumber numberWithInt:NO]];

    [_expandableTableView reloadSections:[NSIndexPath indexPathWithIndex:indexPath.section]
withRowAnimation:UITableViewRowAnimationAutomatic];
}

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath
*)indexPath
{
    if ([[arrayForBool objectAtIndex:indexPath.section] boolValue]) {
        return 40;
    }
    return 0;
}

- (UIView *)tableView:(UITableView *)tableView viewForHeaderInSection:(NSInteger)section
{
    UIView *sectionView=[[UIView alloc]initWithFrame:CGRectMake(0, 0, 280,40)];
    sectionView.tag=section;
    UILabel *viewLabel=[[UILabel alloc]initWithFrame:CGRectMake(10, 0,
_expandableTableView.frame.size.width-10, 40)];
    viewLabel.backgroundColor=[UIColor clearColor];
    viewLabel.textColor=[UIColor blackColor];
    viewLabel.font=[UIFont systemFontOfSize:15];
    viewLabel.text=[NSString stringWithFormat:@"List of %@",[sectionTitleArray
objectAtIndex:section]];
    [sectionView addSubview:viewLabel];
    /***** Add a custom Separator with Section view *****/
    UIView* separatorLineView = [[UIView alloc] initWithFrame:CGRectMake(15, 40,
_expandableTableView.frame.size.width-15, 1)];
    separatorLineView.backgroundColor = [UIColor blackColor];
    [sectionView addSubview:separatorLineView];

    /***** Add UITapGestureRecognizer to SectionView *****/

    UITapGestureRecognizer *headerTapped = [[UITapGestureRecognizer alloc] initWithTarget:self
action:@selector(sectionHeaderTapped:)];
    [sectionView addGestureRecognizer:headerTapped];

    return sectionView;
}

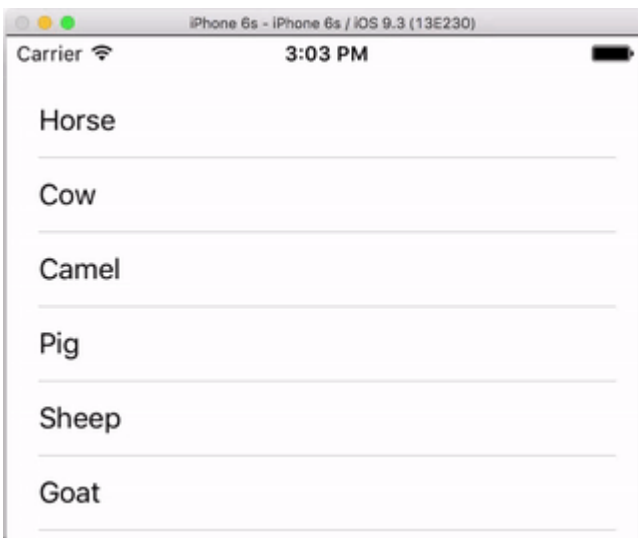
- (void)sectionHeaderTapped:(UITapGestureRecognizer *)gestureRecognizer{
    NSIndexPath *indexPath = [NSIndexPath indexPathForRow:0 inSection:gestureRecognizer.view.tag];
    if (indexPath.row == 0) {
        BOOL collapsed = [[arrayForBool objectAtIndex:indexPath.section] boolValue];
        for (int i=0; i<[sectionTitleArray count]; i++) {
            if (indexPath.section==i) {
                [arrayForBool replaceObjectAtIndex:i withObject:[NSNumber
numberWithBool:!collapsed]];
            }
        }
    }
}

```

```
    }  
  }  
  [_expandableTableView reloadData:[NSIndexSet  
indexSetWithIndex:gestureRecognizer.view.tag]  
withRowAnimation:UITableViewRowAnimationAutomatic];  
}  
}
```

Deslizar para eliminar filas

Siempre creo que es bueno tener un ejemplo muy simple y autónomo para que no se suponga nada cuando estoy aprendiendo una nueva tarea. Esta respuesta es para borrar filas de `UITableView` . El proyecto se comporta así:



Este proyecto se basa en el [ejemplo de UITableView para Swift](#) .

Agrega el código

Cree un nuevo proyecto y reemplace el código `ViewController.swift` con lo siguiente.

```
import UIKit  
class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource {  
  
    // These strings will be the data for the table view cells  
    var animals: [String] = ["Horse", "Cow", "Camel", "Pig", "Sheep", "Goat"]  
  
    let reuseIdentifier = "cell"  
  
    @IBOutlet var tableView: UITableView!  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
  
        // It is possible to do the following three things in the Interface Builder  
        // rather than in code if you prefer.  
        self.tableView.registerClass(UITableViewCell.self, forCellReuseIdentifier:  
        reuseIdentifier)  
    }  
}
```

```

        tableView.delegate = self
        tableView.dataSource = self
    }

    // number of rows in table view
    func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return self.animals.count
    }

    // create a cell for each table view row
    func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
    UITableViewCell {

        let cell:UITableViewCell =
self.tableView.dequeueReusableCellWithIdentifier(cellReuseIdentifier) as UITableViewCell!

        cell.textLabel?.text = self.animals[indexPath.row]

        return cell
    }

    // method to run when table view cell is tapped
    func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIndexPath) {
        print("You tapped cell number \(indexPath.row).")
    }

    // this method handles row deletion
    func tableView(tableView: UITableView, commitEditingStyle editingStyle:
    UITableViewCellEditingStyle, forRowAtIndexPath indexPath: NSIndexPath) {

        if editingStyle == .Delete {

            // remove the item from the data model
            animals.removeAtIndex(indexPath.row)

            // delete the table view row
            tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Fade)

        } else if editingStyle == .Insert {
            // Not used in our example, but if you were adding a new row, this is where you
            would do it.
        }
    }
}

```

El método de clave única en el código anterior que permite la eliminación de filas es el último. Aquí está de nuevo por énfasis:

```

func tableView(tableView: UITableView, commitEditingStyle editingStyle:
UITableViewCellEditingStyle, forRowAtIndexPath indexPath: NSIndexPath) {

    if editingStyle == .Delete {

        // remove the item from the data model
        animals.removeAtIndex(indexPath.row)

        // delete the table view row
        tableView.deleteRowsAtIndexPaths([indexPath], withRowAnimation: .Fade)
    }
}

```

```
}
```

Guión gráfico

Agregue un `UITableView` al controlador de vista en el guión gráfico. Utilice el diseño automático para fijar los cuatro lados de la vista de tabla a los bordes del Controlador de vista. Controle el arrastre desde la vista de tabla en el guión gráfico a `@IBOutlet var tableView: UITableView!` línea en el código.

Terminado

Eso es todo. Debería poder ejecutar su aplicación ahora y eliminar filas al deslizar hacia la izquierda y tocar "Eliminar".

Notas

- Esto solo está disponible en iOS 8. Vea [esta respuesta](#) para más detalles.
- Si necesita cambiar la cantidad de botones que se muestran o el texto del botón, consulte [esta respuesta](#) para obtener más detalles.

Otras lecturas

- [Cómo hacer una celda de vista de tabla deslizable con acciones - Sin nueces con vistas de desplazamiento](#)
- [Documentación de Apple](#)

Líneas separadoras

Edición del ancho de las líneas de separación

Puede establecer que las líneas separadoras de su vista de tabla se extiendan a varios anchos a través de la tabla cambiando la propiedad `layoutMargins:` en su (s) celda (s). Esto se puede lograr de varias maneras.

Cambio de las líneas de separación para celdas específicas

En el método `cellForRowAtIndexPath:` fuente de datos de la vista de tabla o en el método `willDisplayCell:` establezca la propiedad `layoutMargins:` propiedad de la `UIEdgeInsetsZero` en `UIEdgeInsetsZero` (se extiende hasta el ancho completo de la tabla), o lo que desee aquí.

C objetivo

```
[cell setLayoutMargins:UIEdgeInsetsZero];  
  
// May also use separatorInset  
[cell setSeparatorInset:UIEdgeInsetsZero];
```

Rápido

```
func tableView(tableView: UITableView, willDisplayCell cell: UITableViewCell,  
forRowAtIndexPath indexPath: NSIndexPath) {  
    cell.separatorInset = UIEdgeInsetsZero  
    cell.layoutMargins = UIEdgeInsetsZero  
}  
  
func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->  
UITableViewCell  
{  
    cell.separatorInset = UIEdgeInsetsZero  
    cell.layoutMargins = UIEdgeInsetsZero  
}
```

Eliminar todas las líneas de separación

Las líneas grises finas entre cada celda pueden no ser exactamente el aspecto que está buscando. Es bastante sencillo ocultarlos de la vista.

En su abarcando `UIViewController`'s `viewDidLoad`: método de añadir el siguiente código. También puede establecer esta propiedad en cualquier momento antes de cargar o volver a cargar la vista de tabla (no necesariamente tiene que estar en el método `viewDidLoad`: .

Rápido:

```
tableView.separatorStyle = .None
```

C objetivo:

```
tableView.separatorStyle = UITableViewCellSeparatorStyleNone;
```

Alternativamente, la propiedad se puede cambiar en su Guión gráfico o XIB seleccionando su `tableView` y configurando el `separator` (bajo el inspector de atributos) en `None` .

Ocultar el exceso de líneas de separación

Puede ocultar las líneas separadoras de `UITableViewCell` para celdas vacías configurando una vista de pie de página vacía en la parte inferior de un `UITableView`:

Rápido

```
tableView.tableFooterView = UIView()
```

C objetivo

```
tableView.tableFooterView = [[UIView alloc] initWithFrame:CGRectZero];
```

 [Add Player](#)

Choose Game

Angry Birds

Chess

Russian Roulette

Spin the Bottle

Texas Hold'em Poker



Tic-Tac-Toe

Capítulo 195: UITableViewCell

Introducción

El archivo xib de celda personalizado utiliza la clase de categoría de celda, sin necesidad de registrar el archivo de plumilla

Examples

Archivo Xib de UITableViewCell

Crear una clase de categoría de celda `UITableViewCell`.

Archivo UITableViewCell + RRCell.h

```
#import <UIKit/UIKit.h>

@interface UITableViewCell (RRCell)

-(id)initWithOwner:(id)owner;

@end
```

Archivo UITableViewCell + RRCell.m

```
#import "UITableViewCell+RRCell.h"

@implementation UITableViewCell (RRCell)

#pragma clang diagnostic push
#pragma clang diagnostic ignored "-Wobjc-designated-initializers"

-(id)initWithOwner:(id)owner {

    if (self = [super init]) {

        NSArray *nib = [[NSBundle mainBundle]loadNibNamed:NSStringFromClass([self class])
owner:self options:nil];
        self = [nib objectAtIndex:0];
    }
    return self;
}

#pragma clang diagnostic pop

@end
```

Importe la clase de categoría de celda para usar este método en el método `cellForRowAtIndexPath`


```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    //Creted custom cell xib file to load by cell category class
    CustomCell *cell = [[CustomCell alloc] initWithOwner:self];

    return cell;
}
```

Lea UITableViewCell en línea: <https://riptutorial.com/es/ios/topic/10101/uitableviewcell>

Capítulo 196: UITableViewController

Introducción

UITableViewController objeto de controlador que administra una vista de tabla. Para un determinado escenario, se recomendará utilizar UITableViewController, por ejemplo, si tiene muchas celdas y algunas tienen campo de extensión.

Examples

TableView con propiedades dinámicas con tableViewCellStyle basic.

```
override func numberOfSections(in tableView: UITableView) -> Int {
    // You need to return minimum one to show the cell inside the tableView
    return 1
}

override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // return the number of rows inside the tableView.
    return 3
}

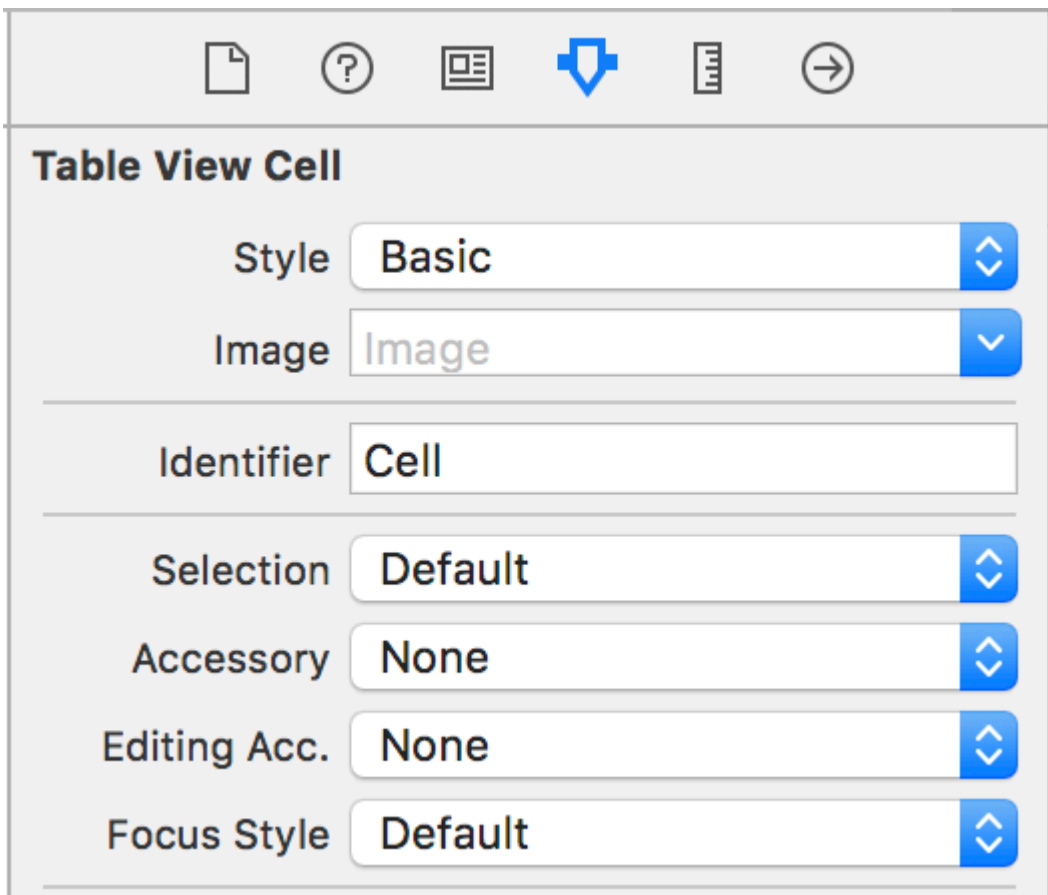
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {

    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath)
    // identifier string should be same as what you have entered in the cell Attribute inspector -
    > identifier (see the image).

    // Configure the cell...
    cell.textLabel?.text = "Cell \(indexPath.row) :" + "Hello"
    //cell have different style Custom, basic, right detail, left detail, subtitle.
    //For custom you can use your own objects and constrains, for other styles all
    //is ready just select according to your design. (see the image for changing the style)

    return cell
}

override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    // this delegate method will trigger when you click a cell
}
```



TableView con celda personalizada

Para la celda de vista de tabla personalizada, necesita una clase que sea subclase de `UITableViewCell`, un ejemplo de clase que puede ver a continuación.

```
class TableViewCell: UITableViewCell {  
  
    @IBOutlet weak var lblTitle: UILabel!  
  
    override func awakeFromNib() {  
        super.awakeFromNib()  
        // Initialization code  
    }  
  
    override func setSelected(_ selected: Bool, animated: Bool) {  
        super.setSelected(selected, animated: animated)  
  
        // Configure the view for the selected state  
    }  
  
}
```

Sus delegados de Tableview

```
override func numberOfSections(in tableView: UITableView) -> Int {  
    // You need to return minimum one to show the cell inside the tableview  
    return 1  
}
```

```
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    // return the number of rows inside the tableview.
    return 3
}

override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
UITableViewCell {

    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath) as!
    TableViewCell
    // identifier string should be same as what you have entered in the cell Attribute
    inspector -> identifier.

    // Configure the cell...
    cell.lblTitle.text = "Cell \(indexPath.row) :" + "Hello"

    return cell
}

override func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
    // this delegate method will trigger when you click a cell
}
```

Lea UITableViewController en línea: <https://riptutorial.com/es/ios/topic/10953/uitableviewController>

Capítulo 197: UITextField

Introducción

UITextField es parte del marco UIKit y se usa para mostrar un área para recopilar entradas de texto del usuario mediante el teclado en pantalla

Sintaxis

- UITextField.text: String // obtiene o establece el texto que muestra el campo.
- UITextField.attributedString: NSAttributedString // obtiene o establece el texto atribuido que muestra el campo.
- UITextField.textColor: UIColor // obtiene o establece el color del texto en el campo
- UITextField.font: UIFont // obtiene o establece la fuente del texto en el campo
- UITextField.textAlignment: NSTextAlignment // el valor predeterminado es NSTextAlignmentLeft
- UITextField.borderStyle: UITextBorderStyle // el valor predeterminado es UITextBorderStyleNone. Si se establece en UITextBorderStyleRoundedRect, las imágenes de fondo personalizadas se ignoran.
- UITextField.placeholder: String // el valor predeterminado es nil. La cuerda está dibujada al 70% de gris.
- UITextField.attributedStringPlaceholder: NSAttributedString // obtiene o establece el marcador de posición atribuido del campo
- UITextField.clearsOnBeginEditing: Bool // default es NO, lo que mueve el cursor a la ubicación donde se hizo clic. Si es SÍ, todo el texto borrado
- UITextField.adjustsFontSizeToFitWidth: Bool // el valor predeterminado es NO. Si es SÍ, el texto se reducirá a minFontSize a lo largo de la línea de base
- UITextField.minimumFontSize: CGFloat // el valor predeterminado es 0.0. el min real puede ser fijado a algo legible. se utiliza si ajustaFontSizeToFitWidth es YES
- UITextField.delegate: UITextFieldDelegate? // el valor predeterminado es nil. referencia débil
- UITextField.clearButtonMode: UITextFieldViewMode // se establece cuando aparece el botón Borrar. el valor predeterminado es UITextFieldViewModeNever
- UITextField.leftView: UIView? // p. ej. lupa
- UITextField.leftViewMode: UITextFieldViewMode // se establece cuando aparece la vista izquierda. el valor predeterminado es UITextFieldViewModeNever
- UITextField.rightView: UIView? // ej. botón de marcadores
- UITextField.rightViewMode: UITextFieldViewMode // se establece cuando aparece la vista correcta. el valor predeterminado es UITextFieldViewModeNever
- UITextField.inputView: UIView? // Presentado cuando el objeto se convierte en el primer respondedor. Si se establece en nulo, vuelve a la siguiente cadena de respondedores. Si se configura durante la primera respuesta, no tendrá efecto hasta que se llame a reloadInputViews.
- UITextField.inputAccessoryView: UIView?
- UITextField.isSecureTextEntry: Bool // p. Ej., Si el campo contiene información confidencial

como contraseña o número de tarjeta

Examples

Inicializar campo de texto

Rápido

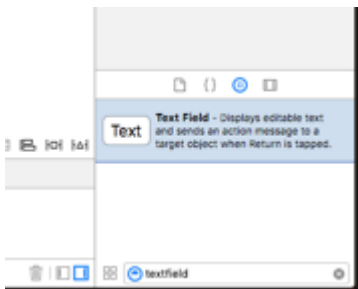
```
let frame = CGRect(x: 0, y: 0, width: 100, height: 100)
let textField = UITextField(frame: frame)
```

C objetivo

```
CGRect *frame = CGRectMake(0, 0, 100, 100);
UITextField *textField = [[UITextField alloc] initWithFrame:frame];
```

Generador de interfaz

También puede agregar un `UITextField` a un guión gráfico arrastrándolo desde la Biblioteca de objetos.



Entrada de vista de accesorios (barra de herramientas)

Añadir una vista de accesorios por encima del teclado. Esto se usa comúnmente para agregar botones siguientes / anteriores, o botones adicionales como Listo / Enviar (especialmente para los tipos de teclado numérico / teléfono / teclado decimal que no tienen una tecla de retorno integrada).

Rápido

```
let textField = UITextField() // initialized however

let toolbar = UIToolbar(frame: CGRect(x: 0, y: 0, width: view.frame.size.width, height: 0)

let flexibleSpace = UIBarButtonItem(barButtonItemSystemItem: .FlexibleSpace, target: nil, action: nil)

let doneButton = UIBarButtonItem(barButtonItemSystemItem: .Done, target: self, action:
```

```
Selector("done"))

let items = [flexibleSpace, doneButton] // pushes done button to right side

toolbar.setItems(items, animated: false) // or toolbar.items = ...
toolbar.sizeToFit()

textField.inputAccessoryView = toolbar
```

C objetivo

```
UITextField *textField = [[UITextField alloc] init];

UIToolbar *toolbar = [[UIToolbar alloc] initWithFrame:CGRectMake(0, 0,
self.view.frame.size.width, 0)];

UIBarButtonItem *flexibleSpace = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemFlexibleSpace target:nil action:nil];
UIBarButtonItem *doneButton = [[UIBarButtonItem alloc]
initWithBarButtonSystemItem:UIBarButtonSystemItemDone target:self action:@selector(done)];
NSArray *items = @[
    flexibleSpace,
    doneButton
];

[toolbar setItems:items];
[toolbar sizeToFit];

textField.inputAccessoryView = toolbar;
```

Auto capitalización

Rápido

```
textField.autocapitalizationType = .None
```

C objetivo

```
textField.autocapitalizationType = UITextAutocapitalizationTypeNone;
```

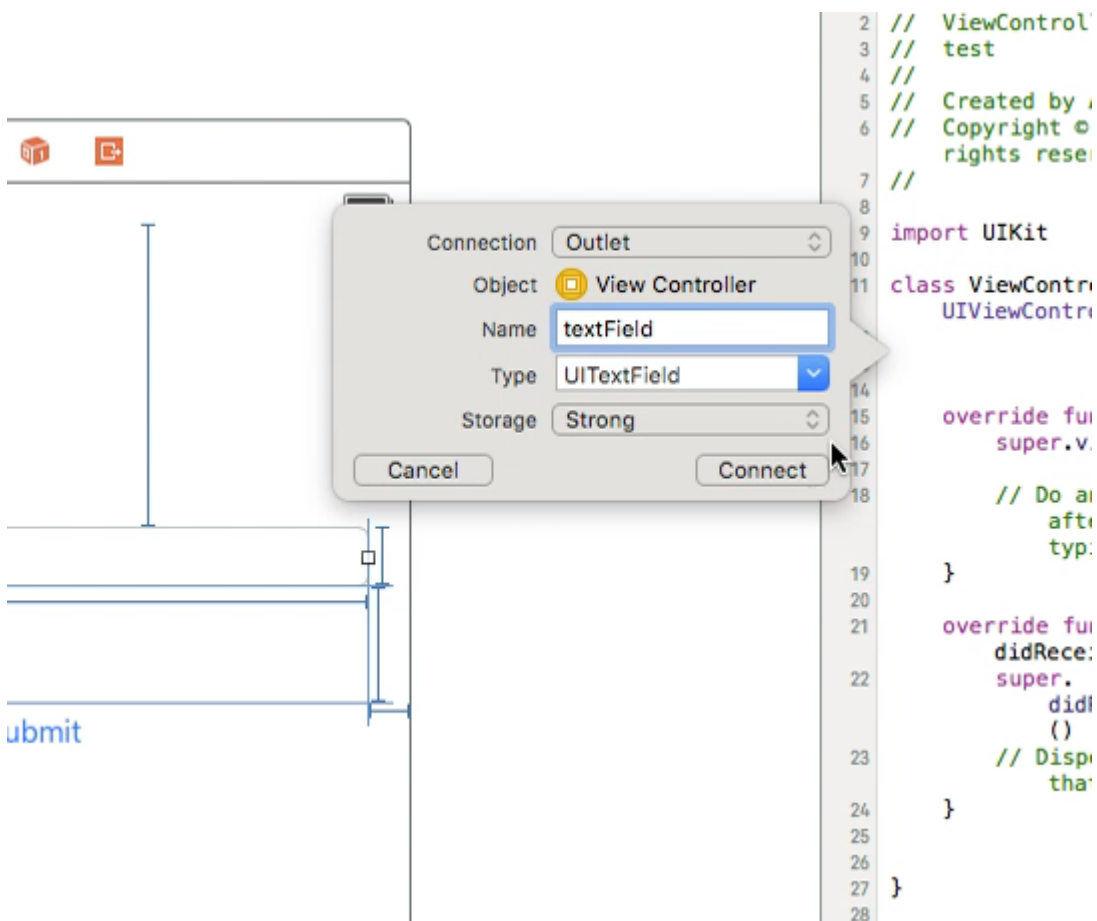
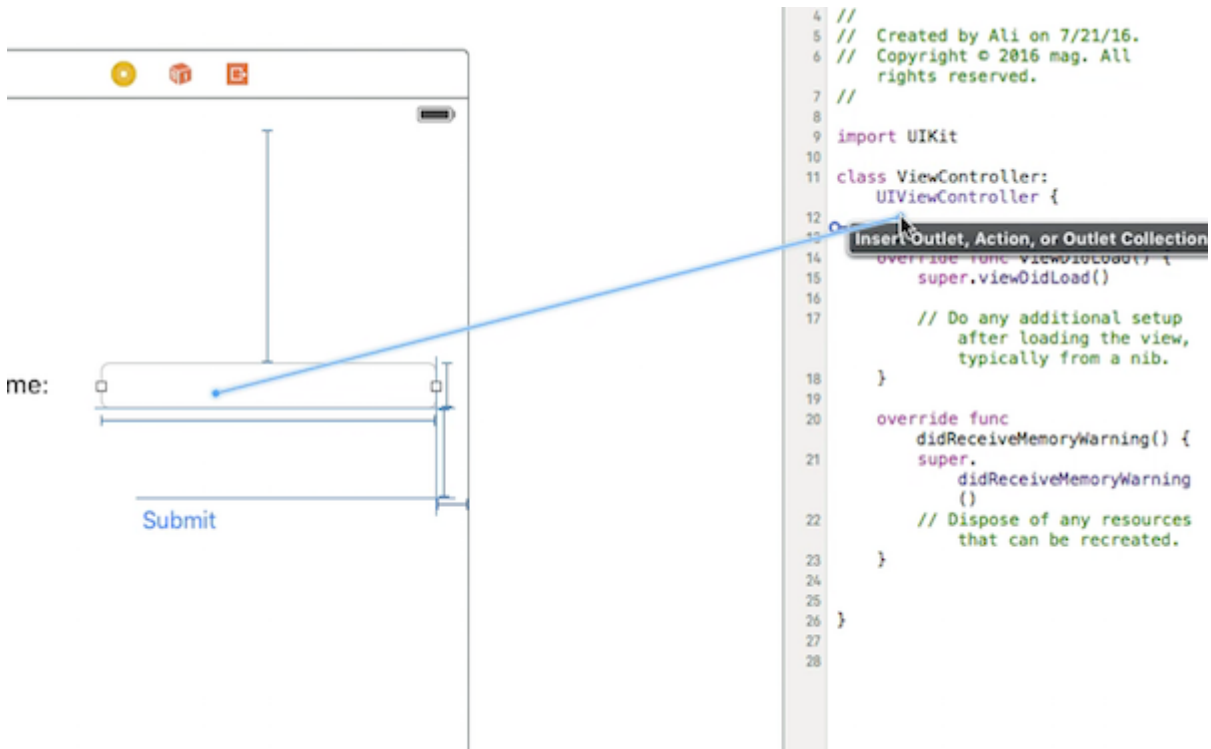
Todas las opciones:

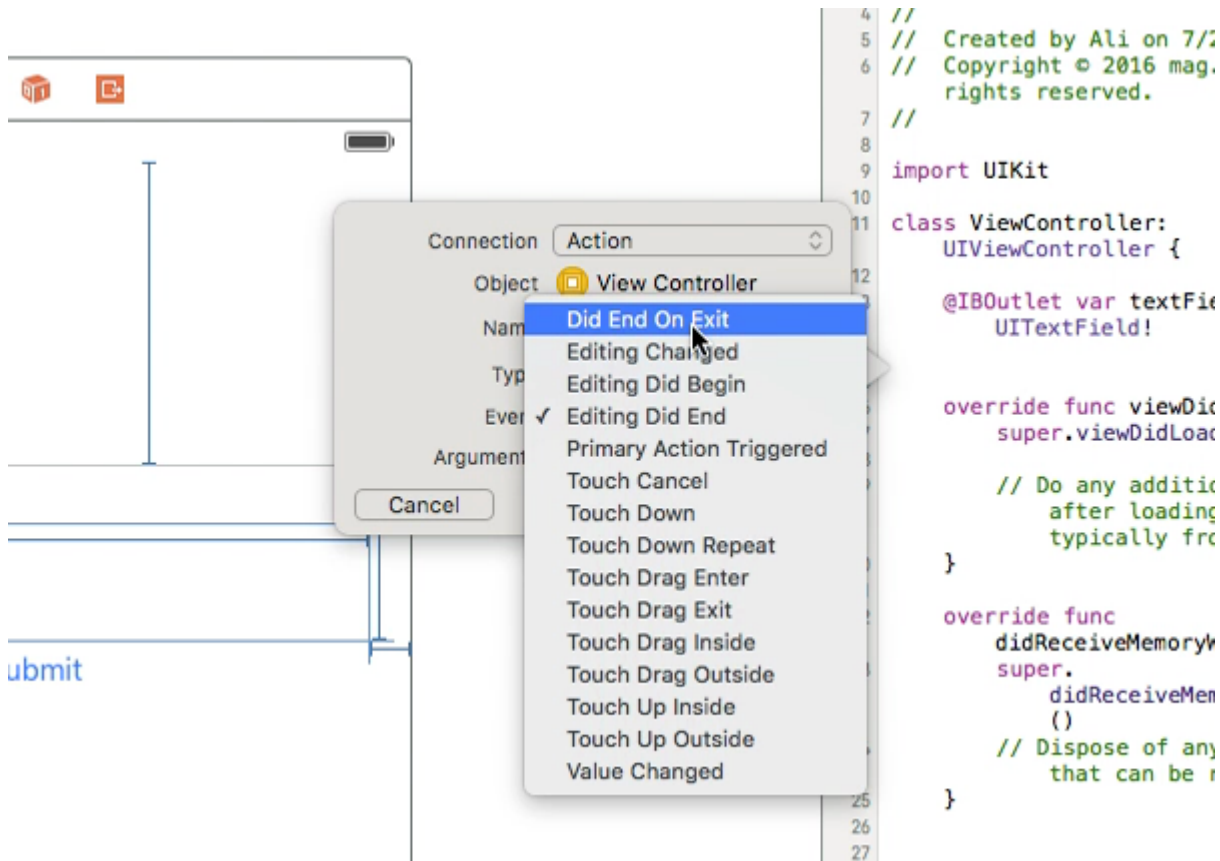
- `.None \ UITextAutocapitalizationTypeNone` : no autocapitalizar nada
- `.Words \ UITextAutocapitalizationTypeWords` : autocapitalizar cada palabra
- `.Sentences \ UITextAutocapitalizationTypeSentences` : autocapitaliza la primera palabra en una oración
- `.AllCharacters \ UITextAutocapitalizationTypeAllCharacters` : autocapitaliza cada letra (es decir, bloqueo de mayúsculas)

Descartar teclado

Rápido

Ctrl + Arrastrar desde el campo de texto UI en MainStoryboard a la clase ViewController y crear un Outlet de UITextField





Después de eso, seleccione nuevamente UITextField y Ctrl + arrastre en la clase ViewController, pero esta vez seleccione la conexión de **Acción** y en el almacenamiento seleccione **Did End On Exit** y luego haga clic en conectar.

en la acción que acaba de crear, escriba el nombre de su UITextField `.resignFirstResponder()`

```
@IBAction func textFieldResign(sender: AnyObject) {
    yourTextFieldName.resignFirstResponder()
}
```

Esto se encargará de ocultar el teclado al presionar la tecla de retorno en el teclado.

Otro ejemplo de ocultar el teclado cuando se presiona la tecla de retorno:

UITextFieldDelegate protocolo UITextFieldDelegate junto a UIViewController

en la función `self.yourTextFieldName.delegate = self` agregamos `self.yourTextFieldName.delegate = self`

Y por último añadimos esto.

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
    yourTextFieldName.resignFirstResponder()
    return true
}
```

El código final es este:

```

class ViewController: UIViewController, UITextFieldDelegate {

@IBOutlet var textField: UITextField!

    func textFieldShouldReturn(textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }

    override func touchesBegan(touches: Set<UITouch>, withEvent event: UIEvent?) {
        view.endEditing(true)
        super.touchesBegan(touches, withEvent: event)
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        self.textField.delegate = self
    }

}

```

C objetivo

```
[textField resignFirstResponder];
```

Establecer alineación

Rápido

```
textField.textAlignment = .Center
```

C objetivo

```
[textField setTextAlignment: NSTextAlignmentCenter];
```

En el ejemplo, hemos establecido la `NSTextAlignment` en el centro. También puede establecer en `.Left`, `.Right`, `.Justified` y `.Natural`.

`.Natural` es la alineación predeterminada para la localización actual. Eso significa que para los idiomas de izquierda a derecha (por ejemplo, inglés), la alineación es `.Left`; para idiomas de derecha a izquierda, es `.Right`.

Tipo de teclado

Para cambiar la apariencia del teclado, los siguientes tipos se pueden configurar individualmente en cada propiedad de `UITextFields`: `keyboardType`

```
typedef NS_ENUM(NSInteger, UIKeyboardType) {
```

```

    UIKeyboardTypeDefault, // Default type for the current input method.
    UIKeyboardTypeASCIICapable, // Displays a keyboard which can enter ASCII
characters, non-ASCII keyboards remain active
    UIKeyboardTypeNumbersAndPunctuation, // Numbers and assorted punctuation.
    UIKeyboardTypeURL, // A type optimized for URL entry (shows . / .com
prominently).
    UIKeyboardTypeNumberPad, // A number pad (0-9). Suitable for PIN entry.
    UIKeyboardTypePhonePad, // A phone pad (1-9, *, 0, #, with letters under the
numbers).
    UIKeyboardTypeNamePhonePad, // A type optimized for entering a person's name or
phone number.
    UIKeyboardTypeEmailAddress, // A type optimized for multiple email address entry
(shows space @ . prominently).
    UIKeyboardTypeDecimalPad NS_ENUM_AVAILABLE_IOS(4_1), // A number pad with a decimal
point.
    UIKeyboardTypeTwitter NS_ENUM_AVAILABLE_IOS(5_0), // A type optimized for twitter
text entry (easy access to @ #)
    UIKeyboardTypeWebSearch NS_ENUM_AVAILABLE_IOS(7_0), // A default keyboard type with
URL-oriented addition (shows space . prominently).

    UIKeyboardTypeAlphabet = UIKeyboardTypeASCIICapable, // Deprecated
};

```

Desplazando el desplazamiento cuando UITextView se convierte en el primer respondedor

Observe las notificaciones `UIKeyboardWillShowNotification` y `UIKeyboardWillHideNotification`, actualice las inserciones de contenido `scrollView` acuerdo con la altura del teclado, luego desplácese hasta el control enfocado.

```

- (void)viewDidLoad
{
    [super viewDidLoad];

    // register for keyboard notifications
    [[NSNotificationCenter defaultCenter] addObserver:self
                                           selector:@selector(keyboardWillShow:)
                                           name:UIKeyboardWillShowNotification
                                           object:self.view.window];

    // register for keyboard notifications
    [[NSNotificationCenter defaultCenter] addObserver:self
                                           selector:@selector(keyboardWillHide:)
                                           name:UIKeyboardWillHideNotification
                                           object:self.view.window];
}

// Called when UIKeyboardWillShowNotification is sent
- (void)keyboardWillShow:(NSNotification*)notification
{
    // if we have no view or are not visible in any window, we don't care
    if (!self.isViewLoaded || !self.view.window) {
        return;
    }

    NSDictionary *userInfo = [notification userInfo];

    CGRect keyboardFrameInWindow;
    [[userInfo objectForKey:UIKeyboardFrameEndUserInfoKey] getValue:&keyboardFrameInWindow];

```

```

// the keyboard frame is specified in window-level coordinates. this calculates the frame
as if it were a subview of our view, making it a sibling of the scroll view
CGRect keyboardFrameInView = [self.view convertRect:keyboardFrameInWindow fromView:nil];

CGRect scrollViewKeyboardIntersection = CGRectIntersection(_scrollView.frame,
keyboardFrameInView);
UIEdgeInsets newContentInsets = UIEdgeInsetsMake(0, 0,
scrollViewKeyboardIntersection.size.height, 0);

// this is an old animation method, but the only one that retains compaitibility between
parameters (duration, curve) and the values contained in the userInfo-Dictionary.
[UIView beginAnimations:nil context:NULL];
[UIView setAnimationDuration:[userInfo
objectForKey:UIKeyboardAnimationDurationUserInfoKey] doubleValue]];
[UIView setAnimationCurve:[userInfo objectForKey:UIKeyboardAnimationCurveUserInfoKey]
intValue]];

_scrollView.contentInset = newContentInsets;
_scrollView.scrollIndicatorInsets = newContentInsets;

/*
 * Depending on visual layout, _focusedControl should either be the input field
(UITextField,..) or another element
 * that should be visible, e.g. a purchase button below an amount text field
 * it makes sense to set _focusedControl in delegates like -textFieldShouldBeginEditing:
if you have multiple input fields
 */
if (_focusedControl) {
    CGRect controlFrameInScrollView = [_scrollView convertRect:_focusedControl.bounds
fromView:_focusedControl]; // if the control is a deep in the hierarchy below the scroll view,
this will calculate the frame as if it were a direct subview
    CGRect controlFrameInset = CGRectInset(controlFrameInScrollView, 0, -10); // replace
10 with any nice visual offset between control and keyboard or control and top of the scroll
view.

    CGFloat controlVisualOffsetToTopOfScrollview = controlFrameInset.origin.y -
_scrollView.contentOffset.y;
    CGFloat controlVisualBottom = controlVisualOffsetToTopOfScrollview +
controlFrameInset.size.height;

    // this is the visible part of the scroll view that is not hidden by the keyboard
    CGFloat scrollViewVisibleHeight = _scrollView.frame.size.height -
scrollViewKeyboardIntersection.size.height;

    if (controlVisualBottom > scrollViewVisibleHeight) { // check if the keyboard will
hide the control in question
        // scroll up until the control is in place
        CGPoint newContentOffset = _scrollView.contentOffset;
        newContentOffset.y += (controlVisualBottom - scrollViewVisibleHeight);

        // make sure we don't set an impossible offset caused by the "nice visual offset"
// if a control is at the bottom of the scroll view, it will end up just above the
keyboard to eliminate scrolling inconsistencies
        newContentOffset.y = MIN(newContentOffset.y, _scrollView.contentSize.height -
scrollViewVisibleHeight);

        [_scrollView setContentOffset:newContentOffset animated:NO]; // animated:NO
because we have created our own animation context around this code
    } else if (controlFrameInset.origin.y < _scrollView.contentOffset.y) {
        // if the control is not fully visible, make it so (useful if the user taps on a

```

```

partially visible input field
    CGPoint newContentOffset = _scrollView.contentOffset;
    newContentOffset.y = controlFrameInScrollView.origin.y;

    [_scrollView setContentOffset:newContentOffset animated:NO]; // animated:NO
because we have created our own animation context around this code
    }
}

[UIView commitAnimations];
}

// Called when the UIKeyboardWillHideNotification is sent
- (void)keyboardWillHide:(NSNotification*)notification
{
    // if we have no view or are not visible in any window, we don't care
    if (!self.isViewLoaded || !self.view.window) {
        return;
    }

    NSDictionary *userInfo = notification.userInfo;

    [UIView beginAnimations:nil context:NULL];
    [UIView setAnimationDuration:[userInfo
valueForKey:UIKeyboardAnimationDurationUserInfoKey] doubleValue]];
    [UIView setAnimationCurve:[userInfo valueForKey:UIKeyboardAnimationCurveUserInfoKey]
intValue]];

    // undo all that keyboardWillShow-magic
    // the scroll view will adjust its contentOffset appropriately
    _scrollView.contentInset = UIEdgeInsetsZero;
    _scrollView.scrollIndicatorInsets = UIEdgeInsetsZero;

    [UIView commitAnimations];
}

```

Obtener el enfoque del teclado y ocultar el teclado

Obtener enfoque

Rápido

```
textField.becomeFirstResponder()
```

C objetivo

```
[textField becomeFirstResponder];
```

Renunciar

Rápido

```
textField.resignFirstResponder()
```

C objetivo

```
[textField resignFirstResponder];
```

Reemplace el teclado con UIPickerView

En algunos casos, desea mostrar a sus usuarios un `UIPickerView` con contenidos predefinidos para un `UITextField` lugar de un teclado.

Crear un UIPickerView personalizado

Al principio, necesita una clase envoltura personalizada para `UIPickerView` conforme a los protocolos `UIPickerViewDataSource` y `UIPickerViewDelegate`.

```
class MyPickerView: UIPickerView, UIPickerViewDataSource, UIPickerViewDelegate
```

Debe implementar los siguientes métodos para `DataSource` y `Delegate`:

```
public func pickerView(_ pickerView: UIPickerView, numberOfRowsInComponent component: Int) -> Int {
    if data != nil {
        return data!.count
    } else {
        return 0
    }
}

public func numberOfComponents(in pickerView: UIPickerView) -> Int {
    return 1
}

public func pickerView(_ pickerView: UIPickerView, titleForRow row: Int, forComponent component: Int) -> String? {
    if data != nil {
        return data![row]
    } else {
        return ""
    }
}
```

Para manejar los datos, `MyPickerView` necesita los `data` propiedades, `selectedValue` y `textFieldBeingEdited`:

```
/**
 * The data for the `UIPickerViewDelegate`
 *
 * Always needs to be an array of `String`! The `UIPickerView` can ONLY display Strings
 */
public var data: [String]? {
    didSet {
```

```

        super.delegate = self
        super.dataSource = self
        self.reloadData()
    }
}

/**
 Stores the UITextField that is being edited at the moment
 */
public var textFieldBeingEdited: UITextField?

/**
 Get the selected Value of the picker
 */
public var selectedValue: String {
    get {
        if data != nil {
            return data![selectedRow(inComponent: 0)]
        } else {
            return ""
        }
    }
}
}

```

Prepare su ViewController

El `ViewController` que contiene su campo de texto, necesita tener una propiedad para su `UIPickerView` personalizado. (Suponiendo que ya tiene otra propiedad o `@IBOutlet` contiene su campo de texto)

```

/**
 The picker view to present as keyboard
 */
var picker: UIPickerView?

```

En su `viewDidLoad()` , necesita inicializar el `picker` y configurarlo un poco:

```

picker = UIPickerView()
picker?.autoresizingMask = [.flexibleHeight, .flexibleWidth]
picker?.backgroundColor = UIColor.white()

picker?.data = ["One", "Two", "Three", "Four", "Five"] //The data shown in the picker

```

Ahora, puede agregar el `MyPicker` como `inputView` de su `UITextField` :

```

textField.inputView = picker

```

Descartando el selector de teclado

Ahora, ha reemplazado el teclado por un `UIPickerView` , pero no hay posibilidad de descartarlo. Esto se puede hacer con un `.inputAccessoryView` personalizado:

Agregue la propiedad `pickerAccessory` a su `ViewController` .

```
/**
 * A toolbar to add to the keyboard when the `picker` is presented.
 */
var pickerAccessory: UIToolbar?
```

En `viewDidLoad()` , **necesita crear una UIToolbar de UIToolbar para el `inputAccessoryView` :**

```
pickerAccessory = UIToolbar()
pickerAccessory?.autoresizingMask = .flexibleHeight

//this customization is optional
pickerAccessory?.barStyle = .default
pickerAccessory?.barTintColor = UIColor.red()
pickerAccessory?.backgroundColor = UIColor.red()
pickerAccessory?.isTranslucent = false
```

Deberías establecer el marco de tu barra de herramientas. Para encajar en el diseño de iOS, se recomienda usar una altura de 44.0 :

```
var frame = pickerAccessory?.frame
frame?.size.height = 44.0
pickerAccessory?.frame = frame!
```

Para una buena experiencia de usuario, debe agregar dos botones ("Hecho" y "Cancelar"), pero también funcionaría con solo uno que descarta el teclado.

```
let cancelButton = UIBarButtonItem(barButtonItemSystemItem: .cancel, target: self, action:
#selector (ViewController.cancelBtnClicked(_:)))
cancelButton.tintColor = UIColor.white()
let flexSpace = UIBarButtonItem(barButtonItemSystemItem: .flexibleSpace, target: nil, action: nil)
//a flexible space between the two buttons
let doneButton = UIBarButtonItem(barButtonItemSystemItem: .done, target: self, action:
#selector (ViewController.doneBtnClicked(_:)))
doneButton.tintColor = UIColor.white()

//Add the items to the toolbar
pickerAccessory?.items = [cancelButton, flexSpace, doneButton]
```

Ahora puedes agregar la barra de herramientas como `inputAccessoryView`

```
textField.inputAccessoryView = pickerAccessory
```

Antes de que pueda construir su proyecto, necesita implementar los métodos, los botones están llamando:

```
/**
 * Called when the cancel button of the `pickerAccessory` was clicked. Dismisses the picker
 */
func cancelBtnClicked(_ button: UIBarButtonItem?) {
    textField?.resignFirstResponder()
}

/**
 * Called when the done button of the `pickerAccessory` was clicked. Dismisses the picker and
```



```
puts the selected value into the textField
*/
func doneBtnClicked(_ button: UIBarButtonItem?) {
    textField?.resignFirstResponder()
    textField.text = picker?.selectedValue
}
```

Ejecute su proyecto, toque el campo de `textField` y debería ver un selector como este en lugar del teclado:

Cancel

Done

One

Two

Three

Four

Seleccione un valor programáticamente (opcional)

Si no desea que la primera fila se seleccione automáticamente, puede configurar la fila seleccionada como en `UIPickerView`:

```
picker?.selectRow(3, inComponent: 0, animated: false) //Will select the row at index 3
```

Descartar teclado cuando el usuario presiona el botón de retorno

Configure el controlador de vista para administrar la edición de texto para el campo de texto.

```
class MyViewController: UITextFieldDelegate {

    override viewDidLoad() {
        super.viewDidLoad()

        textField.delegate = self
    }

}
```

`textFieldShouldReturn` se llama cada vez que se presiona el botón de retorno en el teclado.

Rápido:

```
func textFieldShouldReturn(textField: UITextField) -> Bool {
    textField.resignFirstResponder()
    return true;
}
```

C objetivo:

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField {
    [textField resignFirstResponder];
    return true;
}
```

Obtención y configuración de la posición del cursor

Información útil

El principio del texto del campo de texto:

```
let startPosition: UITextPosition = textField.beginningOfDocument
```

El final del texto del campo de texto:

```
let endPosition: UITextPosition = textField.endOfDocument
```

El rango seleccionado actualmente:

```
let selectedRange: UITextRange? = textField.selectedTextRange
```

Obtener posición del cursor

```
if let selectedRange = textField.selectedTextRange {
    let cursorPosition = textField.offsetFromPosition(textField.beginningOfDocument,
    toPosition: selectedRange.start)

    print("\(cursorPosition)")
}
```

Establecer la posición del cursor

Para establecer la posición, todos estos métodos están configurando un rango con los mismos

valores de inicio y final.

Al Principio

```
let newPosition = textField.beginningOfDocument
textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

Hasta el final

```
let newPosition = textField.endOfDocument
textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

A una posición a la izquierda de la posición actual del cursor

```
// only if there is a currently selected range
if let selectedRange = textField.selectedTextRange {

    // and only if the new position is valid
    if let newPosition = textField.positionFromPosition(selectedRange.start, inDirection:
UITextLayoutDirection.Left, offset: 1) {

        // set the new position
        textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
    }
}
```

A una posición arbitraria

Comienza por el principio y mueve 5 caracteres a la derecha.

```
let arbitraryValue: Int = 5
if let newPosition = textField.positionFromPosition(textField.beginningOfDocument,
inDirection: UITextLayoutDirection.Right, offset: arbitraryValue) {

    textField.selectedTextRange = textField.textRangeFromPosition(newPosition, toPosition:
newPosition)
}
```

Relacionado

Seleccionar todo el texto

```
textField.selectedTextRange = textField.textRangeFromPosition(textField.beginningOfDocument,
toPosition: textField.endOfDocument)
```

Seleccione un rango de texto

```
// Range: 3 to 7
```

```
let startPosition = textField.positionFromPosition(textField.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: 3)
let endPosition = textField.positionFromPosition(textField.beginningOfDocument, inDirection:
UITextLayoutDirection.Right, offset: 7)

if startPosition != nil && endPosition != nil {
    textField.selectedTextRange = textField.textRangeFromPosition(startPosition!, toPosition:
endPosition!)
}
```

Insertar texto en la posición actual del cursor

```
textField.insertText("Hello")
```

Notas

- Este ejemplo proviene originalmente de [esta respuesta de desbordamiento de pila](#) .
- Esta respuesta utiliza un campo de texto, pero los mismos conceptos se aplican a `UITextView` .
- Use `textField.becomeFirstResponder()` para enfocar el campo de texto y hacer que aparezca el teclado.
- Vea [esta respuesta](#) para [saber](#) cómo obtener el texto en algún rango.

Relacionado

- [Cómo crear un rango en Swift](#) (trata indirectamente el problema de por qué tenemos que usar `selectedTextRange` aquí en lugar de solo `selectedRange`)

Ocultar carete parpadeante

Para ocultar el cursor que parpadea, debe anular `caretRectForPosition` de un `UITextField` y devolver `CGRectZero`.

Swift 2.3 <

```
public override func caretRectForPosition(position: UITextPosition) -> CGRect {
    return CGRectZero
}
```

Swift 3

```
override func caretRect(for position: UITextPosition) -> CGRect {
    return CGRect.zero
}
```

```
}
```

C objetivo

```
- (CGRect) caretRectForPosition:(UITextPosition*) position{
return CGRectZero;
}
```

Cambiar el marcador de posición de color y fuente

Podemos cambiar el estilo del marcador de posición estableciendo el `attributedString` (un `NSAttributedString`).

```
var placeholderAttributes = [String: AnyObject]()
placeholderAttributes[NSForegroundColorAttributeName] = color
placeholderAttributes[NSFontAttributeName] = font

if let placeholder = textField.placeholder {
    let newAttributedString = NSAttributedString(string: placeholder, attributes:
placeholderAttributes)
    textField.attributedString = newAttributedString
}
```

En este ejemplo cambiamos solo el `color` y la `font`. Puede cambiar otras propiedades como el estilo subrayado o tachado. Consulte `NSAttributedString` para conocer las propiedades que se pueden cambiar.

Crear un UITextField

Inicialice el `UITextField` con un `CGRect` como un marco:

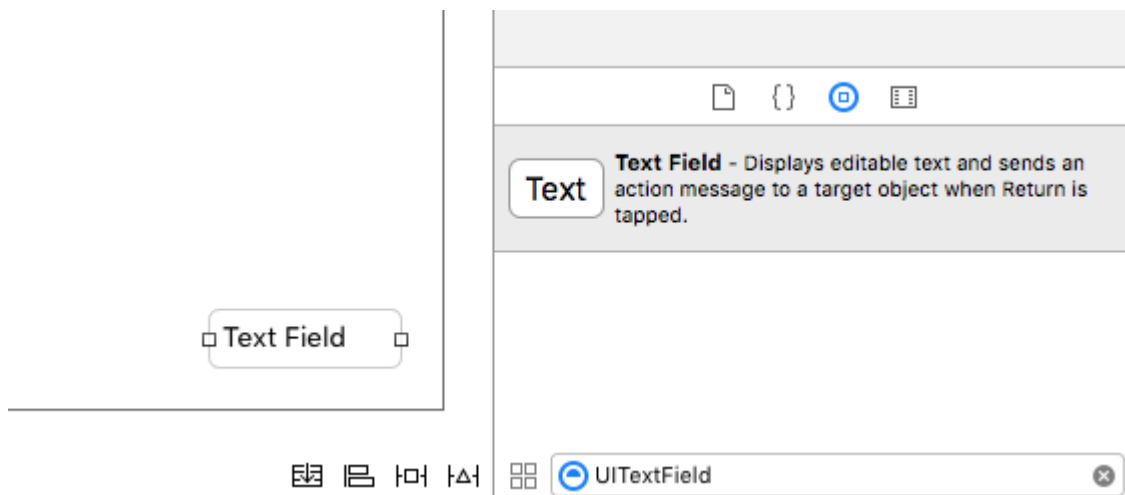
Rápido

```
let textField = UITextField(frame: CGRect(x: 0, y: 0, width: 200, height: 21))
```

C objetivo

```
UITextField *textField = [[UITextField alloc] initWithFrame:CGRectMake(0, 0, 200, 21)];
```

También puede crear un `UITextField` en Interface Builder:



Lea UITextField en línea: <https://riptutorial.com/es/ios/topic/1630/UITextField>

Capítulo 198: UITextField Delegate

Examples

UITextField - Restringir el campo de texto a ciertos caracteres

Si desea realizar una validación de entrada de usuario de su campo de texto, use el siguiente fragmento de código:

```
// MARK: - UITextFieldDelegate

let allowedCharacters =
CharacterSet(charactersIn:"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz").inverted

func textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange,
replacementString string: String) -> Bool {

    let components = string.components(separatedBy: allowedCharacters)
    let filtered = components.joined(separator: "")

    if string == filtered {

        return true

    } else {

        return false

    }
}
```

C objetivo

```
#define ACCEPTABLE_CHARACTERS @"0123456789
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

- (BOOL)textField:(UITextField *)textField shouldChangeCharactersInRange:(NSRange) range
replacementString:(NSString *)string
{
    NSCharacterSet *cs = [[NSCharacterSet
characterSetWithCharactersInString:ACCEPTABLE_CHARACTERS] invertedSet];

    NSString *filtered = [[string componentsSeparatedByCharactersInSet:cs]
componentsJoinedByString:@""];

    return [string isEqualToString:filtered];
}
```

Además, también puede utilizar los conjuntos de caracteres proporcionados por Apple para realizar la validación:

Echa un vistazo a <https://developer.apple.com/reference/foundation/nscharacterset>

```
let allowedCharacters = CharacterSet.alphanumerics.inverted
let allowedCharacters = CharacterSet.capitalizedLetters.inverted
```

Buscar siguiente etiqueta y administrar teclado

El campo de texto llama a diferentes métodos de delegado (solo si se establecen delegados) Uno de los métodos de delegado llamado por campo de texto es * - **(BOOL) textFieldShouldReturn: (UITextField) textField**

Este método se llama siempre que los usuarios tocan el botón de retorno. Al usar este método, podemos implementar cualquier comportamiento personalizado.

Por ejemplo,

En el siguiente ejemplo, el siguiente respondedor se informará sobre la base de la etiqueta y administrar el teclado. Aquí 20 es la constante, Como la etiqueta asignada al campo de texto es así 50,70,90, etc.

Aquí, al encontrar un nuevo objeto de campo de texto como respondedor, creará el campo de texto actual como nuevo respondedor y abrirá el teclado en consecuencia.

```
- (BOOL)textFieldShouldReturn:(UITextField *)textField {
    NSInteger nextTag = textField.tag+20;
    // Try to find next responder
    UIResponder *nextResponder = [textField.superview viewWithTag:nextTag];
    if (nextResponder)
    {
        // Found next responder, so set it.
        [nextResponder becomeFirstResponder];
    }
    else
    {
        // Not found, so remove keyboard.
        [textField resignFirstResponder];
    }
    return YES;
}
```

Acciones cuando un usuario ha comenzado / terminado interactuando con un campo de texto

Para Swift 3.1:

En el primer ejemplo, uno puede ver cómo interceptaría al usuario que interactúa con un campo de texto mientras escribe. De manera similar, hay métodos en el [UITextFieldDelegate](#) que se llaman cuando un usuario ha iniciado y finalizado su interacción con un TextField.

Para poder acceder a estos métodos, debe cumplir con el protocolo [UITextFieldDelegate](#) y, para cada campo de texto sobre el que desee recibir notificaciones, asigne la clase principal como delegado:


```

class SomeClass: UITextFieldDelegate {

    @IBOutlet var textField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        textField.delegate = self
    }

}

```

Ahora podrás implementar todos los métodos de UITextFieldDelegate.

Para recibir una notificación cuando un usuario haya comenzado a editar un campo de texto, puede implementar el [método textFieldDidBeginEditing \(_ :\)](#) así:

```

func textFieldDidBeginEditing(_ textField: UITextField) {
    // now you can perform some action
    // if you have multiple textfields in a class,
    // you can compare them here to handle each one separately
    if textField == emailTextField {
        // e.g. validate email
    }
    else if textField == passwordTextField {
        // e.g. validate password
    }
}

```

De manera similar, si se le notifica si un usuario ha finalizado la interacción con un campo de texto, puede usar el [método textFieldDidEndEditing \(_ :\)](#) así:

```

func textFieldDidEndEditing(_ textField: UITextField) {
    // now you can perform some action
    // if you have multiple textfields in a class,
    // you can compare them here to handle each one separately
    if textField == emailTextField {
        // e.g. validate email
    }
    else if textField == passwordTextField {
        // e.g. validate password
    }
}

```

Si desea tener control sobre si un TextField debe comenzar / finalizar la edición, los [métodos textFieldShouldBeginEditing \(_ :\)](#) y [textFieldShouldEndEditing \(_ :\)](#) se pueden usar devolviendo verdadero / falso según su lógica necesaria.

Lea UITextField Delegate en línea: <https://riptutorial.com/es/ios/topic/7185/uitextfield-delegate>

Capítulo 199: UITextField personalizado

Introducción

Con el uso de UITextField personalizado, podemos manipular el comportamiento del campo de texto.

Examples

UITextField personalizado para filtrar el texto de entrada

Aquí hay un ejemplo de UITextField personalizado que toma solo texto numérico y descarta todos los demás.

NOTA: Para iPhone, es fácil hacerlo usando el teclado de tipo Número, pero para iPad no hay un teclado con números solamente

```
class NumberTextField: UITextField {

    required init(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        registerForTextFieldNotifications()
    }

    override init(frame: CGRect) {
        super.init(frame: frame)
    }

    override func awakeFromNib() {
        super.awakeFromNib()
        keyboardType = .numberPad//useful for iPhone only
    }

    private func registerForTextFieldNotifications() {
        NotificationCenter.default.addObserver(self, selector:
        #selector(NumberTextField.textDidChange), name: NSNotification.Name(rawValue:
        "UITextFieldTextDidChangeNotification"), object: self)
    }

    deinit {
        NotificationCenter.default.removeObserver(self)
    }

    func textDidChange() {
        text = filteredText()
    }

    private func filteredText() -> String {
        let inverseSet = CharacterSet(charactersIn:"0123456789").inverted
        let components = text!.components(separatedBy: inverseSet)
        return components.joined(separator: "")
    }
}
```

Entonces, donde sea que queramos un campo de texto que tome solo números como texto de entrada, entonces podemos usar este campo de campo personalizado.

Personalizar campo de campo para no permitir todas las acciones como copiar, pegar, etc.

Si queremos deshabilitar todas las acciones como Copiar, Pegar, Reemplazar, Seleccionar, etc. desde `UITextField` , podemos usar el siguiente campo de texto personalizado:

```
class CustomTextField: UITextField {  
  
    var enableLongPressActions = false  
  
    required init(coder aDecoder: NSCoder) {  
        super.init(coder: aDecoder)!  
    }  
  
    override init(frame: CGRect) {  
        super.init(frame: frame)  
    }  
  
    override func canPerformAction(_ action: Selector, withSender sender: Any?) -> Bool {  
        return enableLongPressActions  
    }  
}
```

Usando la propiedad `enableLongPressActions` , podemos habilitar todas las acciones en cualquier momento posterior, si es necesario.

Lea `UITextField` personalizado en línea: <https://riptutorial.com/es/ios/topic/9997/uitextfield-personalizado>

Capítulo 200: UITextView

Examples

Cambiar texto

Rápido

```
textView.text = "Hello, world!"
```

C objetivo:

```
textView.text = @"Hello, world!";
```

Establecer texto atribuido

```
// Modify some of the attributes of the attributed string.
let attributedText = NSMutableAttributedString(attributedString: textView.attributedText!)

// Use NSString so the result of rangeOfString is an NSRange.
let text = textView.text! as NSString

// Find the range of each element to modify.
let tintedRange = text.range(of: NSLocalizedString("tinted", comment: ""))
let highlightedRange = text.range(of: NSLocalizedString("highlighted", comment: ""))

// Add tint.
attributedText.addAttribute(NSForegroundColorAttributeName, value: UIColor.blue, range:
tintedRange)

// Add highlight.
attributedText.addAttribute(NSBackgroundColorAttributeName, value: UIColor.yellow, range:
highlightedRange)

textView.attributedText = attributedText
```

Cambiar la alineación del texto

Rápido

```
textView.textAlignment = .left
```

C objetivo

```
textView.textAlignment = NSTextAlignmentLeft;
```

UITextViewDelegate métodos

Respondiendo a las notificaciones de edición

- textViewShouldBeginEditing(_:)
- textViewDidBeginEditing(_:)
- textViewShouldEndEditing(_:)
- textViewDidEndEditing(_:)

Respondiendo a cambios de texto

- textView(_:shouldChangeTextIn:replacementText:)
- textViewDidChange(_:)

Respondiendo a la URL

- textView(_: UITextView, shouldInteractWithURL: NSURL, inRange: NSRange) -> Bool

Cambiar fuente

Rápido

```
//System Font
textView.font = UIFont.systemFont(ofSize: 12)

//Font of your choosing
textView.font = UIFont(name: "Font Name", size: 12)
```

C objetivo

```
//System Font
textView.font = [UIFont systemFontOfSize:12];

//Font of your choosing
textView.font = [UIFont fontWithName:@"Font Name" size:12];
```

Cambiar el color del texto

Rápido

```
textView.textColor = UIColor.red
```

C objetivo

```
textView.textColor = [UIColor redColor];
```

UITextView con texto HTML

```
NSString *htmlString = @"<p> This is an <b>HTML</b> text</p>";
NSAttributedString *attributedString = [[NSMutableAttributedString alloc]
                                         initWithData: [htmlString
                                                         dataUsingEncoding:NSUTF8StringEncoding]
                                         options: @{
```

```
NSDocumentTypeDocumentAttribute: NSHTMLTextDocumentType }
                                documentAttributes: nil
                                error: nil
                                ];
    _yourTextView.attributedString = attributedString;
    // If you want to modify the font
    field.font = [UIFont fontWithName:@"Raleway-Regular" size:15];
```

Detección automática de enlaces, direcciones, fechas y más

`UITextView` ha incorporado soporte para detectar automáticamente una variedad de datos. Los datos que se pueden detectar automáticamente actualmente incluyen:

```
enum {
    UIDataDetectorTypePhoneNumber = 1 << 0,
    UIDataDetectorTypeLink        = 1 << 1,
    UIDataDetectorTypeAddress     = 1 << 2,
    UIDataDetectorTypeCalendarEvent = 1 << 3,
    UIDataDetectorTypeNone       = 0,
    UIDataDetectorTypeAll        = NSUIntegerMax
};
```

Habilitando la autodetección

```
// you may add as many as you like by using the `|` operator between options
textView.dataDetectorTypes = (UIDataDetectorTypeLink | UIDataDetectorTypePhoneNumber);
```

Si está habilitado, el texto aparecerá como un hipervínculo en la `UITextView`

Datos seleccionables

Para permitir que se haga clic en el enlace (que dará lugar a diferentes acciones según el tipo de datos), debe asegurarse de que el `UITextView` sea seleccionable pero no editable y que la interacción del usuario esté habilitada.

```
textView.editable = NO;
textView.selectable = YES;
textView.userInteractionEnabled = YES; // YES by default
```

Compruebe si está vacío o nulo

Rápido

```
if let text = self.textView.text where !text.isEmpty {
    // Do stuff for text
} else {
    // Do stuff for nil text or empty string
}
```

C objetivo

```
if (self.textView.text.length > 0){
    // Do stuff for text
} else {
    // Do stuff for nil text or empty string
}
```

Obtención y configuración de la posición del cursor

Información útil

El principio del texto del campo de texto:

```
let startPosition: UITextPosition = textView.beginningOfDocument
```

El final del texto del campo de texto:

```
let endPosition: UITextPosition = textView.endOfDocument
```

El rango seleccionado actualmente:

```
let selectedRange: UITextRange? = textView.selectedTextRange
```

Obtener posición del cursor

```
if let selectedRange = textView.selectedTextRange {
    let cursorPosition = textView.offsetFromPosition(textView.beginningOfDocument, toPosition:
selectedRange.start)
    print("\(cursorPosition)")
}
```

Establecer la posición del cursor

Para establecer la posición, todos estos métodos están configurando un rango con los mismos valores de inicio y final.

Al Principio

```
let newPosition = textView.beginningOfDocument
textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

Hasta el final

```
let newPosition = textView.endOfDocument
textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
```

A una posición a la izquierda de la posición actual del cursor

```
// only if there is a currently selected range
if let selectedRange = textView.selectedTextRange {

    // and only if the new position is valid
    if let newPosition = textView.positionFromPosition(selectedRange.start, inDirection:
UITextViewLayoutDirection.Left, offset: 1) {

        // set the new position
        textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
    }
}
```

A una posición arbitraria

Comienza por el principio y mueve 5 caracteres a la derecha.

```
let arbitraryValue: Int = 5
if let newPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextViewLayoutDirection.Right, offset: arbitraryValue) {

    textView.selectedTextRange = textView.textRangeFromPosition(newPosition, toPosition:
newPosition)
}
```

Relacionado

Seleccionar todo el texto

```
textView.selectedTextRange = textView.textRangeFromPosition(textView.beginningOfDocument,
toPosition: textView.endOfDocument)
```

Seleccione un rango de texto

```
// Range: 3 to 7
let startPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextViewLayoutDirection.Right, offset: 3)
let endPosition = textView.positionFromPosition(textView.beginningOfDocument, inDirection:
UITextViewLayoutDirection.Right, offset: 7)

if startPosition != nil && endPosition != nil {
    textView.selectedTextRange = textView.textRangeFromPosition(startPosition!, toPosition:
endPosition!)
}
```


Insertar texto en la posición actual del cursor

```
textView.insertText("Hello")
```

Notas

- Este ejemplo proviene originalmente de una adaptación de [esta respuesta de desbordamiento de pila](#) .
- Esta respuesta utiliza un campo de texto, pero los mismos conceptos se aplican a `UITextView` .
- Use `textView.becomeFirstResponder()` para enfocar el campo de texto y hacer que aparezca el teclado.
- Vea [esta respuesta](#) para [saber](#) cómo obtener el texto en algún rango.

Relacionado

- [Cómo crear un rango en Swift](#) (trata indirectamente el problema de por qué tenemos que usar `selectedTextRange` aquí en lugar de solo `selectedRange`)

Elimine los rellenos adicionales para ajustar a un texto medido con precisión.

`UITextView` tiene rellenos adicionales por defecto. A veces es molesto, especialmente si desea medir un texto sin una instancia de vista y ubicarlos en algún área con precisión.

Haga esto para eliminar tales rellenos.

```
messageTextView.textContainerInset = UIEdgeInsetsZero  
messageTextView.textContainer.lineFragmentPadding = 0
```

Ahora puede medir el tamaño del texto usando `NSAttributedString.boundingBoxWithSize(...)` , y cambiar el tamaño de un `UITextView` solo para ajustarlo al texto.

```
let budget = getSomeCGSizeBudget()  
let text = getSomeAttributedString()  
let textSize = text.boundingBoxWithSize(budget, options: [.UsesLineFragmentOrigin,  
.UsesFontLeading], context: nil).size  
messageTextView.frame.size = textSize // Just fits.
```

Lea `UITextView` en línea: <https://riptutorial.com/es/ios/topic/1043/uitextview>

Capítulo 201: UIViewController

Examples

Subclases

Subclase `UIControl` nos da acceso a los siguientes métodos:

- Se llama a `beginTrackingWithTouch` cuando el dedo toca por primera vez los límites del control.
- `continueTrackingWithTouch` se llama repetidamente a medida que el dedo se desliza por el control e incluso fuera de los límites del control.
- `endTrackingWithTouch` se llama cuando el dedo se levanta de la pantalla.

MyCustomControl.swift

```
import UIKit

// These are our self-defined rules for how we will communicate with other classes
protocol ViewControllerCommunicationDelegate: class {
    func myTrackingBegan()
    func myTrackingContinuing(location: CGPoint)
    func myTrackingEnded()
}

class MyCustomControl: UIControl {

    // whichever class wants to be notified of the touch events must set the delegate to
    // itself
    weak var delegate: ViewControllerCommunicationDelegate?

    override func beginTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?) -> Bool {

        // notify the delegate (i.e. the view controller)
        delegate?.myTrackingBegan()

        // returning true means that future events (like continueTrackingWithTouch and
        // endTrackingWithTouch) will continue to be fired
        return true
    }

    override func continueTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?) -> Bool
    {

        // get the touch location in our custom control's own coordinate system
        let point = touch.locationInView(self)

        // Update the delegate (i.e. the view controller) with the new coordinate point
        delegate?.myTrackingContinuing(point)

        // returning true means that future events will continue to be fired
        return true
    }
}
```

```

override func endTrackingWithTouch(touch: UITouch?, withEvent event: UIEvent?) {

    // notify the delegate (i.e. the view controller)
    delegate?.myTrackingEnded()
}
}

```

ViewController.swift

Así es como el controlador de vista está configurado para ser el delegado y responder a eventos táctiles de nuestro control personalizado.

```

import UIKit
class ViewController: UIViewController, ViewControllerCommunicationDelegate {

    @IBOutlet weak var myCustomControl: MyCustomControl!
    @IBOutlet weak var trackingBeganLabel: UILabel!
    @IBOutlet weak var trackingEndedLabel: UILabel!
    @IBOutlet weak var xLabel: UILabel!
    @IBOutlet weak var yLabel: UILabel!

    override func viewDidLoad() {
        super.viewDidLoad()
        myCustomControl.delegate = self
    }

    func myTrackingBegan() {
        trackingBeganLabel.text = "Tracking began"
    }

    func myTrackingContinuing(location: CGPoint) {
        xLabel.text = "x: \(location.x)"
        yLabel.text = "y: \(location.y)"
    }

    func myTrackingEnded() {
        trackingEndedLabel.text = "Tracking ended"
    }
}

```

Notas

- Los métodos alternativos para lograr el mismo resultado sin subclassificar incluyen agregar un objetivo o usar un reconocedor de gestos.
- No es necesario usar un delegado con estos métodos si solo se usan dentro del propio control personalizado. Podríamos haber añadido una declaración de `print` para mostrar cómo se llaman los eventos. En ese caso, el código se simplificaría para

```

import UIKit
class MyCustomControl: UIControl {

    override func beginTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?) ->
    Bool {
        print("Began tracking")
        return true
    }
}

```

```

    }

    override func continueTrackingWithTouch(touch: UITouch, withEvent event: UIEvent?)
-> Bool {
        let point = touch.locationInView(self)
        print("x: \(point.x), y: \(point.y)")
        return true
    }

    override func endTrackingWithTouch(touch: UITouch?, withEvent event: UIEvent?) {
        print("Ended tracking")
    }
}

```

Crear una instancia

Rápido

```
let viewController = UIViewController()
```

C objetivo

```
UIViewController *viewController = [UIViewController new];
```

Establecer la vista programáticamente

Rápido

```
class FooViewController: UIViewController {

    override func loadView() {
        view = FooView()
    }

}

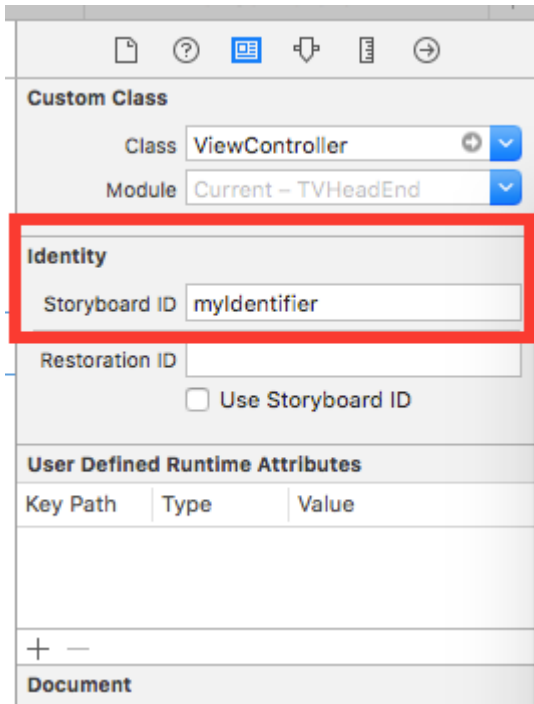
```

Instancia de un guión gráfico

```
UINavigationController *storyboard = [UINavigationController storyboardWithName:@"Main" bundle:nil];
```

Con un identificador :

Déle a la escena una identificación de guión gráfico dentro del inspector de identidad del guión gráfico.

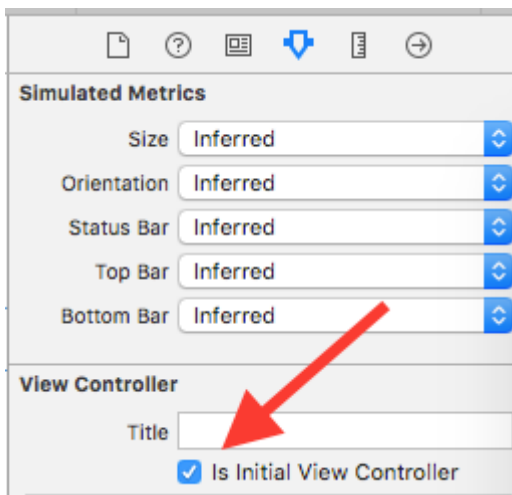


Instancia en el código:

```
UIViewController *controller = [storyboard  
instantiateViewControllerWithIdentifier:@"myIdentifier"];
```

Instalar un controlador de vista inicial :

Dentro del guión gráfico, seleccione el controlador de vista, luego seleccione el inspector de atributos, marque la casilla "Es el controlador de vista inicial".



```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];  
UIViewController *controller = [storyboard instantiateInitialViewController];
```

Accede al controlador de vista de contenedor

Cuando el controlador de vista se presenta dentro de un controlador de barra de pestañas, puede acceder al controlador de barra de pestañas de esta manera:

Rápido

```
let tabBarController = viewController.tabBarController
```

C objetivo

```
UITabBarController *tabBarController = self.tabBarController;
```

Cuando el controlador de vista forma parte de una pila de navegación, puede acceder al controlador de navegación de esta forma:

Rápido

```
let navigationController = viewController.navigationController
```

C objetivo

```
UINavigationController *navigationController = self.navigationController;
```

Agregar / quitar un controlador de vista hijo

Para agregar un controlador de vista secundario:

```
- (void)displayContentController:(UIViewController *)vc {  
    [self addChildViewController:vc];  
    vc.view.frame = self.view.frame;  
    [self.view addSubview:vc.view];  
    [vc didMoveToParentViewController:self];  
}
```

Para quitar un controlador de vista hijo:

```
- (void)hideContentController:(UIViewController *)vc {  
    [vc willMoveToParentViewController:nil];  
    [vc.view removeFromSuperview];  
    [vc removeFromParentViewController];  
}
```

Lea `UIViewController` en línea: <https://riptutorial.com/es/ios/topic/1956/uiviewcontroller>

Capítulo 202: UIViews personalizados de archivos XIB

Observaciones

Desde Apple: [Crear una vista personalizada que se rinda en Interface Builder](#)

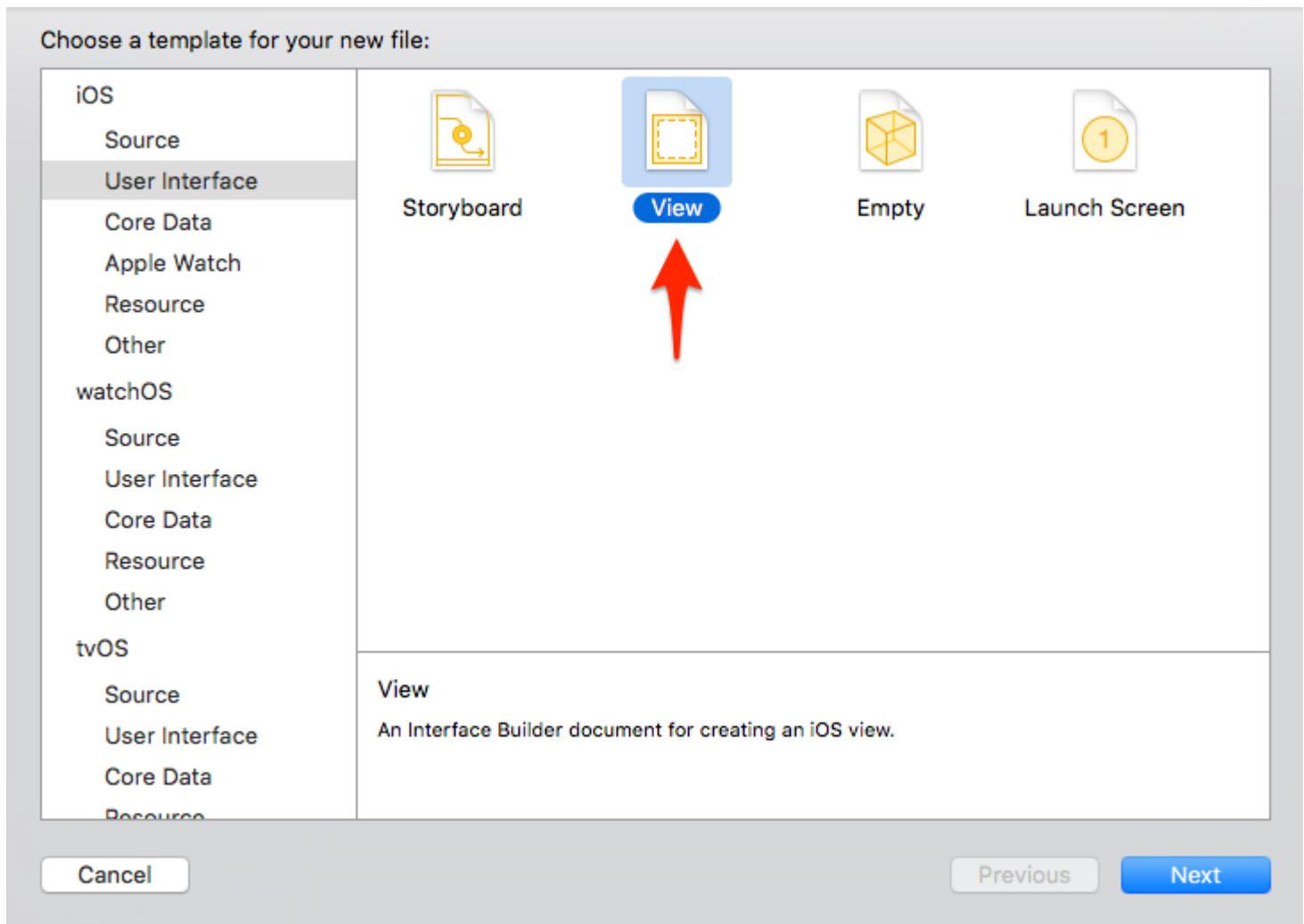
- Nota: tenga en cuenta que si utiliza fuentes 'personalizadas' de fantasía en sus elementos XIB (como UILabel, UITextField, etc.), el tiempo de carga inicial de su XIB será mayor dependiendo de la fuente elegida y la versión del sistema.

Examples

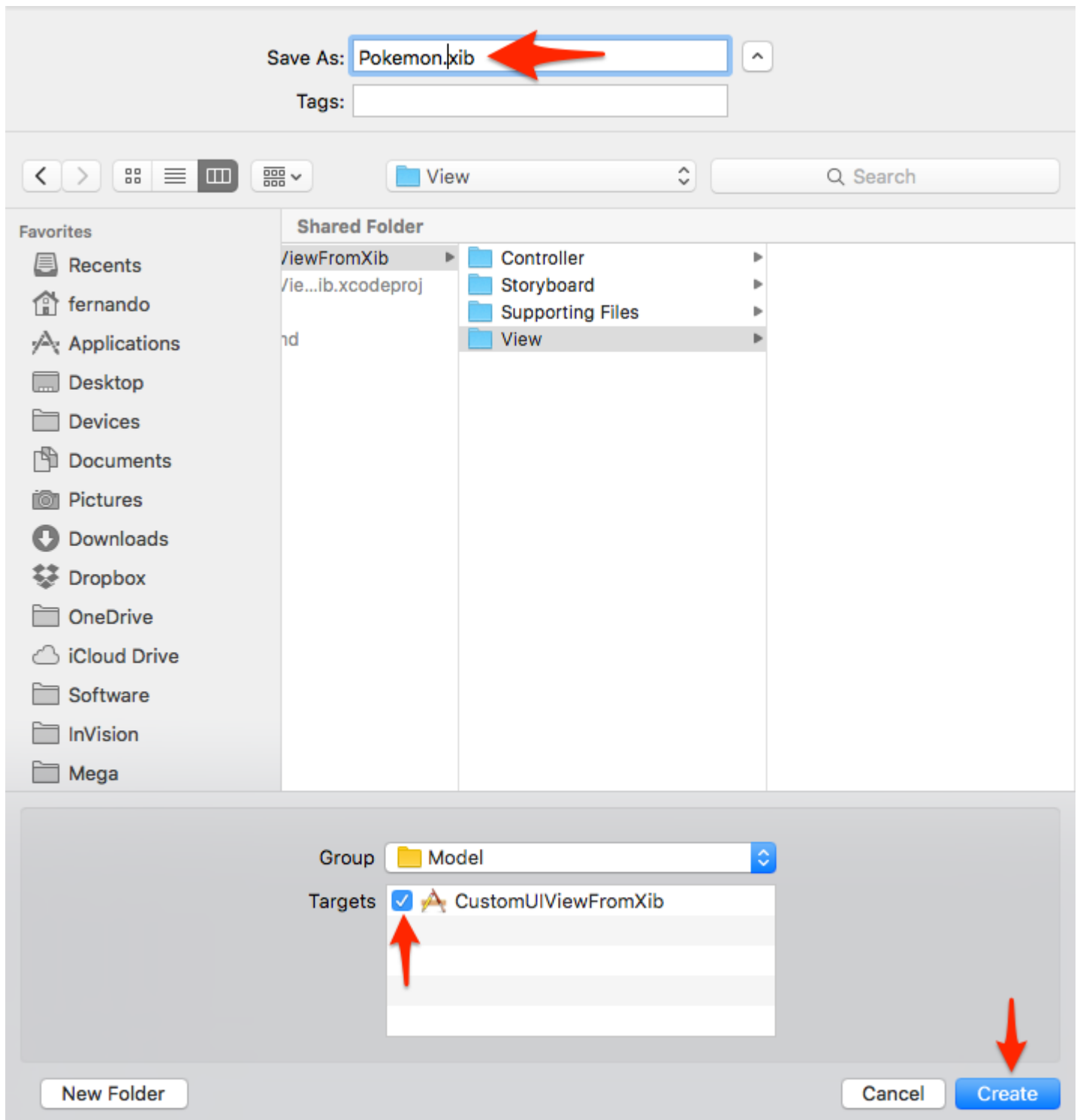
Elementos de cableado

Crear un archivo XIB

Barra de menú de Xcode> Archivo> Nuevo> Archivo.
Seleccione iOS, interfaz de usuario y luego "Ver":



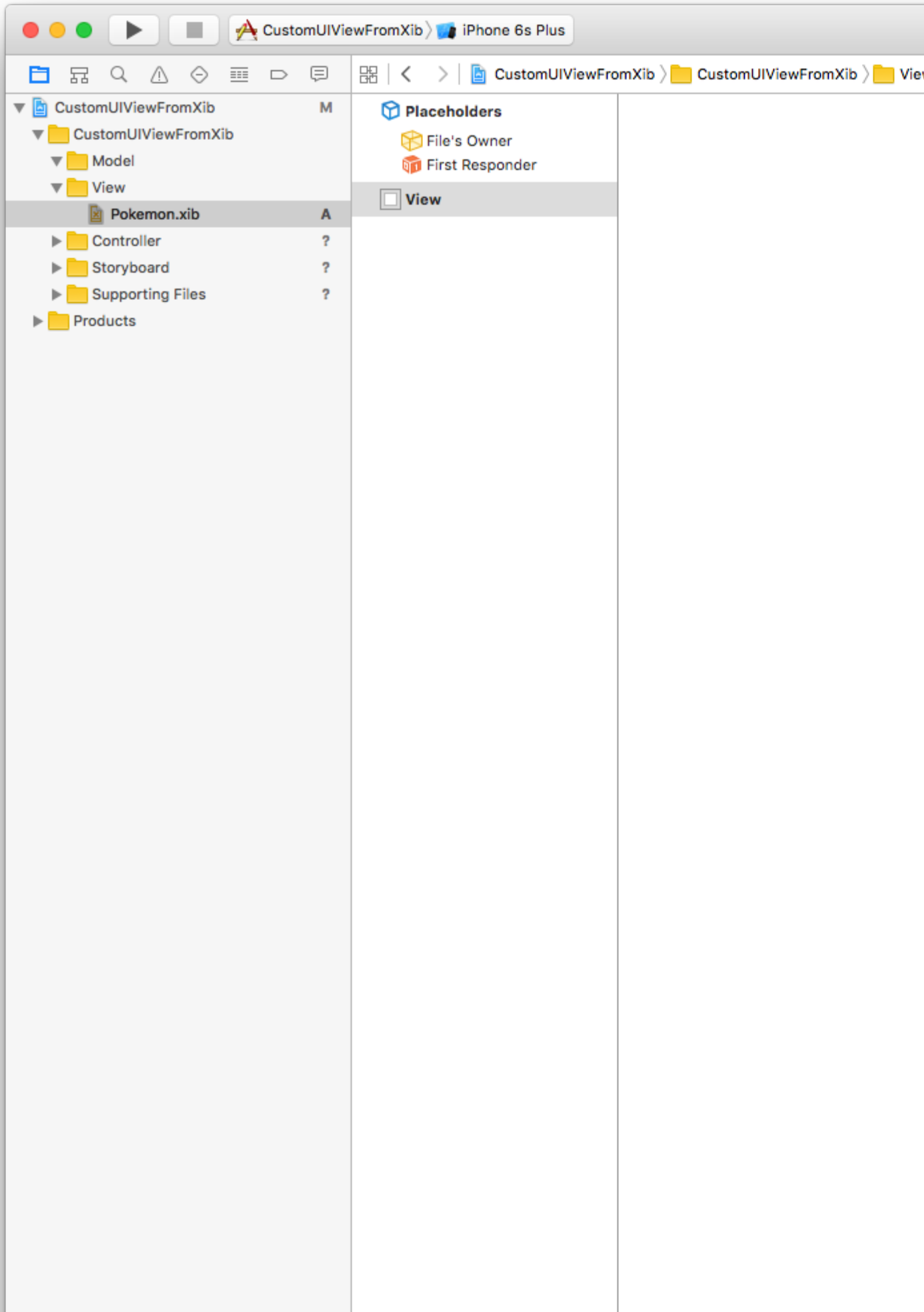
Asígnele un nombre a su XIB (sí, estamos haciendo un ejemplo de Pokémon).
Recuerda revisar tu objetivo y presiona "Crear".



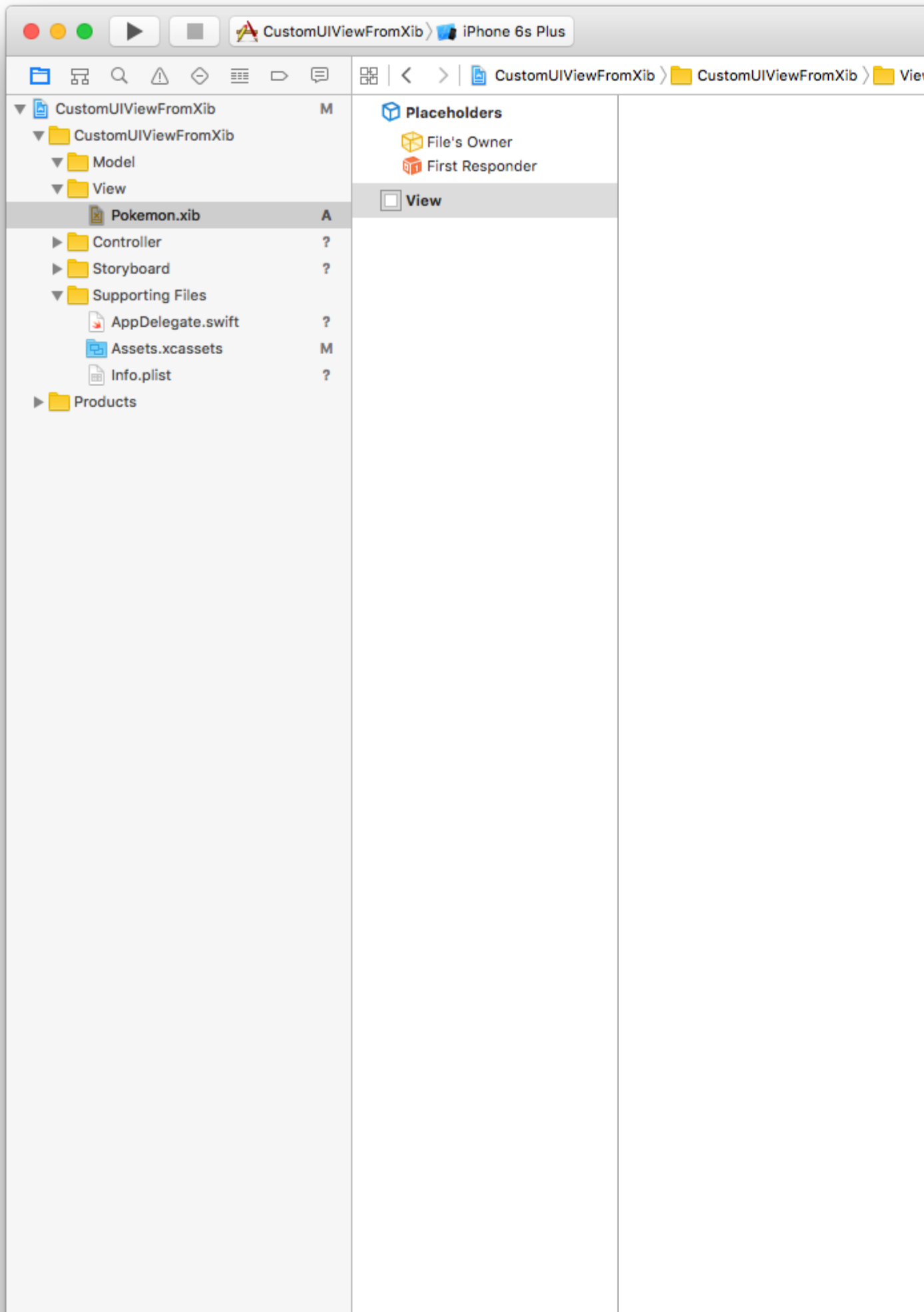
Diseña tu vista

Para hacer las cosas más fáciles, establece:

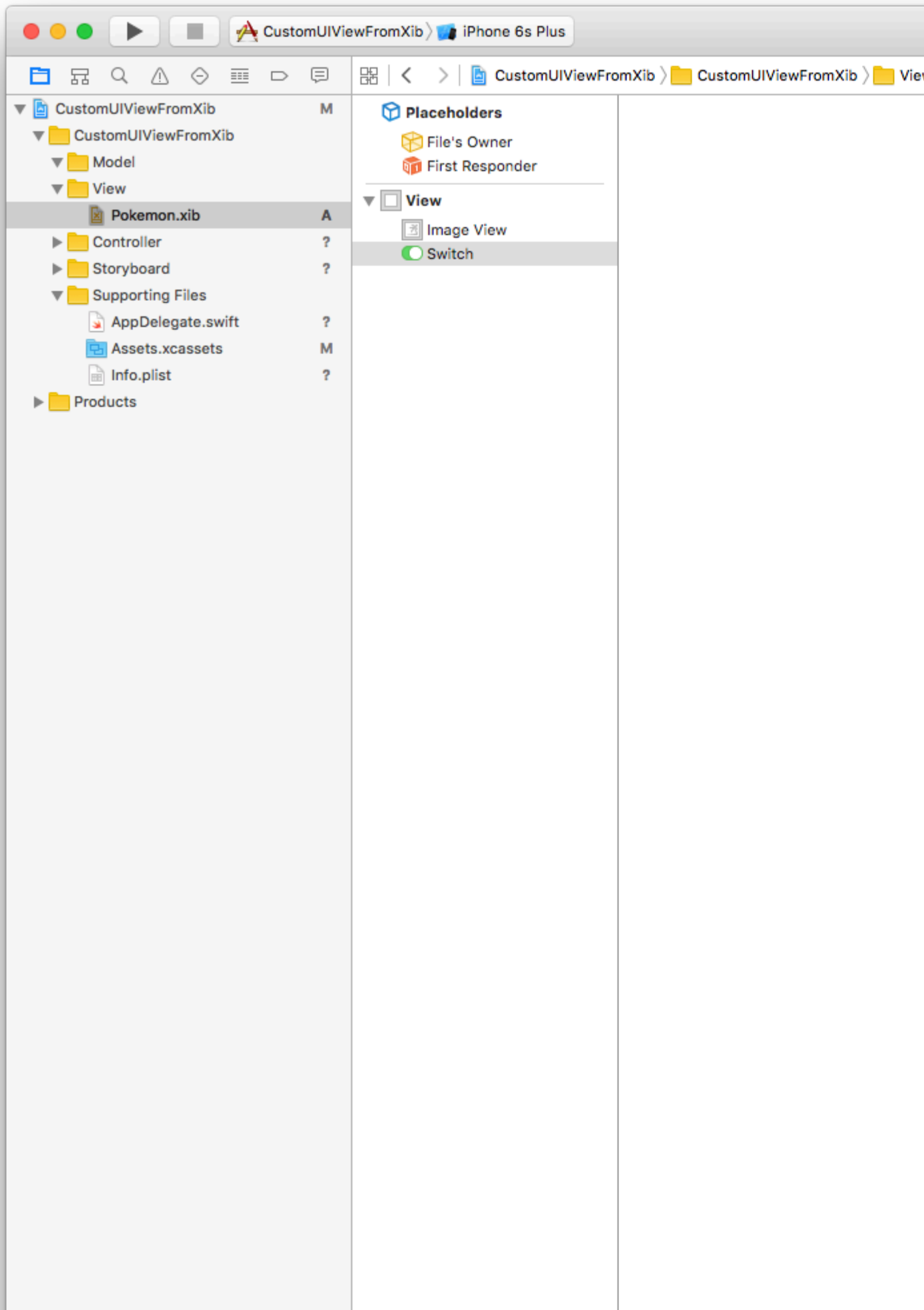
- Tamaño: forma libre
- Barra de estado: Ninguna
- Barra superior: Ninguna
- Barra inferior: Ninguna



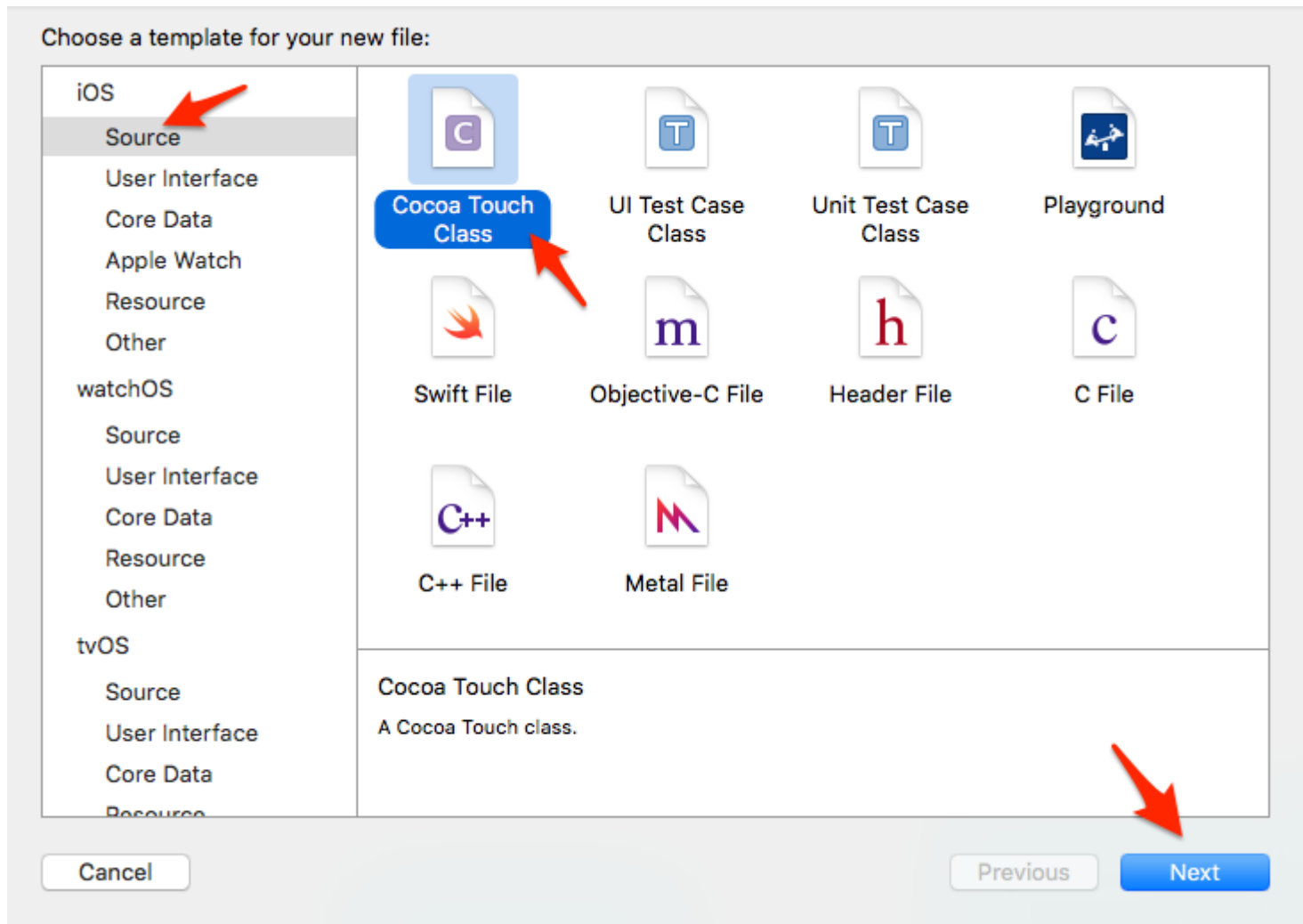
Para este ejemplo usaremos ancho 321 y alto 256.



Aquí agregaremos una **vista de imagen** (256x256) y un **interruptor** .



Barra de menú de Xcode> Archivo> Nuevo> Archivo.
Seleccione iOS / Source / Cocoa Touch Class. Pulse "Siguiente".



Asígnele un nombre a la clase, que debe ser el mismo nombre que el archivo XIB (Pokémon).
Seleccione UIView como el tipo de subclase, luego presione "Siguiente".

Choose options for your new file:

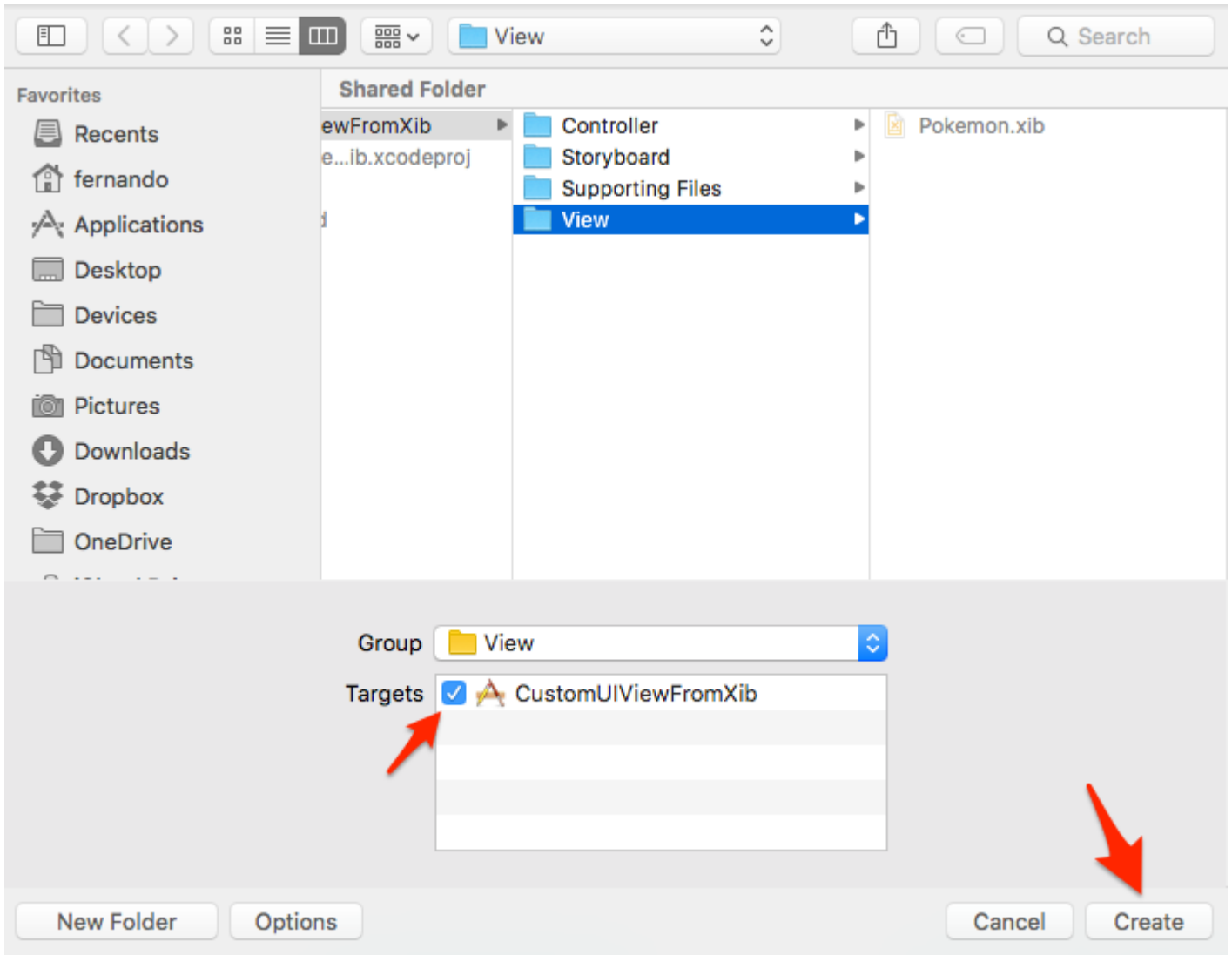
Class:

Subclass of:

Also create XIB file

Language:

En la siguiente ventana, selecciona tu objetivo y presiona "Crear".

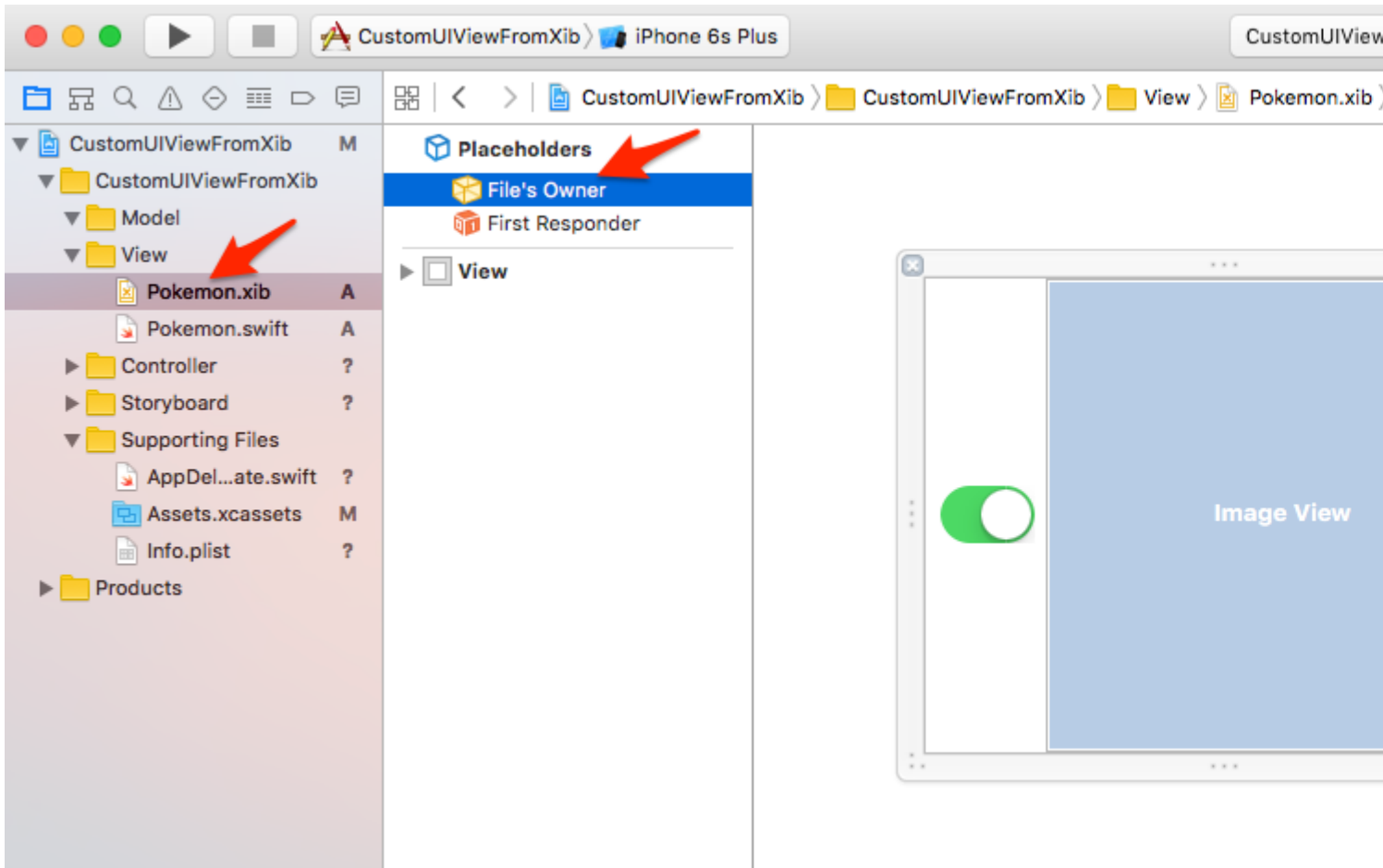


Conecta Pokemon.xib a Pokemon.swift a través del atributo "Propietario del archivo"

Haga clic en el archivo Pokemon.xib en Xcode.

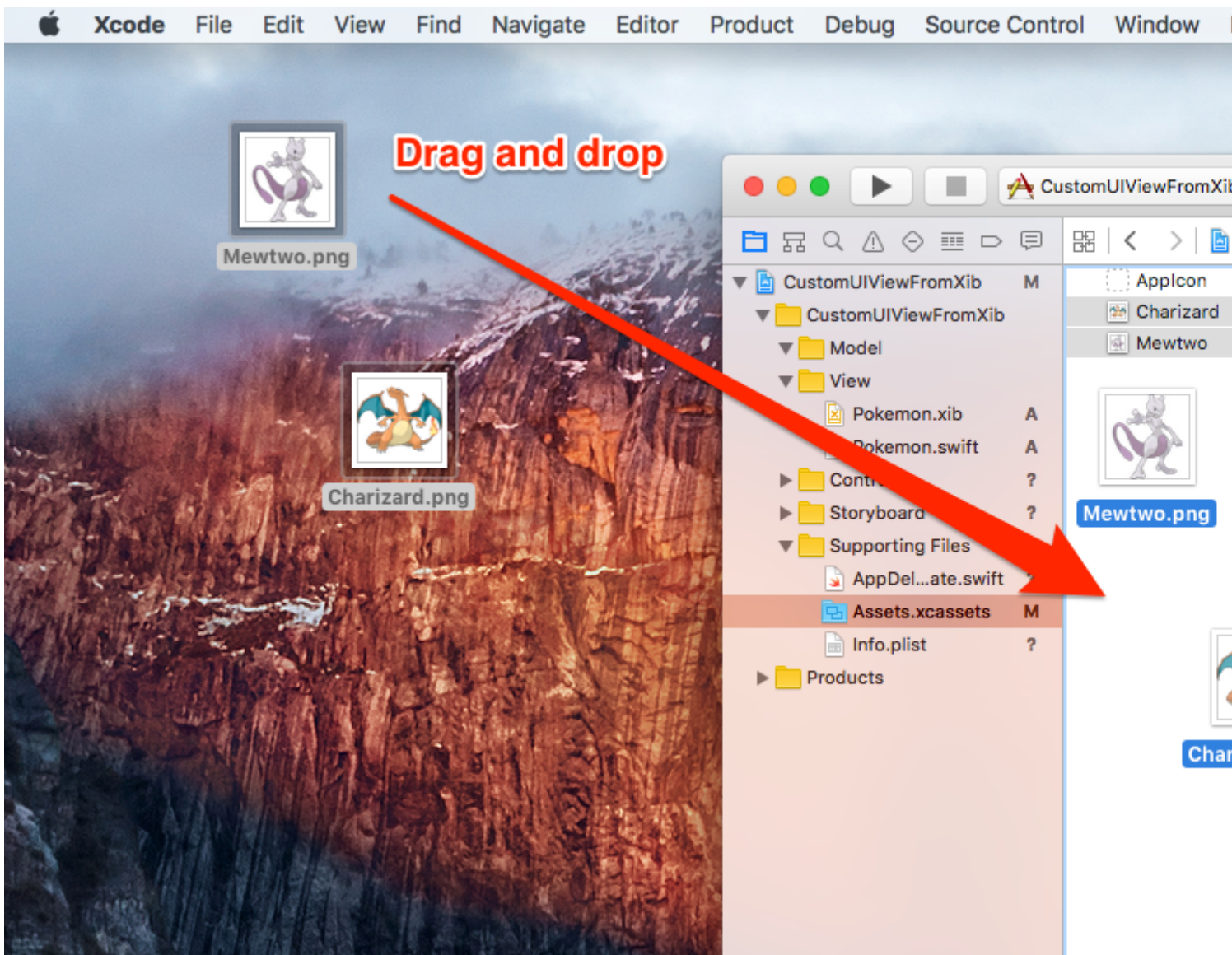
Haga clic en la salida "Propietario del archivo".

En el "inspector de identidad" (arriba a la derecha), configura la Clase en nuestro archivo Pokemon.swift creado recientemente.



POKEMONS !!!

¡Sí! Arrastra y suelta algunos Pokemon en tu proyecto para terminar nuestra "infraestructura". Aquí estamos agregando dos archivos PGN, 256x256, transparentes.



Muéstrame el código ya.

Bien, bien.

Es hora de agregar algo de código a nuestra clase Pokemon.swift.

En realidad es bastante simple:

1. Implementar los inicializadores requeridos.
2. Cargar el archivo XIB
3. Configure la vista que mostrará el archivo XIB
4. Mostrar la vista anterior

Agrega el siguiente código a la clase Pokemon.swift:

```
import UIKit

class Pokemon: UIView {

    // MARK: - Initializers

    override init(frame: CGRect) {
```

```

    super.init(frame: frame)
    setupView()
}

required init?(coder aDecoder: NSCoder) {
    super.init(coder: aDecoder)
    setupView()
}

// MARK: - Private Helper Methods

// Performs the initial setup.
private func setupView() {
    let view = viewFromNibForClass()
    view.frame = bounds

    // Auto-layout stuff.
    view.autoresizingMask = [
        UIViewAutoresizing.flexibleWidth,
        UIViewAutoresizing.flexibleHeight
    ]

    // Show the view.
    addSubview(view)
}

// Loads a XIB file into a view and returns this view.
private func viewFromNibForClass() -> UIView {

    let bundle = Bundle(for: type(of: self))
    let nib = UINib(nibName: String(describing: type(of: self)), bundle: bundle)
    let view = nib.instantiate(withOwner: self, options: nil).first as! UIView

    /* Usage for swift < 3.x
    let bundle = NSBundle(forClass: self.dynamicType)
    let nib = UINib(nibName: String(self.dynamicType), bundle: bundle)
    let view = nib.instantiateWithOwner(self, options: nil)[0] as! UIView
    */

    return view
}
}

```

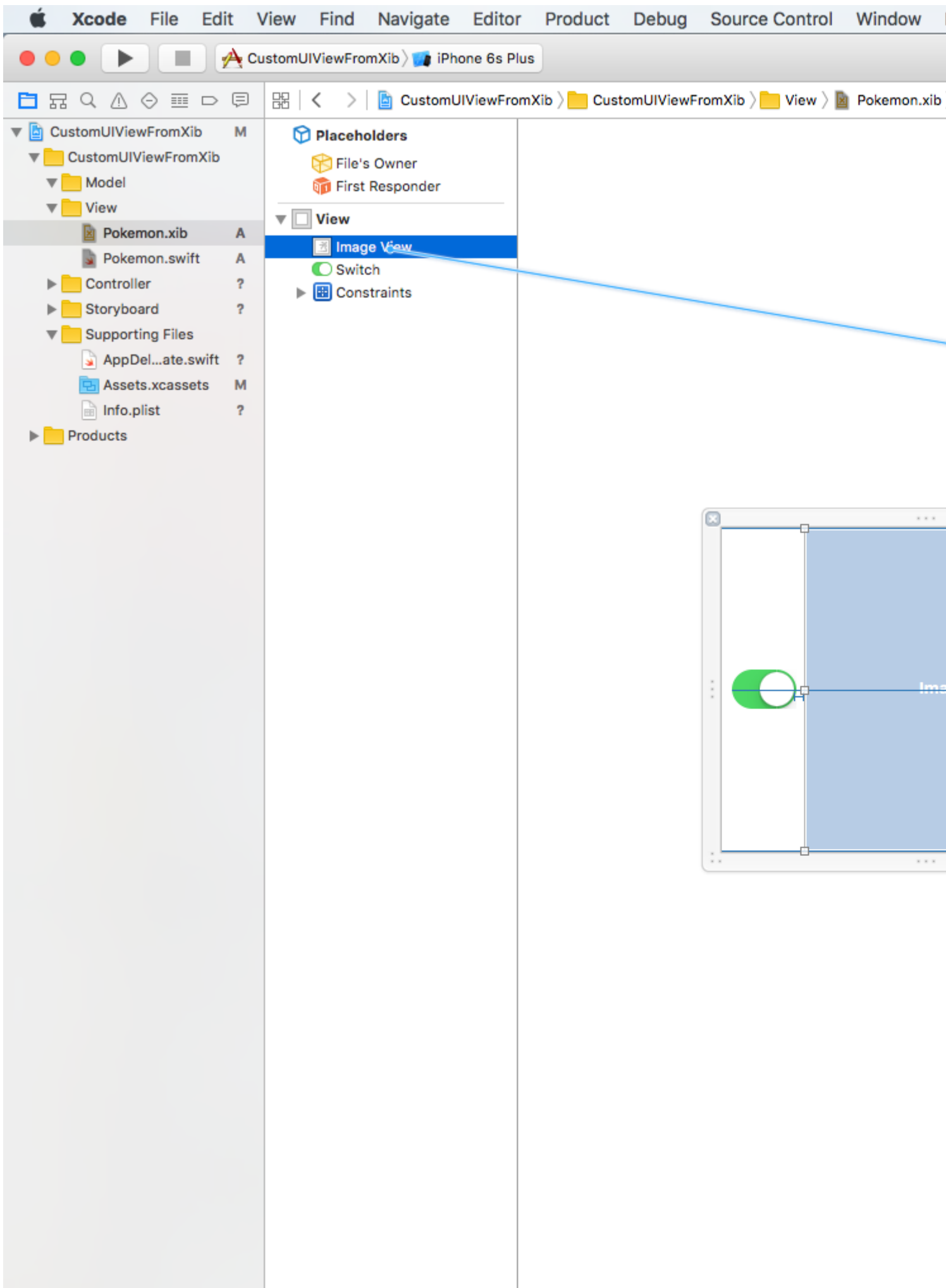
@IBDesignable y @IBInspectable

Al agregar `@IBDesignable` a su clase, hace posible que se `@IBDesignable` en vivo en Interface Builder.

Al agregar `@IBInspectable` a las propiedades de su clase, puede ver cómo cambian sus vistas personalizadas en Interface Builder tan pronto como modifica esas propiedades.

Hagamos la `Image View` de `Image View` de nuestra vista personalizada "Inspeccionable".

Primero, conecta la `Image View` de imagen del archivo `Pokemon.xib` a la clase `Pokemon.swift`.



antes del nombre de la clase):

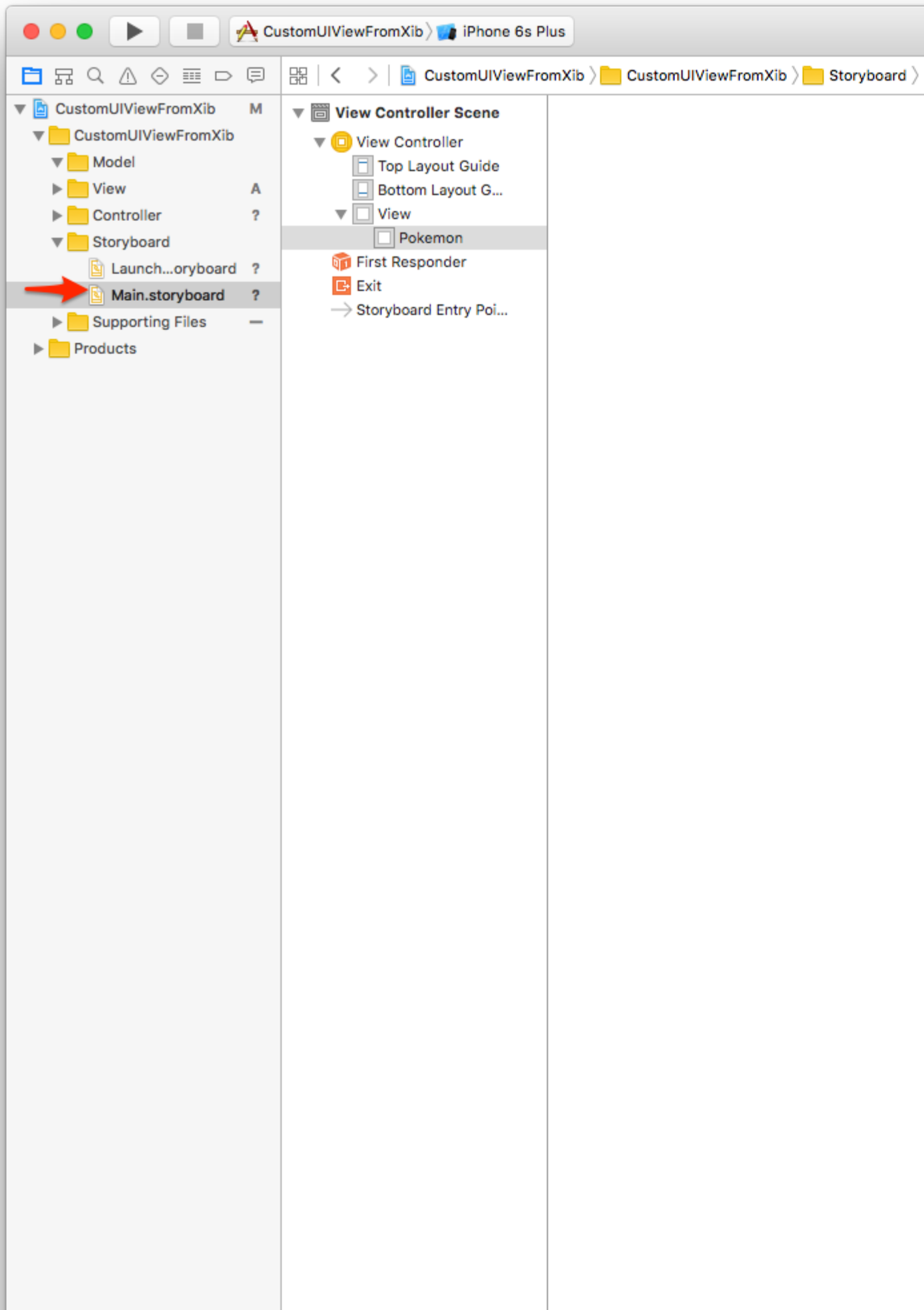
```
@IBDesignable class Pokemon: UIView {  
  
    // MARK: - Properties  
  
    @IBOutlet weak var imageView: UIImageView!  
  
    @IBInspectable var image: UIImage? {  
        get {  
            return imageView.image  
        }  
        set(image) {  
            imageView.image = image  
        }  
    }  
  
    // MARK: - Initializers  
    ...  
}
```

Usando sus vistas personalizadas

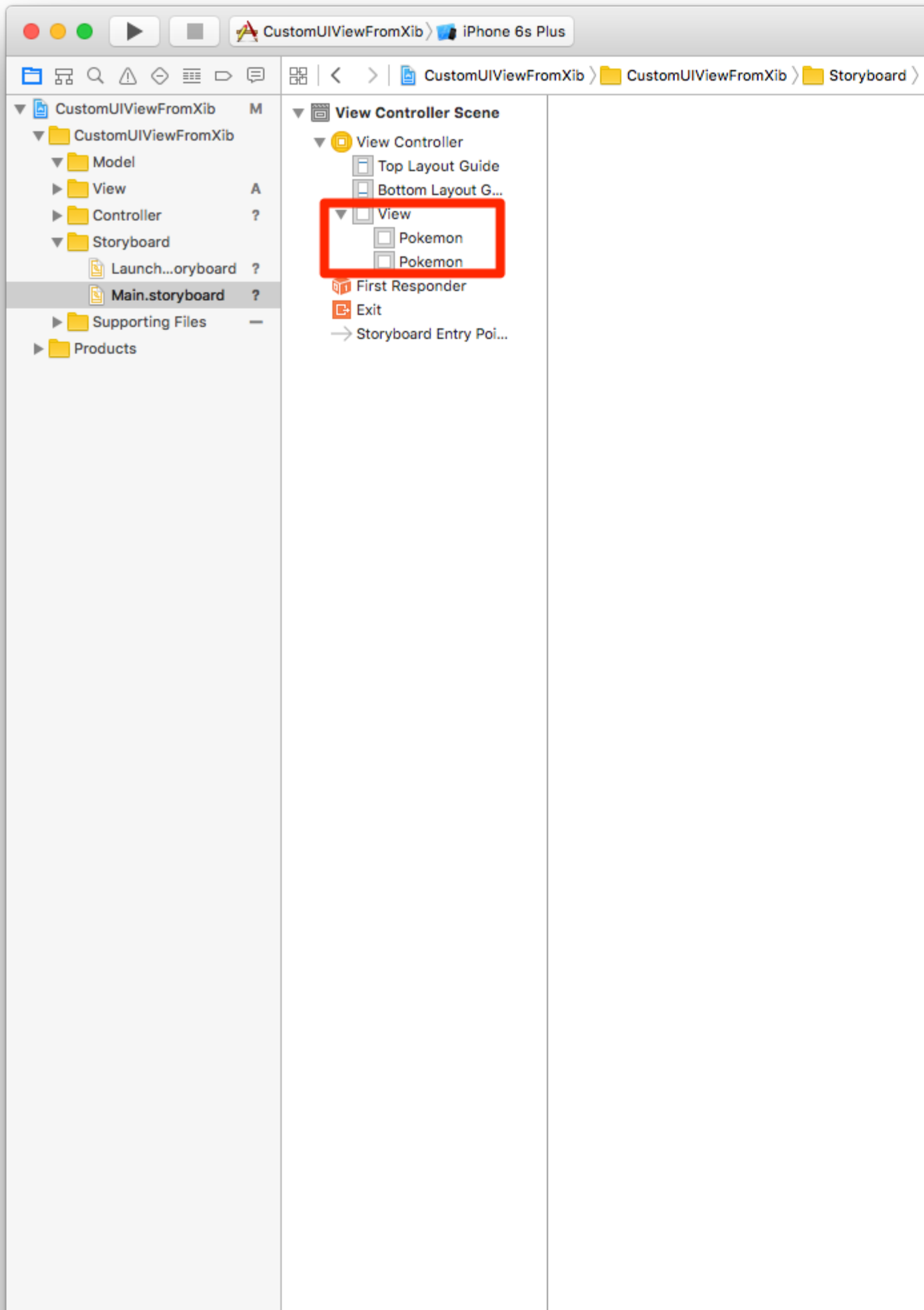
Llegó a su archivo de guión gráfico principal, arrastre una vista UIV en él.

Cambiar el tamaño de la vista a, digamos 200x200. Centralizar.

Ve al inspector de identidad (arriba a la derecha) y configura la clase en Pokémon.



Dale un tamaño diferente, digamos 150x150.
Elige otra imagen de Pokémon, observa:



El botón permitirá que los Pokemon sean habilitados / deshabilitados.

Crea una `IBAction` desde el botón Cambiar a la clase `Pokemon.swift`.

Llama a la acción algo así como `switchTapped`.


Agregue el siguiente código:

```
// MARK: - Actions

@IBAction func switchTapped(sender: UISwitch) {
    imageView.alpha = sender.on ? 1.0 : 0.2
}

// MARK: - Initializers
...
```

Resultado final:

Carrier 

3:54 PM



Game Center



Extras



Watch



CustomUIV...



Ahora puede crear vistas personalizadas complejas y reutilizarlas en cualquier lugar que desee. Esto aumentará la productividad al tiempo que se aísla el código en elementos de la interfaz de usuario autónomos.

[El proyecto final puede ser clonado en Github.](#)

(**Actualizado a Swift 3.1**)

Cómo hacer UIView reutilizable personalizado utilizando XIB

El siguiente ejemplo muestra los pasos involucrados en la inicialización de una vista desde XIB.

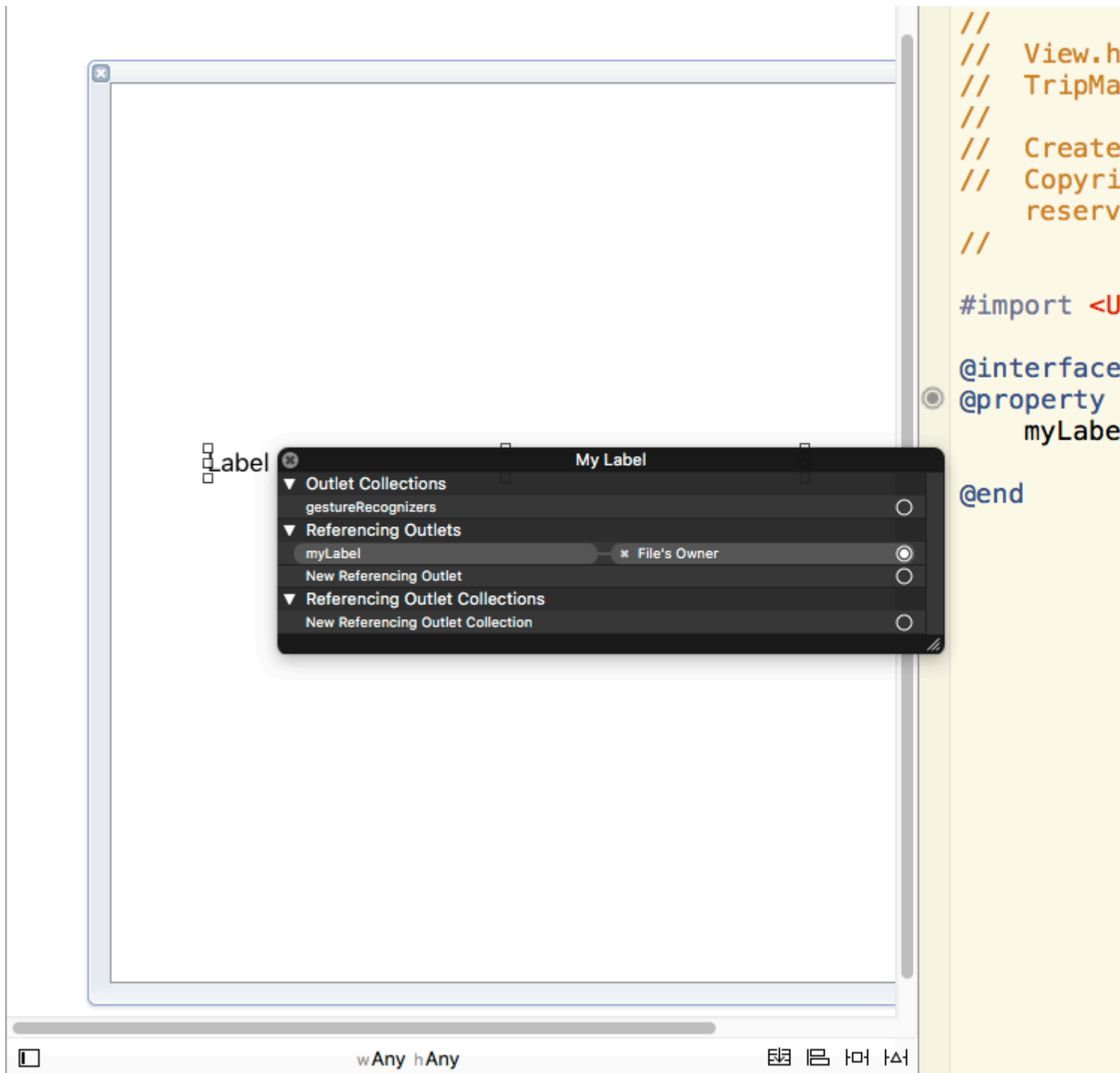
Esta no es una operación compleja, pero se deben seguir los pasos exactos para hacerlo correctamente la primera vez, evitando excepciones.

[¿Cómo funciona loadNibNamed?](#)

Los pasos principales son:

1. Crear XIB
2. Crear clase .h y .m
3. Definir salidas en .h
4. Conecte las salidas entre .h y XIB

Ver captura de pantalla adjunta:



5. Invoque `loadNibNamed` dentro de la función `initWithCoder` del archivo `.m`. Esto es necesario para asegurarse de que puede colocar directamente el objeto `UIView` en el archivo `storyboard` / `Parent UIView XIB` y definirlo como su vista personalizada. No se necesita ningún otro código de inicialización una vez que cargue el `storyboard` / `parent XIB`. Su vista personalizada se puede agregar a otras vistas al igual que otros objetos de vista de Objective C incorporados que se proporcionan en XCode.

Lea [UIViews personalizados de archivos XIB en línea](https://riptutorial.com/es/ios/topic/1362/uiviews-personalizados-de-archivos-xib):

<https://riptutorial.com/es/ios/topic/1362/uiviews-personalizados-de-archivos-xib>

Capítulo 203: UIWebView

Observaciones

Funciones de delegado UIWebView: -

Objective-C Declerations

```
- (BOOL)webView:(UIWebView *)webView
shouldStartLoadWithRequest:(NSURLRequest *)request
navigationType:(UIWebViewNavigationType)navigationType;

- (void)webView:(UIWebView *)webView
didFailLoadWithError:(NSError *)error;

- (void)webViewDidFinishLoad:(UIWebView *)webView;

- (void)webViewDidStartLoad:(UIWebView *)webView;
```

Examples

Crear una instancia de UIWebView

Rápido

```
let webview = UIWebView(frame: CGRect(x: 0, y: 0, width: 320, height: 480))
```

C objetivo

```
UIWebView *webview = [[UIWebView alloc] initWithFrame:CGRectMake(0, 0, 320, 480)];

//Alternative way of defining frame for UIWebView
UIWebView *webview = [[UIWebView alloc] init];
CGRect webviewFrame = webview.frame;
webviewFrame.size.width = 320;
webviewFrame.size.height = 480;
webviewFrame.origin.x = 0;
webviewFrame.origin.y = 0;
webview.frame = webviewFrame;
```

Hacer una solicitud de URL

Cargar contenido en webview desde la `url`

Rápido

```
webview.loadRequest(NSURLRequest(URL: NSURL(string: "http://www.google.com"))) )
```

C objetivo

```
[webView loadRequest:[NSURLRequest requestWithURL:[NSURL
URLWithString:@"http://www.google.com"]]]];
```

Dejar de cargar contenido web

El método `stopLoading()` detiene el proceso de carga actual de la vista web.

Rápido

```
webView.stopLoading()
```

C objetivo

```
[webView stopLoading];
```

Actualizar contenido web actual

Rápido

```
webView.reload()
```

C objetivo

```
[webView reload];
```

Determinar el tamaño del contenido

En muchos casos, por ejemplo, cuando se usan vistas web en celdas de vista de tabla, es importante determinar el tamaño del contenido de la página HTML representada. Después de cargar la página, esto se puede calcular en el método de delegado `UIWebViewDelegate` :

```
- (void) webViewDidFinishLoad:(UIWebView *) aWebView {
    CGRect frame = aWebView.frame;
    frame.size.height = 1;
    aWebView.frame = frame;
    CGSize fittingSize = [aWebView sizeThatFits:CGSizeZero];
    frame.size = fittingSize;
    aWebView.frame = frame;

    NSLog(@"size: %f, %f", fittingSize.width, fittingSize.height);
}
```

El código emplea un truco adicional de configurar brevemente la altura de la vista web en 1 antes de medir el tamaño del accesorio. De lo contrario, simplemente informaría el tamaño del marco actual. Después de la medición, establecemos inmediatamente la altura a la altura del contenido real.

Fuente

Cargar cadena HTML

Las vistas web son útiles para cargar cadenas HTML generadas localmente.

```
NSString *html = @"<!DOCTYPE html><html><body>Hello World</body></html>";  
[webView loadHTMLString:html baseURL:nil];
```

Rápido

```
let htmlString = "<h1>My First Heading</h1><p>My first paragraph.</p>"  
webView.loadHTMLString(htmlString, baseURL: nil)
```

Se puede especificar una URL base local. Esto es útil para hacer referencia a imágenes, hojas de estilo o scripts del paquete de aplicaciones:

```
NSString *html = @"<!DOCTYPE html><html><head><link href='style.css' rel='stylesheet'  
type='text/css'></head><body>Hello World</body></html>";  
[self loadHTMLString:html baseURL:[NSURL fileURLWithPath:[NSBundle mainBundle]  
resourcePath]]];
```

En este caso, `style.css` se carga localmente desde el directorio de recursos de la aplicación. Por supuesto, también es posible especificar una URL remota.

Cargar JavaScript

Podemos ejecutar JavaScript personalizado en un `UIWebView` usando el método `stringByEvaluatingJavaScriptFromString()`. Este método devuelve el resultado de ejecutar el script de JavaScript pasado en el parámetro del script, o `nil` si el script falla.

Rápido

Cargar script desde cadena

```
webView.stringByEvaluatingJavaScriptFromString("alert('This is JavaScript!');")
```

Cargar script desde archivo local

```
//Suppose you have javascript file named "JavaScript.js" in project.  
let filePath = NSBundle.mainBundle().pathForResource("JavaScript", ofType: "js")  
do {  
    let jsContent = try String.init(contentsOfFile: filePath!, encoding:  
NSUTF8StringEncoding)  
    webView.stringByEvaluatingJavaScriptFromString(jsContent)  
}  
catch let error as NSError{  
    print(error.debugDescription)  
}
```

C objetivo

Cargar script desde cadena

```
[webview stringByEvaluatingJavaScriptFromString:@"alert('This is JavaScript!');"];
```

Cargar script desde archivo local

```
//Suppose you have javascript file named "JavaScript.js" in project.  
NSString *filePath = [[NSBundle mainBundle] pathForResource:@"JavaScript" ofType:@"js"];  
NSString *jsContent = [NSString stringWithContentsOfFile:filePath  
encoding:NSUTF8StringEncoding error:nil];  
[webview stringByEvaluatingJavaScriptFromString:jsContent];
```

Nota `stringByEvaluatingJavaScriptFromString:` método `stringByEvaluatingJavaScriptFromString:` espera sincrónicamente a que se complete la evaluación de JavaScript. Si carga contenido web cuyo código JavaScript no ha examinado, invocar este método podría bloquear su aplicación. La mejor práctica es adoptar la clase `WKWebView` y usar su método `WKWebView` `evaluateJavaScript:completionHandler:` lugar. Pero `WKWebView` está disponible desde iOS 8.0 y versiones posteriores.

Cargue archivos de documentos como .pdf, .txt, .doc, etc.

En lugar de páginas web, también podemos cargar los archivos de documentos en iOS WebView como .pdf, .txt, .doc, etc. `loadData` método `loadData` se utiliza para cargar `NSData` en webview.

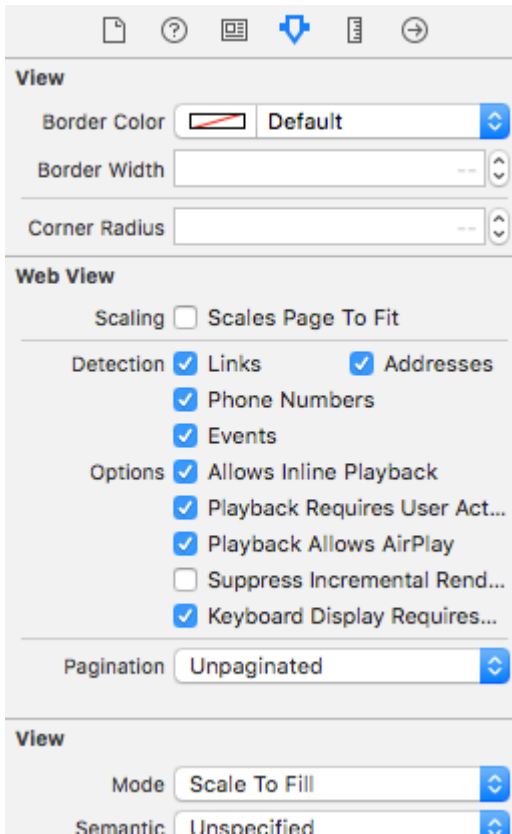
Rápido

```
//Assuming there is a text file in the project named "home.txt".  
let localFilePath = NSBundle.mainBundle().pathForResource("home", ofType:"txt");  
let data = NSFileManager.defaultManager().contentsAtPath(localFilePath!);  
webview.loadData(data!, MIMEType: "application/txt", textEncodingName:"UTF-8" , baseURL:  
NSURL())
```

C objetivo

```
//Assuming there is a text file in the project named "home.txt".  
NSString *localFilePath = [[NSBundle mainBundle] pathForResource:@"home" ofType:@"txt"];  
NSData *data = [[NSFileManager defaultManager] contentsAtPath:localFilePath];  
[webview loadData:data MIMEType:@"application/txt" textEncodingName:@"UTF-8" baseURL:[NSURL  
new]];
```

Hacer enlaces que dentro de UIWebView haga clic en



En vc.h

```
@interface vc : UIViewController<UIWebViewDelegate>
```

en vc.m

```
- (BOOL)webView:(UIWebView *)webView shouldStartLoadWithRequest:(NSURLRequest *)request
navigationType:(UIWebViewNavigationType)navigationType{

    if (navigationType == UIWebViewNavigationTypeLinkClicked){
        //open it on browser if you want to open it in same web view remove return NO;
        NSURL *url = request.URL;
        if ([[UIApplication sharedApplication] canOpenURL:url]) {
            [[UIApplication sharedApplication] openURL:url];
        }
        return NO;
    }

    return YES;
}
```

Cargar archivo HTML local en webView

Primero, agregue el archivo HTML a su proyecto (si se le pide que elija opciones para agregar el archivo, seleccione *Copiar elementos si es necesario*)

La siguiente línea de código carga el contenido del archivo HTML en el webView

```
webView.loadRequest(NSURLRequest(URL: NSURL(fileURLWithPath:  
NSBundle mainBundle().pathForResource("YOUR HTML FILE", ofType: "html")!))
```

- Si su archivo HTML se llama index.html, reemplace **SU ARCHIVO HTML** por **índice**
- Puede usar este código en *viewDidLoad ()* o *viewWillAppear ()* o en cualquier otra función

Lea **UIWebView** en línea: <https://riptutorial.com/es/ios/topic/1452/uiwebview>

Capítulo 204: Usando Aseets de Imagen

Introducción

Los activos de imagen se utilizan para administrar y organizar diferentes tipos de activos de imagen en nuestra aplicación iOS usando Xcode.

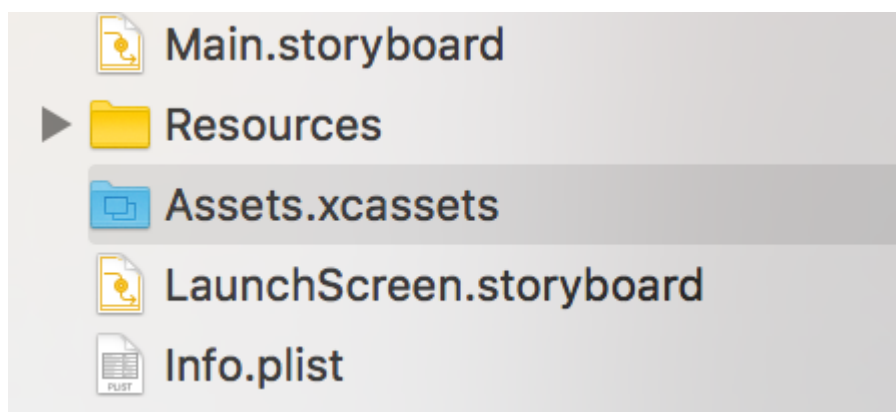
Estos activos pueden ser **iconos de aplicaciones, imágenes de inicio, imágenes utilizadas en toda la aplicación, imágenes de tamaño completo, imágenes de tamaño aleatorio**, etc.

Examples

Icono de la aplicación utilizando los activos de imagen

Cada vez que creamos un nuevo proyecto en Xcode para nuestra nueva aplicación, nos brinda varias clases, objetivos, pruebas, archivos plist, etc. incorporados. Del mismo modo, también nos da como archivo `Assets.xcassets` , que administra todos los recursos de imagen en nuestro proyecto.

Así es como se ve este archivo en el navegador de archivos:

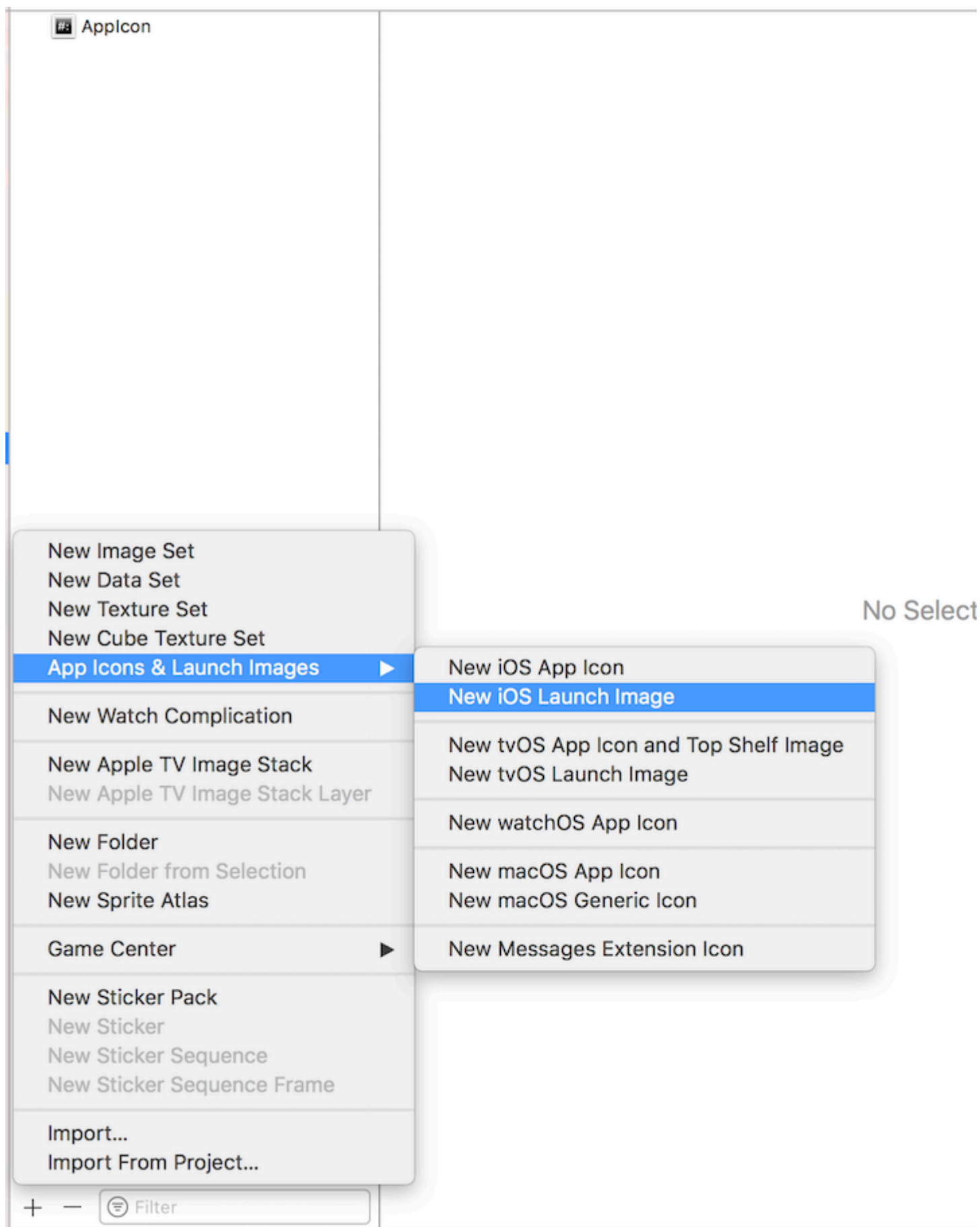


Si hacemos clic en él, se verá así:

Solo tenemos que **arrastrar y soltar la** imagen correspondiente en cada bloque cuadrado vacío. Cada negro nos dirá qué tamaño debe tener esa imagen, está escrito justo debajo. Después de arrastrar y soltar todas las imágenes en todos los cuadrados, se verá así:



en la parte inferior como:



Después de esto, de acuerdo con nuestro requisito, podemos cambiar las casillas vacías a

dispositivos que admitimos mediante el uso del inspector de atributos marcando / desmarcando casillas.

Llené estas imágenes para iPhones de 4 "a 5.5" y para todos los iPads como:



AppIcon



LaunchImage

LaunchImage

Aquí están los tamaños de todas las imágenes de lanzamiento:

```
Retina HD 5.5" iPhone Portrait - iPhone (6, 6S, 7)Plus - 1242x2208px
Retina HD 4.7" iPhone Portrait - iPhone 6, 6S, 7 - 750x1334px
Retina HD 5.5" iPhone Landscape - iPhone (6, 6S, 7)Plus - 2208x1242px
2x iPhone Portrait - (3.5") iPhone 4S - 640x960px
Retina 4 iPhone Portrait - (4") iPhone 5, 5S, 5C, iPod Touch, SE - 640x1136px
2x iPad Portrait - All Retina iPads - 1536x2048px
2x iPad Landscape - All Retina iPads - 2048x1536px
```

Notas:

1 iPad sin retina: Deje en blanco `1x iPad Portrait and Landscape` porque los iPad sin retina usarán imágenes de lanzamiento `2x` al escalar

2 iPad Pro de 12.9 " : no hay una casilla para este iPad porque este iPad también usará imágenes de `2x iPad` al escalarlas

3 Retina HD 5.5 ": los iPad deben tener `1920x1080px` para el retrato y `1080x1920px` para el paisaje, pero Xcode dará warning y la imagen de lanzamiento no se mostrará en esos dispositivos

4 SplitView: como estamos usando `LaunchImage Asset` lugar de `LaunchScreen XIB` , nuestra aplicación no admite `SplitView` en iPads y iPhones horizontales de 5.5 "

5 Reinstalar: si nuestra aplicación ya está instalada en el dispositivo e intentamos ejecutar con estos recursos de imágenes de lanzamiento recién agregados, a veces el dispositivo no mostrará las imágenes de inicio al iniciar la aplicación. En este caso, simplemente elimine la aplicación del dispositivo, limpie y compile el proyecto y ejecútelo, mostrará nuevas imágenes de inicio

Lea Usando Aseets de Imagen en línea: <https://riptutorial.com/es/ios/topic/10087/usando-aseets-de-imagen>

Capítulo 205: UUID (identificador único universal)

Observaciones

Para guardar el UUID podemos usar [SSKeychainUtility](#) . El ejemplo se puede encontrar en la página de Github

Examples

Generando UUID

UUID aleatorio

Rápido

```
func randomUUID() -> NSString{
    return NSUUID.UUID().UUIDString()
}
```

C objetivo

```
+ (NSString *)randomUUID {
    if(NSClassFromString(@"NSUUID")) { // only available in iOS >= 6.0
        return [[NSUUID UUID] UUIDString];
    }
    CFUUIDRef uuidRef = CFUUIDCreate(kCFAllocatorDefault);
    CFStringRef cfuuid = CFUUIDCreateString(kCFAllocatorDefault, uuidRef);
    CFRelease(uuidRef);
    NSString *uuid = [((__bridge NSString *) cfuuid) copy];
    CFRelease(cfuuid);
    return uuid;
}
```

Identificador para el vendedor

ios 6

Dentro de una sola línea, podemos obtener un UUID como a continuación:

Rápido

```
let UDIDString = UIDevice.currentDevice().identifierForVendor?.UUIDString
```

C objetivo

```
NSString *UDIDString = [[[UIDevice currentDevice] identifierForVendor] UUIDString];
```

El `identifierForVendor` es un identificador único que permanece igual para cada aplicación de un solo proveedor en un solo dispositivo, a menos que todas las aplicaciones del proveedor se eliminen de este dispositivo. Consulte [la documentación de Apple](#) sobre cuándo cambia este `UUID`.

IFA de Apple frente a IFV (Identificador de Apple para anunciantes frente a Identificador de proveedores)

- Puede utilizar el IFA para medir clics de anuncios y el IFV para medir instalaciones de aplicaciones.
- IFA tiene mecanismos de privacidad incorporados que lo hacen perfecto para la publicidad. En contraste, el IFV es para que los desarrolladores lo utilicen internamente para medir a los usuarios que instalan sus aplicaciones.

IFA

- `ASIdentifierManager` proporciona la clase
 - **advertisingIdentifier: UUID** : una cadena alfanumérica única para cada dispositivo, utilizada solo para publicar anuncios.
 - **isAdvertisingTrackingEnabled** : un valor booleano que indica si el usuario tiene un seguimiento de anuncios limitado.

IFV

- `ASIdentifierManager` proporciona la clase
 - **identifierForVendor: UUID** : una cadena alfanumérica que identifica de forma única un dispositivo al proveedor de la aplicación.

Encuentra tu dispositivo IFA y IFV [aquí](#) .

Crear una cadena UUID para dispositivos iOS

Aquí podemos crear una `UUID String` con una línea.

Representa cadenas UUID, que se pueden usar para identificar de forma única los tipos, interfaces y otros elementos.

Swift 3.0

```
print(UUID().uuidString)
```

Es muy útil para identificar múltiples dispositivos con identificación única.

Lea UUID (identificador único universal) en línea: <https://riptutorial.com/es/ios/topic/3629/uuid--identificador-unico-universal->

Capítulo 206: Vista

Sintaxis

1. // C objetivo
2. [UIView new] // Obtener un objeto de vista asignado e inicializado
3. [[UIView alloc] initWithFrame: (Pass CGRect)] // Obtenga la vista asignada e inicializada con un marco
4. [[UIView alloc] init] // Obtener un objeto de vista asignado e inicializado
5. // Swift
6. UIView () // Crea una instancia de UIView con el marco CGRect.zero
7. UIView (frame: CGRect) // Crea una instancia de UIView especificando el frame
8. UIView.addSubview (UIView) // agrega otra instancia de UIView como subview
9. UIView.hidden // Obtener o configurar la visibilidad de la vista
10. UIView.alpha // Obtener o configurar la opacidad de la vista
11. UIView.setNeedsLayout () // Obliga a la vista a actualizar su diseño

Observaciones

La clase **UIView** define un área rectangular en la pantalla y las interfaces para administrar el contenido en esa área. En tiempo de ejecución, un objeto de vista maneja la representación de cualquier contenido en su área y también maneja cualquier interacción con ese contenido.

Examples

Crear una vista UIV

C objetivo

```
CGRect myFrame = CGRectMake(0, 0, 320, 35)
UIView *view = [[UIView alloc] initWithFrame:myFrame];

//Alternative way of defining the frame
UIView *view = [[UIView alloc] init];
CGRect myFrame = view.frame;
myFrame.size.width = 320;
myFrame.size.height = 35;
myFrame.origin.x = 0;
```

```
myFrame.origin.y = 0;
view.frame = myFrame;
```

Rápido

```
let myFrame = CGRect(x: 0, y: 0, width: 320, height: 35)
let view = UIView(frame: myFrame)
```

Hacer la vista redondeada

Para hacer una `UIView` redondeada, especifique un `cornerRadius` para la `layer` la vista.

Esto también se aplica a cualquier clase que herede de `UIView`, como `UIImageView`.

Programáticamente

código SWIFT

```
someImageView.layoutIfNeeded()
someImageView.clipsToBounds = true
someImageView.layer.cornerRadius = 10
```

Código Objective-C

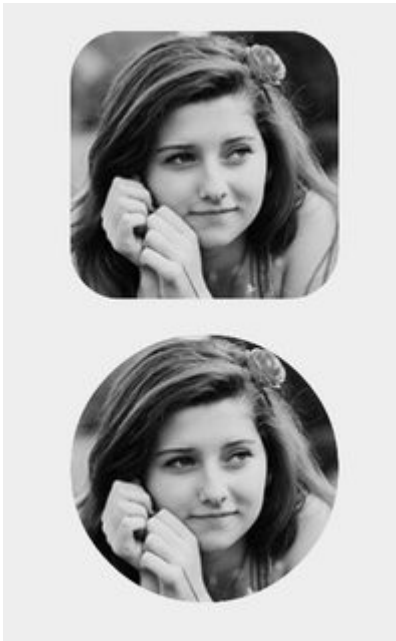
```
[someImageView layoutIfNeeded];
someImageView.clipsToBounds = YES;
someImageView.layer.cornerRadius = 10;
```

Ejemplo

```
//Swift code
topImageView.layoutIfNeeded()
bottomImageView.layoutIfNeeded()
topImageView.clipsToBounds = true
topImageView.layer.cornerRadius = 10
bottomImageView.clipsToBounds = true
bottomImageView.layer.cornerRadius = bottomImageView.frame.width / 2

//Objective-C code
[topImageView layoutIfNeeded]
[bottomImageView layoutIfNeeded];
topImageView.clipsToBounds = YES;
topImageView.layer.cornerRadius = 10;
bottomImageView.clipsToBounds = YES;
bottomImageView.cornerRadius = CGRectGetGetWidth(bottomImageView.frame) / 2;
```

Aquí está el resultado, mostrando el efecto de vista redondeada utilizando el radio de esquina especificado:



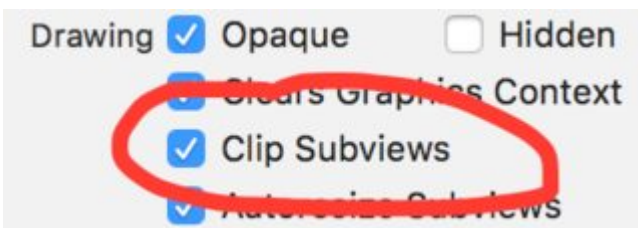
Nota

Para hacer esto necesitas incluir el framework QuartzCore.

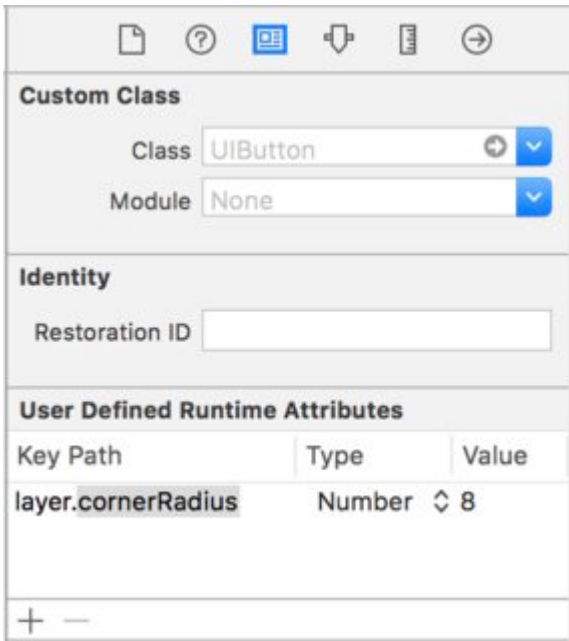
```
#import <QuartzCore/QuartzCore.h>
```

Configuración de Storyboard

También se puede lograr un efecto de vista redondeada *non-programmatically* mediante el establecimiento de las propiedades correspondientes en el **Guión gráfico** .



Como las propiedades de la *layer* no están expuestas en el Guión gráfico, debe modificar el atributo *cornerRadius* través de la sección Atributos de tiempo de ejecución definidos por el usuario.



Extensión rápida

Puede usar esta útil extensión para aplicar la vista redondeada siempre que tenga el mismo ancho y alto.

```
extension UIView {
    @discardableResult
    public func setAsCircle() -> Self {
        self.clipsToBounds = true
        let frameSize = self.frame.size
        self.layer.cornerRadius = min(frameSize.width, frameSize.height) / 2.0
        return self
    }
}
```

Para usarlo:

```
yourView.setAsCircle()
```

Tomando una instantánea

Puedes tomar una instantánea de una vista en `UIView` como esta:

Rápido

```
let snapshot = view.snapshotView(afterScreenUpdates: true)
```

C objetivo

```
UIView *snapshot = [view snapshotViewAfterScreenUpdates: YES];
```

Usando `IBInspectable` y `IBDesignable`

Una (o dos) de las nuevas características más `IBInspectable` en las últimas versiones de Xcode son las propiedades de `IBDesignable` y las `IBDesignable UIView`. Estos no tienen nada que ver con la funcionalidad de su aplicación, sino que afectan la experiencia del desarrollador en Xcode. El objetivo es poder inspeccionar visualmente las vistas personalizadas en su aplicación iOS sin ejecutarlas. Así que suponga que tiene una vista personalizada llamada `CustomView` que hereda de `UIView`. En esta vista personalizada, mostrará una cadena de texto con un color designado. También puede optar por no mostrar ningún texto. Necesitaremos tres propiedades:

```
var textColor: UIColor = UIColor.blackColor()
var text: String?
var showText: Bool = true
```

Entonces podemos anular la función `drawRect` en la clase:

```
if showText {
    if let text = text {
        let s = NSString(string: text)
        s.drawInRect(rect,
            withAttributes: [
                NSForegroundColorAttributeName: textColor,
                NSFontAttributeName: UIFont(name: "Helvetica Neue", size: 18)!
            ])
    }
}
```

Suponiendo que la propiedad de `text` esté establecida, esto dibujará una cadena en la esquina superior izquierda de la vista cuando se ejecute la aplicación. El problema es que no sabremos qué aspecto tendrá sin ejecutar la aplicación. Aquí es donde `IBInspectable` e `IBDesignable`. `IBInspectable` nos permite establecer visualmente los valores de propiedad de la vista en Xcode, al igual que con los controles incorporados. `IBDesignable` nos mostrará una vista previa visual en el guión gráfico. Aquí es cómo debe verse la clase:

```
@IBDesignable
class CustomView: UIView {
    @IBInspectable var textColor: UIColor = UIColor.blackColor()
    @IBInspectable var text: String?
    @IBInspectable var showText: Bool = true

    override func drawRect(rect: CGRect) {
        // ...
    }
}
```

O en el Objetivo C:

```
IB_DESIGNABLE
@interface CustomView: UIView

@property (nonatomic, strong) IBInspectable UIColor* textColor;
@property (nonatomic, strong) IBInspectable NSString* text;
@property (nonatomic, assign) IBInspectable BOOL showText;

@end
```

```

@implementation CustomView

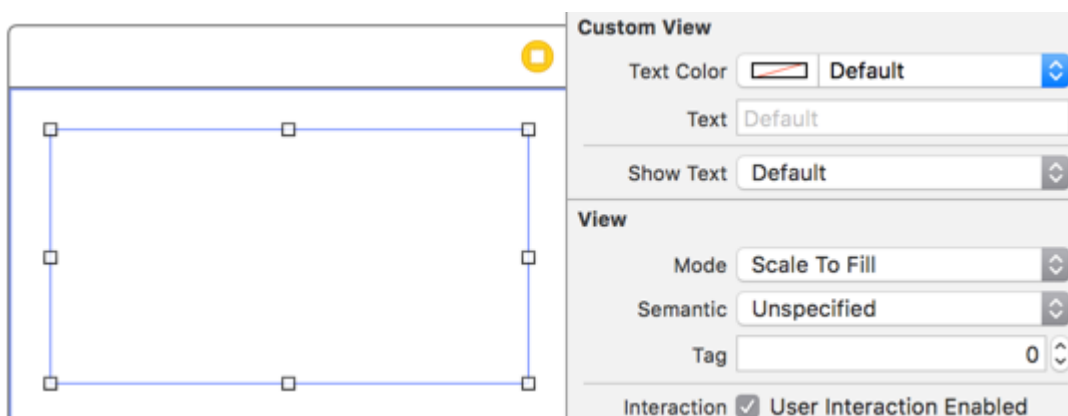
- (instancetype)init {
    if(self = [super init]) {
        self.textColor = [UIColor blackColor];
        self.showText = YES;
    }
    return self;
}

- (void)drawRect:(CGRect)rect {
    //...
}

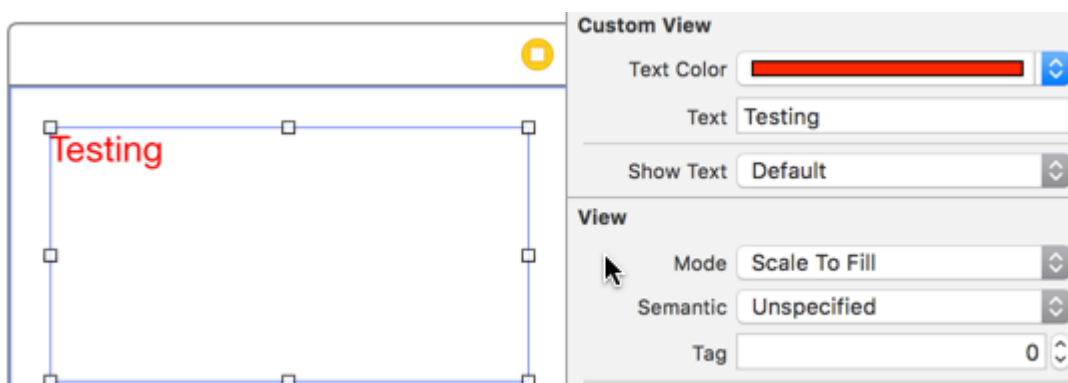
@end

```

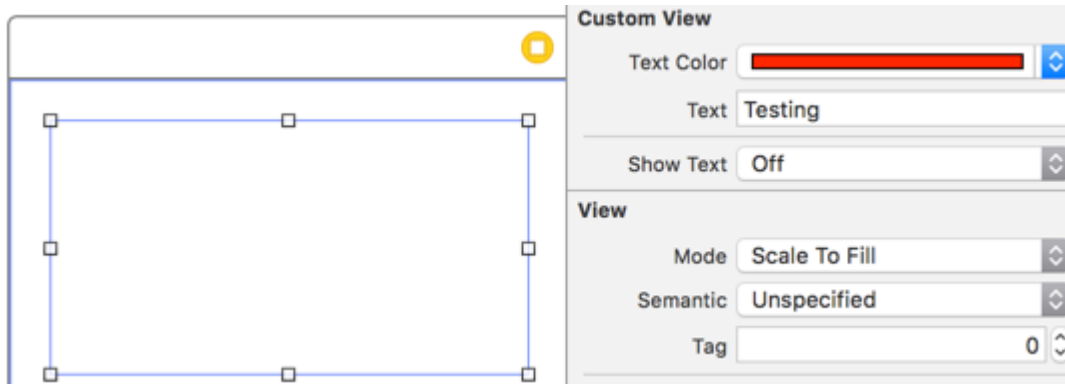
Las siguientes capturas de pantalla muestran lo que sucede en Xcode. El primero es lo que sucede después de agregar la clase revisada. Tenga en cuenta que hay tres nuevos elementos de interfaz de usuario para las tres propiedades. El *color del texto* mostrará un selector de color, el *texto* es solo un cuadro de entrada y *Mostrar texto* nos dará las opciones de *Off* y *On* que son `false` y `true` respectivamente.



Lo siguiente es después de cambiar el *Color* del *texto* a rojo usando el selector de color. Además, se ha proporcionado algo de texto para hacer que la función `drawRect` muestre. Observe que la vista en Interface Builder también se ha actualizado.

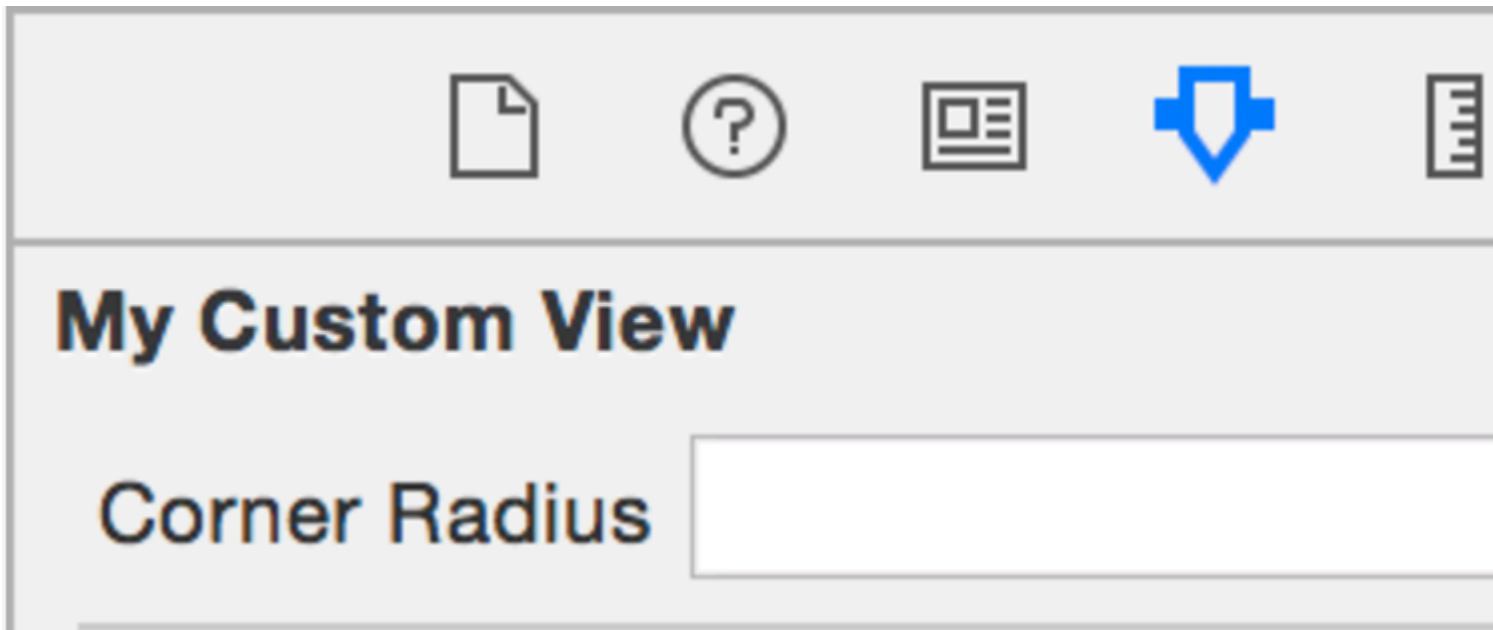


Por último, la configuración de *Mostrar texto* en *Off* en el inspector de propiedades hace que desaparezca la visualización de texto en Interface Builder.



Sin embargo, todos tenemos una situación en la que necesitamos crear `UIView` redondeado en múltiples vistas en su `Storyboard` lugar de declarar `IBDesignable` para cada vista de `Storyboard`, es mejor crear una `Extension` de `UIView` y obtener una interfaz de usuario creada solo para cada `UIView` a través del proyecto para crear una vista redondeada estableciendo el radio de la esquina. Un radio de borde configurable en cualquier `UIView` que cree en el guión gráfico.

```
extension UIView {
    @IBInspectable var cornerRadius:CGFloat {
        set {
            layer.cornerRadius = newValue
            clipsToBounds = newValue > 0
        }
        get {
            return layer.cornerRadius
        }
    }
}
```



Animando una vista

```
let view = UIView(frame: CGRect(x: 0, y: 0, width: 100, height: 100))
view.backgroundColor = UIColor.orange
```

```
self.view.addSubview(view)
UIView.animate(withDuration: 0.75, delay: 0.5, options: .curveEaseIn, animations: {
    //This will cause view to go from (0,0) to
    // (self.view.frame.origin.x,self.view.frame.origin.y)
    view.frame.origin.x = self.view.frame.origin.x
    view.frame.origin.y = self.view.frame.origin.y
}) { (finished) in
    view.backgroundColor = UIColor.blueColor()
}
```

Extensión UIView para atributos de tamaño y marco

Si queremos obtener la coordenada x de origen de la vista, debemos escribir como:

```
view.frame.origin.x
```

Para el ancho, tenemos que escribir:

```
view.frame.size.width
```

Pero si agregamos una extensión simple a una `UIView`, podemos obtener todos los atributos de manera muy simple, como:

```
view.x
view.y
view.width
view.height
```

También ayudará a establecer estos atributos como:

```
view.x = 10
view.y = 10
view.width = 100
view.height = 200
```

Y la simple extensión sería:

```
extension UIView {

    var x: CGFloat {
        get {
            return self.frame.origin.x
        }
        set {
            self.frame = CGRect(x: newValue, y: self.frame.origin.y, width:
self.frame.size.width, height: self.frame.size.height)
        }
    }

    var y: CGFloat {
        get {
            return self.frame.origin.y
        }
    }
}
```

```

        set {
            self.frame = CGRect(x: self.frame.origin.x, y: newValue, width:
self.frame.size.width, height: self.frame.size.height)
        }
    }

    var width: CGFloat {
        get {
            return self.frame.size.width
        }
        set {
            self.frame = CGRect(x: self.frame.origin.x, y: self.frame.origin.y, width:
newValue, height: self.frame.size.height)
        }
    }

    var height: CGFloat {
        get {
            return self.frame.height
        }
        set {
            self.frame = CGRect(x: self.frame.origin.x, y: self.frame.origin.y, width:
self.frame.size.width, height: newValue)
        }
    }
}

```

¡Necesitamos agregar este archivo de clase en un proyecto y estará disponible para usarlo durante todo el proyecto!

Administre mediante programación la inserción y eliminación de UIView en y desde otra UIView

Supongamos que tiene un `parentView` en el que desea insertar un nuevo `subView` programación (por ejemplo, cuando desea insertar un `UIImageView` en una vista de `UIViewController`), puede hacerlo de la siguiente manera.

C objetivo

```
[parentView addSubview:subView];
```

Rápido

```
parentView.addSubview(subView)
```

También puede agregar el `subView` debajo de otro `subView2`, que ya es una vista secundaria de `parentView` utilizando el siguiente código:

C objetivo

```
[parentView insertSubview:subView belowSubview:subView2];
```

Rápido

```
parentView.insertSubview(subView, belowSubview: subView2)
```

Si quieres insertarlo por encima de `subView2` puedes hacerlo de esta manera:

C objetivo

```
[parentView insertSubview:subView aboveSubview:subView2];
```

Rápido

```
parentView.insertSubview(subView, aboveSubview: subView2)
```

Si en algún lugar de su código necesita traer un cierto `subView` al frente, así que por encima de todos los `parentView` de `parentView`, puede hacerlo así:

C objetivo

```
[parentView bringSubviewToFront:subView];
```

Rápido

```
parentView.bringSubviewToFront(subView)
```

Finalmente, si desea eliminar `subView` de `parentView`, puede hacer lo siguiente:

C objetivo

```
[subView removeFromSuperview];
```

Rápido

```
subView.removeFromSuperview()
```

Crear UIView utilizando Autolayout

```
UIView *view = [[UIView alloc] init];

[self.view addSubview:view];

//Use the function if you want to use height as constraint
[self addSubview:view onParentView:self.view withHeight:200.f];

//Use this function if you want to add view with respect to parent and should resize with it
[self addFullResizeConstraintForSubview:view addedOnParentView:self.view];
```

Funciones

Función para agregar vista con altura fija usando restricciones de autolayout


```

-(void)addView:(UIView*)subView onParentView:(UIView*)parentView withHeight:(CGFloat)height {
    subView.translatesAutoresizingMaskIntoConstraints = NO;

    NSLayoutConstraint *trailing = [NSLayoutConstraint
                                    constraintWithItem:subView
                                    attribute:NSLayoutAttributeTrailing
                                    relatedBy:NSLayoutRelationEqual
                                    toItem:parent
                                    attribute:NSLayoutAttributeTrailing
                                    multiplier:1.0
                                    constant:10.f];

    NSLayoutConstraint *top = [NSLayoutConstraint
                                constraintWithItem:subView
                                attribute:NSLayoutAttributeTop
                                relatedBy:NSLayoutRelationEqual
                                toItem:parent
                                attribute:NSLayoutAttributeTop
                                multiplier:1.0
                                constant:10.f];

    NSLayoutConstraint *leading = [NSLayoutConstraint
                                    constraintWithItem:subView
                                    attribute:NSLayoutAttributeLeading
                                    relatedBy:NSLayoutRelationEqual
                                    toItem:parent
                                    attribute:NSLayoutAttributeLeading
                                    multiplier:1.0
                                    constant:10.f];

    [parent addConstraint:trailing];
    [parent addConstraint:top];
    [parent addConstraint:leading];

    NSLayoutConstraint *heightConstraint = [NSLayoutConstraint
                                              constraintWithItem:subView
                                              attribute:NSLayoutAttributeHeight
                                              relatedBy:NSLayoutRelationEqual
                                              toItem:nil
                                              attribute:0
                                              multiplier:0.0
                                              constant:height];

    [subView addConstraint:heightConstraint];
}

```

Función agregar restricción de tamaño completo para UIView creado.

```

-(void)addFullResizeConstraintForSubview:(UIView*)subView
addedOnParentView:(UIView*)parentView{

    subView.translatesAutoresizingMaskIntoConstraints = NO;

    NSLayoutConstraint *trailing = [NSLayoutConstraint
                                    constraintWithItem:subView
                                    attribute:NSLayoutAttributeTrailing
                                    relatedBy:NSLayoutRelationEqual
                                    toItem:parent

```

```

        attribute:NSLayoutAttributeTrailing
        multiplier:1.0
        constant:10.f];

NSLayoutConstraint *top = [NSLayoutConstraint
    constraintWithItem:subView
    attribute:NSLayoutAttributeTop
    relatedBy:NSLayoutRelationEqual
    toItem:parent
    attribute:NSLayoutAttributeTop
    multiplier:1.0
    constant:10.f];

NSLayoutConstraint *leading = [NSLayoutConstraint
    constraintWithItem:subView
    attribute:NSLayoutAttributeLeading
    relatedBy:NSLayoutRelationEqual
    toItem:parent
    attribute:NSLayoutAttributeLeading
    multiplier:1.0
    constant:10.f];

NSLayoutConstraint *bottom = [NSLayoutConstraint
    constraintWithItem:subView
    attribute:NSLayoutAttributeBottom
    relatedBy:NSLayoutRelationEqual
    toItem:parent
    attribute:NSLayoutAttributeBottom
    multiplier:1.0
    constant:0.f];

[parent addConstraint:trailing];
[parent addConstraint:top];
[parent addConstraint:leading];
[parent addConstraint:bottom];
}

```

Utilizando Tamaño de Contenido Intrínseco

Al crear una subclase `UIView`, el tamaño del contenido intrínseco ayuda a evitar la configuración de restricciones de altura y anchura codificadas

Un vistazo básico de cómo una clase puede utilizar esto.

```

class ImageView: UIView {
    var image: UIImage {
        didSet {
            invalidateIntrinsicContentSize()
        }
    }
    // omitting initializers
    // convenience init(image: UIImage)

    override func intrinsicContentSize() -> CGSize {
        return CGSize(width: image.size.width, height: image.size.height)
    }
}

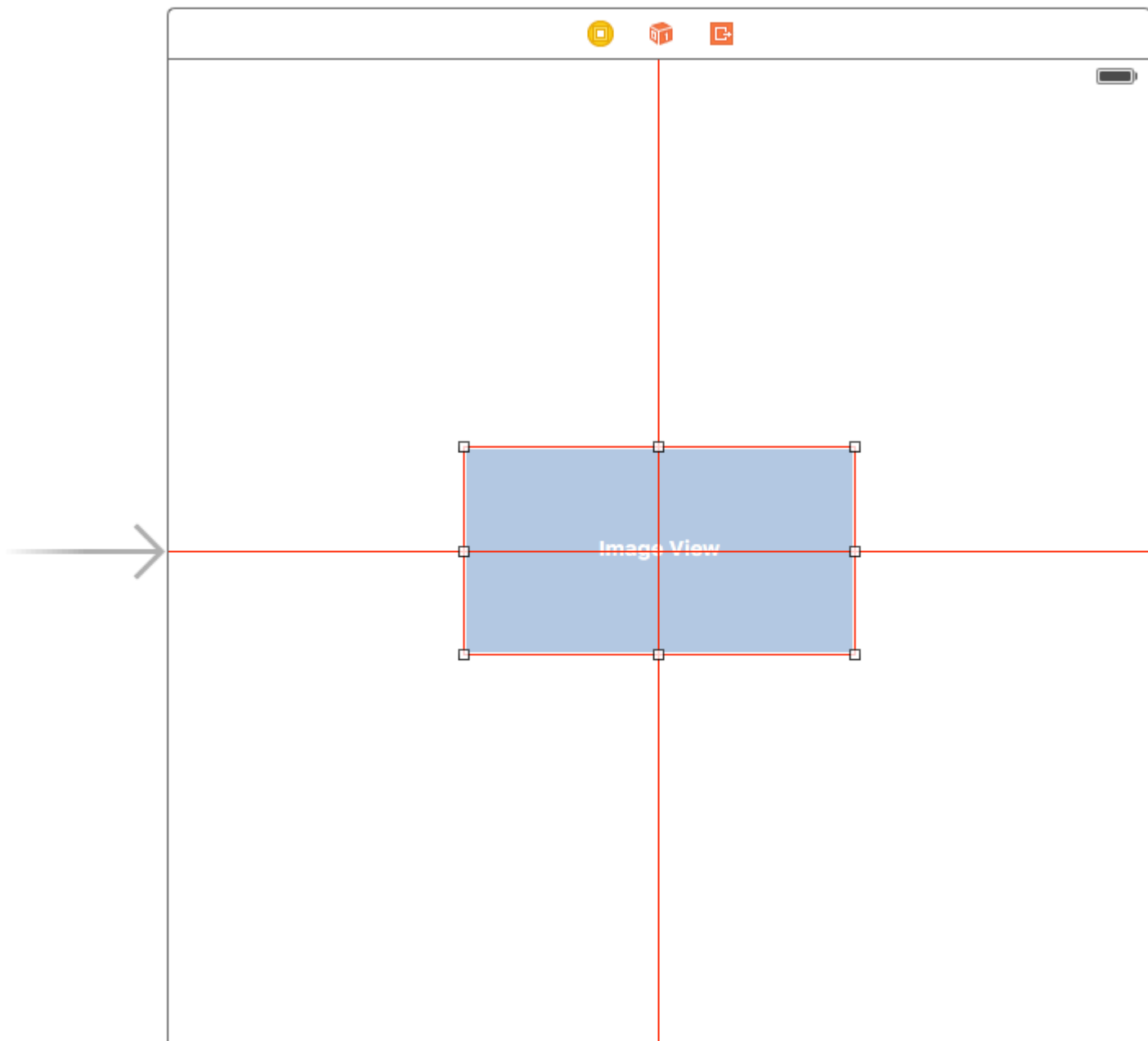
```

Si solo desea proporcionar un tamaño intrínsecamente, puede proporcionar el valor `UIViewNoIntrinsicMetric` para el valor que desea ignorar.

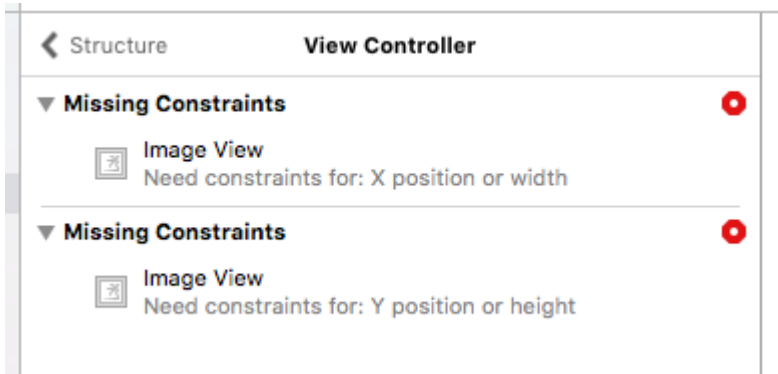
```
override func intrinsicContentSize() -> CGSize {
    return CGSize(width: UIViewNoIntrinsicMetric, height: image.size.width)
}
```

Beneficios cuando se utiliza con AutoLayout y Interface Builder

Uno podría tomar este `ImageView` (o `UIImageView`) y establecer la alineación horizontal para supervisar el centro X y la alineación vertical para supervisar el centro Y.

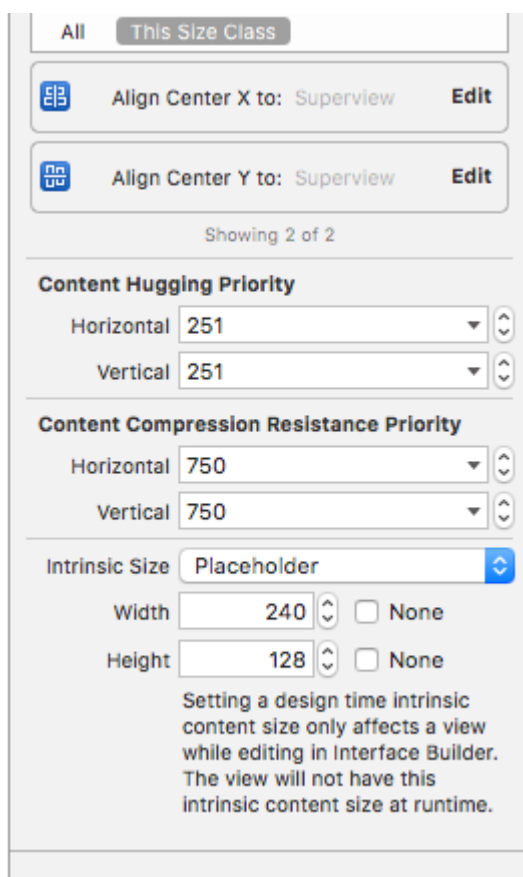


El constructor de la interfaz se quejará con usted en este punto dando la siguiente advertencia:



Aquí es donde entra en `Placeholder Intrinsic Size` de posición.

Al entrar en el panel del inspector de tamaño, y bajar a la lista desplegable Tamaño intrínseco, puede cambiar este valor de Predeterminado a Marcador de posición.



y ahora el constructor de interfaces eliminará las advertencias anteriores y puede usar este tamaño para tener vistas de tamaño dinámico dispuestas en el creador de interfaces.

Sacude una vista

```
extension UIView {
    func shake() {
        let animation = CAKeyframeAnimation(keyPath: "transform.translation.x")
        animation.timingFunction = CAMediaTimingFunction(name: kCAMediaTimingFunctionLinear)
        animation.duration = 0.6
        animation.values = [-10.0, 10.0, -7.0, 7.0, -5.0, 5.0, 0.0 ]
        layer.add(animation, forKey: "shake")
    }
}
```

```
}
```

Esta función se puede utilizar para llamar la atención sobre una vista específica agitándola un poco.

Lea Vista en línea: <https://riptutorial.com/es/ios/topic/858/vista>

Capítulo 207: WCSSessionDelegate

Introducción

WCSessionDelegate funciona con watch OS2 + utilizando WatchConnectivity. var watchSession: WCSession? func startWatchSession () {if (WCSession.isSupported ()) {watchSession = WCSession.default () watchSession!.delegate = self watchSession!.activate ()}} Implemente el método requerido: - didReceiveApplicationContext

Examples

Controlador del kit de reloj (WKInterfaceController)

```
import WatchConnectivity

var watchSession : WCSession?

override func awake(withContext context: Any?) {
    super.awake(withContext: context)
    // Configure interface objects here.
    startWatchSession()
}

func startWatchSession(){

    if(WCSession.isSupported()){
        watchSession = WCSession.default()
        watchSession!.delegate = self
        watchSession!.activate()
    }
}

//Callback in below delegate method when iOS app triggers event
func session(_ session: WCSession, didReceiveApplicationContext applicationContext: [String : Any]) {
    print("did ReceiveApplicationContext at watch")
}
```

Lea WCSSessionDelegate en línea: <https://riptutorial.com/es/ios/topic/8289/wcssessiondelegate>

Capítulo 208: WKWebView

Introducción

WKWebView es la pieza central de la moderna API WebKit introducida en iOS 8 y OS X Yosemite. Reemplaza UIWebView en UIKit y WebView en AppKit, ofreciendo una API consistente en las dos plataformas.

WKWebView es uno de los anuncios más significativos de WWDC 2014 con un desplazamiento de 60 fps, gestos integrados, comunicación optimizada entre la aplicación y la página web, y el mismo motor de JavaScript que Safari.

Examples

Creando un WebBrowser simple

```
import UIKit
import WebKit

class ViewController: UIViewController, UISearchBarDelegate, WKNavigationDelegate, WKUIDelegate {

    var searchBar: UISearchBar! //All web-browsers have a search-bar.
    var webView: WKWebView! //The WKWebView we'll use.
    var toolbar: UIToolbar! //Toolbar at the bottom just like in Safari.
    var activityIndicator: UIActivityIndicatorView! //Activity indicator to let the user know the page is loading.

    override func viewDidLoad() {
        super.viewDidLoad()

        self.initControls()
        self.setTheme()
        self.doLayout()
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
    }

    func initControls() {
        self.searchbar = UISearchBar()

        //WKUserController allows us to add Javascript scripts to our webView that will run either at the beginning of a page load OR at the end of a page load.

        let configuration = WKWebViewConfiguration()
        let contentController = WKUserController()
        configuration.userContentController = contentController

        //create the webView with the custom configuration.
```



```

self.webView = WKWebView(frame: .zero, configuration: configuration)

self.toolbar = UIToolbar()
self.layoutToolBar()

self.activityIndicator = UIActivityIndicatorView(activityIndicatorStyle: .gray)
self.activityIndicator.hidesWhenStopped = true
}

func setTheme() {
    self.edgesForExtendedLayout = UIRectEdge(rawValue: 0)
    self.navigationController?.navigationBar.barTintColor = UIColor.white()

    //Theme the keyboard and searchBar. Setup delegates.
    self.searchbar.delegate = self
    self.searchbar.returnKeyType = .go
    self.searchbar.searchBarStyle = .prominent
    self.searchbar.placeholder = "Search or enter website name"
    self.searchbar.autocapitalizationType = .none
    self.searchbar.autocorrectionType = .no

    //Set the WebView's delegate.
    self.webView.navigationDelegate = self //Delegate that handles page navigation
    self.webView.uiDelegate = self //Delegate that handles new tabs, windows, popups,
layout, etc..

    self.activityIndicator.transform = CGAffineTransform(scaleX: 1.5, y: 1.5)
}

func layoutToolBar() {
    //Browsers typically have a back button, forward button, refresh button, and
newTab/newWindow button.

    var items = Array<UIBarButtonItem>()

    let space = UIBarButtonItem(barButtonItemSystemItem: .flexibleSpace, target: nil, action:
nil)

    items.append(UIBarButtonItem(title: "<", style: .plain, target: self, action:
#selector(onBackButtonPressed)))
    items.append(space)
    items.append(UIBarButtonItem(title: ">", style: .plain, target: self, action:
#selector(onForwardButtonPressed)))
    items.append(space)
    items.append(UIBarButtonItem(barButtonItemSystemItem: .refresh, target: self, action:
#selector(onRefreshPressed)))
    items.append(space)
    items.append(UIBarButtonItem(barButtonItemSystemItem: .organize, target: self, action:
#selector(onTabPressed)))

    self.toolbar.items = items
}

func doLayout() {
    //Add the searchBar to the navigationBar.
    self.navigationItem.titleView = self.searchbar

    //Add all other subViews to self.view.
    self.view.addSubview(self.webView)
    self.view.addSubview(self.toolbar)
    self.view.addSubview(self.activityIndicator)
}

```

```

//Setup which views will be constrained.

let views: [String: AnyObject] = ["webView": self.webView, "toolbar": self.toolbar,
"activityIndicator": self.activityIndicator];
var constraints = Array<String>();

constraints.append("H:|-0-[webView]-0-|")
constraints.append("H:|-0-[toolbar]-0-|")
constraints.append("V:|-0-[webView]-0-[toolbar(50)]-0-|")

//constrain the subviews using the above visual constraints.

for constraint in constraints {
    self.view.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutConstraintOptions(rawValue: 0), metrics: nil, views: views))
}

for view in self.view.subviews {
    view.translatesAutoresizingMaskIntoConstraints = false
}

//constraint the activity indicator to the center of the view.
self.view.addConstraint(NSLayoutConstraint(item: self.activityIndicator, attribute:
.centerX, relatedBy: .equal, toItem: self.view, attribute: .centerX, multiplier: 1.0,
constant: 0.0))
self.view.addConstraint(NSLayoutConstraint(item: self.activityIndicator, attribute:
.centerY, relatedBy: .equal, toItem: self.view, attribute: .centerY, multiplier: 1.0,
constant: 0.0))
}

//Searchbar Delegates

func searchBarSearchButtonClicked(_ searchBar: UISearchBar) {
    self.searchbar.resignFirstResponder()

    if let searchText = self.searchbar.text, url = URL(string: searchText) {
        //Get the URL from the search bar. Create a new NSURLRequest with it and tell the
webView to navigate to that URL/Page. Also specify a timeout for if the page takes too long.
Also handles cookie/caching policy.

        let request = URLRequest(url: url, cachePolicy: .useProtocolCachePolicy,
timeoutInterval: 30)
        self.webView.load(request)
    }
}

//Toolbar Delegates

func onBackButtonPressed(button: UIBarButtonItem) {
    if (self.webView.canGoBack) { //allow the user to go back to the previous page.
        self.webView.goBack()
    }
}

func onForwardButtonPressed(button: UIBarButtonItem) {
    if (self.webView.canGoForward) { //allow the user to go forward to the next page.
        self.webView.goForward()
    }
}

```

```

    }
}

func onRefreshPressed(button: UIBarButtonItem) {
    self.webView.reload() //reload the current page.
}

func onTabPressed(button: UIBarButtonItem) {
    //TODO: Open a new tab or web-page.
}

//WebView Delegates

func webView(_ webView: WKWebView, decidePolicyFor navigationAction: WKNavigationAction,
decisionHandler: (WKNavigationActionPolicy) -> Void) {

    decisionHandler(.allow) //allow the user to navigate to the requested page.
}

func webView(_ webView: WKWebView, decidePolicyFor navigationResponse:
WKNavigationResponse, decisionHandler: (WKNavigationResponsePolicy) -> Void) {

    decisionHandler(.allow) //allow the webView to process the response.
}

func webView(_ webView: WKWebView, didStartProvisionalNavigation navigation:
WKNavigation!) {
    self.activityIndicator.startAnimating()
}

func webView(_ webView: WKWebView, didFailProvisionalNavigation navigation: WKNavigation!,
withError error: NSError) {
    self.activityIndicator.stopAnimating()

    //Handle the error. Display an alert to the user telling them what happened.

    let alert = UIAlertController(title: "Error", message: error.localizedDescription,
preferredStyle: .alert)
    let action = UIAlertAction(title: "OK", style: .default) { (action) in
        alert.dismiss(animated: true, completion: nil)
    }
    alert.addAction(action)
    self.present(alert, animated: true, completion: nil)
}

func webView(_ webView: WKWebView, didFinish navigation: WKNavigation!) {
    self.activityIndicator.stopAnimating()

    //Update our search bar with the webPage's final endpoint-URL.
    if let url = self.webView.url {
        self.searchbar.text = url.absoluteString ?? self.searchbar.text
    }
}

func webView(_ webView: WKWebView, didReceiveServerRedirectForProvisionalNavigation
navigation: WKNavigation!) {
    //When the webview receives a "Redirect" to a different page or endpoint, this is
called.
}

```

```

func webView(_ webView: WKWebView, didCommit navigation: WKNavigation!) {
    //When the content for the webpage starts arriving, this is called.
}

func webView(_ webView: WKWebView, didFail navigation: WKNavigation!, withError error:
NSError) {

}

func webView(_ webView: WKWebView, didReceive challenge: URLAuthenticationChallenge,
completionHandler: (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {

    completionHandler(.performDefaultHandling, .none) //Handle SSL connections by default.
    We aren't doing SSL pinning or custom certificate handling.

}

//WebView's UINavigationController Delegates

//This is called when a webView or existing loaded page wants to open a new window/tab.
func webView(_ webView: WKWebView, createWebViewWith configuration:
WKWebViewConfiguration, for navigationAction: WKNavigationAction, windowFeatures:
WKWindowFeatures) -> WKWebView? {

    //The view that represents the new tab/window. This view will have an X button at the
    top left corner + a webView.
    let container = UIView()

    //New tabs need an exit button.
    let XButton = UIButton()
    XButton.addTarget(self, action: #selector(onWebViewExit), for: .touchUpInside)
    XButton.layer.cornerRadius = 22.0

    //Create the new webView window.
    let webView = WKWebView(frame: .zero, configuration: configuration)
    webView.navigationDelegate = self
    webView.uiDelegate = self

    //Layout the tab.
    container.addSubview(XButton)
    container.addSubview(webView)

    let views: [String: AnyObject] = ["XButton": XButton, "webView": webView];
    var constraints = Array<String>()

    constraints.append("H:|-(22)-[XButton(44)]")
    constraints.append("H:|-0-[webView]-0-|")
    constraints.append("V:|-(22)-[XButton(44)]-0-[webView]-0-|")

    //constrain the subviews.
    for constraint in constraints {
        container.addConstraints(NSLayoutConstraint.constraints(withVisualFormat:
constraint, options: NSLayoutFormatOptions(rawValue: 0), metrics: nil, views: views))
    }

    for view in container.subviews {
        view.translatesAutoresizingMaskIntoConstraints = false
    }
}

```

```
        //TODO: Add the containerView to self.view or present it with a new controller. Keep
track of tabs..

        return webView
    }

    func onWebViewExit(button: UIButton) {
        //TODO: Destroy the tab. Remove the new tab from the current window or controller.
    }
}
```

Mostrando el botón `GO` personalizado en el teclado:

 <https://stackoverflow.com/>



All Questions 

Show [Interesting](#)

0 

0 

Mysql error, "Specified key was too long; max key length is 767 bytes" need workaround

mysql

6 secs ago [rerat](#)

0 

0 

Error Code: 1305. FUNCTION or PROCEDURE does not exist

q w e r t y u i o p

a s d f g h j k l

 z x c v b n m 

.AtDocumentEnd

Envía mensajes desde JavaScript y los maneja desde el lado nativo.

Los mensajes pueden enviarse desde JavaScript utilizando el siguiente código

```
window.webkit.messageHandlers.{NAME}.postMessage()
```

Aquí se explica cómo crear un controlador de mensajes de script para manejar los mensajes:

```
class NotificationScriptMessageHandler: NSObject, WKScriptMessageHandler {
    func userContentController(userContentController: WKUserContentController,
didReceiveScriptMessage message: WKScriptMessage!) {
        if message.name == "{NAME}" {
            // to be sure of handling the correct message
            print(message.body)
        }
    }
}
```

Aquí se explica cómo configurar el controlador de mensajes de script en WKWebView:

```
let configuration = WKWebViewConfiguration()
let userContentController = WKUserContentController()
let handler = NotificationScriptMessageHandler()
userContentController.addScriptMessageHandler(handler, name: "{NAME}")
configuration.userContentController = userContentController
let webView = WKWebView(frame: self.view.bounds, configuration: configuration)
```

NOTA: agregar el mismo controlador "{NAME}" con `addScriptMessageHandler(name: más` de una vez, da como resultado la excepción `NSInvalidArgumentException` .

Lea WKWebView en línea: <https://riptutorial.com/es/ios/topic/3602/wkwebview>

Capítulo 209: Xcode Build & Archive desde la línea de comandos

Sintaxis

- `xcodebuild` `[-project name.xcodeproj]` `-scheme schemename` `[[-destination destinationspecifier] ...]` `[-destination-timeout value]` `[-configuration configurationname]` `[-sdk [sdkfullpath | sdkname]]` `[action ...]` `[buildsetting=value ...]` `[-userdefault=value ...]`

Parámetros

Opción	Descripción
<code>-proyecto</code>	Construye el nombre del proyecto.xcodeproj.
<code>-esquema</code>	Se requiere si se construye un espacio de trabajo.
<code>-destino</code>	Usa el dispositivo de destino
<code>-configuración</code>	Usa la configuración de compilación
<code>-sdk</code>	SDK especificado

Observaciones

Ejecute `xcodebuild` desde el directorio que contiene su proyecto para construir un proyecto de Xcode. Para construir un espacio de trabajo de Xcode, debe pasar las opciones **-workspace** y **-scheme** para definir la compilación. Los parámetros del esquema controlarán qué objetivos se crean y cómo se crean, aunque puede pasar otras opciones a `xcodebuild` para anular algunos parámetros del esquema.

Examples

Construir y archivar

Construir:

```
xcodebuild -exportArchive -exportFormat ipa \  
-archivePath "/Users/username/Desktop/MyiOSApp.xcarchive" \  
-exportPath "/Users/username/Desktop/MyiOSApp.ipa" \  
-exportProvisioningProfile "MyCompany Distribution Profile"
```

Archivo:


```
xcodebuild -project <ProjectName.xcodeproj>
  -scheme <ProjectName>
  -sdk iphonesimulator
  -configuration Debug
  -destination "platform=iOS Simulator,name=<Device>,OS=9.3"
clean build
```

Lea Xcode Build & Archive desde la línea de comandos en línea:

<https://riptutorial.com/es/ios/topic/5027/xcode-build--amp--archive-desde-la-linea-de-comandos>

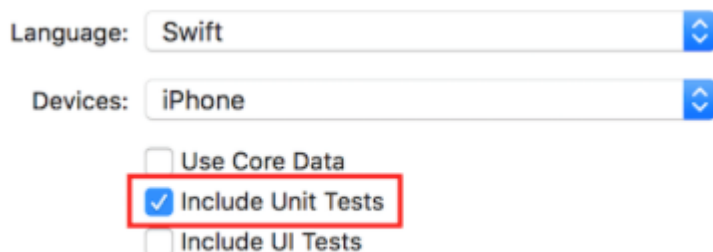
Capítulo 210: XCTest framework - Unit Testing

Examples

Agregando archivos de prueba a Xcode Project

Al crear el proyecto.

Debe marcar "Incluir pruebas unitarias" en el cuadro de diálogo de creación de proyecto.



Después de crear el proyecto.

Si no revisó ese elemento mientras creaba su proyecto, siempre podría agregar archivos de prueba más tarde. Para hacerlo:

- 1- Ve a la configuración de tu proyecto en Xcode
- 2- Ir a "Objetivos"
- 3- Haga clic en "Agregar objetivo"
- 4- En "Otro", seleccione "Paquete de prueba de prueba de unidad de cacao táctil"

Al final, debe tener un archivo llamado `[Your app name]Tests.swift` . En Objective-C, debe tener dos archivos llamados `[Your app name]Tests.h` `[Your app name]Tests.m` en `[Your app name]Tests.m` lugar.

`[Your app name]Tests.swift` or `.m` archivo `[Your app name]Tests.swift` or `.m` incluirá de forma predeterminada:

- Una importación del módulo `XCTest`
- Una clase de `[Your app name]Tests` que amplía `XCTestCase`
- `setUp` , `tearDown` , `testExample` , `testPerformanceExample`

Rápido

```
import XCTest

class MyProjectTests: XCTestCase {

    override func setUp() {
        super.setUp()
        // Put setup code here. This method is called before the invocation of each test method in
        the class.
    }

    override func tearDown() {
        // Put teardown code here. This method is called after the invocation of each test method
        in the class.
        super.tearDown()
    }

    func testExample() {
        // This is an example of a functional test case.
        // Use XCTAssert and related functions to verify your tests produce the correct results.
    }

    func testPerformanceExample() {
        // This is an example of a performance test case.
        self.measure {
            // Put the code you want to measure the time of here.
        }
    }

}
```

C objetivo

```
#import <XCTest/XCTest.h>

@interface MyProjectTests : XCTestCase

@end

@implementation MyProjectTests

- (void)setUp {
    [super setUp];
    // Put setup code here. This method is called before the invocation of each test method in the
    class.
}

- (void)tearDown {
    // Put teardown code here. This method is called after the invocation of each test method in
    the class.
    [super tearDown];
}

- (void)testExample {
    // This is an example of a functional test case.
```

```
// Use XCTAssert and related functions to verify your tests produce the correct results.
}

- (void)testPerformanceExample {
// This is an example of a performance test case.
    [self measureBlock:^(
        // Put the code you want to measure the time of here.
        )];
}

@end
```

Agregando Storyboard y View Controller como instancias al archivo de prueba

Para comenzar con las pruebas unitarias, que se realizarán en el archivo de pruebas y se probarán el Controlador de vista y el Guión gráfico, debemos introducir estos dos archivos en el archivo de prueba.

Definiendo el controlador de vista

Rápido

```
var viewController : ViewController!
```

Presentando el Storyboard e inicializando el controlador de vista

Agregue este código al método `setUp()` :

Rápido

```
let storyboard = UIStoryboard(name: "Main", bundle: nil)
viewController = storyboard.instantiateInitialViewController() as! ViewController
```

C objetivo

```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:"Main" bundle:nil];
viewController = (ViewController *) [storyboard instantiateInitialViewController];
```

De esta manera, podría escribir métodos de prueba, y ellos sabrán dónde verificar errores. En este caso, hay View Controller y el Storyboard.

Añadiendo métodos de prueba.

Según Apple:

Métodos de prueba

Un método de prueba es un método de instancia de una clase de prueba que comienza con la prueba de prefijo, no toma parámetros y devuelve vacío, por ejemplo, (void) testColorIsRed (). Un método de prueba ejerce código en su proyecto y, si ese código no produce el resultado esperado, informa fallas usando un conjunto de API de aserción. Por ejemplo, el valor de retorno de una función podría compararse con un valor esperado o su prueba podría afirmar que el uso incorrecto de un método en una de sus clases produce una excepción.

Así que agregamos un método de prueba usando "prueba" como el prefijo del método, como:

Rápido

```
func testSomething() {  
}
```

C objetivo

```
- (void)testSomething {  
}
```

Para probar realmente los resultados, usamos el método `XCTAssert()`, que toma una expresión booleana, y si es verdadero, marca la prueba como exitosa, de lo contrario, la marcará como fallida.

Digamos que tenemos un método en la clase View Controller llamado `sum()` que calcula la suma de dos números. Para probarlo, utilizamos este método:

Rápido

```
func testSum(){  
    let result = viewController.sum(4, and: 5)  
    XCTAssertEqual(result, 9)  
}
```

C objetivo

```
- (void)testSum {
    int result = [viewController sum:4 and:5];
    XCTAssertEqual(result, 9);
}
```

Nota

De forma predeterminada, no puede acceder a la etiqueta, el cuadro de texto u otros elementos de la interfaz de usuario de la clase View Controller desde la clase de prueba si se crean por primera vez en un archivo de Storyboard. Esto se debe a que se inicializan en el método `loadView()` de la clase View Controller, y no se llamará al realizar la prueba. La mejor manera de llamar a `loadView()` y todos los demás métodos requeridos es acceder a la propiedad de `view` de nuestra propiedad `viewController`. Debe agregar esta línea antes de probar los elementos de la interfaz de usuario:

```
XCTAssertNotNil(viewController.view)
```

Empezar a probar

Probando un método específico

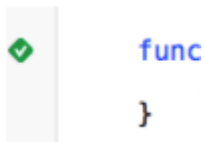
Para probar un método específico, haga clic en el cuadrado junto a la definición del método.

Probando todos los métodos

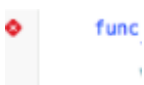
Para probar todos los métodos, haga clic en el cuadrado junto a la definición de clase.

Ver el resultado de la prueba.

Si hay una marca verde al lado de la definición, la prueba ha tenido éxito.



Si hay una cruz roja junto a la definición, la prueba ha fallado.



Ejecutando todas las pruebas

```
Product -> Test OR Cmd + U
```

¡Se ejecutarán todas las pruebas de todos los objetivos de prueba!

Importar un módulo que pueda ser probado.

Las clases, estructuras, enumeraciones y todos sus métodos son `internal` por defecto. Esto significa que solo se puede acceder desde el mismo módulo. Los casos de prueba están en un objetivo diferente y esto significa que están en un módulo diferente. Para poder acceder al método que desea probar, necesita importar el módulo a probar usando la palabra clave `@testable`.

Digamos que tenemos un módulo principal llamado `ToDo` y queremos escribir pruebas para él. Importaríamos ese módulo así:

```
@testable import ToDo
```

Todos los métodos de prueba en el archivo con esta declaración de importación ahora pueden acceder a todas `internal` clases `internal`, estructuras, enumeraciones y todos sus métodos `internal` del módulo de `ToDo`.

Nunca debe agregar los archivos con los elementos que desea probar al objetivo de la prueba, ya que esto puede llevar a errores difíciles de depurar.

Vista de gatillo de carga y apariencia.

Ver cargando

En una prueba para un controlador de vista, a veces desea activar la ejecución de `loadView()` o `viewDidLoad()`. Esto se puede hacer accediendo a la vista. Digamos que tiene una instancia de controlador de vista en su prueba llamada `sut` (sistema bajo prueba), entonces el código se vería así:

```
XCTAssertNotNil(sut.view)
```

Ver apariencia

También puede activar los métodos `viewWillAppear(_:)` y `viewDidAppear(_:)` agregando el siguiente código:

```
sut.beginAppearanceTransition(true, animated: true)
sut.endAppearanceTransition()
```

Escribiendo una clase de prueba

```
import XCTest
@testable import PersonApp

class PersonTests: XCTestCase {
    func test_completeName() {
        let person = Person(firstName: "Josh", lastName: "Brown")
        XCTAssertEqual(person.completeName(), "Josh Brown")
    }
}
```

Ahora vamos a discutir lo que está pasando aquí. La línea de `import XCTest` nos permitirá extender `XCTestCase` y usar `XCTAssertEqual` (entre otras afirmaciones). Extender `XCTestCase` y prefijar nuestro nombre de `test` con `test` garantizará que Xcode ejecute automáticamente esta prueba cuando ejecute las pruebas en el proyecto (**U** o **Producto** > **Prueba**). La línea `@testable import PersonApp` importará nuestro destino de `PersonApp` para que podamos probar y usar clases desde él, como la `Person` en nuestro ejemplo anterior. Y finalmente, nuestro `XCTAssertEqual` se asegurará de que `person.completeName()` sea igual a la cadena "Josh Brown" .

Lea **XCTest framework - Unit Testing** en línea: <https://riptutorial.com/es/ios/topic/5075/xctest-framework---unit-testing>

Creditos

S. No	Capítulos	Contributors
1	Empezando con iOS	Ali Beadle , Allan Burleson , Anand Nimje , Anatoliy , Ashutosh Dave , Bhadresh Kathiriya , bjtitus , Blachshma , bmike , Charlie H , Cin316 , Community , Dair , dan , deyanm , Efraim Weiss , Erik Godard , FelixSFD , Fogmeister , Hudson Taylor , Irfan , J F , Jack Ngai , James , Josh Brown , jrf , Kampai , Kevin , Losiowaty , M. Galban , Maddyツヅ , Matthew Cawley , Md. Ibrahim Hassan , Midhun MP , Miguel Cabezas , Muhammad Zohaib Ehsan , Pro Q , PSN , RamenChef , Sam Fischer , Seyyed Parsa Neshaei , shim , Skeleton Bow , Stephen Leppik , Steve Moser , Suragch , SuzGupta , The_Curry_Man , ThrowingSpoon , Undo , user3480295 , user6939352 , Vignan
2	Abrazar contenido / Compresión de contenido en Autolayout	Mehul Chuahan
3	Accesibilidad	Harshal Bhavsar , Justin , Ruby , Stephen Leppik , Zev Eisenberg
4	Actualizando dinámicamente un UIStackView	Harshal Bhavsar , Rahul , Stephen Leppik
5	Alamofire	Alex Koshy , Josh Caswell , Sour LeangChhean , yogesh wadhwa
6	AÑADIR A UN LÍDER DE SWIFT BRIDGING	yogesh wadhwa
7	Aparición de la UIA	azimov , Harshal Bhavsar , Stephen Leppik , Undo
8	API de Google Places para iOS	Cyril Ivar Garcia , Vignan
9	API de reconocimiento de voz de iOS 10	rohit90 , Stephen Leppik
10	Aplicación de Seguridad de Transporte (ATS)	breakingobstacles , D4ttatraya , esthepiking , FelixSFD , Mehul Chuahan , nathan
11	Aplicación en la compra	Cyril Ivar Garcia , Martin , rigdonmr , WMios

12	AppDelegate	CodeChanger , Oleh Zayats , Saumil Shah
13	ARC (Conteo Automático de Referencia)	4444 , Irfan , John Militer , Ketan P , Tricertops
14	Archivo de texto básico de E / S	Idan
15	Arquitectura MVP	Oleh Zayats
16	atribuidoTexto en UILabel	vp2698
17	AVPlayer y AVPlayerViewController	Bonnie , Chirag Desai , Gazi Alankus , Harshal Bhavsar , Konda Yadav , Stephen Leppik
18	AWS SDK	OhadM
19	Barra de navegación	Md. Ibrahim Hassan , Mehul Chuahan
20	Bloquear	4444 , animuson , Joshua , Mehul Chuahan , Ruby , Tamarous , user459460
21	Bloques de cadena en una cola (con MKBlockQueue)	StackUnderflow
22	CAAnimación	Bhavin Ramani , James P , Mr. Xcoder , Narendra Pandey , Rahul , Rob , Undo
23	CAGradientLayer	Bhavin Ramani , Harshal Bhavsar , Sam Fischer , Stephen Leppik , Undo
24	CALayer	Alistra , Dunja Lalic , HariKrishnan.P , Harshal Bhavsar , ignotusverum , iOS BadBoy , Kamil Harasimowicz , Luiz Henrique Guimaraes , Stephen Leppik , Suragch , Viktor Simkó , william205
25	Calificación de solicitud / solicitud de revisión	Abhijit
26	Cambiar color de barra de estado	Alex Rouse , danshevluk , Harshal Bhavsar , Mr. Xcoder , shim , Stephen Leppik , Steve Moser , william205 , WMios
27	Cambiar el tamaño de UIImage	Rahul
28	Cargar imagenes async	J.Paravicini
29	Carril rápido	J F , KrauseFx , SM18 , tharkay
30	CAShapeLayer	Filip Radelic , HariKrishnan.P , Harshal Bhavsar , Narendra Pandey , Stephen Leppik

31	Categorías	Faran Ghani , simple_code
32	Clases de tamaño y adaptabilidad	Tim
33	CLLocation	amar , Duly Kinsky , FelixSFD , Siddharth Sunil , Sujania , That lazy iOS Guy , void , Zee
34	CloudKit	Seyyed Parsa Neshaei
35	Codificable	Ashish Kakkad
36	Codificación del valor clave-Observación del valor clave	D4ttatraya , Harshal Bhavsar , Mehul Chuahan , Mihriban Minaz , Mithrandir , Muhammad Zohaib Ehsan , Pärserk , sanman
37	Comprobando la conectividad de la red	ajmccall , breakingobstacles , Mick MacCallum , pableiros , sushant jagtap
38	Comprobando la versión de iOS	Bhavin Ramani , byJeevan , James P , Joshua , njuri , Samuel Teferra , Sandy
39	Concurrencia	Doc , Fonix , Juan Campa , Kevin DiTraglia , Tien
40	Configuración de iOS de Cartago	Md. Ibrahim Hassan
41	Configurar balizas con CoreBluetooth	Beto Caldas
42	Control UISegmentado	Kamil Harasimowicz
43	Convertir HTML a cadena NSAttributedString y viceversa	Md. Ibrahim Hassan
44	Convertir NSAttributedString a UIImage	Md. Ibrahim Hassan
45	Core Graphics	Dunja Lalic , Josh Caswell , Seyyed Parsa Neshaei , Sunil Sharma , Unheilig
46	Core Motion	Md. Ibrahim Hassan , RamenChef
47	Core Spotlight en iOS	Md. Ibrahim Hassan
48	Cortar un UIImage en un círculo	Md. Ibrahim Hassan
49	Creación de PDF en iOS	Mansi Panchal , Narendra Pandey

50	Creación de una ID de aplicación	yogesh wadhwa
51	Crear un marco personalizado en iOS	Saeed-rz
52	Crear un video a partir de imágenes.	Tiko
53	Cree un archivo .ipa para cargar en la tienda de aplicaciones con Applicationloader	Anuj Joshi
54	CTCallCenter	MANI , Md. Ibrahim Hassan , OhadM
55	CydiaSubstrate tweak	gkpln3
56	Datos básicos	Ankit chauhan , Md. Ibrahim Hassan
57	Delegados de multidifusión	Rahul
58	Depuración se bloquea	NobodyNada
59	Detección de rostro utilizando CoreImage / OpenCV	Md. Ibrahim Hassan
60	Diseño automático	alaphao , amar , Anuj Joshi , Bean , Bhumit Mehta , BlackDeveraux , dasdom , Dennis , Dima Deplov , Dinesh Raja , Đông An , Harshal Bhavsar , Hasintha Janka , Irfan , Jano , juanjo , keithbhunter , Mahesh , Mert Buran , Mr. Xcoder , NSNoob , ozgur , Pärserk , Rajesh , Sally , Sandy , Stephen Leppik , Suragch , Undo , user3480295 , Vignan
61	Enlace profundo en iOS	bryanjclark , Dunja Lalic , FelixSFD , sanman
62	Enlaces universales	Harshal Bhavsar , Irfan , satheeshwaran , Stephen Leppik , Vineet Choudhary
63	Entrega por paracaídas	FelixSFD , Md. Ibrahim Hassan
64	Escáner de códigos QR	Bluewings , Efraim Weiss
65	Establecer fondo de vista	Adriana Carelli , Andreas , Bhadresh Kathiriya , Harshal Bhavsar , Md. Ibrahim Hassan , user459460
66	EventKit	Seyyed Parsa Neshaei
67	Extensión para	Oleh Zayats

	notificaciones push enriquecidas - iOS 10.	
68	FacebookSDK	Brian , Harshal Bhavsar , Irfan , Mehul Chuahan , OhadM , Ravi Prakash Verma , Stephen Leppik
69	FileHandle	Nikhlesh Bagdiya
70	Filtros de CoreImage	Md. Ibrahim Hassan
71	Firma de código	HaemEternal
72	Fuentes personalizadas	Alexi , Dima Deplov , Harshal Bhavsar , Maddy ヅヅ , njuri , Stephen Leppik , Tommie C.
73	GCD (Grand Central Dispatch)	Andrea Antonioni , DS Dharma , Fonix , Md. Ibrahim Hassan , skyline75489
74	Gráfico (Coreplot)	MarmiK , Md. Ibrahim Hassan
75	Grupo de despacho	Brandon , Fonix
76	Guía para elegir los mejores patrones de arquitectura iOS	Phani Sai
77	Guión gráfico	Harshal Bhavsar , Kirit Vaghela , Stephen Leppik , Tommie C.
78	Hacer selectivas esquinas UIView redondeadas.	Md. Ibrahim Hassan
79	iBeacon	amar , Arefly , Harshal Bhavsar , Stephen Leppik
80	IBOutlets	Fabio , SharkbaitWhohaha
81	Imágenes de caché en línea	Md. Ibrahim Hassan
82	Instantánea de UIView	Bright Future , Darshit Shah , Md. Ibrahim Hassan , SpaceDog
83	Integración SqlCipher	Nirav
84	Interoperabilidad rápida y objetiva-C	Harshal Bhavsar , njuri , Stephen Leppik
85	iOS - Implementación de XMPP con el framework Robbie Hanson	Saheb Roy

86	iOS TTS	Ali Abbas , Stephen Leppik
87	Kit de juego	BennX , Seyyed Parsa Neshaei
88	Kit de salud	Md. Ibrahim Hassan
89	Llavero	abjurato , avojak , Matthew Seaman , Mehul Chuahan
90	Localización	4444 , animuson , Joshua , Ruby , WMios
91	Manejando el teclado	Alexander Tkachenko , Greg , Harshal Bhavsar , Mr. Xcoder , Richard Ash , Shog9 , sxl , Stephen Leppik , Steve Moser , Suragch , V1P3R , william205 , WMios
92	Manejar múltiples entornos usando macro	Tien
93	Manejo de esquemas de URL	azimov , Brian , Dunja Lalic , Harshal Bhavsar , James P , Stephen Leppik
94	Marco de contactos	Md. Ibrahim Hassan , Seyyed Parsa Neshaei
95	Mensajería FCM en Swift	Saeed-rz
96	Métodos personalizados de selección de UITableViewCells.	Kamil Harasimowicz
97	MKDistanceFormatter	Harshal Bhavsar , Md. Ibrahim Hassan , Stephen Leppik , Undo
98	MKMapView	Arnon Rodrigues , Brian , FelixSFD , Harshal Bhavsar , Kosuke Ogawa , Mahesh , Mehul Thakkar , Ortwin Gentz , Reinier Melian , Stephen Leppik
99	ModeloPresentaciónEstilos	Dishant Kapadiya
100	Modismos de inicialización	Jano
101	Modos de fondo	Seyyed Parsa Neshaei
102	Modos de fondo y eventos	Ashish Kakkad
103	MPMediaPickerDelegate	FelixSFD , George Lee
104	MPVolumeView	lostAtSeaJoshua
105	MVVM	JPetric
106	MyLayout	

107	Notificaciones enriquecidas	Koushik
108	Notificaciones push	Amanpreet , Anh Pham , Ashish Kakkad , Bhadresh Kathiriya , BloodWoork , Bonnie , Honey , Hossam Ghareeb , iOS BadBoy , J F , Patrick Beard , Pavel Gurov , sanman , Seyyed Parsa Neshaei , tilo
109	NSArray	Krunal , user5553647
110	NSAttributedString	Bhavin Ramani , Harshal Bhavsar , Jinhuan Li , Kirit Modi , Luiz Henrique Guimaraes , Mansi Panchal , Stephen Leppik , Tim , Tim Ebenezer , Undo
111	NSBundle	wdywayne
112	NSData	Felipe Cypriano , maxkonovalov , Seyyed Parsa Neshaei
113	NSDate	Bonnie , Charles , dasdom , Dunja Lalic , ERbittuu , FelixSFD , Harshal Bhavsar , Jon Snow , Josh Caswell , lostAtSeaJoshua , maxkonovalov , Mehul Thakkar , NSNoob , Nykholas , OhadM , Sally , Samuel Teferra , Sandy , Seyyed Parsa Neshaei , Stephen Leppik , tharkay , tobeiosdeveloper
114	NSHTTPCookieStorage	balagurubaran
115	NSInvocación	Md. Ibrahim Hassan
116	NSNotificationCenter	Alex Kallam , Alex Koshy , Anand Nimje , Bence Pattogato , Bright Future , Ichthyocentaurs , Jacopo Penzo , James P , Kirit Modi , Tarun Seera
117	NSPredicate	Brendon Roberto , Joshua , Mehul Chuahan
118	NSTimer	AJ9 , James P , Maddy`ヅヅ` , Samuel Teferra , tfrank377 , That lazy iOS Guy , Undo , william205
119	NSURConexión	byJeevan
120	NSURL	Adnan Aftab , ApolloSoftware , tharkay
121	NSUserActivity	Samuel Spencer
122	NSUserDefaults	Anand Nimje , Emptyless , Harshal Bhavsar , Husein Behboodi Rad , J F , James P , Josh Caswell , Kirit Modi , Mr. Xcoder , Roland Keesom , Seyyed Parsa Neshaei , user3760892 , william205
123	Objetos asociados a	Noam

	Objective-C	
124	OpenGL	Fonix
125	Operaciones de toda la aplicación	midori
126	Pasando datos entre los controladores de vista	Arulkumar , Ashish Kakkad , BorisE , Bright Future , Dima Deplov , dispute , FelixSFD , Honey , ignotusverum , Irfan , Jake Runzer , juanjo , Kasun Randika , Kendall Lister , Kyle KIM , Luca D'Alberti , muazhud , OhadM , RamenChef , rustproofFish , salabaha , StackUnderflow , Steve Moser , Suragch , Tamarous , timbroder , Undo , WMios , Yagnesh Dobariya
127	Pasar datos entre los controladores de vista (con MessageBox-Concept)	StackUnderflow
128	Perfil con instrumentos	Vinod Kumar
129	plist iOS	SNarula
130	Proceso de envío de aplicaciones	Nermin Sehic
131	Pruebas de interfaz de usuario	P. Pawluś
132	Red de redes	4444 , Mayank Patel , OhadM , Ruby
133	Referencia CGContext	4444 , Narendra Pandey
134	Reino	subv3rsion
135	Segues	Daniel Ormeño
136	Seguridad	D4ttatraya
137	Servicios de safari	Arnon Rodrigues , Harshal Bhavsar , Kilian Koeltzsch , Md. Ibrahim Hassan , Stephen Leppik
138	SESIÓN	bluey31 , dasdom , dgatwood , Duly Kinsky , Harshal Bhavsar , Narendra Pandey , Otávio , R P , sage444 , Stephen Leppik
139	Simulador	Seyyed Parsa Neshaei
140	Simulador construye	Durai Amuthan.H
141	Simulando Ubicación	Uma

	Usando archivos GPX iOS	
142	Sintetizador AVSpeech	Ali Beadle , Bhumit Mehta , Harshal Bhavsar , Midhun MP , Stephen Leppik
143	SiriKit	Seyyed Parsa Neshaei
144	SLComposeViewController	Md. Ibrahim Hassan
145	StoreKit	askielboe
146	Swift: cambiando el control rootViewController en AppDelegate para presentar el flujo principal o de inicio de sesión / incorporación	cleverbit
147	SWRevealViewController	Reinier Melian , tharkay
148	Tablas de clasificación de GameCenter	4444 , Cyril Ivar Garcia , Harshal Bhavsar , Stephen Leppik
149	Teclado personalizado	Md. Ibrahim Hassan
150	Tiempo de ejecución en Objective-C	halil_g
151	Tipo dinámico	Alvin Abia , H. M. Madrone , Harshal Bhavsar , James P , Stephen Leppik
152	Toque 3D	4444 , Harshal Bhavsar , LinusGeffarth , Md. Ibrahim Hassan , Onur Tuna , Stephen Leppik , tobeiosdeveloper
153	Tutorial de AirPrint en iOS	Md. Ibrahim Hassan
154	Ubicación del núcleo	Harshal Bhavsar , Mayuri R Talaviya , Mehul Chuahan , mtso , quant24 , Stephen Leppik , sushant jagtap , william205
155	UIActivityViewController	Amandeep , Harshal Bhavsar , Stephen Leppik , Vivek Molkar
156	UIAlertController	Andrii Chernenko , Arefly , Bhavin Ramani , FelixSFD , Harshal Bhavsar , Irfan , juliand665 , Kirit Modi , Muhammad Zohaib Ehsan , Narendra Pandey , Nikita Kurtin , NSNoob , pableiros , Senseful , Seyyed Parsa Neshaei , shim , Stephen Leppik , Sunil Sharma , Suragch , user3480295

157	UIBarButtonItem	Ahmed Khalaf, Dunja Lalic, hgwhittle, Suragch, william205
158	UIBezierPath	Bean, Igor Bidiniuc, Suragch, Teja Nandamuri
159	UIButton	Aleksei Minaev, Arefly, dasdom, ddb, Fabio Berger, FelixSFD, fredpi, James, James P, Jojodmo, Joshua, mattblessed, Mr. Xcoder, mtso, Nate Lee, NSNoob, P. Pawluś, Quantm, RamenChef, Roland Keesom, Sachin S P, tharkay, Viktor Simkó, william205, WMios
160	UICollectionView	Adam Eberbach, AJ9, Alex Koshy, Anand Nimje, Anh Pham, Bhavin Ramani, Bhumit Mehta, Brian, Dalija Prasnikaar, ddb, Dima Deplov, Harshal Bhavsar, Kevin DiTraglia, Koushik, Mark, Rodrigo de Santiago, Stephen Leppik, Suragch, Undo
161	UIColor	Amanpreet, Anh Pham, Avineet Gupta, Brett Ponder, Cin316, Community, dasdom, DeyaEldeen, Douglas Hill, Elias Datler, Fabio Berger, FelixSFD, Gary Riches, Harshal Bhavsar, Honey, ing0, iphonic, Irfan, JAL, Jaleel Nazir, Jojodmo, Luca D'Alberti, maxkonovalov, mtso, nielsbot, NSNoob, pableiros, Reinier Melian, Rex, Sally, Samer Murad, Sandy, shim, The_Curry_Man, Tommie C. , Viktor Simkó, WMios, Yagnesh Dobariya
162	UIControl - Manejo de eventos con bloques	Brandon
163	UIDatePicker	Pavel Gatilov
164	UIDevice	Bhavin Ramani, FelixSFD, Md. Ibrahim Hassan, Mehul Chuahan, Nef10, pableiros, Ramkumar chintala
165	UIFeedbackGenerator	beyowulf
166	UIFont	Mr. Xcoder
167	UIGestureRecognizer	Adam Preble, dannyzlo, Dunja Lalic, Harshal Bhavsar, John Leonardo, Josh Caswell, Md. Ibrahim Hassan, Ruby , Stephen Leppik, Sujania, Suragch, Undo
168	UIImage	Adrian Schönig, Alexander Tkachenko, Bean, Bhavin Ramani, Dipen Panchasara, Dunja Lalic, Emptyless, FelixSFD, Harshal Bhavsar, Heberti Almeida, Jimmy James, Mahmoud Adam, maxkonovalov, Md. Ibrahim Hassan, Muhammad Zeeshan, RamenChef, Reinier Melian, Rex, rob180, Ronak Chaniyara, sage444, Sandy, Seyyed Parsa Neshaei, Sujania, Sunil Sharma,

The_Curry_Man , user3480295 , Vineet Choudhary		
169	UIImagePickerController	Brian , stonybrooklyn , william205
170	UIImageView	Adam Eberbach , Anh Pham , Bean , Caleb Kleveter , DeyaEldeen , Dunja Lalic , FelixSFD , il Malvagio Dottor Prosciutto , Irfan , Joshua , mattblessed , Md. Ibrahim Hassan , njuri , Quantm , Reinier Melian , Rex , Rob , Samuel Spencer , Sunil Sharma , Suragch , william205
171	UIKit Dynamics	beyowulf , Mark Stewart , Md. Ibrahim Hassan
172	UIKit Dynamics con UICollectionView	beyowulf
173	UILabel	4oby , Akilan Arasu , Alex Koshy , alvarolopez , Andres Canella , Andrii Chernenko , Anh Pham , Ashwin Ramaswami , AstroCB , Barlow Tucker , bentford , Bhumit Mehta , Brian , byJeevan , Caleb Kleveter , Chathuranga Silva , Chris Brandsma , Cin316 , Code.Warrior , Community , Daniel Bocksteger , Daniel Stradowski , danshevluk , dasdom , ddb , DeyaEldeen , Dunja Lalic , Eric , Erwin , esthepiking , Fabio Berger , Fahim Parkar , Felix , FelixSFD , Franck Dernoncourt , gadu , ggrana , GingerHead , gvuksic , HaemEternal , hankide , Hans Sjunnesson , Harshal Bhavsar , Hossam Ghareeb , idobn , Imanou Petit , iOS BadBoy , iphonic , Irfan , J F , Jacky , Jacobbanks , johnpenning , Jojodmo , Josh Brown , Joshua , Joshua J. McKinnon , jtbandes , juanjo , kabioberai , Kai Engelhardt , KANGKANG , Khanh Nguyen , Kireyin , leni , Luca D'Alberti , Iufritz , Lukas , Luke Patterson , Lumialxk , Mad Burea , Mahmoud Adam , Md. Ibrahim Hassan , Moshe , Nadzeya , Narendra Pandey , Nathan Levitt , Nirav D , njuri , noelicus , NSNoob , Ollie , Quantm , Radagast the Brown , Rahul Vyas , RamenChef , ramsserio , rfarry , sage444 , Scotow , Seyyed Parsa Neshaei , Shahabuddin Vansiwala , solidcell , Sravan , stackptr , Sunil Sharma , Suragch , sushant jagtap , TDM , tharkay , The_Curry_Man , Tibor Molnár , Tyler , Undo , user3480295 , vasili111 , Vignan , Viktor Simkó , william205 , WMios , Yagnesh Dobariya
174	UILabel texto subrayado	Md. Ibrahim Hassan
175	UILocalNotification	Bhumit Mehta , Brian , Byte1518 , D4ttatraya , David , ElonChan , Harshal Bhavsar , hgwhittle , kamwysoc , KrishnaCA , rajesh sukumaran , Rex , Samuel Spencer , themathsrobot , tksubota , william205 , Wolverine , Xenon

176	UINavigationController	dasdom , Oleh Zayats , sage444 , Suragch , william205 , WMios
177	UIPageViewController	azimov , Bright Future , Harshal Bhavsar , Mayuri R Talaviya , Stephen Leppik , stonybrooklyn , Victor M
178	UIPheonix: marco de IU fácil, flexible, dinámico y altamente escalable	StackUnderflow
179	UIPickerView	FelixSFD , Hasintha Janka , MCMatan , Md. Ibrahim Hassan , Moritz , NinjaDeveloper
180	UIRefreshControl TableView	Md. Ibrahim Hassan , Mohammad Rana
181	UIScrollView	Bhavin Ramani , LinusGeffarth , maxkonovalov , Rex , sanman , Sujania , Sunil Sharma , Suragch , tharkay , torinpitchers
182	UIScrollView AutoLayout	Aaron , Brandon , Shrikant K
183	UIScrollView con niño StackView	mourodrigo
184	UISearchController	Harshal Bhavsar , Mehul Chuahan , mtso , Stephen Leppik , Tarvo Mäesepp
185	UISlider	Andreas , Md. Ibrahim Hassan
186	UISplitViewController	Cerbrus , Koushik
187	UIStackView	Anuj Joshi , danshevluk , Harshal Bhavsar , Kof , Lior Pollak , Sally , sasquatch , Stephen Leppik , william205
188	UIStoryboard	Adriana Carelli , Mr. Xcoder , Vignan
189	UISwitch	Bhavin Ramani , FelixSFD , Md. Ibrahim Hassan , Mr. Xcoder , RamenChef , Sujay
190	UITabBarController	Alexi , Anand Nimje , Cristina , Mehul Chuahan , Quantm , Srinija
191	UITableView	AJ9 , Alex Koshy , Andres Kievsky , Anh Pham , animuson , Bean , Brendon Roberto , Brian , dasdom , DeyaEldeen , Dima Deplov , Dunja Lalic , Erik Godard , Glorfindel , Harshal Bhavsar , Jojodmo , Kof , Luca D'Alberti , Luis , Meng Zhang , Nathan , Nirav Bhatt , Nirav D , RamenChef , Rex , RodolfoAntonici , Ruby , Samuel Spencer , Seslyn , simple_code , Srinija , Steve Moser , Sujania , Sujay

		Suragch , Tamarous , user3480295
192	UITableViewCell	Rahul
193	UITableViewController	Aju
194	UITextField	Alex Koshy , Ali Elsokary , Ashvinkumar , Duly Kinsky , Fabio Berger , FelixSFD , J F , Joshua , Kof , Luiz Henrique Guimaraes , Maddy ヽヾ, P. Pawluś , RamenChef , Reinier Melian , Ruby , samwize , sasquatch , shim , SourabhV , Suragch , sushant jagtap , tharkay , william205 , WMios
195	UITextField Delegate	Andreas , animuson , Md. Ibrahim Hassan , midori , Ruby
196	UITextField personalizado	D4ttatraya
197	UITextView	Anh Pham , animuson , Bole Tzar , Bright Future , Cris , Dunja Lalic , Eonil , gadu , Harshal Bhavsar , Hejazi , Md. Ibrahim Hassan , njuri , Roland Keesom , Ruby , Suragch , sushant jagtap , william205 , WMios
198	UIViewController	dasdom , Dunja Lalic , shim , Suragch , tassinari , william205
199	UIViews personalizados de archivos XIB	backslash-f , Code.Warrior , Harshal Bhavsar , idocode , Nirav Bhatt , Sharpkits Innovations , Stephen Leppik
200	UIWebView	Allan Burleson , dchar4life80X , iOS BadBoy , J F , Julian135 , KANGKANG , Kevin DiTraglia , maxkonovalov , Md. Ibrahim Hassan , Ortwin Gentz , Ramkumar chintala , Sunil Sharma
201	Usando Aseets de Imagen	D4ttatraya
202	UUID (identificador único universal)	Anand Nimje , FelixSFD , Harshal Bhavsar , James P , Mehul Chuahan , Rahul Vyas , Seyyed Parsa Neshaei , shim , Stephen Leppik , sushant jagtap
203	Vista	Adam Preble , alaphao , Anh Pham , Caleb Kleveter , Community , Cory Wilhite , D4ttatraya , ddb , DeyaEldeen , Douglas Starnes , hgwhittle , iphonic , Irfan , James , Jojodmo , Jota , Kotha Sai Ram , Luca D'Alberti , maxkonovalov , Md. Ibrahim Hassan , muazhud , Narendra Pandey , Nikhil Manapure , NSNoob , pableiros , pckill , Peter DeWeese , Rahul Vyas , sasquatch , shallowThought , Sunil Sharma , That lazy iOS Guy , The_Curry_Man , Viktor Simkó , william205
204	WCSessionDelegate	pkc456
205	WKWebView	Brandon , byJeevan , Mahmoud Adam , Yevhen Dubinin

206	Xcode Build & Archive desde la línea de comandos	Kyle Decot , Shardul
207	XCTest framework - Unit Testing	D4ttatraya , dasdom , Jan ATAC , Josh Brown , msohng , Raphael Silva , Seyyed Parsa Neshaei , Tarun Seera