EBook Gratis

APRENDIZAJE Xamarin.iOS

Free unaffiliated eBook created from
Stack Overflow contributors.

#xamarin.io

S

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: xamarin-ios

It is an unofficial and free Xamarin.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 Xamarin.iOS.

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 Xamarin.iOS

Observaciones

Xamarin.iOS le permite crear aplicaciones iOS nativas utilizando los mismos controles de interfaz de usuario que tendría en Objective-C y Xcode, pero con la flexibilidad y la elegancia de un lenguaje moderno (C #), la potencia de la Biblioteca de clases base .NET (BCL) y dos IDEs de primera clase, Xamarin Studio y Visual Studio, al alcance de su mano.

Para obtener más información sobre la instalación de Xamarin.iOS en su máquina Mac o Windows, consulte las guías de introducción en el centro de desarrolladores de Xamarin

Versiones

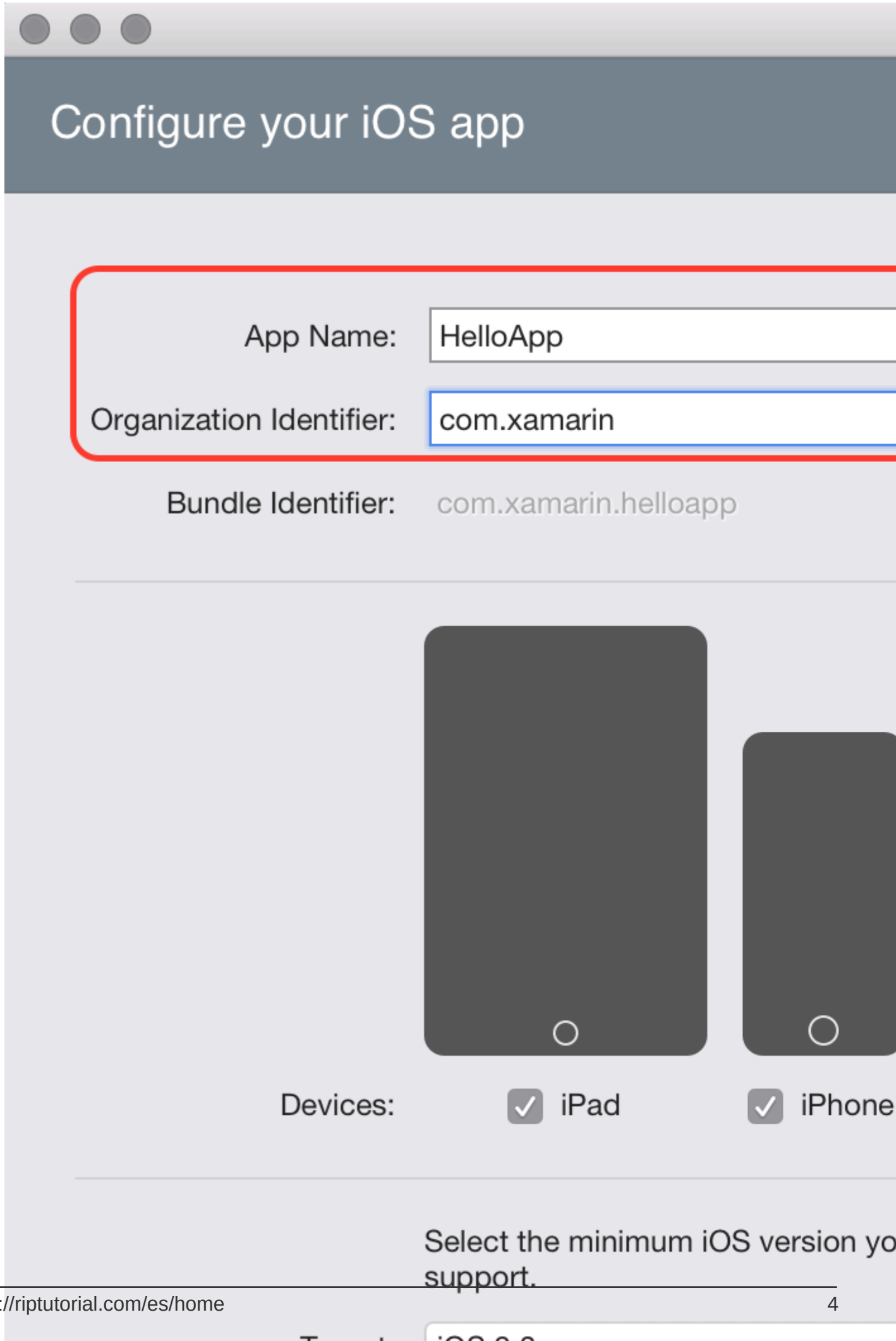
Versión	Fecha de lanzamiento
1.0	2009-09-14
2.0	2010-04-05
3.0	2010-04-16
4.0	2011-04-06
5.0	2011-10-12
6.0	2012-09-19
7.0	2013-09-18
8.0	2014-09-10
9.0	2015-09-17
9.2	2015-11-17
9.4	2015-12-09
9.6	2016-03-22

La información detallada de cada versión se puede encontrar aquí: https://developer.xamarin.com/releases/ios/

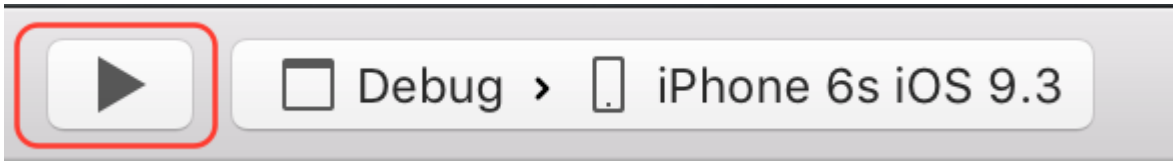
Examples

Empieza en Xamarin Studio

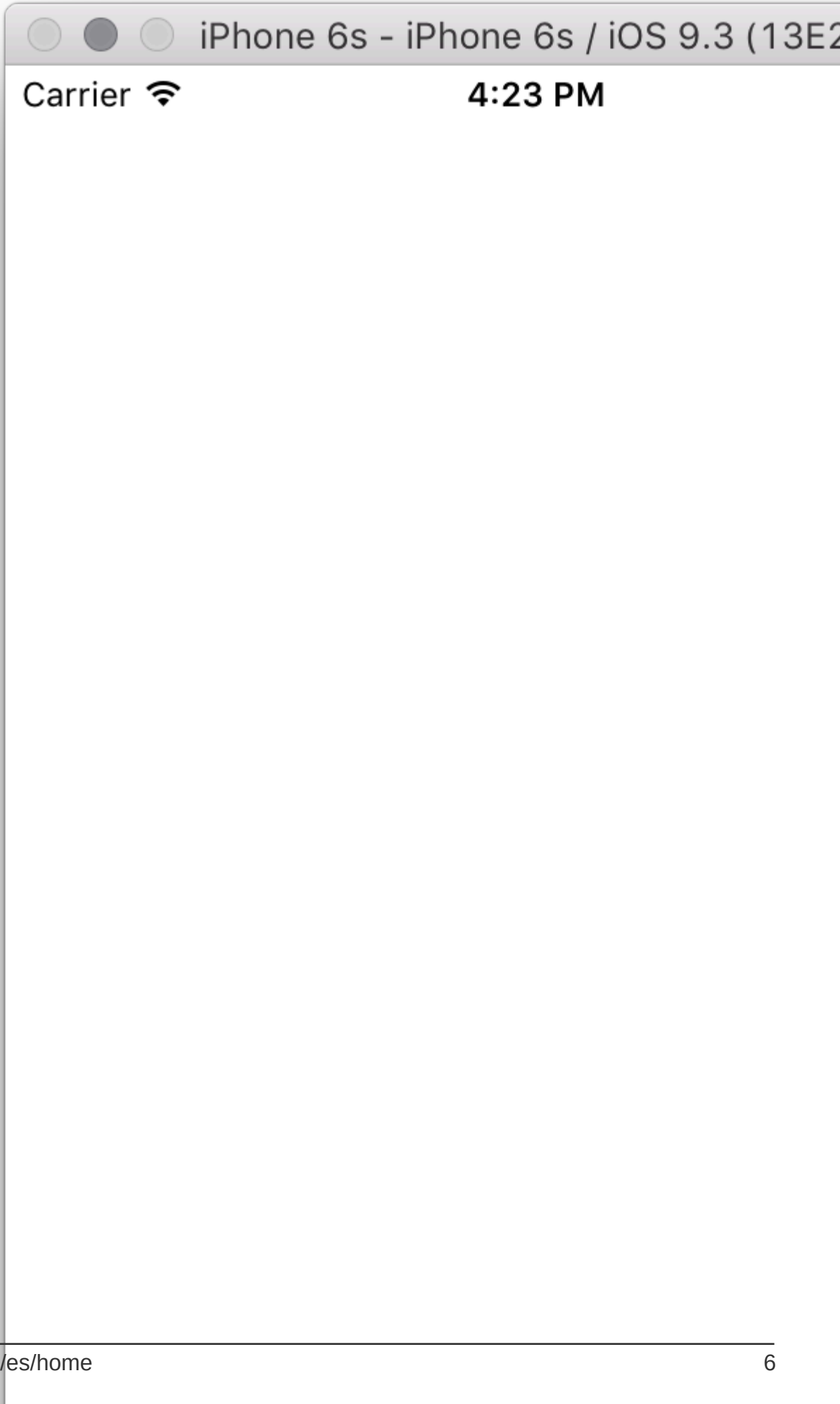
1. Vaya a **Archivo> Nuevo> Solución** para abrir el diálogo del nuevo proyecto
2. Seleccione la **aplicación de vista única** y presione **Siguiente**
3. Configure su aplicación configurando su nombre de aplicación y su ID de organización, y presione **Siguiente** :


Configure your iOS app
App Name:
HelloApp
Organization Identifier:
com.xamarin
Bundle Identifier:
com.xamarin.helloapp
Devices:
iPad
iPhone
Select the minimum iOS version yo
support.

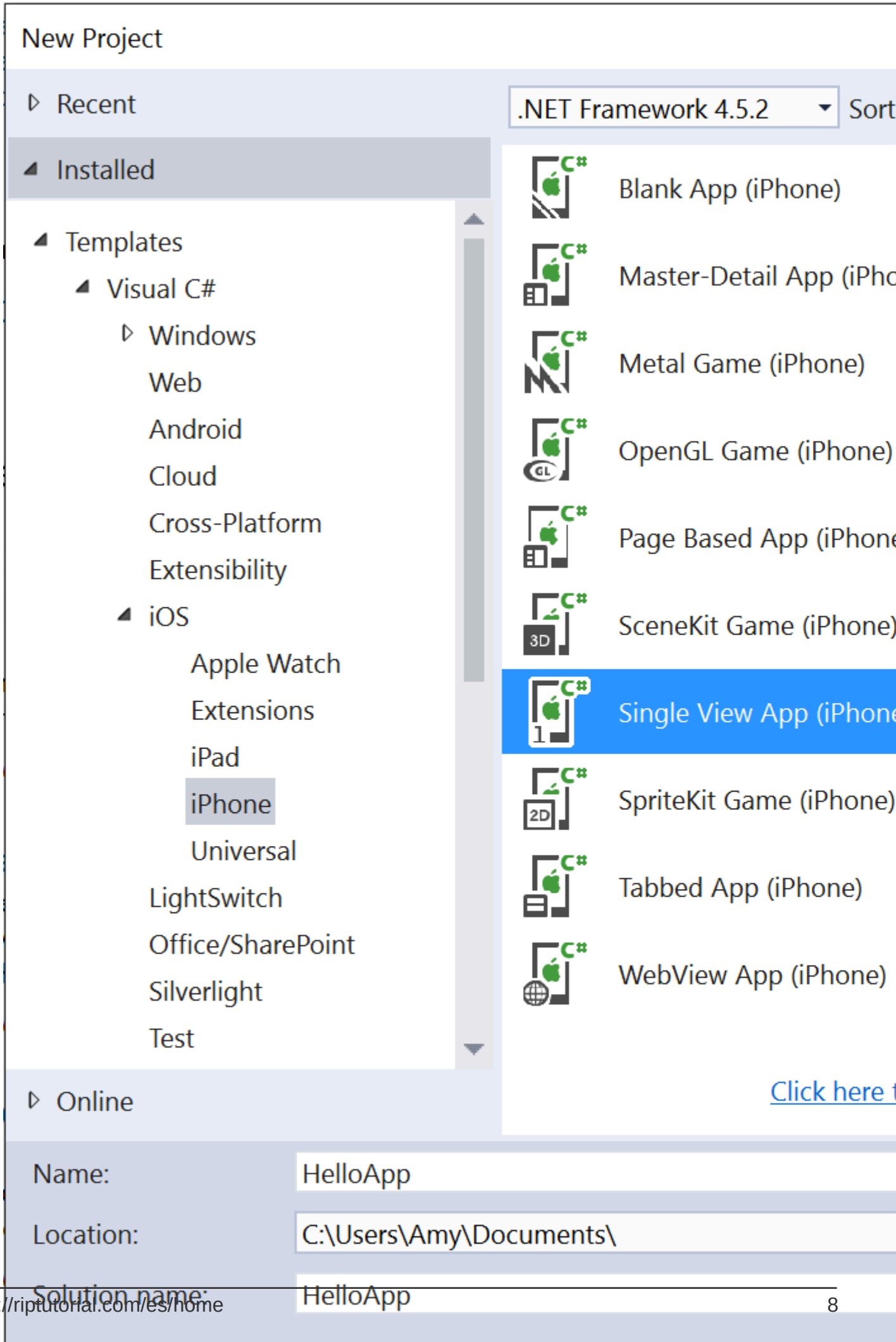
5. Para ejecutar su aplicación, seleccione la opción Depurar | iPhone 6s iOS 9.x configuración, y presione el botón **Play** :



6. Esto iniciará el simulador de iOS y mostrará su aplicación vacía:

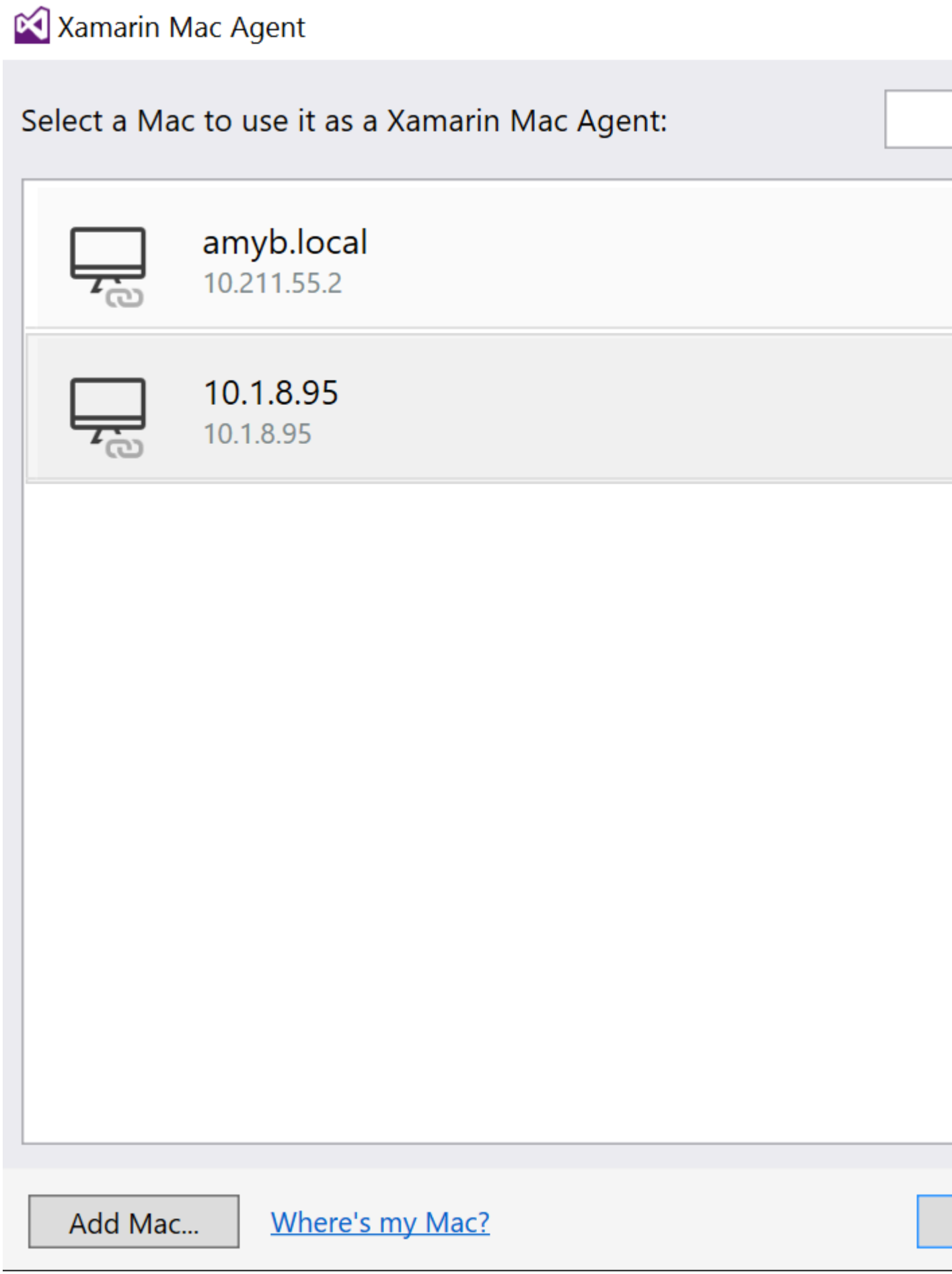

iPhone 6s - iPhone 6s / iOS 9.3 (13E2
Carrier
4:23 PM

2. Vaya a Visual C #> iOS> iPhone y seleccione la aplicación de vista única:


New Project
Recent
.NET Framework 4.5.2
Sort
Installed
Templates
Visual C#
Windows
Web
Android
Cloud
Cross-Platform
Extensibility
iOS
Apple Watch
Extensions
iPad
iPhone
Universal
LightSwitch
Office/SharePoint
Silverlight
Test
Online
Blank App (iPhone)
Master-Detail App (iPho
Metal Game (iPhone)
OpenGL Game (iPhone)
Page Based App (iPhone
SceneKit Game (iPhone)
Single View App (iPhone
SpriteKit Game (iPhone)
Tabbed App (iPhone)
WebView App (iPhone)
Click here t
Name:
HelloApp
Location:
C:\Users\Amy\Documents\
Solution name:
HelloApp

3. Asigne un **nombre a** su aplicación y presione **OK** para crear su proyecto.
4. Seleccione el icono de Mac Agent en la barra de herramientas, como se ilustra a continuación:

5. Seleccione la Mac que construirá su aplicación de la lista (asegúrese de que su Mac esté configurada para recibir la conexión) y presione **Conectar** :

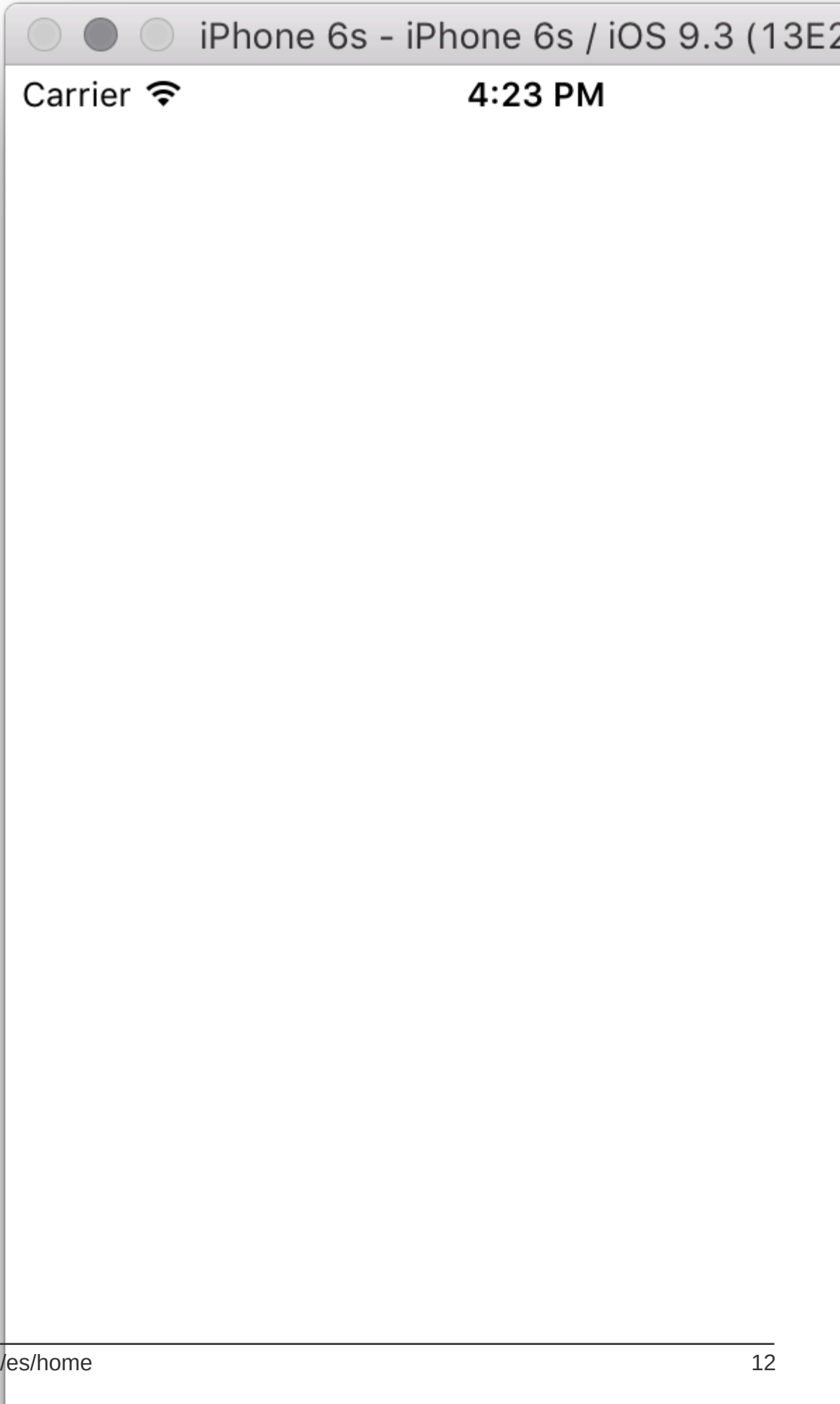


6. Para ejecutar su aplicación, seleccione la **opción Depurar |** Configuración de **iPhoneSimulator** , y presione el botón Play:

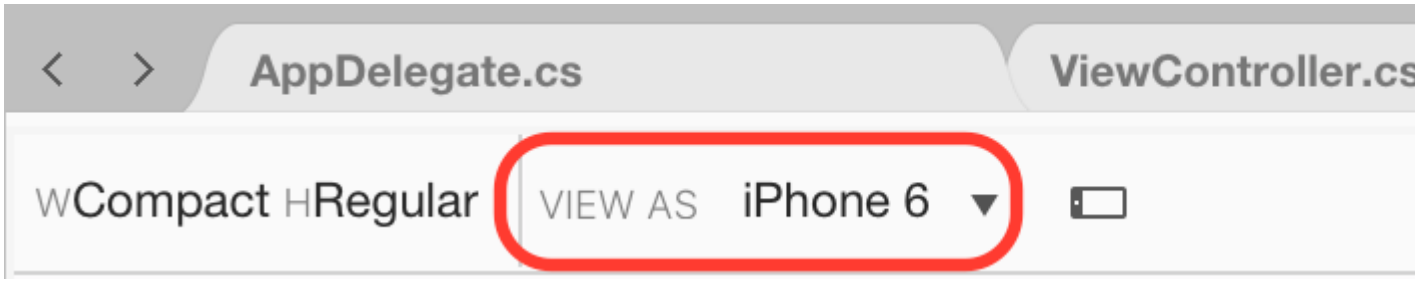
iPhoneSimulator , y presione el botón Play:

iPhoneSimulator , y presione el botón Play:

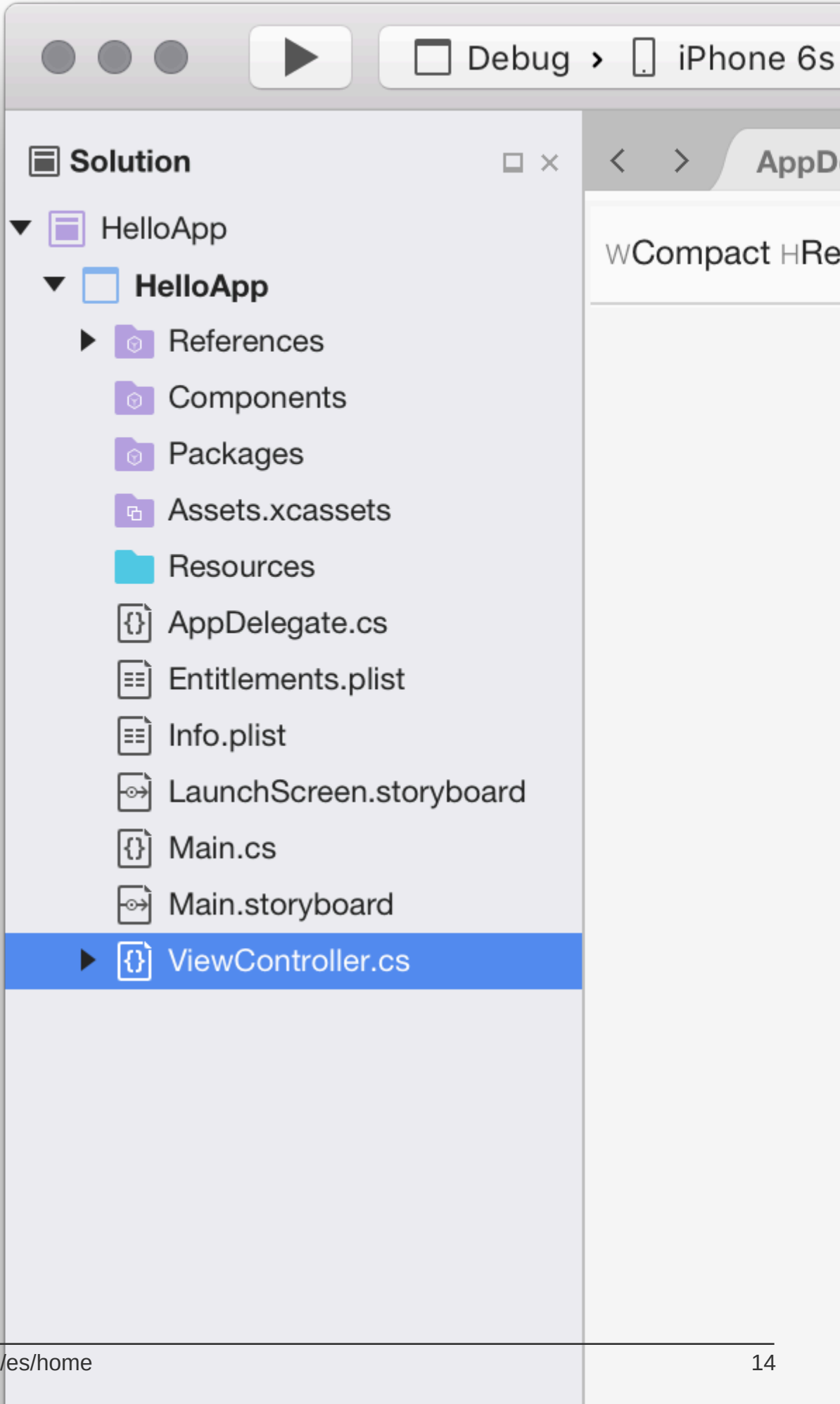
7. Esto iniciará el simulador de iOS en la Mac y mostrará su aplicación vacía:


iPhone 6s - iPhone 6s / iOS 9.3 (13E2
Carrier
4:23 PM

2. Establecer **vista como** para iPhone 6:



3. Arrastre una etiqueta y un botón desde la Caja de herramientas hasta la superficie de diseño para que se vea como la imagen a continuación:


Debug › iPhone 6s
Solution
HelloApp
HelloApp
References
Components
Packages
Assets.xcassets
Resources
AppDelegate.cs
Entitlements.plist
Info.plist
LaunchScreen.storyboard
Main.cs
Main.storyboard
ViewController.cs
AppD
wCompact HRe

4. En el panel de propiedades, asigne a la etiqueta y al botón las siguientes propiedades:

nada	Nombre	Título
Etiqueta	lblClicks	[blanco]
Botón	Haz click en mi	¡Haz click en mi!

5. Agregue el siguiente código al método **ViewDidLoad** dentro de la clase **ViewController** :

```
clickMe.TouchUpInside += (sender, e) =>
{
    totalClicks++;
    if (totalClicks == 1)
    {
        lblClicks.Text = totalClicks + " Click";
    }

   else {
       lblClicks.Text = totalClicks + " Clicks";
   }
};
```

6. Ejecutar la aplicación

Lea Empezando con Xamarin.iOS en línea: https://riptutorial.com/es/xamarin-ios/topic/402/empezando-con-xamarin-ios

Capítulo 2: Agregar UIRefreshControl a una vista de tabla

Examples

Agregar un control UIRefresh a un TableView

Suposiciones

TableView - referencia a la TableView

DataSource - es una clase que hereda UITableViewSource

DataSource.Objects: es una lista pública <object> (), accesible para UIViewController

```
private UIRefreshControl refreshControl;

public override void ViewDidLoad()
{
    base.ViewDidLoad();

    // Set the DataSource for the TableView
    TableView.Source = dataSource = new DataSource(this);

    // Create the UIRefreshControl
    refreshControl = new UIRefreshControl();

    // Handle the pullDownToRefresh event
    refreshControl.ValueChanged += refreshTable;

    // Add the UIRefreshControl to the TableView
    TableView.AddSubview(refreshControl);
}

private void refreshTable(object sender, EventArgs e)
{
    fetchData();
    refreshControl.EndRefreshing();
    TableView.ReloadData();
}

private void fetchData()
{
    var objects = new List<object>();
    // fetch data and store in objects.
    dataSource.Objects = objects;
}
```

Lea Agregar UIRefreshControl a una vista de tabla en línea: https://riptutorial.com/es/xamarin-ios/topic/4642/agregar-uirefreshcontrol-a-una-vista-de-tabla

Capítulo 3: Agregar UIRefreshControl a una vista de tabla

Examples

Agregar un simple UIRefreshControl a un UIScrollView

Asumimos una `UIScrollview` completamente funcional llamada `_scrollView` ;

Tenga en cuenta que `UITableView` , `UICollectionView` también son vistas de desplazamiento, por lo que los siguientes ejemplos funcionarán en esos elementos de la interfaz de usuario.

Primero, creación y asignación.

```
UIRefreshControl refreshControl = new UIRefreshControl();
```

Segundo, conectando el evento de actualización a un método. Hay diferentes maneras de hacer eso.

Estilo 1:

```
refreshControl.ValueChanged += (object sender, EventArgs e) => MyMethodCall();
```

Estilo 2:

```
refreshControl.ValueChanged += (object sender, EventArgs e) =>
{
    //Write code here
};
```

Estilo 3:

```
refreshControl.ValueChanged += HandleRefreshValueChanged;

void HandleRefreshValueChanged(object sender, EventArgs e)
{
    //Write code here
}
```

Tercero y último, agregando el control de actualización a nuestra vista de desplazamiento.

```
_scrollView.AddSubview(refreshControl);
```

Lea Agregar UIRefreshControl a una vista de tabla en línea: https://riptutorial.com/es/xamarin-ios/topic/8371/agregar-uirefreshcontrol-a-una-vista-de-tabla

Capítulo 4: Añadir barra de búsqueda a UITableView

Observaciones

Referencias de objetos:

Tabla de UITableView;
TableSource tableSource;
Listar artículos de tabla;
UISearchBar searchBar;

Para obtener la muestra completa: https://github.com/adiiaditya/Xamarin.iOS-Samples/tree/master/SearchBarWithTableView

Examples

Añadir UISearchBar a UITableView

```
public override void ViewDidLoad()
{
    base.ViewDidLoad();
    // Perform any additional setup after loading the view, typically from a nib.

    //Declare the search bar and add it to the header of the table
    searchBar = new UISearchBar();
    searchBar.SizeToFit();
    searchBar.AutocorrectionType = UITextAutocorrectionType.No;
    searchBar.AutocapitalizationType = UITextAutocapitalizationType.None;
    searchBar.TextChanged += (sender, e) =>
    {
        //this is the method that is called when the user searches
        searchTable();
    };

    Title = "SearchBarWithTableView Sample";
    table = new UITableView(new CGRect(0, 20, View.Bounds.Width, View.Bounds.Height - 20));
    //table.AutoresizingMask = UIViewAutoresizing.All;
    tableItems = new List<TableItem>();

    tableItems.Add(new TableItem("Vegetables") { ImageName = "Vegetables.jpg" });
    tableItems.Add(new TableItem("Fruits") { ImageName = "Fruits.jpg" });
    tableItems.Add(new TableItem("Flower Buds") { ImageName = "Flower Buds.jpg" });
    tableItems.Add(new TableItem("Legumes") { ImageName = "Legumes.jpg" });
    tableItems.Add(new TableItem("Tubers") { ImageName = "Tubers.jpg" });
    tableSource = new TableSource(tableItems);
    table.Source = tableSource;
    table.TableHeaderView = searchBar;
    Add(table);
}
```

```
private void searchTable()
{
    //perform the search, and refresh the table with the results
    tableSource.PerformSearch(searchBar.Text);
    table.ReloadData();
}
```

La clase TableSource se verá así:

```
public class TableSource : UITableViewSource
{
    private List<TableItem> tableItems = new List<TableItem>();
    private List<TableItem> searchItems = new List<TableItem>();
    protected string cellIdentifier = "TableCell";

    public TableSource(List<TableItem> items)
    {
        this.tableItems = items;
        this.searchItems = items;
    }

    public override nint RowsInSection(UITableView tableview, nint section)
    {
        return searchItems.Count;
    }

    public override UITableViewCell GetCell(UITableView tableView, NSIndexPath indexPath)
    {
        // request a recycled cell to save memory
        UITableViewCell cell = tableView.DequeueReusableCell(cellIdentifier);


        var cellStyle = UITableViewCellStyle.Default;

        // if there are no cells to reuse, create a new one
        if (cell == null)
        {
            cell = new UITableViewCell(cellStyle, cellIdentifier);
        }

        cell.TextLabel.Text = searchItems[indexPath.Row].Title;
        cell.ImageView.Image = UIImage.FromFile("Images/" +
searchItems[indexPath.Row].ImageName);

        return cell;
    }

    public override nint NumberOfSections(UITableView tableView)
    {
        return 1;
    }

    public void PerformSearch(string searchText)
    {
        searchText = searchText.ToLower();
        this.searchItems = tableItems.Where(x =>
x.Title.ToLower().Contains(searchText)).ToList();
    }
}
```

Lea Añadir barra de búsqueda a UITableView en línea: https://riptutorial.com/es/xamarin-ios/topic/6540/anadir-barra-de-busqueda-a-uitableview

Capítulo 5: Añadir PullToRefresh a UITableView

Observaciones

Referencias de objetos:

Tabla de UITableView;
TableSource tableSource;
bool useRefreshControl = falso;
UIRefreshControl RefreshControl;
Listar artículos de tabla;

TableSource y TableItem son clases definidas por el usuario

Para obtener el ejemplo completo, puede obtener: https://github.com/adiiaditya/Xamarin.iOS-Samples/tree/master/PullToRefresh

Examples

Añadiendo UIRefreshControl a UITableView

```
public override async void ViewDidLoad(){
base.ViewDidLoad();
// Perform any additional setup after loading the view, typically from a nib.

Title = "Pull to Refresh Sample";
table = new UITableView(new CGRect(0, 20, View.Bounds.Width, View.Bounds.Height - 20));
//table.AutoresizingMask = UIViewAutoresizing.All;
tableItems = new List<TableItem>();
tableItems.Add(new TableItem("Vegetables") { ImageName = "Vegetables.jpg" });
tableItems.Add(new TableItem("Fruits") { ImageName = "Fruits.jpg" });
tableItems.Add(new TableItem("Flower Buds") { ImageName = "Flower Buds.jpg" });
tableItems.Add(new TableItem("Legumes") { ImageName = "Legumes.jpg" });
tableItems.Add(new TableItem("Tubers") { ImageName = "Tubers.jpg" });
tableSource = new TableSource(tableItems);
table.Source = tableSource;

await RefreshAsync();

AddRefreshControl();

Add(table);
table.Add(RefreshControl);
}

async Task RefreshAsync()
{
// only activate the refresh control if the feature is available
if (useRefreshControl)
```

```
    RefreshControl.BeginRefreshing();

if (useRefreshControl)
    RefreshControl.EndRefreshing();

    table.ReloadData();
}

#region * iOS Specific Code
// This method will add the UIRefreshControl to the table view if
// it is available, ie, we are running on iOS 6+
void AddRefreshControl()
{
if (UIDevice.CurrentDevice.CheckSystemVersion(6, 0))
{
    // the refresh control is available, let's add it
    RefreshControl = new UIRefreshControl();
    RefreshControl.ValueChanged += async (sender, e) =>
    {
        tableItems.Add(new TableItem("Bulbs") { ImageName = "Bulbs.jpg" });
        await RefreshAsync();
    };
    useRefreshControl = true;
   }
}
#endregion
```

Lea Añadir PullToRefresh a UITableView en línea: https://riptutorial.com/es/xamarin-ios/topic/6565/anadir-pulltorefresh-a-uitableview

Capítulo 6: Cálculo de altura de fila variable en GetHeightForRow

Observaciones

Calcular la altura de las filas puede ser costoso y el rendimiento del desplazamiento puede verse afectado si tiene grandes cantidades de datos. En ese escenario, anule `UITableViewSource.EstimatedHeight(UITableView, NSIndexPath)` para proporcionar rápidamente un número suficiente para el desplazamiento rápido, por ejemplo:

```
public override nfloat EstimatedHeight(UITableView tableView, NSIndexPath indexPath)
{
    return 44.0f;
}
```

Examples

Usando GetHeightForRow

Para establecer un alto de fila personalizado, anule `UITableViewSource.GetHeightForRow(UITableView,NSIndexPath)` :

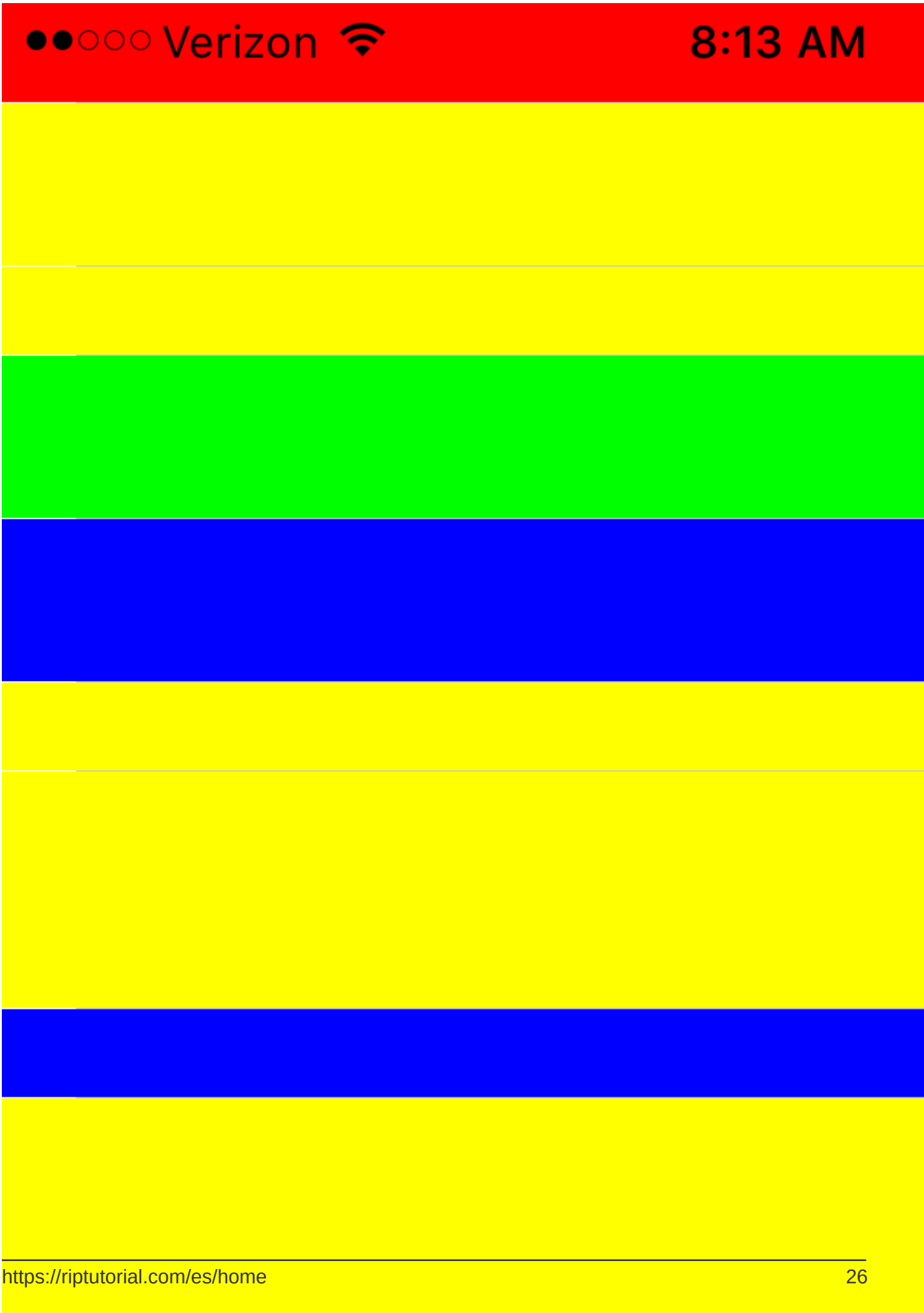
```
public class ColorTableDataSource : UITableViewSource
{
    List<DomainClass> Model { get; set; }

    public override nfloat GetHeightForRow(UITableView tableView, NSIndexPath indexPath)
    {
        var height = Model[indexPath.Row % Model.Count].Height;
        return height;
    }

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

La clase de dominio para la tabla (en este caso, tiene 1 de 3 colores aleatorios y 1 de 3 alturas aleatorias):

```
public class DomainClass
{
    static Random rand = new Random(0);
    public UIColor Color { get; protected set; }
    public float Height { get; protected set; }

    static UIColor[] Colors = new UIColor[]
    {
        UIColor.Red,
        UIColor.Green,
        UIColor.Blue,
```

```
        UIColor.Yellow
    };

    public DomainClass()
    {
        Color = Colors[rand.Next(Colors.Length)];
        switch (rand.Next(3))
        {
            case 0:
                Height = 24.0f;
                break;
            case 1:
                Height = 44.0f;
                break;
            case 2:
                Height = 64.0f;
                break;
            default:
                throw new ArgumentOutOfRangeException();
        }
    }

    public override string ToString()
    {
        return string.Format("[DomainClass: Color={0}, Height={1}]", Color, Height);
    }
}
```

Que se parece a


Verizon
8:13 AM

https://riptutorial.com/es/xamarin-ios/topic/6515/calculo-de-altura-de-fila-variable-en-getheightforrow

Capítulo 7: Cómo utilizar los catálogos de activos de activos

Examples

Uso de catálogos de activos

Para usar un Catálogo de Activos, necesita hacer lo siguiente:

1. Haga doble clic en el archivo Info.plist en el Explorador de soluciones para abrirlo y editarlo.
2. Desplázate hacia abajo hasta la sección Iconos de aplicaciones.
3. En la lista desplegable Fuente, asegúrese de que AppIcons esté seleccionado.
4. Desde el Explorador de soluciones, haga doble clic en el archivo Assets.xcassets para abrirlo y editarlo.
5. Seleccione AppIcons de la lista de activos para mostrar el Editor de iconos.
6. Haga clic en el tipo de icono dado y seleccione un archivo de imagen para el tipo / tamaño requerido o arrastre una imagen desde una carpeta y suéltela en el tamaño deseado.
7. Haga clic en el botón Abrir para incluir la imagen en el proyecto y configurarla en xcasset.

Lea Cómo utilizar los catálogos de activos de activos en línea: https://riptutorial.com/es/xamarin-ios/topic/6539/como-utilizar-los-catalogos-de-activos-de-activos

Capítulo 8: Conectando con Microsoft Cognitive Services

Observaciones

En este ejemplo se utilizó el paquete Microsoft.ProjectOxford.Vision NuGet: https://www.nuget.org/packages/Microsoft.ProjectOxford.Vision/

Para leer más acerca de los servicios de Microsoft cognitivas por favor refiérase a la documentación oficial: https://www.microsoft.com/cognitive-services/en-us/computer-vision-api

Encuentre muestra cargado en mi GitHub: https://github.com/Daniel-Krzyczkowski/XamarinIOS/tree/master/XamariniOS_CognitiveServices

También hay que adjuntar un enlace a mi blog donde presenté cómo utilizar los servicios cognitivos con Xamarin Forms: http://mobileprogrammer.pl

Examples

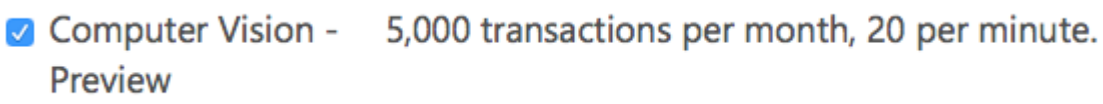
Conectando con Microsoft Cognitive Services

En este ejemplo, aprenderá a usar los Servicios cognitivos de Microsoft con la aplicación móvil Xamarin iOS. Usaremos Computer Vision API para detectar lo que hay en la imagen.

Una vez que cree el proyecto Xamarin.iOS, a continuación agregue el paquete NuGet al proyecto:

https://www.nuget.org/packages/Microsoft.ProjectOxford.Vision/

Con esta biblioteca podremos utilizar los Servicios cognitivos en nuestra aplicación iOS. Supongo que ya tiene una cuenta de Microsoft registrada para usarla y ha habilitado Computer Vision Api como en la siguiente pantalla:



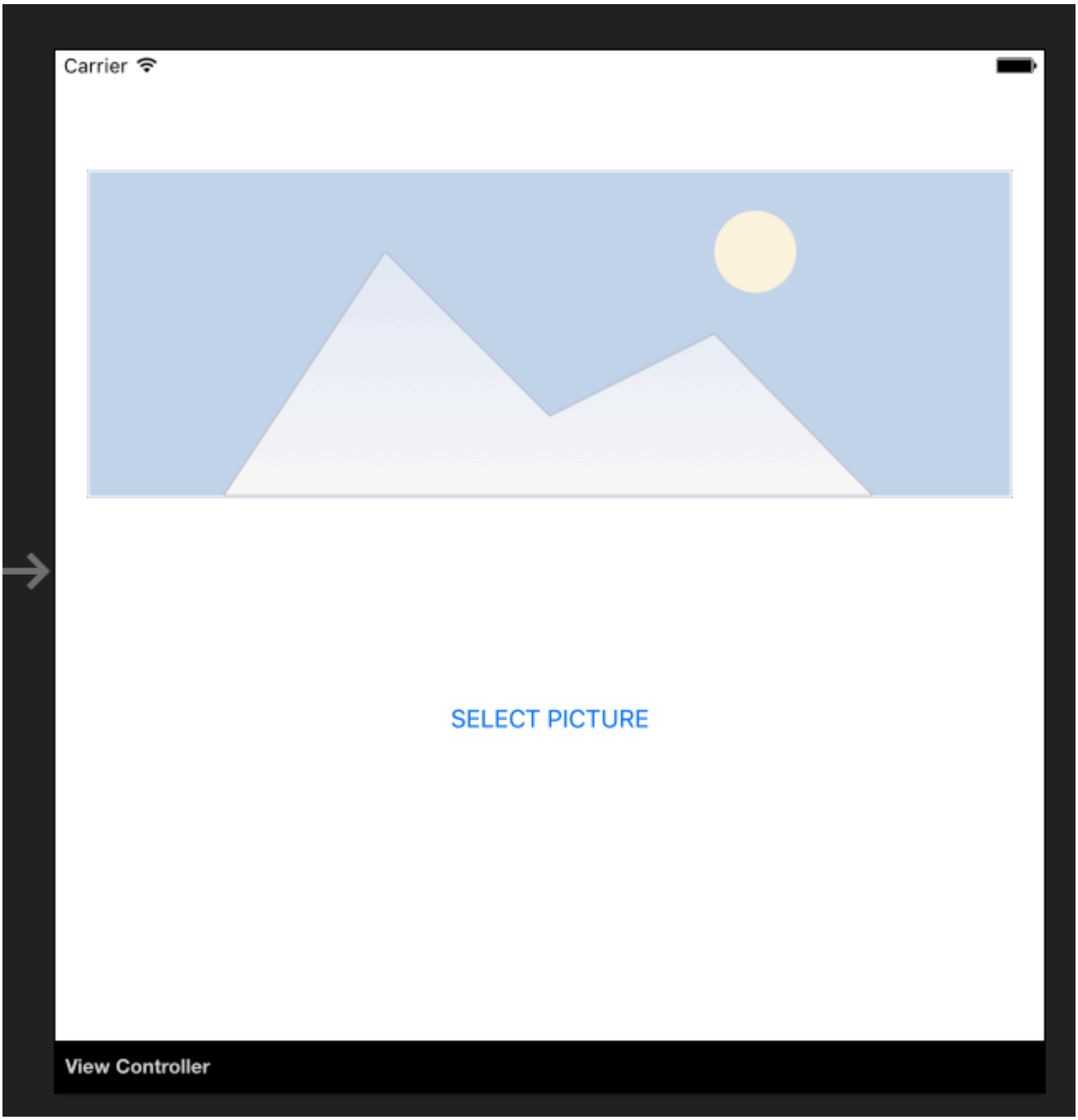
Una vez que haga clic en "Suscribirse" en la parte inferior, se generará la clave Api:

Computer Vision - Preview	5,000 transactions per month, 20 per minute.	Key 1: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate | Show | Copy Key 2: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate | Show | Copy

Ahora podemos comenzar a configurar el acceso a los Servicios cognitivos desde la aplicación iOS. En primer lugar, necesitamos obtener una imagen para el análisis. Para hacerlo, podemos usar Xamarin Media Component disponible a continuación:
https://components.xamarin.com/view/mediaplugin

Una vez que se instale correctamente, vamos a crear una interfaz de usuario simple con la imagen y el botón para seleccionar la imagen de la galería. El tamaño de los controles depende de usted.

Abra Main.storyboard y agregue los controles UIImageView y UIButton que hacen el ViewController predeterminado. Agregue los nombres: "SelectedPictureImageView" y "SelectButton":



Ahora debemos agregar el controlador de eventos "Touch Up Inside" para manejar la selección de imágenes:

```
partial void SelectButtonClick(UIButton sender)
{
    selectImage();
}

async void selectImage()
{
    var selectedImage = await CrossMedia.Current.PickPhotoAsync();
    SelectedPictureImageView.Image =  new
UIImage(NSData.FromStream(selectedImage.GetStream()));
}
```

Ahora nos gustaría mostrar información de análisis una vez que Cognitive Services devuelva la

información. Añadir etiqueta debajo del botón llamado "AnalysisLabel":

SELECT PICTURE

Analysis result....

¡Es hora de conectar Computer Vision API!

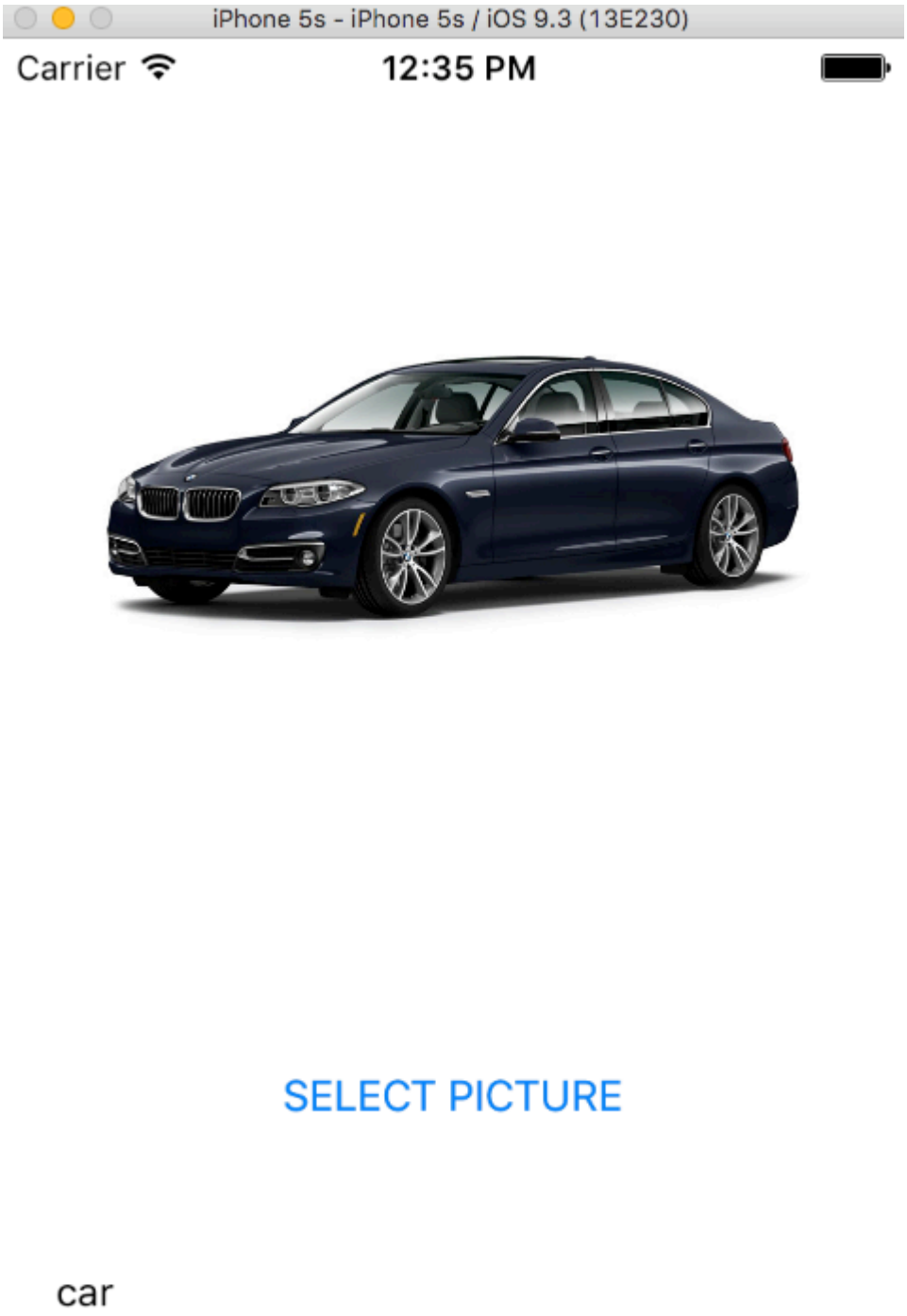
Para obtener información sobre la imagen seleccionada, agregue el siguiente método. Recuerda pegar tu clave API!

```
async Task analyseImage(Stream imageStream)
{
    try
    {
        VisionServiceClient visionClient = new VisionServiceClient("<<YOUR API KEY HERE>>");
        VisualFeature[] features = { VisualFeature.Tags, VisualFeature.Categories,
VisualFeature.Description };
        var analysisResult = await visionClient.AnalyzeImageAsync(imageStream,
features.ToList(), null);
        AnalysisLabel.Text = string.Empty;
        analysisResult.Description.Tags.ToList().ForEach(tag => AnalysisLabel.Text =
AnalysisLabel.Text + tag + "\n");
    }
    catch (Microsoft.ProjectOxford.Vision.ClientException ex)
    {
        AnalysisLabel.Text = ex.Error.Message;
    }
}
```

Ahora puedes invocarlo en el método "selectImage":

```
async void selectImage()
{
    var selectedImage = await CrossMedia.Current.PickPhotoAsync();
    SelectedPictureImageView.Image =  new
UIImage(NSData.FromStream(selectedImage.GetStream()));
    await analyseImage(selectedImage.GetStream());
}
```

Una vez que seleccione la imagen, Microsoft Cognitive Services la analizará y devolverá el resultado:



Recuerde que su imagen no puede ser demasiado grande, en este caso recibirá información como la siguiente:

SELECT PICTURE

Input image is too large.

Hay muchos otros servicios que puedes intentar usar. Por favor, consulte la documentación oficial (enlace adjunto) para descubrir más.

Lea Conectando con Microsoft Cognitive Services en línea: https://riptutorial.com/es/xamarin-ios/topic/6122/conectando-con-microsoft-cognitive-services

Capítulo 9: Controlando la captura de pantalla en el conmutador multitarea de iOS

Introducción

En la Guía de programación de aplicaciones para iOS :

Elimine la información confidencial de las vistas antes de pasar al fondo.

Cuando una aplicación pasa a un segundo plano, el sistema toma una instantánea de la ventana principal de la aplicación, que luego presenta brevemente al hacer la transición de la aplicación al primer plano.

Observaciones

Adaptado de la pregunta de StackOverflow real sobre el control de la captura de pantalla en el conmutador de tareas múltiples iOS7 y la respuesta Obj-c Answer

Examples

Mostrar una imagen para la instantánea

```
public override void DidEnterBackground(UIApplication application)
 {
    //to add the background image in place of 'active' image
    var backgroundImage = new UIImageView();
    backgroundImage.Tag = 1234;
    backgroundImage.Image = UIImage.FromBundle("Background");
    backgroundImage.Frame = this.window.Frame;
    this.window.AddSubview(backgroundImage);
    this.window.BringSubviewToFront(backgroundImage);
}

public override void WillEnterForeground(UIApplication application)
{
    //remove 'background' image
    var backgroundView = this.window.ViewWithTag(1234);
    if(null != backgroundView)
        backgroundView.RemoveFromSuperview();
}
```

Lea Controlando la captura de pantalla en el conmutador multitarea de iOS en línea: https://riptutorial.com/es/xamarin-ios/topic/8681/controlando-la-captura-de-pantalla-en-el-conmutador-multitarea-de-ios

Capítulo 10: Crea y usa celdas de la tabla de prototipos personalizados en xamarin.iOS usando el guión gráfico

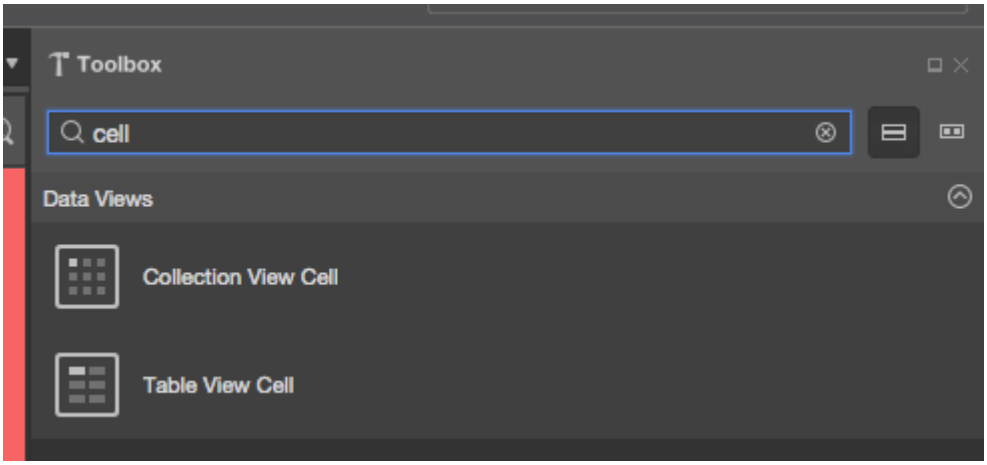
Examples

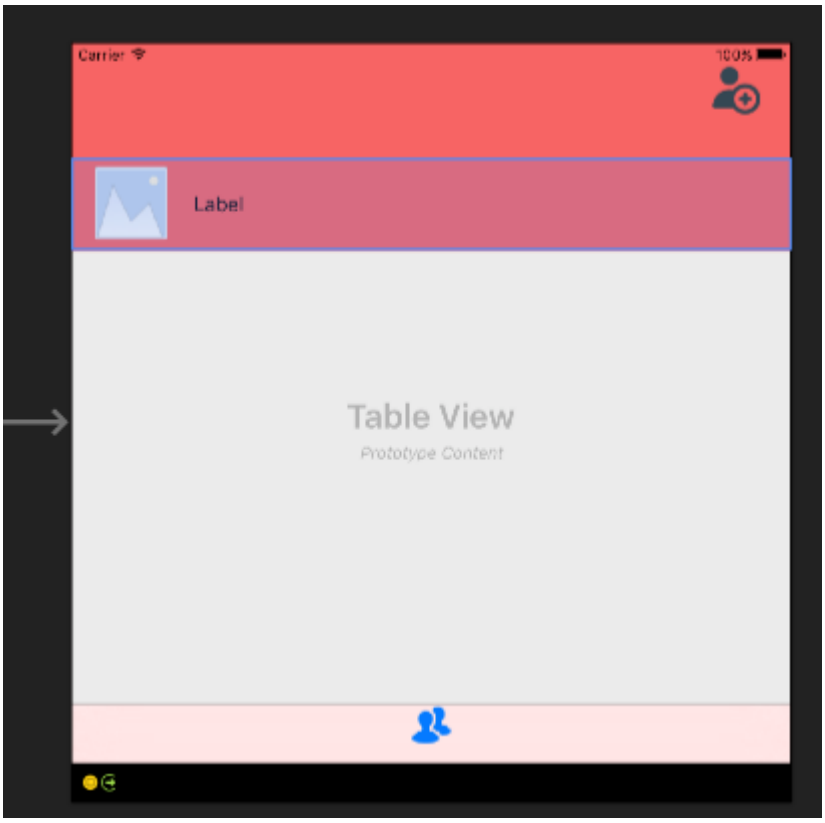
Crear celda personalizada usando Storyboard

Abra el Guión gráfico donde tiene su ViewController con TableView:

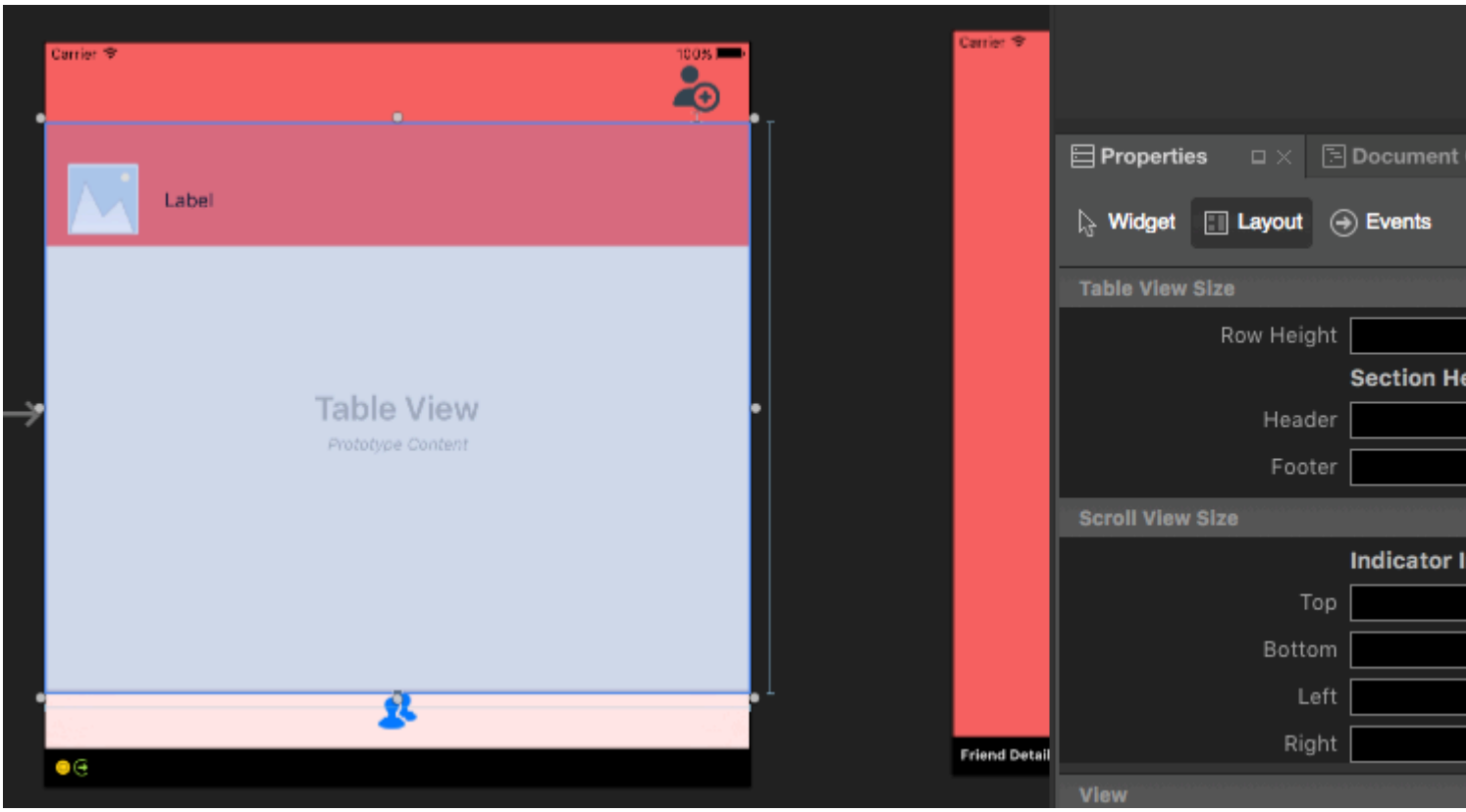
Agregar celda prototipo (si no hay celda agregada antes):

Personaliza la celda como quieras (en mi caso hay UIImage y Label personalizados):



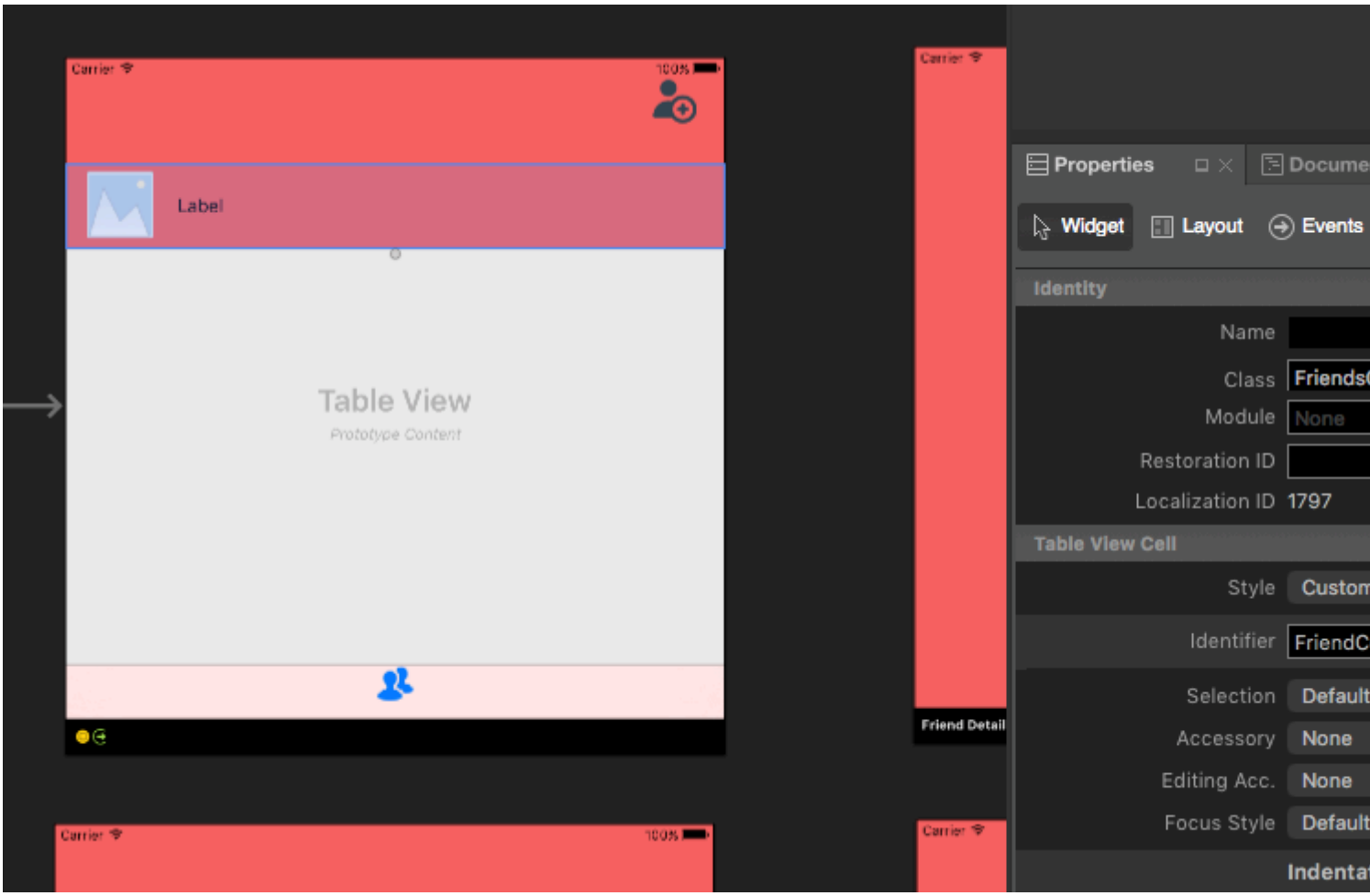


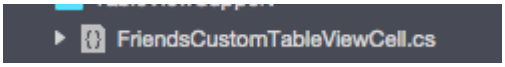
Recuerda ajustar la altura de la celda. Para hacerlo, seleccione su TableView completo y desde la ventana de Propiedades seleccione la pestaña "Diseño". En la parte superior de la ventana de propiedades, debería ver el "alto de fila" - ponga el valor apropiado:



Ahora seleccione la célula prototipo una vez más. En la ventana Propiedades, escriba el nombre de la clase (creará una clase de código subyacente). En mi caso esto es "FriendsCustomTableViewCell". Después de eso proporcione "Identificador" para su celular.

Como puedes ver mi es "FriendCell". Lo último que se establece es la propiedad "Estilo" establecida en personalizada. El campo "Nombre" debe estar vacío. Una vez que haga clic en "entrar" después de escribir "Clase", se creará automáticamente el archivo de código subyacente:





Ahora el código detrás de la celda debería verse a continuación:

```
public partial class FriendsCustomTableViewCell : UITableViewCell
{
    public FriendsCustomTableViewCell (IntPtr handle) : base (handle)
    {
    }

    public FriendsCustomTableViewCell(NSString cellId, string friendName, UIImage friendPhoto)
: base (UITableViewCellStyle.Default, cellId)
    {
        FriendNameLabel.Text = friendName;
        FriendPhotoImageView.Image = friendPhoto;
    }

    //This methods is to update cell data when reuse:
    public void UpdateCellData(string friendName, UIImage friendPhoto)
    {
        FriendNameLabel.Text = friendName;
        FriendPhotoImageView.Image = friendPhoto;
    }
}
```

En UITableViewSource tienes que declarar cellIdentifier en la parte superior de la clase (en mi caso es "FriendCell") y en el método "GetCell" tienes que lanzar celdas y establecer datos para ellos:

```
string cellIdentifier = "FriendCell";

public override UITableViewCell GetCell(UITableView tableView, NSIndexPath indexPath)
{
    FriendsCustomTableViewCell cell = (FriendsCustomTableViewCell)
tableView.DequeueReusableCell(cellIdentifier);
    Friend friend = _friends[indexPath.Row];

    //---- if there are no cells to reuse, create a new one
    if (cell == null)
    { cell = new FriendsCustomTableViewCell(new NSString(cellIdentifier), friend.FriendName,
new UIImage(NSData.FromArray(friend.FriendPhoto))); }

    cell.UpdateCellData(friend.UserName, new UIImage(NSData.FromArray(friend.FriendPhoto)));

    return cell;
}
```

Lea Crea y usa celdas de la tabla de prototipos personalizados en xamarin.iOS usando el guión gráfico en línea: https://riptutorial.com/es/xamarin-ios/topic/5907/crea-y-usa-celdas-de-la-tabla-de-prototipos-personalizados-en-xamarin-ios-usando-el-guion-grafico

Capítulo 11: Disposición automática en Xamarin.iOS

Examples

Agregar restricciones con iOS 9+ Anclajes de diseño

9.0

```
// Since the anchor system simply returns constraints, you still need to add them somewhere.
View.AddConstraints(
    new[] {
        someLabel.TopAnchor.ConstraintEqualTo(TopLayoutGuide.GetBottomAnchor()),
        anotherLabel.TopAnchor.ConstraintEqualTo(someLabel.BottomAnchor, 6),
        oneMoreLabel.TopAnchor.ConstraintEqualTo(anotherLabel.BottomAnchor, 6),

oneMoreLabel.BottomAnchor.ConstraintGreaterThanOrEqualTo(BottomLayoutGuide.GetTopAnchor(), -
10),
    }
);
```

Agregar restricciones usando el lenguaje de formato visual (VFL)

```
// Using Visual Format Language requires a special look-up dictionary of names<->views.
var views = new NSDictionary(
    nameof(someLabel), someLabel,
    nameof(anotherLabel), anotherLabel,
    nameof(oneMoreLabel), oneMoreLabel
);
// It can also take a look-up dictionary for metrics (such as size values).
// Since we are hard-coding those values in this example, we can give it a `null` or empty
dictionary.
var metrics = (NSDictionary)null;

// Add the vertical constraints to stack everything together.
// `V:` = vertical
// `|…|` = constrain to super view (`View` for this example)
// `-10-` = connection with a gap of 10 pixels (could also be a named parameter from the
metrics dictionary)
// `-[viewName]-` = connection with a control by name looked up in views dictionary (using C#
6 `nameof` for refactoring support)
var verticalConstraints = NSLayoutConstraint.FromVisualFormat(
    $"V:|-20-[{nameof(someLabel)}]-6-[{nameof(anotherLabel)}]-6-[{nameof(oneMoreLabel)}]->=10-
|",
    NSLayoutFormatOptions.AlignAllCenterX,
    metrics,
    views
);
View.AddConstraints(verticalConstraints);
```

Es posible que algunos tipos de restricciones, como las relaciones de aspecto , no se puedan transmitir en la sintaxis de Visual Format Language (VFL) y deben llamarse directamente a los

métodos apropiados.

Utilizando Cirrious.FluentLayout

Usando NuGet

```
Install-Package Cirrious.FluentLayout
```

Un ejemplo expandido basado en el ejemplo de inicio en la página de GitHub , un nombre simple, etiquetas de apellido y campos todos apilados uno encima del otro:

```
public override void ViewDidLoad()
{
    //create our labels and fields
    var firstNameLabel = new UILabel();
    var lastNameLabel = new UILabel();
    var firstNameField = new UITextField();
    var lastNameField = new UITextField();

    //add them to the View
    View.AddSubviews(firstNameLabel, lastNameLabel, firstNameField, lastNameField);

    //create constants that we can tweak if we do not like the final layout
    const int vSmallMargin = 5;
    const int vMargin = 20;
    const int hMargin = 10;

    //add our constraints
    View.SubviewsDoNotTranslateAutoresizingMaskIntoConstraints();
    View.AddConstraints(
        firstNameLabel.WithSameTop(View).Plus(vMargin),
        firstNameLabel.AtLeftOf(View).Plus(hMargin),
        firstNameLabel.WithSameWidthOf(View),

        firstNameField.WithSameWidth(firstNameLabel),
        firstNameField.WithSameLeft(firstNameLabel),
        firstNameField.Below(firstNameLabel).Plus(vSmallMargin),

        lastNameLabel.Below(firstNameField).Plus(vMargin),
        lastNameLabel.WithSameLeft(firstNameField),
        lastNameLabel.WithSameWidth(firstNameField),

        lastNameField.Below(lastNameLabel).Plus(vSmallMargin),
        lastNameField.WithSameWidth(lastNameLabel),
        lastNameField.WithSameLeft(lastNameLabel));
}
```

Agregando restricciones con mampostería

Masonry es una biblioteca para objetivo-c, pero xamarin ha creado un enlace para él y lo creó como un paquete nuget https://www.nuget.org/packages/Masonry/ .

Instalación de nuget

```
Install-Package Masonry
```

Esto centra un botón a 100 puntos debajo del punto central de la vista contenedora y establece un ancho entre 200 y 400 puntos

```
this.loginBtn.MakeConstraints(make =>
{
    make.Width.GreaterThanOrEqualTo(new NSNumber(200));
    make.Width.LessThanOrEqualTo(new NSNumber(400));
    make.Center.EqualTo(this.View).CenterOffset(new CGPoint(0, 100));
});
```

Esto establece una imagen escalada de 100 puntos sobre el punto central de la vista que contiene y luego establece el ancho al ancho de la vista que contiene con un multiplicador de 0,5 que significa 50% del ancho. Luego establece la altura al ancho multiplicado por la relación de aspecto que hace que la imagen se amplíe pero mantiene su relación de aspecto correcta.

```
this.logo.MakeConstraints(make =>
{
    make.Center.EqualTo(this.View).CenterOffset(new CGPoint(0, -100));
    make.Width.EqualTo(this.View).MultipliedBy(0.5f);
    make.Height.EqualTo(this.logo.Width()).MultipliedBy(0.71f);
});
```

Lea Disposición automática en Xamarin.iOS en línea: https://riptutorial.com/es/xamarin-ios/topic/1317/disposicion-automatica-en-xamarin-ios

Capítulo 12: Encuadernación de bibliotecas rápidas

Introducción

Una guía fácil de seguir que lo guiará a través del proceso de enlace de archivos Swift .framework para su uso en un proyecto de Xamarin.

Observaciones

1. Cuando se construye una biblioteca en Xcode, tiene una opción para incluir las bibliotecas swift. No lo hagas Se incluirán en su aplicación final como NAME.app/Frameworks/LIBRARY.framework/Frameworks/libswift*.dylib pero deben incluirse como NAME.app/Frameworks/libswift*.dylib

2. Puede encontrar esta información en otra parte, pero vale la pena mencionarlo: no incluya Bitcode en la biblioteca. A partir de este momento, Xamarin no incluye Bitcode para iOS y Apple requiere que todas las bibliotecas admitan las mismas arquitecturas.

Examples

Encuadernación de una Biblioteca Swift en Xamarin.iOS

El enlace de una Biblioteca Swift en Xamarin.iOS sigue el mismo proceso para Objective-C como se muestra en https://developer.xamarin.com/guides/ios/advanced_topics/binding_objective-c/ , pero con algunas advertencias.

1. Una clase swift debe heredar de NSObject para ser enlazada.
2. El compilador Swift traducirá los nombres de clase y protocolo a otra cosa a menos que use la anotación @objc (por ejemplo, @objc (MyClass)) en sus clases swift para especificar el nombre c objetivo explícito.
3. En tiempo de ejecución, su APP debe incluir algunas bibliotecas centrales rápidas junto con su marco enlazado en una carpeta llamada Marcos;
4. Cuando la aplicación se envía a AppStore, debe incluir una carpeta SwiftSupport junto con su carpeta de carga útil. Esos están dentro del archivo IPA.

Aquí puede encontrar un enlace de muestra simple:
https://github.com/Flash3001/Xamarin.BindingSwiftLibrarySample

Y un ejemplo de enlace completo: https://github.com/Flash3001/iOSCharts.Xamarin

Por favor encuentre los pasos a continuación:

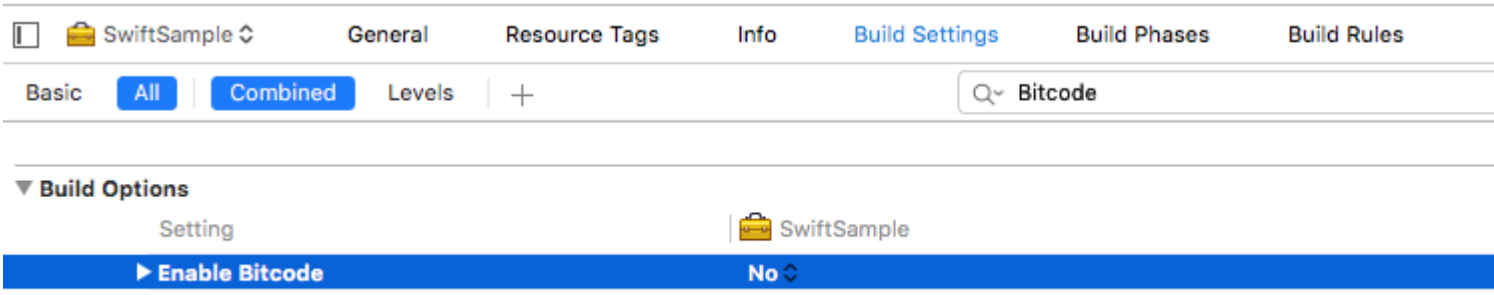
1.1 Prepara las clases Swift que quieres exportar.

Para cualquier clase de Swift que quiera usar, debe heredar de NSObject y hacer explícito el nombre de Objective-C usando la anotación objc. De lo contrario, el compilador Swift generará diferentes nombres. A continuación se muestra un código de ejemplo de cómo podría verse una clase Swift. Tenga en cuenta que no importa qué clase hereda mientras la clase raíz hereda de NSObject.

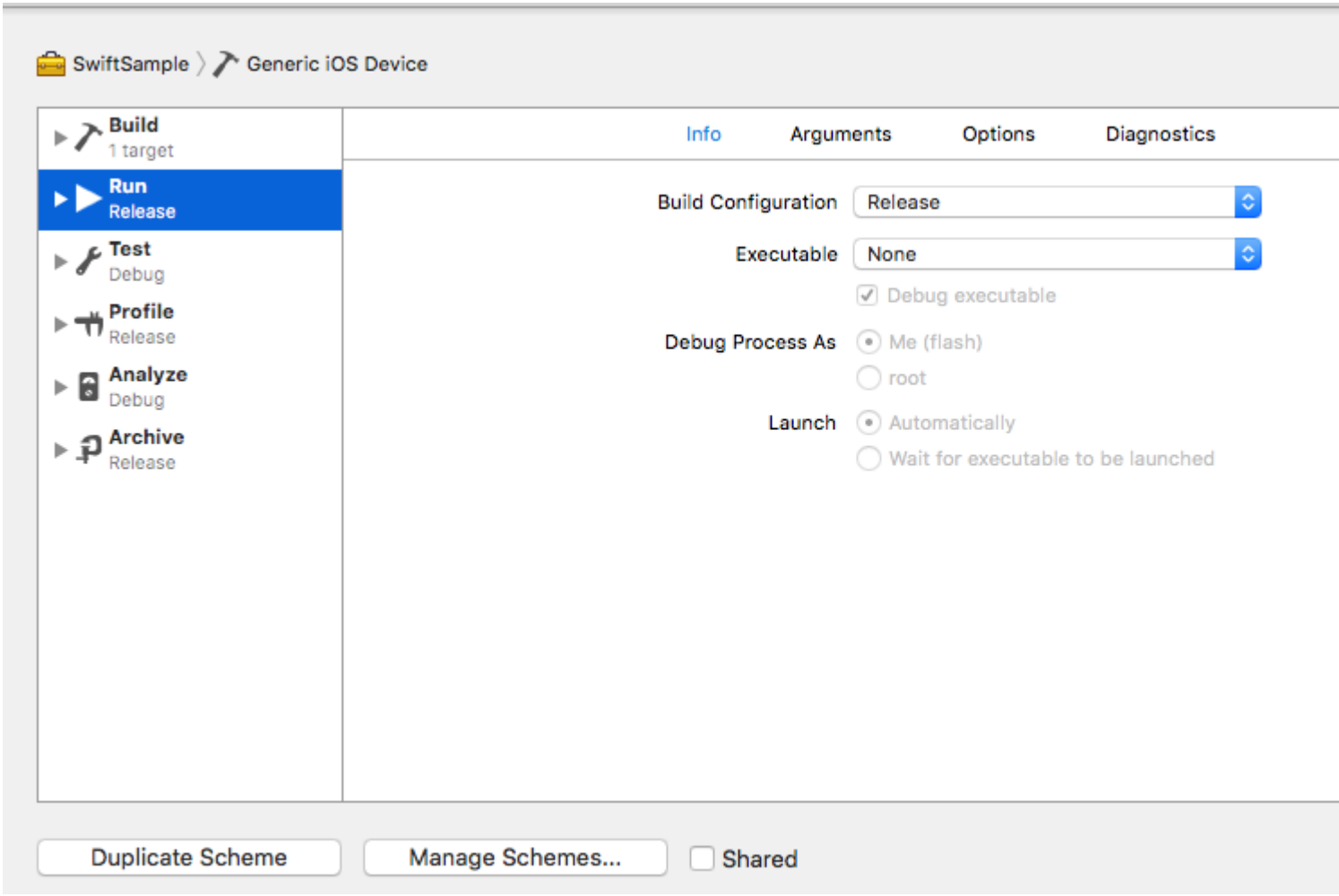
```
//Add this to specify explicit objective c name
@objc(MyClass)
open class MyClass: NSObject {
    open func getValue() -> String
    {
        return "Value came from MyClass.swift!";
    }
}
```

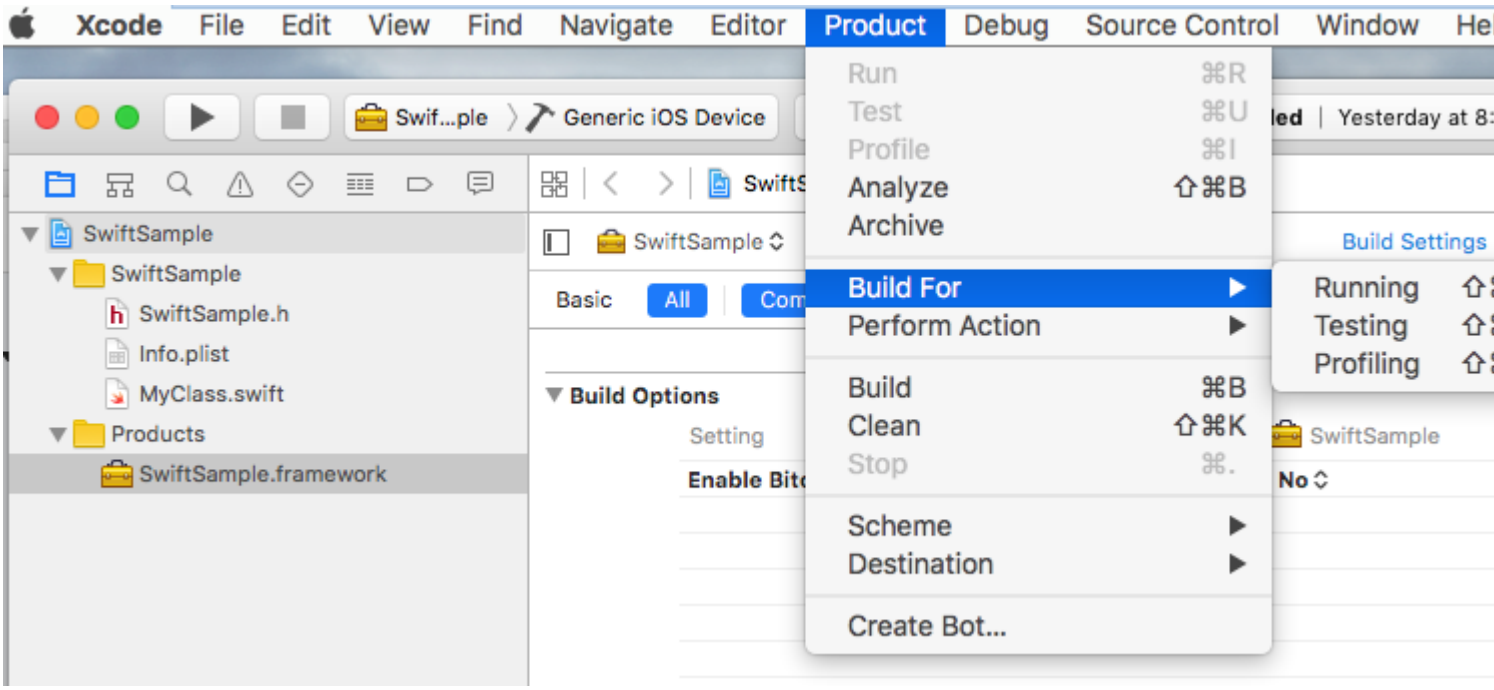
1.2 Construir el marco

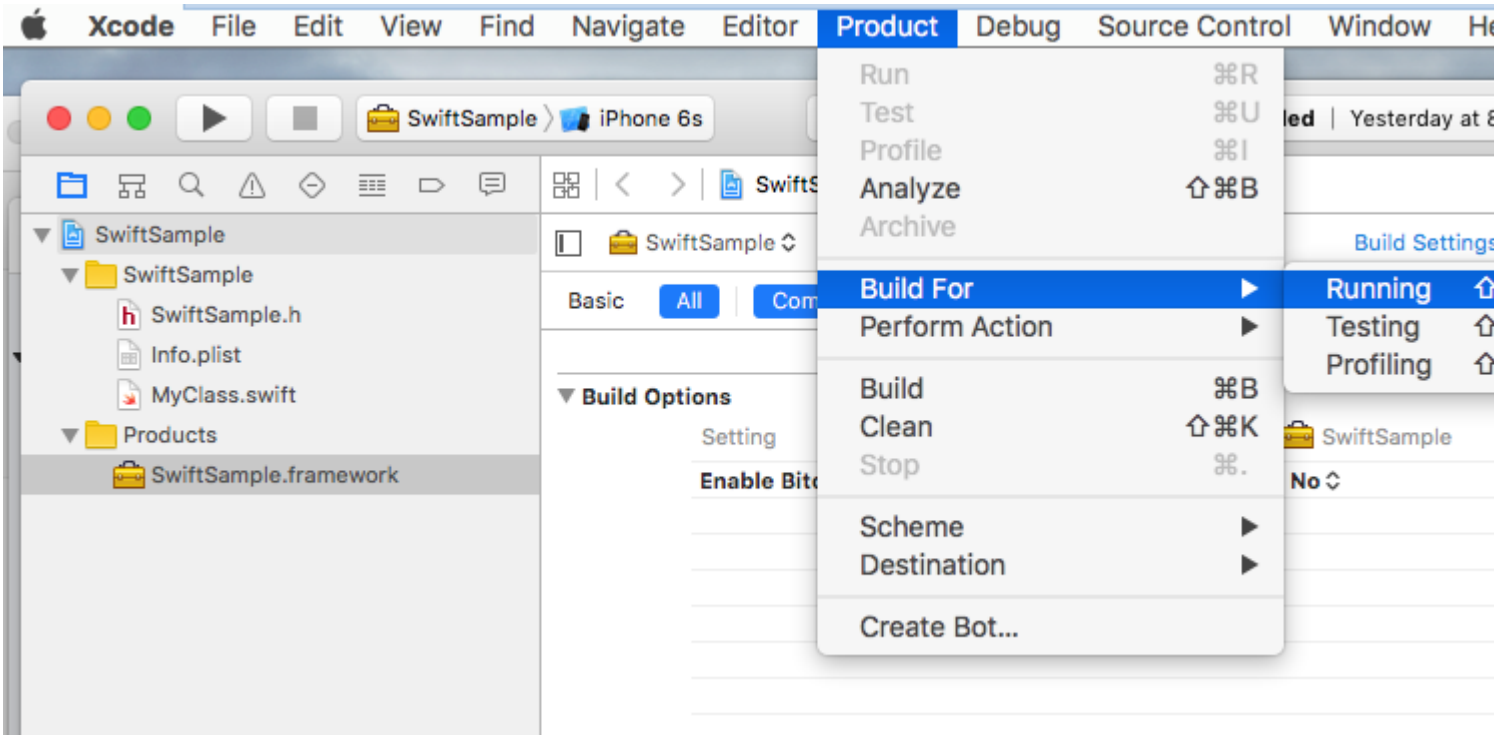
Desactivar el código de bits. *

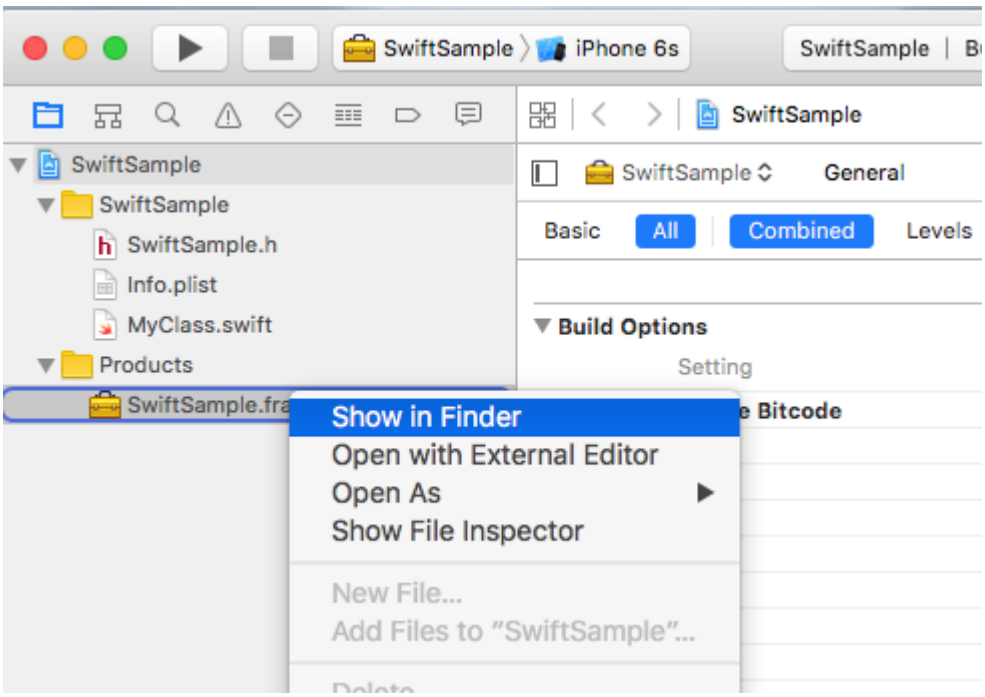


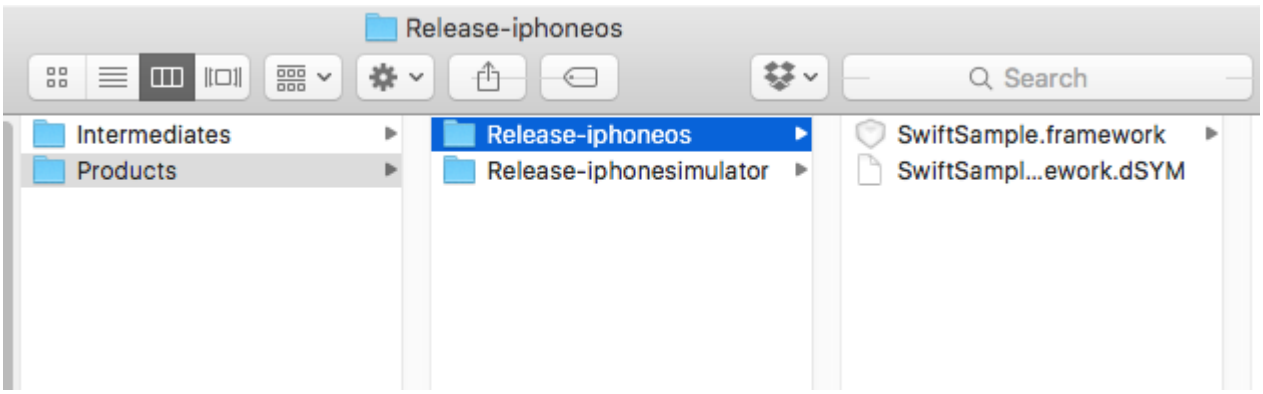
Construir para el lanzamiento de dispositivo y simulador. *


SwiftSample
Generic iOS Device
Build
1 target
Run
Release
Test
Debug
Profile
Release
Analyze
Debug
Archive
Release
Info
Arguments
Options
Diagnostics
Build Configuration
Release
Executable
None
Debug executable
Debug Process As
Me (flash)
root
Launch
Automatically
Wait for executable to be launched
Duplicate Scheme
Manage Schemes...
Shared


Xcode
File
Edit
View
Find
Navigate
Editor
Product
Debug
Source Control
Window
Run ⌘R
Test ⌘U
Profile ⌘I
Analyze ⇧⌘B
Archive
Build For
Perform Action
Build ⌘B
Clean ⇧⌘K
Stop ⌘.
Scheme
Destination
Create Bot...
Swif...ple
Generic iOS Device
SwiftSample
SwiftSample.h
Info.plist
MyClass.swift
Products
SwiftSample.framework
Basic
All
Build Options
Setting
Build Settings
Running
Testing
Profiling





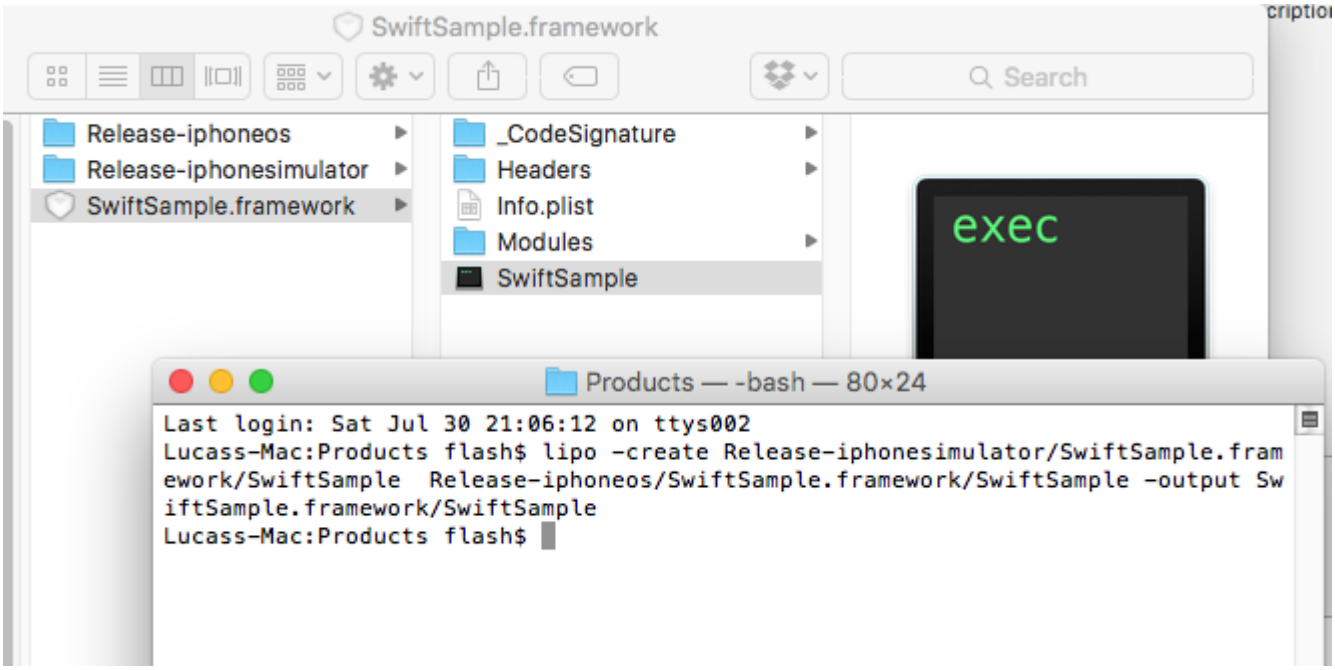


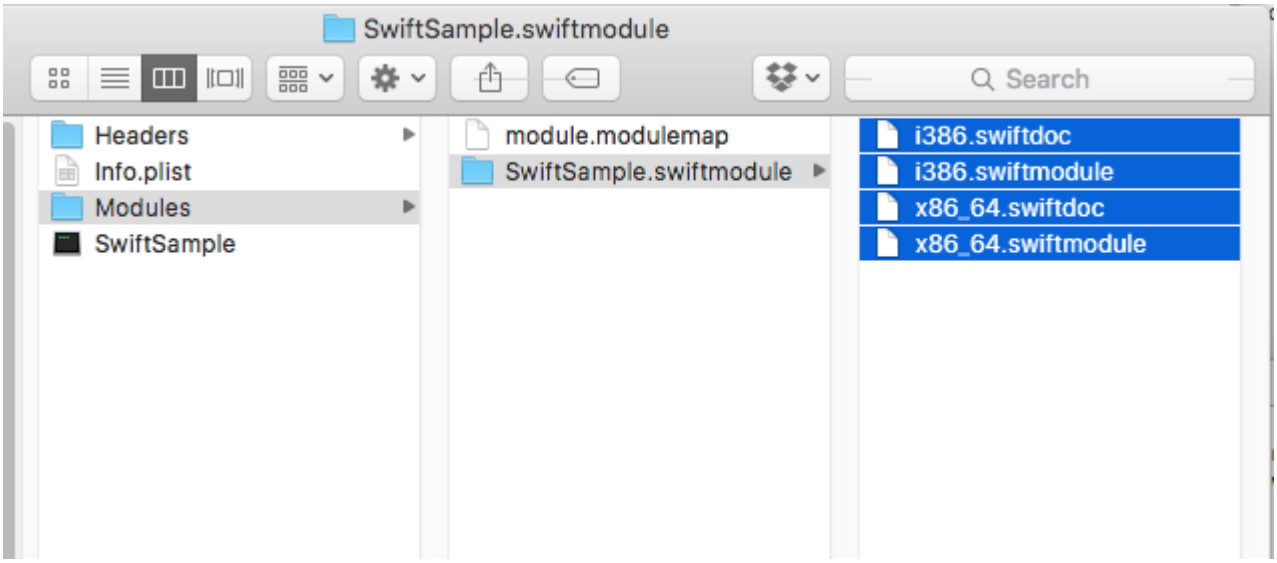
- No relacionado solo con Swift binding.

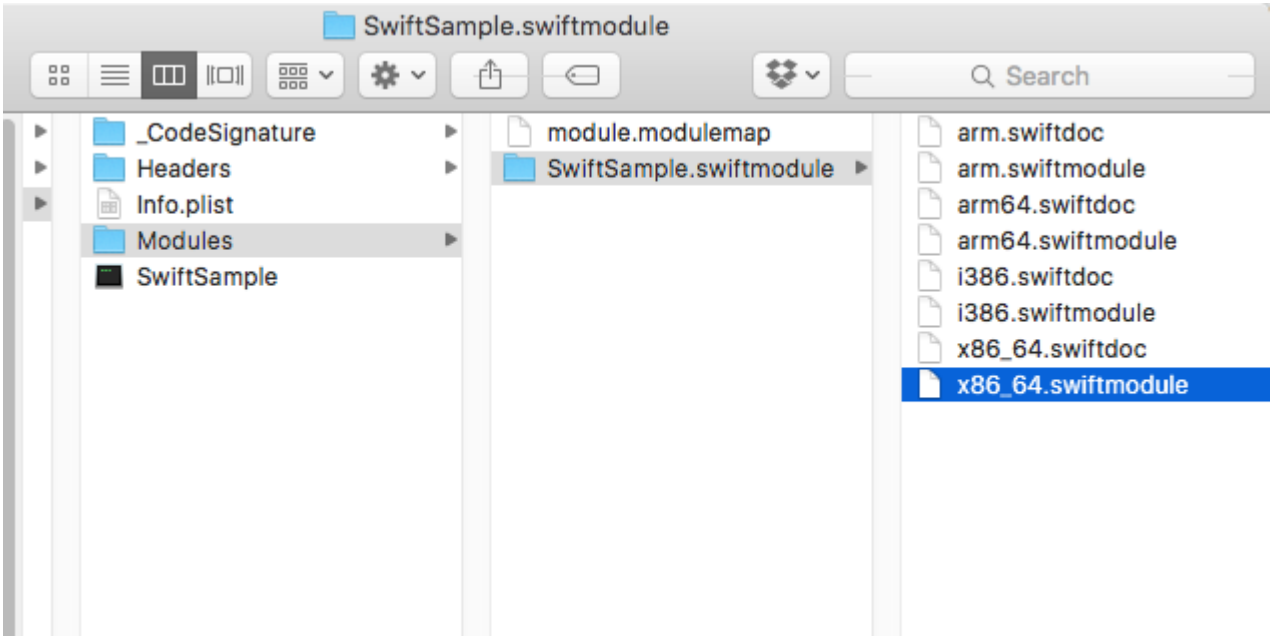
2. Crear una biblioteca de grasa

Un marco contiene varios archivos, el que necesita comer un poco es NAME.framework / NAME (sin extensión).

- Copie Release-iphoneos / NAME.framework a NAME.framework
- Crea la biblioteca FAT usando:
    - **lipo -create Release-iphonesimulator / NAME.framework / NAME Release-iphoneos / NAME.framework / NAME -output NAME.framework / NAME**
- Copie los archivos en Release-iphonesimulator / NAME.framework / Modules / NAME.swiftmodule en NAME.framework / Modules / NAME.swiftmodule (hasta ahora solo contenía archivos del iPhoneOS)



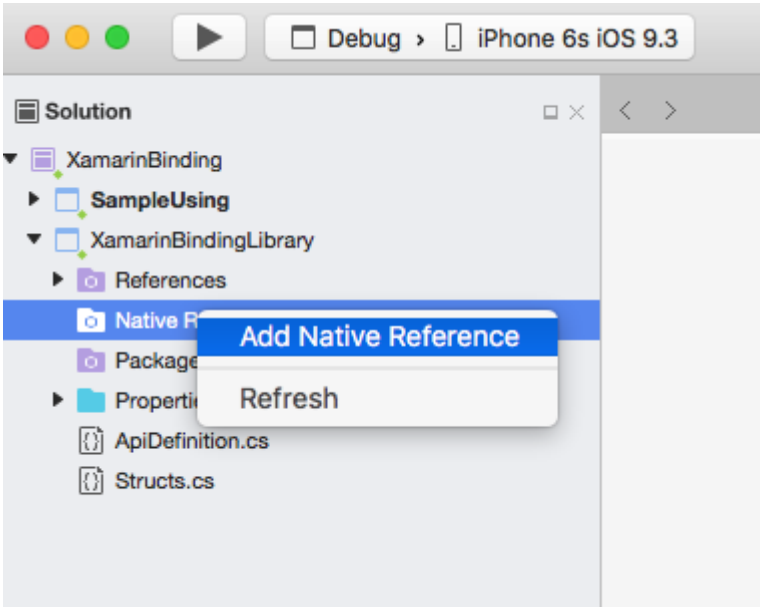


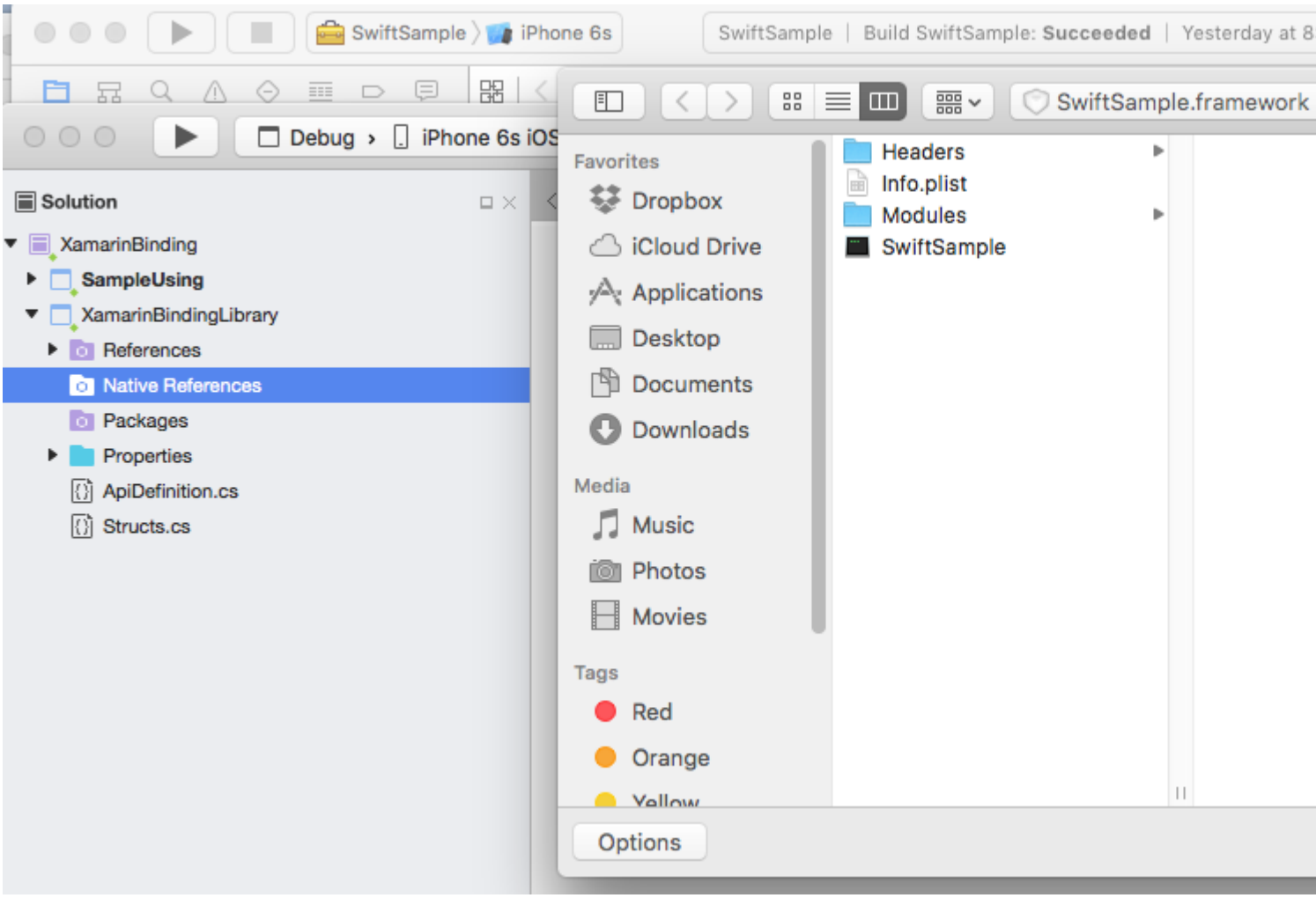


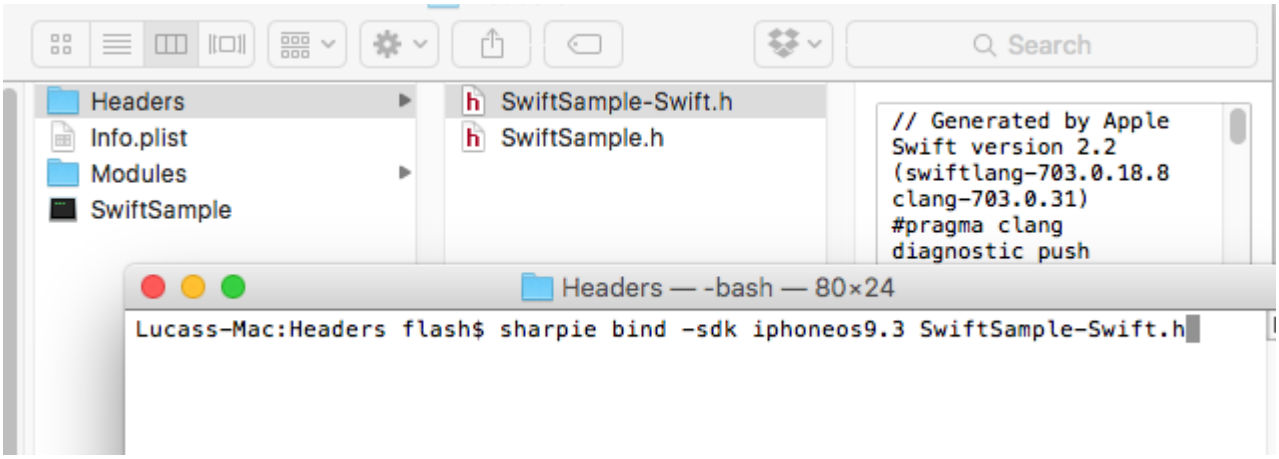
3. Importar la biblioteca

Asumiré que ya creó el proyecto de enlace en Archivo -> Nuevo -> iOS -> Biblioteca de enlaces.

Soporte para Xamarin importando .frameworks. Simplemente haga clic derecho en 'Referencias nativas' y haga clic en 'Agregar referencia nativa'. Encuentra el marco de grasa recién creado y agregarlo.







4. Cree la ApiDefinition basada en el archivo LIBRARY-Swift.h dentro de los encabezados.

Puedes hacerlo manualmente, pero no estará bien. Puedes usar Objetivo Sharpie. La herramienta que Xamarin utiliza para enlazar sus propias bibliotecas.

Cómo usarlo en https://developer.xamarin.com/guides/cross-platform/macios/binding/objective-sharpie/

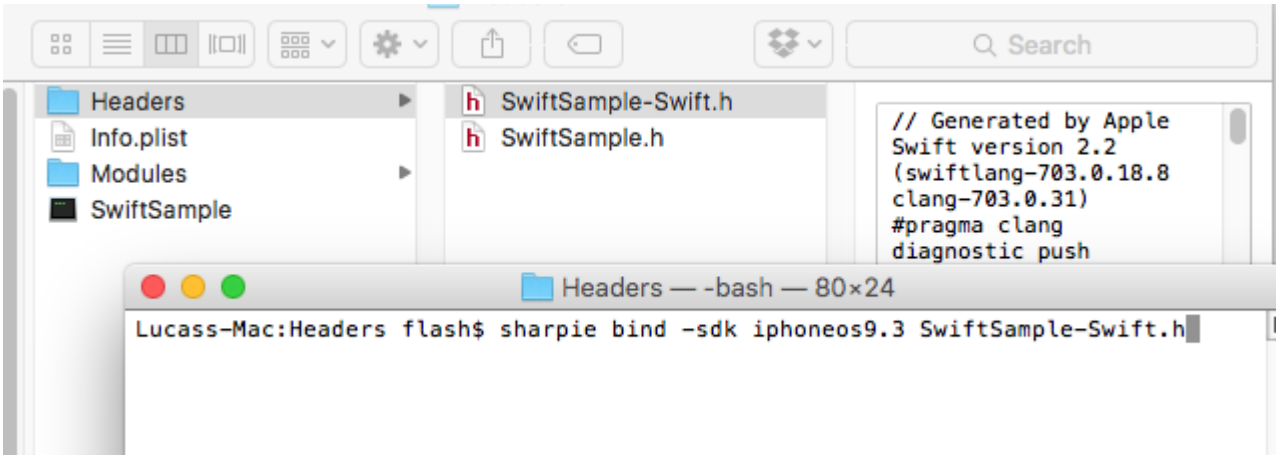
El comando básico será algo así como: ***sharpie bind -sdk iphoneos9.3 NAME-Swift.h***

Si obtiene una `System.Reflection.TargetInvocationException` probablemente se deba a que tiene instalada una versión diferente de SDK. Ejecute el siguiente comando para verificar con el iPhone OS SDK que ha instalado:

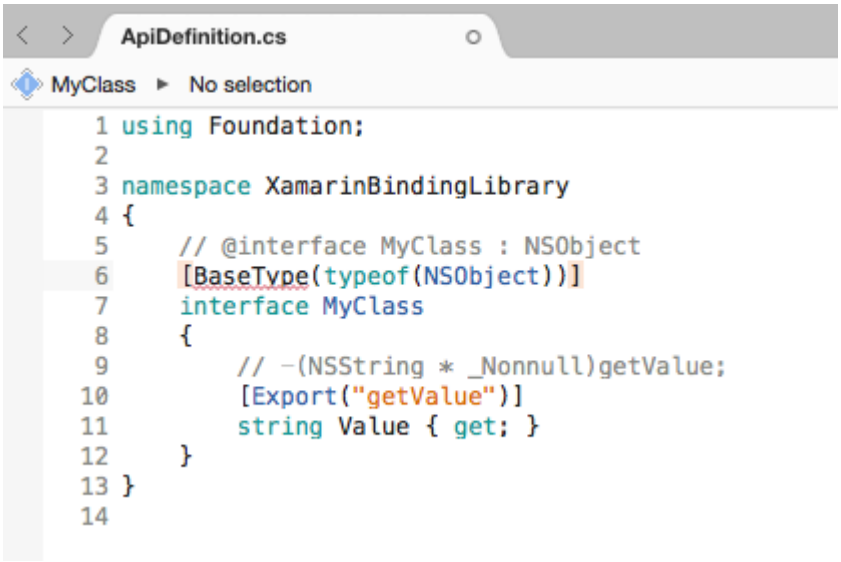
```
sharpie xcode -sdks
```

El archivo ***NAME-Swift.h*** se encuentra en ***NAME.framework / Headers / NAME-Swift.h***

Nota: las clases swift deben heredarse de "NSObject", de lo contrario, ***NAME-Swift.h*** no importará sus clases y Objetivo Sharpie no convertirá nada.



Reemplace los contenidos de su proyecto de enlace ApiDefinition.cs con el recién creado.



5. Cambie todos [Protocolo] y [Tipo de base] para incluir el nombre de la clase en el tiempo de ejecución de Objective-C.

En caso de que la clase o el protocolo Swift original no incluya la anotación @objc (MyClass) como se especifica en el paso 1.1, se cambiarán sus nombres internos de Objective-C, por lo que debe asignarlos al correcto.

Todos los nombres están disponibles en el archivo NAME-Swift.h en el siguiente formato:

```
SWIFT_CLASS("_TtC11SwiftSample7MyClass")
@interface MyClass : NSObject
```

Y

```
SWIFT_PROTOCOL("_TtP6Charts17ChartDataProvider_")
@protocol ChartDataProvider
```

Para establecer el nombre, use BaseTypeAttribute.Name https://developer.xamarin.com/guides/cross-platform/macios/binding/binding-types-reference/#BaseType.Name propiedad para las clases y ProcotolAttribute.Name https: // developer.xamarin.com/api/property/MonoTouch.Foundation.ProtocolAttribute.Name/ para protocolos.

```
[BaseType(typeof(NSObject), Name = "_TtC11SwiftSample7MyClass")]
interface MyClass
```

Hacerlo manualmente no está bien. Puede utilizar esta herramienta https://github.com/Flash3001/SwiftClassify para insertar todos los nombres. (Es un trabajo en progreso. Pero es bastante simple, solo con mirar el código obtendrá cómo funciona).

6.1 Incluir todas las dependencias Swift para ejecutar.

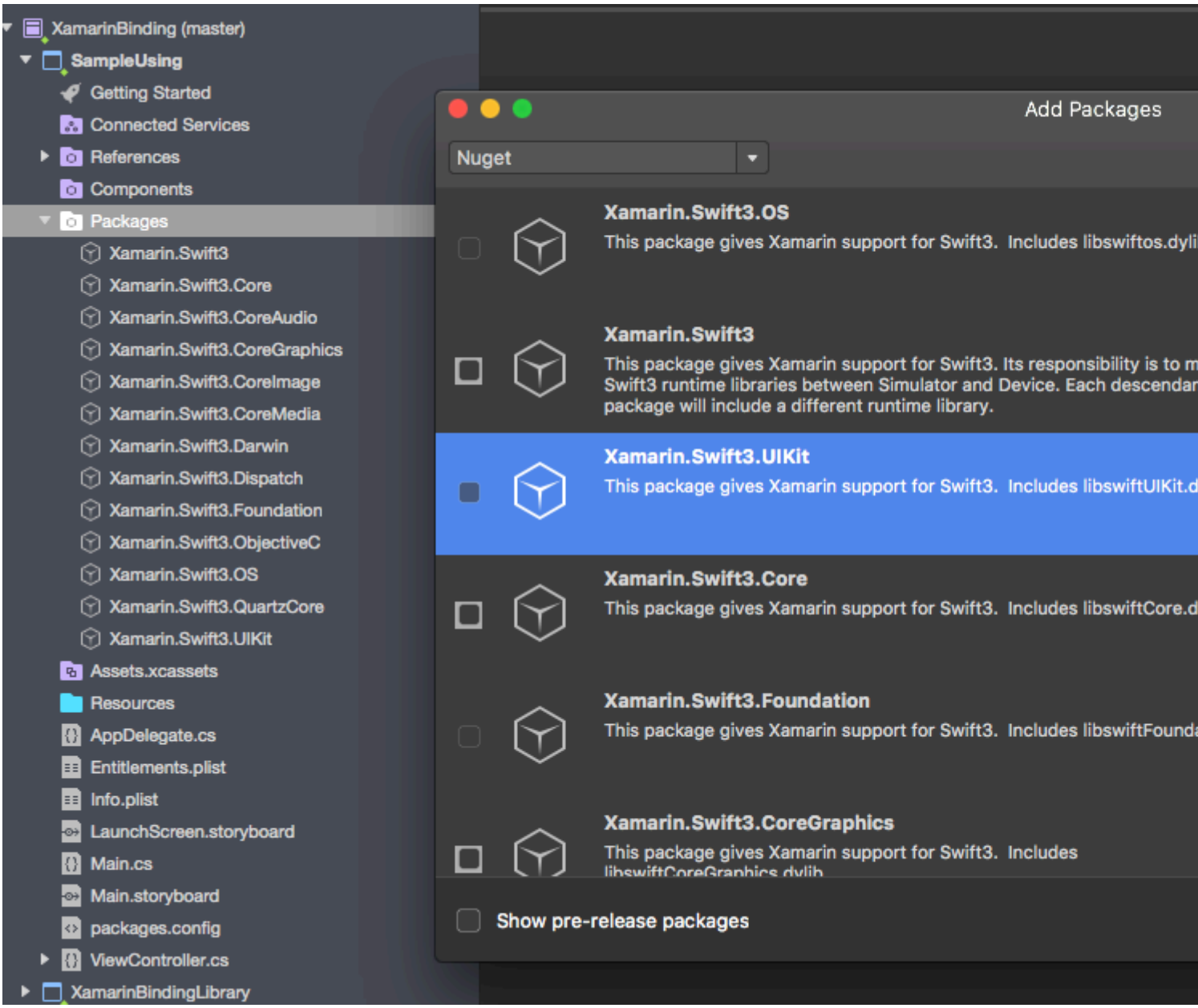
Si intenta consumir la biblioteca en una aplicación e intenta ejecutarla ahora mismo, se bloqueará. El error se debe a la falta de libswiftCore.dylib

Algo como esto:

```
Dyld Error Message:
  Library not loaded: @rpath/libswiftCore.dylib
  Referenced from: /Users/USER/Library/Developer/CoreSimulator/Devices/AC440891-C819-4050-
8CAB-CE15AB4B3830/data/Containers/Bundle/Application/27D2EC87-5042-4FA7-9B80-
A24A8971FB48/SampleUsing.app/Frameworks/SwiftSample.framework/SwiftSample
  Reason: image not found
```

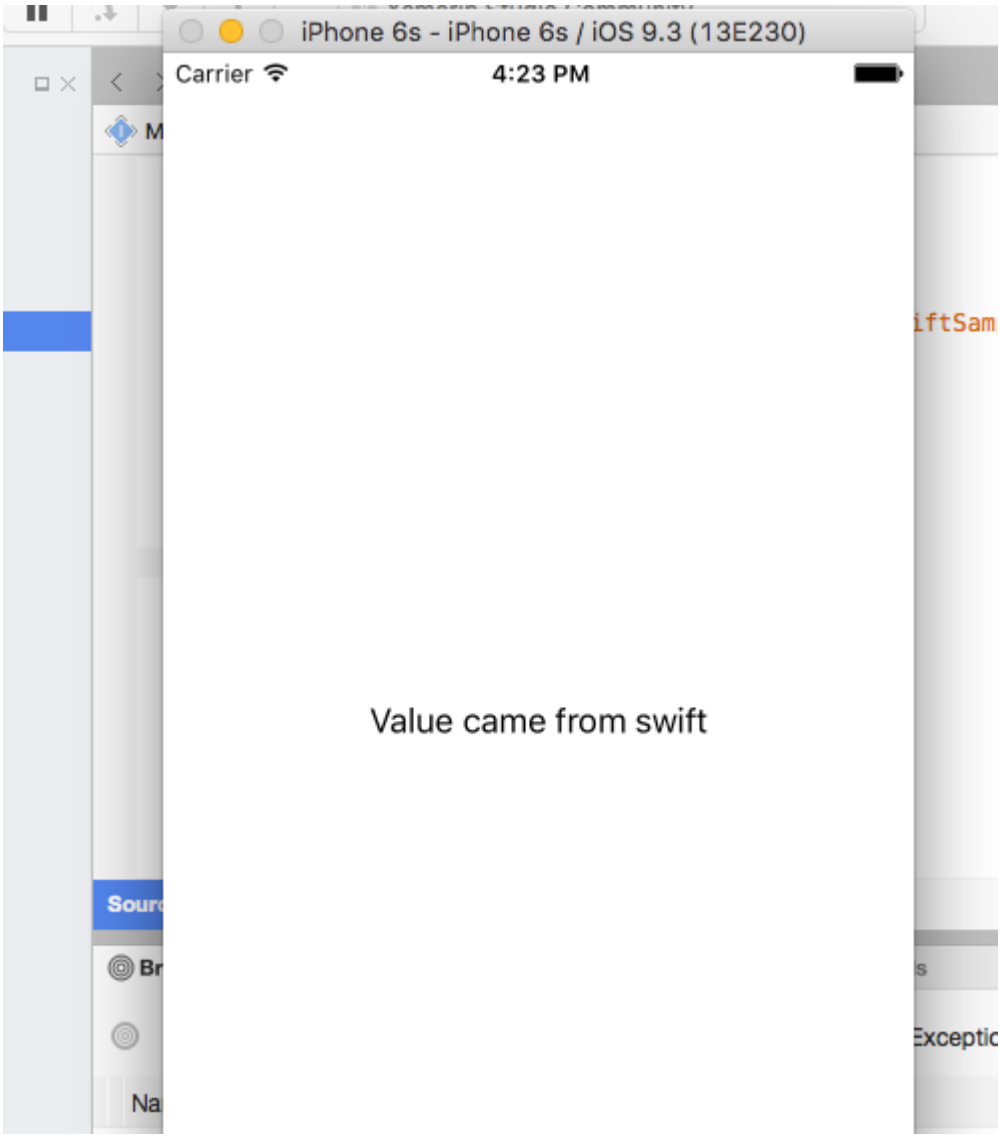
Xamarin.iOS no ofrece soporte oficial para vincular una biblioteca Swift. Por lo tanto, debe incluir manualmente las bibliotecas de Swift Core en las carpetas Frameworks y SwiftSupport. Los archivos para la carpeta Frameworks son diferentes para Simulator y Device. Se pueden encontrar en /Applications/Xcode.app/Contents/Developer//XcodeDefault.xctoolchain/usr/lib/swift.Toolchains

En lugar de copiar manualmente los archivos dentro de la carpeta de Framework, puede usar esta biblioteca https://github.com/Flash3001/Xamarin.Swift3.Support . Incluye cada una de las necesidades de Swift 3.1, cada una en un solo paquete NuGet.



Como puede ver, el paquete Nuget está incluido en la aplicación del consumidor, no en el enlace en sí. Si intentas incluirlo en el enlace obtendrás errores de compilación.

Si está creando un paquete Nuget, puede indicar a Nuget que lo incluya como una dependencia.



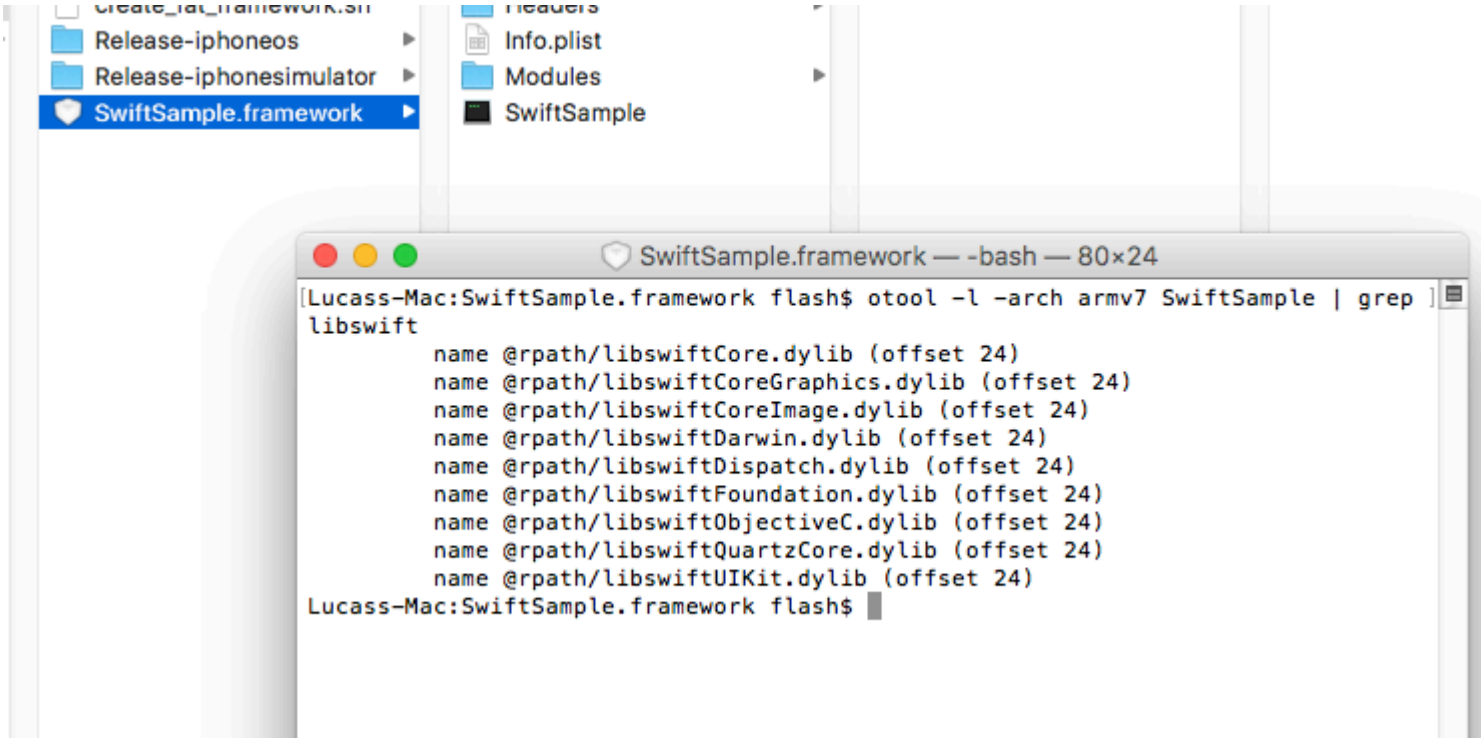
6.2. Averiguar qué dependencias Swift incluir.

Una cosa importante que hacer es averiguar cada paquete que necesita incluir en su proyecto. Un enlace simple usualmente necesitará:

```
    libswiftCore.dylib
    libswiftCoreGraphics.dylib
    libswiftCoreImage.dylib
    libswiftDarwin.dylib
    libswiftDispatch.dylib
    libswiftFoundation.dylib
    libswiftObjectiveC.dylib
    libswiftQuartzCore.dylib
    libswiftUIKit.dylib
```

Para enumerar cada dependencia, puede ejecutar el siguiente comando dentro de LibraryName.framework

```
otool -l -arch armv7 LibraryName | grep libswift
```



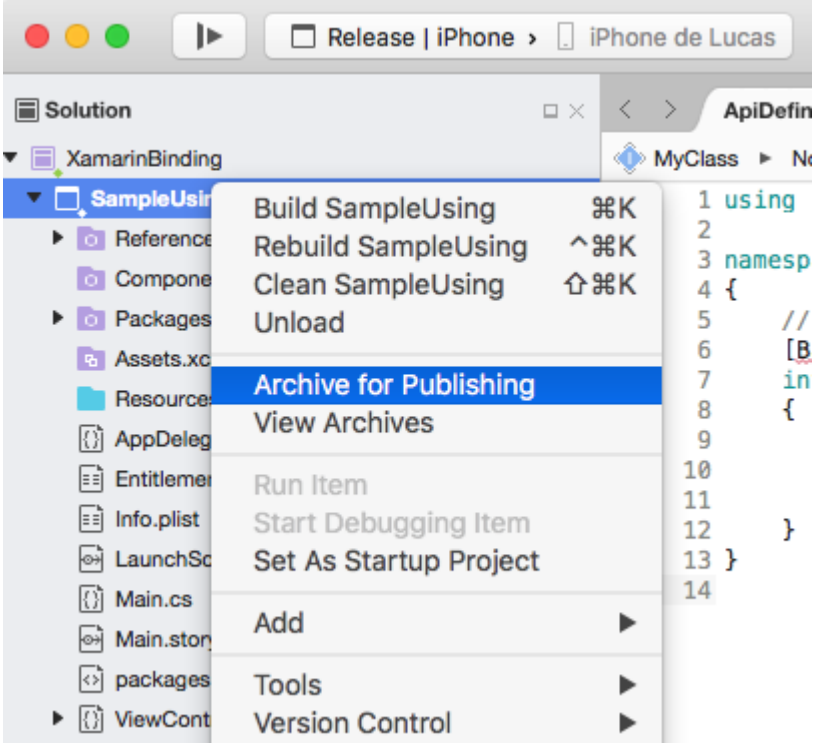
No incluya todos los paquetes disponibles en NuGet para Swift3, ya que pueden aumentar el tamaño de su aplicación.

7. Incluya SwiftSupport para llevar la aplicación a AppStore.

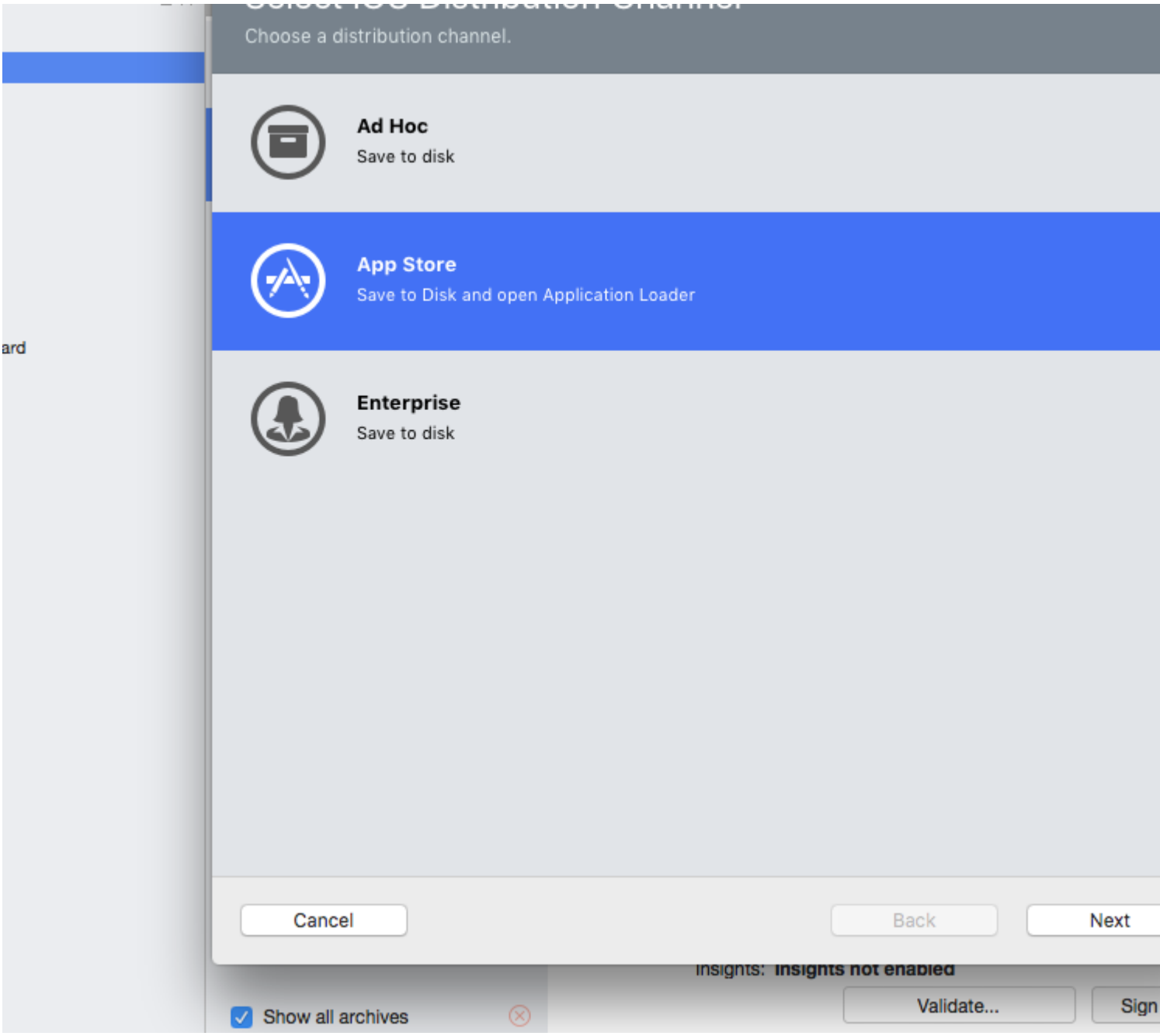
Apple requiere que su aplicación se envíe con una carpeta SwiftSupport junto con su carpeta de carga útil. Ambos están dentro de su paquete de IPA.

Puede usar este script https://github.com/bq/ipa-packager para hacer este trabajo por usted.

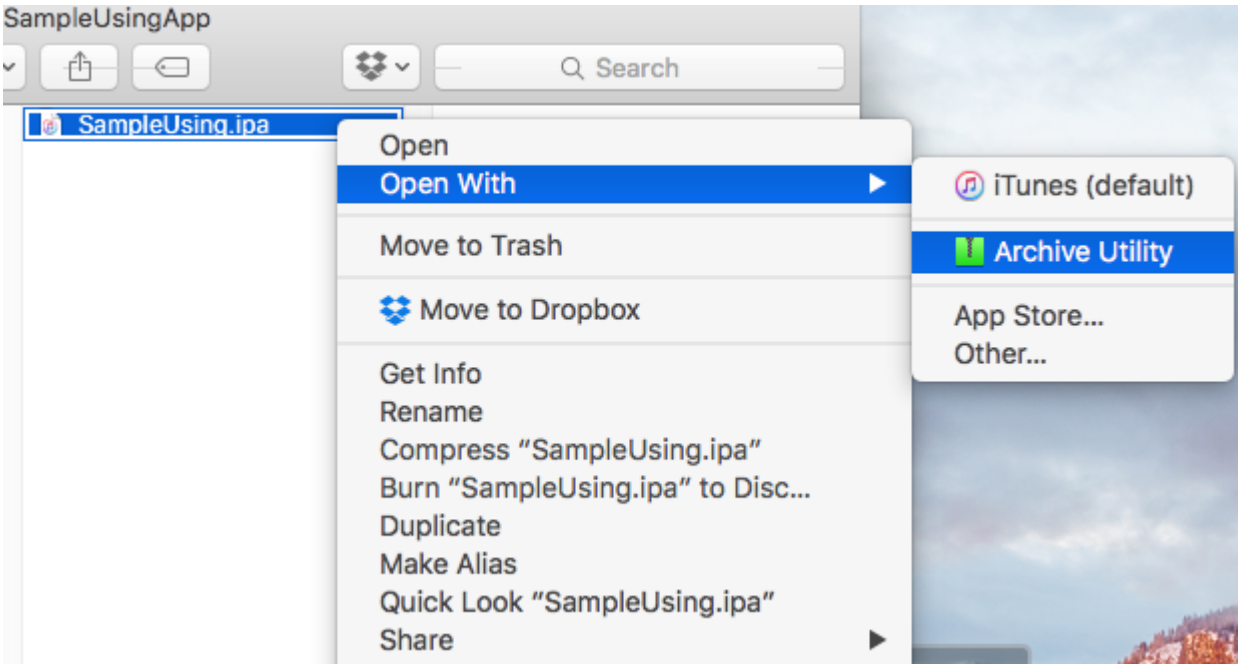
Este proceso es el único que el consumidor de la biblioteca tendrá que hacer manualmente. Cada vez que él / ella intenta empujar la aplicación a AppStore.



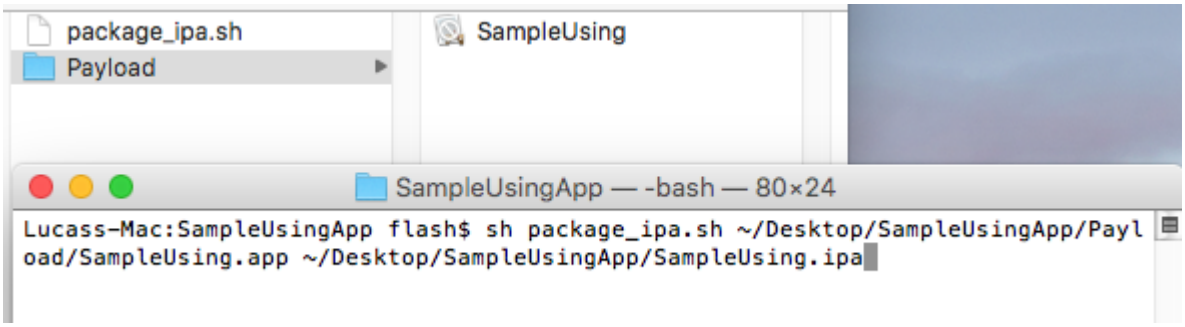
Haga clic en 'Firmar y distribuir' y guardar en disco



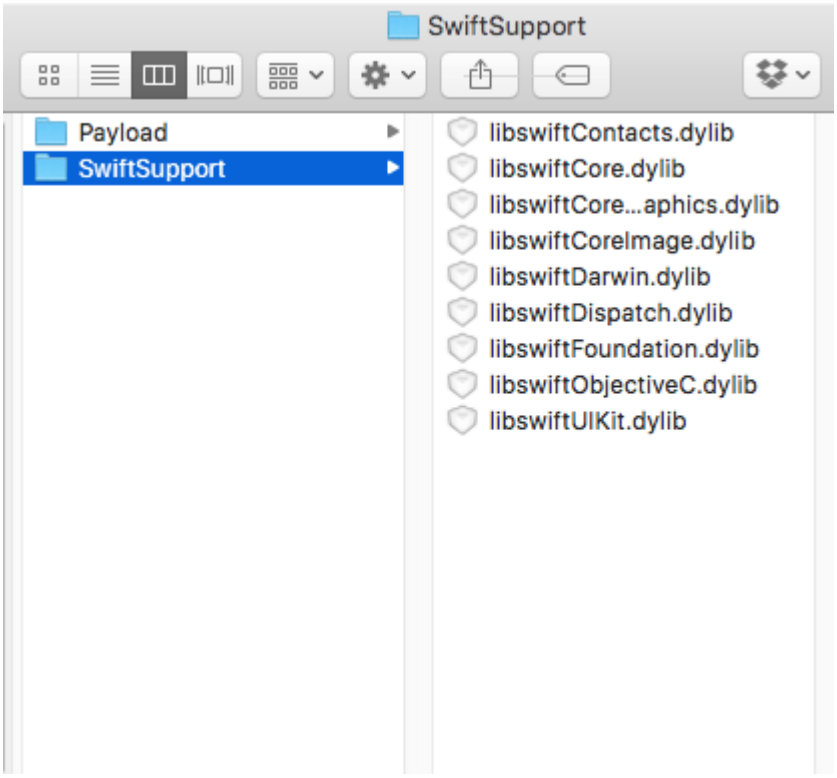
Descomprime tu .IPA



Crea la nueva API usando el script antes mencionado.



Si descomprime el archivo, este contendrá la carpeta SwiftSupport.



Observaciones

Cuando se construye una biblioteca en Xcode, tiene una opción para incluir las bibliotecas swift. No lo hagas Se incluirán en su aplicación final como NAME.app/Frameworks/LIBRARY.framework/Frameworks/libswift*.dylib pero deben incluirse como NAME.app/Frameworks/libswift*.dylib

Puede encontrar esta información en otra parte, pero vale la pena mencionarlo: no incluya Bitcode en la biblioteca. A partir de este momento, Xamarin no incluye Bitcode para iOS y Apple requiere que todas las bibliotecas admitan las mismas arquitecturas.

Renuncia

Esta guía fue creada originalmente por Lucas Teixeira . Todos los créditos le pertenecen. Gracias, Lucas.

Lea Encuadernación de bibliotecas rápidas en línea: https://riptutorial.com/es/xamarin-ios/topic/6091/encuadernacion-de-bibliotecas-rapidas

Capítulo 13: identifición de toque

Parámetros

Columna	Columna
Célula	Célula

Observaciones

Primero, establezca si el dispositivo es capaz de aceptar la entrada Touch ID.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
```

Si lo hace, entonces podemos mostrar la interfaz de usuario de Touch ID utilizando:

```
context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
```

Hay tres datos que debemos pasar a `EvaluatePolicy` : la política en sí misma, una cadena que explica por qué es necesaria la autenticación y un controlador de respuestas. El controlador de respuestas le dice a la aplicación lo que debe hacer en el caso de una autenticación exitosa o no exitosa.

Una de las advertencias de la autenticación local es que debe ejecutarse en primer plano, así que asegúrese de usar `InvokeOnMainThread` para el controlador de respuesta:

```
var replyHandler = new LAContextReplyHandler((success, error) =>
    {

        this.InvokeOnMainThread(() =>
        {
            if (success)
            {
                Console.WriteLine("You logged in!");
                PerformSegue("AuthenticationSegue", this);
            }
            else {
                //Show fallback mechanism here
            }
        });

    });
```

Para determinar si la base de datos de huellas dactilares autorizadas se ha modificado, puede verificar la estructura opaca (NSData) que devuelve `context.EvaluatedPolicyDomainState` . Su aplicación deberá almacenar y comparar el estado de la política para detectar cambios. Una cosa

a tener en cuenta que Apple dice:

> Sin embargo, la naturaleza del cambio no se puede determinar a partir de estos datos.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
{
    var policyState = context.EvaluatedPolicyDomainState;

    var replyHandler = new LAContextReplyHandler((success, error) =>
    {

        this.InvokeOnMainThread(() =>
        {
            if (success)
            {
                Console.WriteLine("You logged in!");
                PerformSegue("AuthenticationSegue", this);
            }
            else {
                //Show fallback mechanism here
            }
        });

    });
    context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
};
```

Examples

Añade Touch ID a tu aplicación

Primero, establezca si el dispositivo es capaz de aceptar la entrada Touch ID.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
```

Si lo hace, entonces podemos mostrar la interfaz de usuario de Touch ID utilizando:

```
context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
```

Hay tres datos que debemos pasar a `EvaluatePolicy` : la política en sí misma, una cadena que explica por qué es necesaria la autenticación y un controlador de respuestas. El controlador de respuestas le dice a la aplicación lo que debe hacer en el caso de una autenticación exitosa o no exitosa.

Una de las advertencias de la autenticación local es que debe ejecutarse en primer plano, así que asegúrese de usar `InvokeOnMainThread` para el controlador de respuesta:

```
var replyHandler = new LAContextReplyHandler((success, error) =>
```

```
{
    this.InvokeOnMainThread(() =>
    {
        if (success)
        {
            Console.WriteLine("You logged in!");
            PerformSegue("AuthenticationSegue", this);
        }
        else {
            //Show fallback mechanism here
        }
    });
});
```

Para determinar si la base de datos de huellas dactilares autorizadas se ha modificado, puede verificar la estructura opaca (NSData) que devuelve `context.EvaluatedPolicyDomainState` . Su aplicación deberá almacenar y comparar el estado de la política para detectar cambios. Una cosa a tener en cuenta que Apple dice:

> Sin embargo, la naturaleza del cambio no se puede determinar a partir de estos datos.

```
if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
{
    var policyState = context.EvaluatedPolicyDomainState;

    var replyHandler = new LAContextReplyHandler((success, error) =>
    {

        this.InvokeOnMainThread(() =>
        {
            if (success)
            {
                Console.WriteLine("You logged in!");
                PerformSegue("AuthenticationSegue", this);
            }
            else {
                //Show fallback mechanism here
            }
        });

    });
    context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);
};
```

Ejemplo de botón

```
partial void AuthenticateMe(UIButton sender)
{
    var context = new LAContext();
    //Describes an authentication context
    //that allows apps to request user authentication using Touch ID.
    NSError AuthError;
    //create the reference for error should it occur during the authentication.
    var myReason = new NSString("To add a new chore");
    //this is the string displayed at the window for touch id
```

```
    if (context.CanEvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, out
AuthError))
    // check if the device have touchId capabilities.
    {
        var replyHandler = new LAContextReplyHandler((success, error) =>
        {

            this.InvokeOnMainThread(() =>
            {
                if (success)
                {
                    Console.WriteLine("You logged in!");
                    PerformSegue("AuthenticationSegue", this);
                }
                else {
                    //Show fallback mechanism here
                }
            });

        });
        context.EvaluatePolicy(LAPolicy.DeviceOwnerAuthenticationWithBiometrics, myReason,
replyHandler);//send touch id request
    };
}
```

Usando llavero

Fuente de trabajo: https://github.com/benhysell/V.TouchIdExample

Descripción larga de la forma: http://benjaminhysell.com/archive/2014/11/authentication-in-xamarin-ios-with-touch-id-or-passcode/

```
//Simple View with a switch to enable / disable Touch ID and
//a button to invoke authentication

/// <summary>
/// Enable/Disable Touch ID
/// </summary>
/// <param name="sender">Sender.</param>
partial void TouchIdEnableDisable(UISwitch sender)
{
    if (sender.On)
    {
        //enable Touch ID
        //set our record
        //note what you fill in here doesn't matter, just needs to be
        //consistent across all uses of the record
        var secRecord = new SecRecord(SecKind.GenericPassword)
        {
            Label = "Keychain Item",
            Description = "fake item for keychain access",
            Account = "Account",
            Service = "com.yourcompany.touchIdExample",
            Comment = "Your comment here",
            ValueData = NSData.FromString("my-secret-password"),
            Generic = NSData.FromString("foo")
        };
```

```
        secRecord.AccessControl = new
SecAccessControl(SecAccessible.WhenPasscodeSetThisDeviceOnly,
SecAccessControlCreateFlags.UserPresence);
        SecKeyChain.Add(secRecord);

        authenticateButton.Enabled = true;
    }
    else
    {
        //disable Touch ID
        var record = new SecRecord(SecKind.GenericPassword)
        {
            Service = "com.yourcompany.touchIdExample",
            UseOperationPrompt = "Authenticate to Remove Touch ID / Passcode from Test App"
        };

        SecStatusCode result;

        //query one last time to ensure they can remove it
        SecKeyChain.QueryAsRecord(record, out result);
        if (SecStatusCode.Success == result || SecStatusCode.ItemNotFound == result)
        {
            //remove the record
            SecKeyChain.Remove(record);
            authenticateButton.Enabled = false;
        }
        else
        {
            //could not authenticate, leave switch on
            sender.On = true;
         }
    }
}

/// <summary>
/// Show Touch ID to user and evaluate authentication
/// </summary>
/// <param name="sender">Sender.</param>
partial void AuthenticateUser(UIButton sender)
{
    var rec = new SecRecord(SecKind.GenericPassword)
    {
        Service = "com.yourcompany.touchIdExample",
        UseOperationPrompt = "Authenticate to access Test App"
    };
    SecStatusCode res;
    SecKeyChain.QueryAsRecord(rec, out res);
    if (SecStatusCode.Success == res || SecStatusCode.ItemNotFound == res)
    {
        //Success!!
        //add your code here to continue into your application
        AuthenticatedLabel.Hidden = false;
    }
    else
    {
        //Failure
        AuthenticatedLabel.Hidden = true;
    }
}
```

Lea identifición de toque en línea: https://riptutorial.com/es/xamarin-ios/topic/577/identificion-de-toque

Capítulo 14: Las alertas

Examples

Mostrar una alerta

Para Alertas desde iOS 8, usaría un `UIAlertController` pero para las versiones anteriores, habría usado un `UIAlertView` , que ahora está en desuso.

8.0

```
var alert = UIAlertController.Create(title, message, UIAlertControllerStyle.Alert);
alert.AddAction(UIAlertAction.Create(otherTitle, UIAlertActionStyle.Destructive, (action) => {
    // otherTitle();
}));
alert.AddAction(UIAlertAction.Create(cancelTitle, UIAlertActionStyle.Cancel, null));
this.PresentViewController(alert, true, null);
```

8.0

```
var alert = new UIAlertView (title, message, null, cancelTitle, otherTitle);
alert.Clicked += (object sender, UIButtonEventArgs e) => {
    if(e.ButtonIndex == 1)
       // otherTitle();
};
alert.Show ();
```

Mostrar una alerta de inicio de sesión

El siguiente código es para iOS 8 e inferior para crear una alerta de inicio de sesión.

```
// Create the UIAlertView
var loginAlertView = new UIAlertView(title, message, null, cancelTitle, okTitle);

// Setting the AlertViewStyle to UIAlertViewStyle.LoginAndPasswordInput
loginAlertView.AlertViewStyle = UIAlertViewStyle.LoginAndPasswordInput;

// Getting the fields Username and Password
var usernameTextField = loginAlertView.GetTextField(0);
var passwordTextField = loginAlertView.GetTextField(1);

// Setting a placeholder
usernameTextField.Placeholder = "user@stackoverflow.com";
passwordTextField.Placeholder = "Password";

// Adding the button click handler.
loginAlertView.Clicked += (alertViewSender, buttonArguments) =>
{

    // Check if cancel button is pressed
    if (buttonArguments.ButtonIndex == loginAlertView.CancelButtonIndex)
    {
        // code
```

```
    }

    // In our case loginAlertView.FirstOtherButtonIndex is equal to the OK button
    if (buttonArguments.ButtonIndex == loginAlertView.FirstOtherButtonIndex)
    {
        // code
    }
};

// Show the login alert dialog
loginAlertView.Show();
```

Mostrar una hoja de acción

El `UIAlertController` disponible desde iOS8 le permite usar el mismo objeto de alerta para hojas de acción o para alertas más clásicas. La única diferencia es el `UIAlertControllerStyle` pasado como parámetro al crear.

Esta línea cambia de un AlertView a un ActionSheet, en comparación con algunos otros ejemplos disponibles aquí:

```
var alert = UIAlertController.Create(title, message, UIAlertControllerStyle.ActionSheet);
```

La forma en que agrega acciones al controlador sigue siendo la misma:

```
alert.AddAction(UIAlertAction.Create(otherTitle, UIAlertActionStyle.Destructive, (action) => {
    // ExecuteSomeAction();
}));
alert.AddAction(UIAlertAction.Create(cancelTitle, UIAlertActionStyle.Cancel, null));

//Add additional actions if necessary
```

Tenga en cuenta que si tiene un método de anulación sin parámetros, puede usarlo como el último parámetro de `.AddAction()` .

Por ejemplo, supongamos que quiero que se ejecute el código de `private void DoStuff(){...}` cuando `private void DoStuff(){...}` "Aceptar":

```
UIAlertAction action = UIAlertAction.Create("OK", UIAlertActionStyle.Cancel, DoStuff);
alert.AddAction(action);
```

Note que no estoy usando el () después de DoStuff en la creación de la acción.

La forma en que presenta el controlador se realiza de la misma manera que cualquier otro controlador:

```
this.PresentViewController(alert, true, null);
```

Mostrar el diálogo de alerta modal

Era una práctica común usar `NSRunLoop` para mostrar `UIAlertView` modal para bloquear la ejecución

del código hasta que la entrada del usuario se procesa en iOS; Hasta que Apple lanzó el iOS7, rompió algunas aplicaciones existentes. Afortunadamente, hay una mejor manera de implementarlo con async / await de C #.

Aquí está el nuevo código que aprovecha el patrón async / await para mostrar UIAlertView modal:

```
Task ShowModalAletViewAsync (string title, string message, params string[] buttons)
{
    var alertView = new UIAlertView (title, message, null, null, buttons);
    alertView.Show ();
    var tsc = new TaskCompletionSource ();

    alertView.Clicked += (sender, buttonArgs) => {
        Console.WriteLine ("User clicked on {0}", buttonArgs.ButtonIndex);
        tsc.TrySetResult(buttonArgs.ButtonIndex);
    };
    return tsc.Task;
}

//Usage
async Task PromptUser() {
    var result = await ShowModalAletViewAsync
                ("Alert", "Do you want to continue?", "Yes", "No"); //process the result
}
```

Lea Las alertas en línea: https://riptutorial.com/es/xamarin-ios/topic/433/las-alertas

Capítulo 15: Métodos de cambio de tamaño para UIImage

Examples

Cambiar el tamaño de la imagen - con relación de aspecto

```
// resize the image to be contained within a maximum width and height, keeping aspect ratio
public static UIImage MaxResizeImage(this UIImage sourceImage, float maxWidth, float
maxHeight)
{
    var sourceSize = sourceImage.Size;
    var maxResizeFactor = Math.Min(maxWidth / sourceSize.Width, maxHeight /
sourceSize.Height);
    if (maxResizeFactor > 1) return sourceImage;
    var width = maxResizeFactor * sourceSize.Width;
    var height = maxResizeFactor * sourceSize.Height;
    UIGraphics.BeginImageContext(new CGSize(width, height));
    sourceImage.Draw(new CGRect(0, 0, width, height));
    var resultImage = UIGraphics.GetImageFromCurrentImageContext();
    UIGraphics.EndImageContext();
    return resultImage;
}
```

Cambiar el tamaño de la imagen - sin relación de aspecto

```
// resize the image (without trying to maintain aspect ratio)
public static UIImage ResizeImage(this UIImage sourceImage, float width, float height)
{
    UIGraphics.BeginImageContext(new SizeF(width, height));
    sourceImage.Draw(new RectangleF(0, 0, width, height));
    var resultImage = UIGraphics.GetImageFromCurrentImageContext();
    UIGraphics.EndImageContext();
    return resultImage;
}
```

Recortar imagen sin cambiar el tamaño

```
// crop the image, without resizing
public static UIImage CropImage(this UIImage sourceImage, int crop_x, int crop_y, int width,
int height)
{
    var imgSize = sourceImage.Size;
    UIGraphics.BeginImageContext(new SizeF(width, height));
    var context = UIGraphics.GetCurrentContext();
    var clippedRect = new RectangleF(0, 0, width, height);
    context.ClipToRect(clippedRect);
    var drawRect = new CGRect(-crop_x, -crop_y, imgSize.Width, imgSize.Height);
    sourceImage.Draw(drawRect);
    var modifiedImage = UIGraphics.GetImageFromCurrentImageContext();
    UIGraphics.EndImageContext();
```

```
    return modifiedImage;
}
```

Lea Métodos de cambio de tamaño para UIImage en línea: https://riptutorial.com/es/xamarin-ios/topic/6542/metodos-de-cambio-de-tamano-para-uiimage

Capítulo 16: Prácticas recomendadas para migrar de UILocalNotification al marco de notificaciones de usuario

Examples

Notificaciones de usuario

1. Tendrás que importar UserNotifications

```
 @import UserNotifications;
```

2. Solicitud de autorización para la notificación local.

```
let center = UNUserNotificationCenter.current()
center.requestAuthorization([.alert, .sound]) { (granted, error) in
    // Enable or disable features based on authorization.
}
```

3. Ahora vamos a actualizar el icono de la aplicación número de placa.

```
 @IBAction  func triggerNotification(){
 let content = UNMutableNotificationContent()
 content.title = NSString.localizedUserNotificationString(forKey: "Tom said:", arguments:
nil)
 content.body = NSString.localizedUserNotificationString(forKey: "Hello Mike Let's go.",
arguments: nil)
 content.sound = UNNotificationSound.default()
 content.badge = UIApplication.shared().applicationIconBadgeNumber + 1;
 content.categoryIdentifier = "com.mike.localNotification"
 //Deliver the notification in two seconds.
 let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval: 1.0, repeats: true)
 let request = UNNotificationRequest.init(identifier: "TwoSecond", 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()
 }
```

Lea Prácticas recomendadas para migrar de UILocalNotification al marco de notificaciones de usuario en línea: https://riptutorial.com/es/xamarin-ios/topic/6382/practicas-recomendadas-para-migrar-de-uilocalnotification-al-marco-de-notificaciones-de-usuario

Capítulo 17: Programación concurrente en Xamarin.iOS

Examples

Manipulación de la interfaz de usuario de hilos de fondo

Subprocesos de fondo no pueden modificar la interfaz de usuario; casi todos los métodos UIKit deben ser llamados en el hilo principal.

Desde una subclase de `NSObject` (incluido cualquier `UIViewController` o `UIView`):

```
InvokeOnMainThread(() =>
{
    // Call UI methods here
});
```

De una clase estándar de C #:

```
UIApplication.SharedApplication.InvokeOnMainThread(() =>
{
    // Call UI methods here
});
```

`InvokeOnMainThread` espera a que se ejecute su código en el subproceso principal antes de continuar. Si no necesita esperar, use `BeginInvokeOnMainThread` .

Usando Async y espera

Puede usar métodos asíncronos para manejar ejecuciones asíncronas. Por ejemplo, las solicitudes POST y GET. Vamos a decir a continuación es su método de obtención de datos.

```
Task<List> GetDataFromServer(int type);
```

Puede llamar a ese método como se muestra a continuación

```
var result = await GetDataFromServer(1);
```

Sin embargo, en la práctica real, este método estará en una interfaz de capa de servicio. La mejor manera de hacerlo es crear un método separado para llamar a esto y actualizar la IU que se muestra a continuación.

```
//Calling from viewDidLoad
void async ViewDidLoad()
{
    await GetDataListFromServer(1);
```

```
    //Do Something else
}

//New method call to handle the async task
private async Task GetArchivedListFromServer(int type)
{
    var result = await GetDataFromServer(type);
    DataList.AddRange(result.toList());
    tableView.ReloadData();
}
```

En el fragmento de código anterior, se llamará al método GetDataListFromServer y enviará la solicitud web. Sin embargo, no bloqueará el subproceso de la interfaz de usuario hasta que reciba la respuesta del servidor. Se moverá hacia abajo de la línea después de `await GetDataListFromServer(1)` . Sin embargo, dentro del método `private async Task GetArchivedListFromServer(int type)` , esperará hasta que reciba la respuesta del servidor para ejecutar las líneas después de `var result = await GetDataFromServer(type);` .

Lea Programación concurrente en Xamarin.iOS en línea: https://riptutorial.com/es/xamarin-ios/topic/1364/programacion-concurrente-en-xamarin-ios

Capítulo 18: Trabajando con Xib y Storyboards en Xamarin.iOS

Examples

Abrir Xib / Sotryboard en Xcode Interface Builder en su lugar

Xamarin sturio abre el archivo Xib y los guiones gráficos de forma predeterminada en el Diseñador Xamarin. El usuario puede hacer clic derecho en el archivo y `Open With` -> `Xcode Interface Builder '

Lea Trabajando con Xib y Storyboards en Xamarin.iOS en línea: https://riptutorial.com/es/xamarin-ios/topic/6182/trabajando-con-xib-y-storyboards-en-xamarin-ios

Capítulo 19: UIImageView zoom en combinación con UIScrollView

Observaciones

El UIImageView debe estar dentro de una vista de desplazamiento para que esto funcione.

El método DoubleTap alternará entre minScale y doubleTapScale.

Examples

Doble toque

```
private float minScale = 1f;
private float doubleTapScale = 2f;
private float maxScale = 4f;

private void SetUpDoubleTapZoom()
{
    imageViewToZoom.ContentMode = UIViewContentMode.ScaleAspectFit;
    scrollView.MaximumZoomScale = maxScale;
    scrollView.MinimumZoomScale = minScale;

    var doubleTap = new UITapGestureRecognizer(OnDoubleTap)
    {
        NumberOfTapsRequired = 2
    };

    scrollView.AddGestureRecognizer(doubleTap);
}

private void OnDoubleTap(UIGestureRecognizer gesture)
{
    scrollView.ZoomScale = (scrollView.ZoomScale.Equals(minScale)) ? doubleTapScale :
minScale;
}
```

Gesto de pellizco zoom

```
private float minScale = 1f;
private float maxScale = 4f;

private void SetUpPinchGestureZoom()
{
    imageViewToZoom.ContentMode = UIViewContentMode.ScaleAspectFit;

    scrollView.MaximumZoomScale = maxScale;
    scrollView.MinimumZoomScale = minScale;

    scrollView.ViewForZoomingInScrollView += (UIScrollView sv) => { return imageViewToZoom; };
}
```

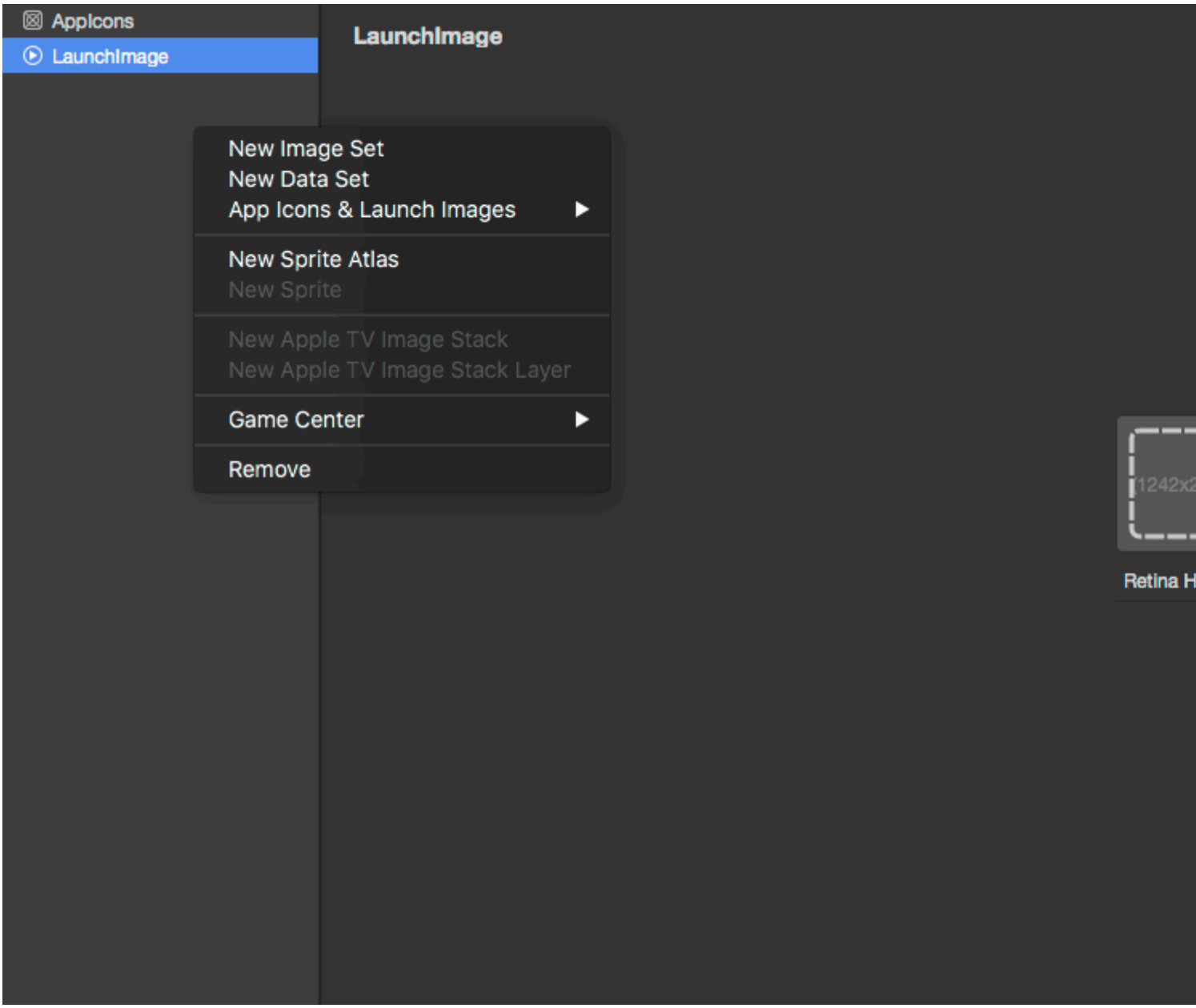
Lea UIImageView zoom en combinación con UIScrollView en línea: https://riptutorial.com/es/xamarin-ios/topic/6346/uiimageview-zoom-en-combinacion-con-uiscrollview

Capítulo 20: Uso de catálogos de activos

Examples

Adición de activos de imagen al catálogo de activos

Así es como se ve el Catálogo de Activos en Xamarin Studio,



Como se muestra en la imagen anterior, hay 5 tipos de recursos que puede crear dentro del catálogo.

Cubriré solo el conjunto de imágenes, porque es el más simple.

Cuando creas un nuevo conjunto de imágenes. Obtendrás opciones como esta

Para agregar imágenes al catálogo, simplemente haga clic en los cuadrados discontinuos y

seleccione la imagen que desea configurar para la opción particular.

En XCode tienes opciones de 1x, 2x y 3x para cubrir los tamaños de pantalla más recientes del dispositivo iOS. Sin embargo, Xamarin tiene una opción adicional Vector, que permite cargar imágenes vectoriales en formato PDF, que se escalarían automáticamente dependiendo del dispositivo en el que se esté ejecutando la aplicación.

Para las imágenes de iPhone, Xamarin conserva el tamaño de imagen especial R4 de iOS7 que se usa para iPhone de tamaño de pantalla de 4 pulgadas (5, 5S y SE).

Consulte la documentación de Xamarin sobre cómo agregar imágenes a la aplicación de iOS para obtener más información.

Lea Uso de catálogos de activos en línea: https://riptutorial.com/es/xamarin-ios/topic/6630/uso-de-catalogos-de-activos

Capítulo 21: Uso de catálogos de activos de iOS para gestionar imágenes

Observaciones

Los catálogos de activos son una forma de administrar varias resoluciones de activos de imagen de iOS. Para mostrar imágenes óptimas, iOS usa versiones de 1x, 2x y 3x de cada imagen según la densidad de pantalla del dispositivo. La versión 1x es solo para dispositivos muy antiguos, sin retina, por lo que no es necesaria para aplicaciones que solo admiten iOS 9.

Los catálogos de activos ayudarán a respaldar el adelgazamiento y la división de aplicaciones, optimizando los recursos que los usuarios deben descargar para instalar una aplicación desde la App Store.

Examples

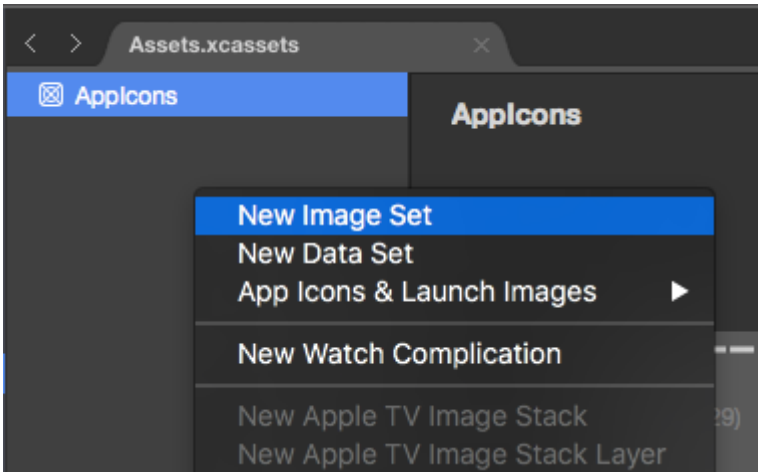
Cargando una imagen de catálogo de activos

Cargue una imagen de un catálogo de activos utilizando `UIImage.FromBundle(string imageName)`

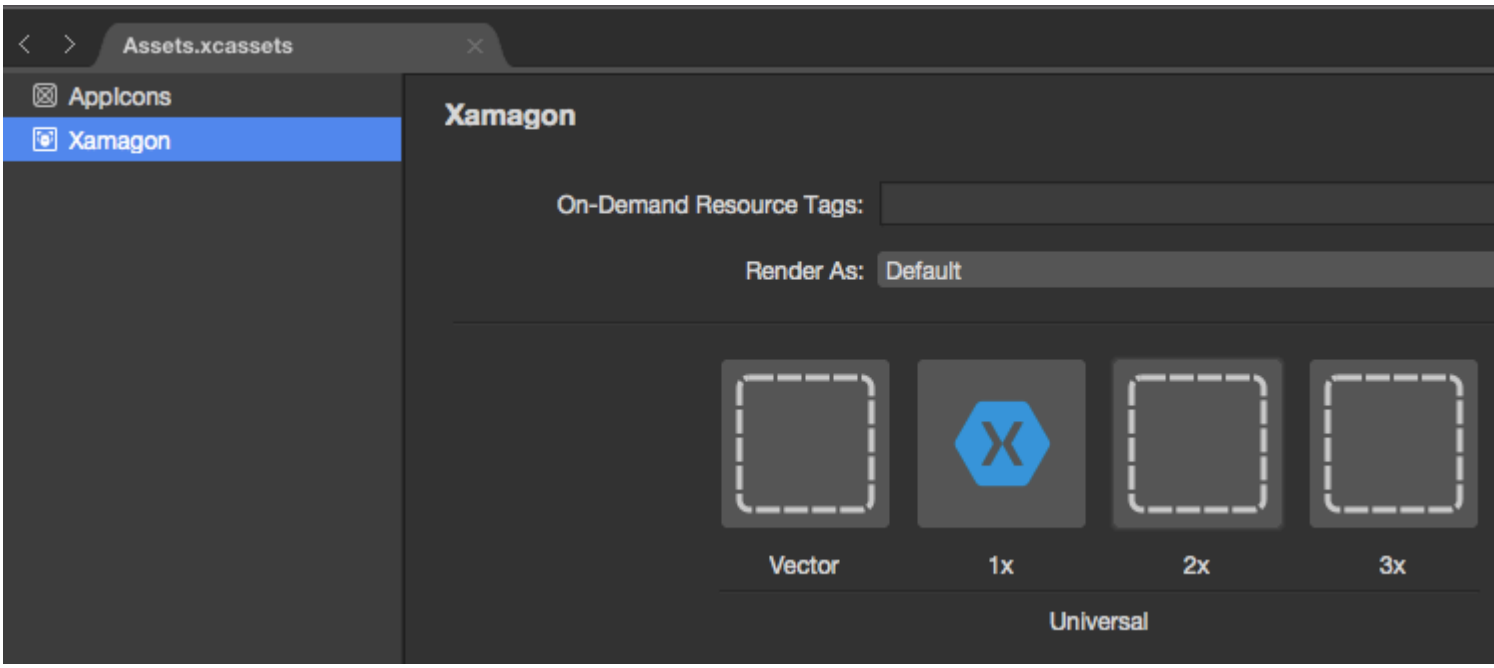
```
UIImage image = UIImage.FromBundle("ImageName");
// use the name of the image set from the asset catalog
```

Puedes usar la imagen para un `UIImageView` o cualquier otra cosa que necesites hacer.

Gestión de imágenes en un catálogo de activos.



Los catálogos de activos permiten administrar imágenes, íconos de aplicaciones e iniciar imágenes. Conjunto de imágenes se utiliza para las imágenes que se muestran en la aplicación. Las imágenes universales suelen ser la mejor opción. Puede usar una imagen basada en vectores (como PDF) que se escalará para todas las pantallas, o incluir una variante de 1x, 2x y 3x e iOS seleccionará la versión adecuada de la imagen para el dispositivo actual del usuario.



Puede cambiar el nombre de cualquier conjunto en el catálogo de activos haciendo doble clic en el nombre. Las imágenes se pueden agregar arrastrando y soltando o haciendo clic en la imagen que desea completar para un selector de archivos.

Adición de imágenes del Catálogo de Activos en el guión gráfico

Las imágenes del catálogo de activos se pueden utilizar desde guiones gráficos como cualquier otro tipo de imagen agregada al proyecto. Se rellenarán automáticamente como una opción en `UIImageView` y otras vistas que admiten agregar una imagen.

Lea Uso de catálogos de activos de iOS para gestionar imágenes en línea: https://riptutorial.com/es/xamarin-ios/topic/6241/uso-de-catalogos-de-activos-de-ios-para-gestionar-imagenes

Capítulo 22: Xamarin iOS Google Places Autocompletar

Introducción

Desde que empecé a trabajar con Xamarin, ha habido muchas cosas que desearía que alguien más ya hubiera documentado. Aquí explico cómo usar 1 aspecto del autocompletado de Google Places, el control de la interfaz de usuario que utiliza un controlador de resultados. Si bien este código se basa en los propios ejemplos de google convertidos a C # desde Swift, he intentado todo lo posible para asegurarme de que sea un ejemplo completamente funcional. Espero mucho que esta documentación ayude a otros.

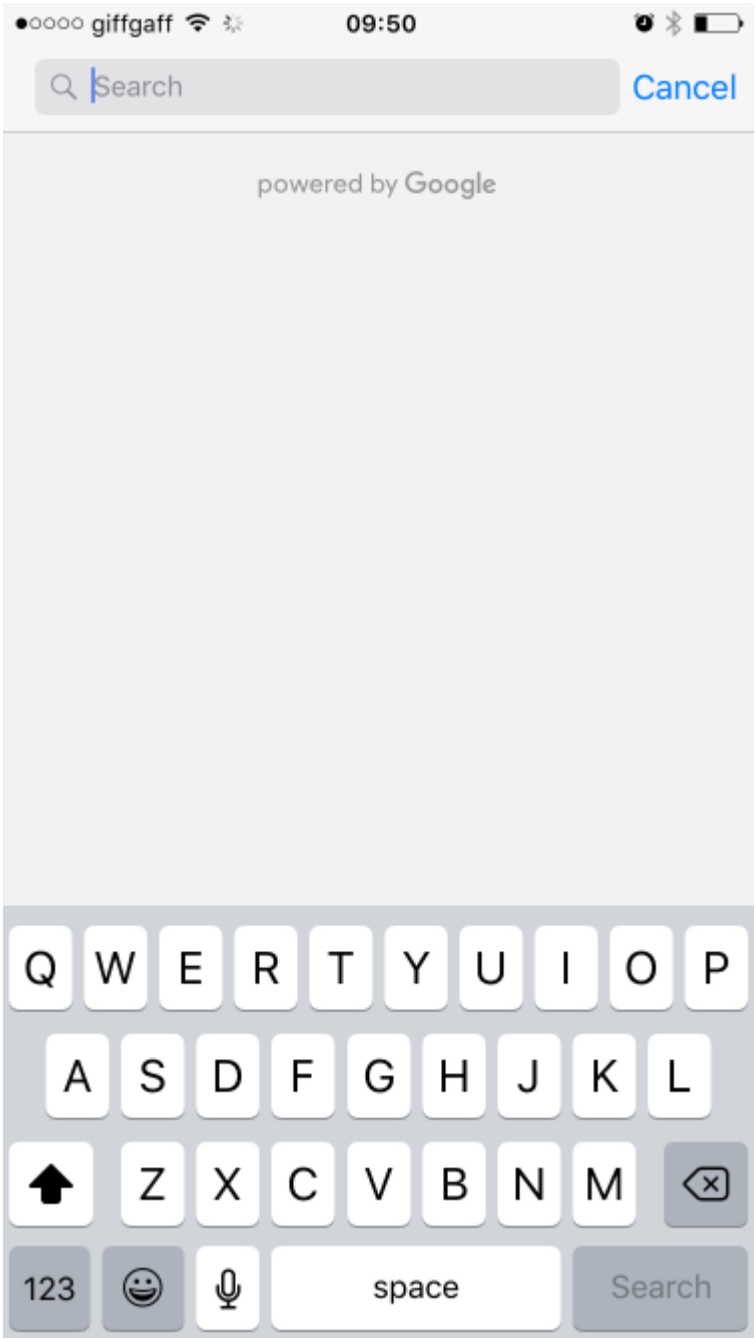
Examples

Agregue un control de interfaz de usuario autocompletado con el controlador de resultados.

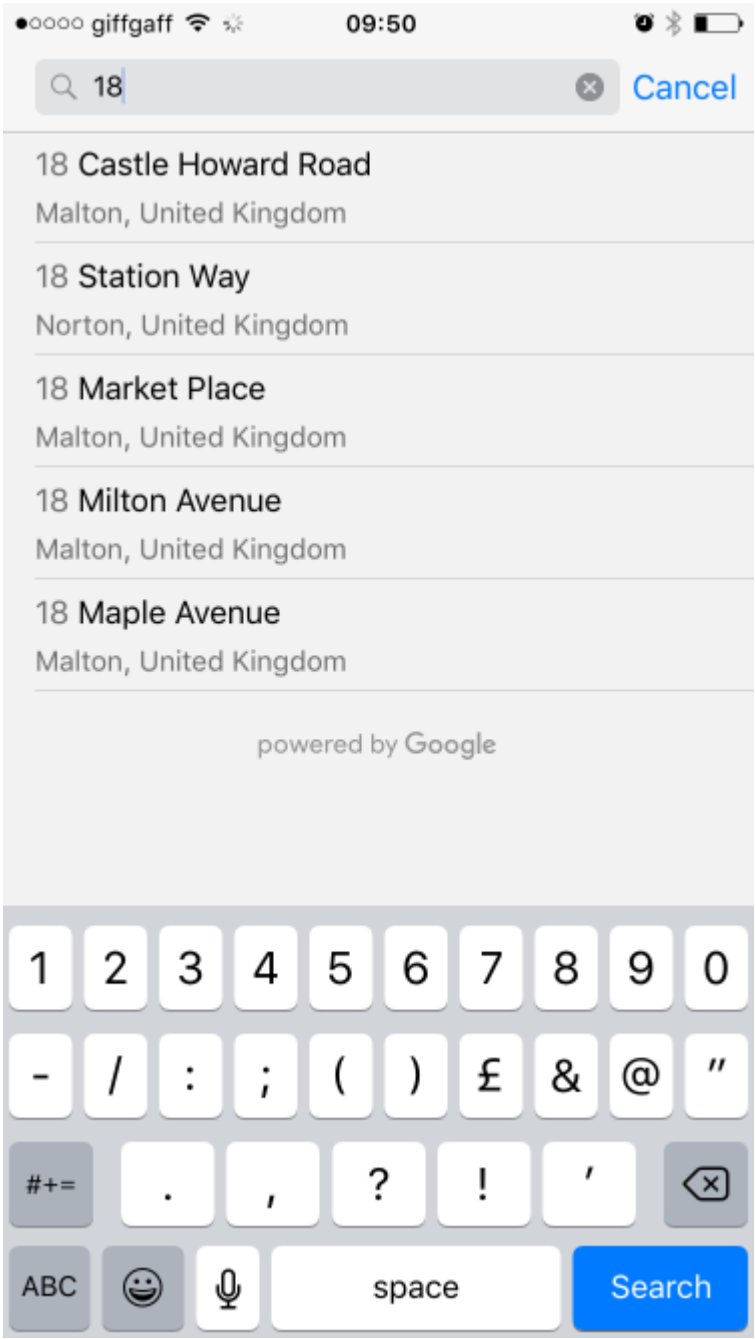
El control de IU de autocompletado es un cuadro de diálogo de búsqueda con una funcionalidad de autocompletar incorporada. Cuando un usuario ingresa los términos de búsqueda, el control presenta una lista de lugares pronosticados para elegir. Cuando el usuario realiza una selección, se devuelve una instancia de GMSPlace (Lugar en Xamarin), que su aplicación puede usar para obtener detalles sobre el lugar seleccionado.

Como se mencionó anteriormente, este ejemplo utiliza un controlador de resultados que permite un mayor control sobre la IU de entrada de texto. El controlador de resultados alternará dinámicamente la visibilidad de la lista de resultados según el enfoque de la IU de entrada.

El objetivo de este código es mostrar una pantalla como la siguiente:



Esto autocompletará su dirección cuando comience a escribir, como en la siguiente imagen:



Instrucciones:

1. Primero necesitamos agregar la API de Google Maps a nuestro Visual Studio, está disponible a través de Nuget, solo busca Xamarin.Google.iOS.Maps, agrégalo a tu proyecto de iOS, o puedes descargarlo de Xamarin Xamarin Google Maps iOS SDK

2. Necesitamos algo como un botón para activar el controlador de vista de autocompletar de Google. En este ejemplo, hago esto con un panel de historia y he agregado un botón llamado GoogleButton, podría activarlo con un código que en realidad no importa.

3. Bajo ViewDidLoad en su clase de controladores de vista agregue el siguiente código. En mi ejemplo a continuación, no estoy usando la ubicación real de los dispositivos móviles, por supuesto, mi solución final haría esto, pero esto fue una prueba y no quise implementar código adicional hasta que probé que esto funcionó o diluyo lo que estoy intentando para mostrarte:

// Código para que aparezca el controlador de vista automática de Google Places.

```
GoogleButton.TouchUpInside += (sender, ea) =>
{

    var FakeCoordinates = new CLLocationCoordinate2D()
    {
        Latitude = 54.135364,
        Longitude = -0.797888
    };

    var north = LocationWithBearing(45, 3000, FakeCoordinates);
    var east = LocationWithBearing(225, 3000, FakeCoordinates);

    var autocompleteController = new AutocompleteViewController();
    autocompleteController.Delegate = new AutoCompleteDelegate();
    autocompleteController.AutocompleteBounds = new CoordinateBounds(north, east);
    PresentViewController(autocompleteController, true, null);

};
```

4. Esto es opcional, pero he agregado una función para calcular los límites locales, estoy pasando en 3000, este número está en metros, por lo que si desea un límite inicial mayor, no dude en ajustar, tenga en cuenta que la búsqueda de Google seguirá encontrando cualquier dirección en el mundo, solo carga primero los resultados iniciales a los límites de estas áreas locales. Esta función fue tomada de una publicación de desbordamiento de pila, la he convertido de Swift a C # para nuestros propósitos:

```
public CLLocationCoordinate2D LocationWithBearing(Double bearing, Double distanceMeters,
CLLocationCoordinate2D origin)
{
    var distRadians = distanceMeters/(6372797.6);

    var rbearing = bearing*Math.PI/180.0;

    var lat1 = origin.Latitude*Math.PI/180;
    var lon1 = origin.Longitude*Math.PI/180;

    var lat2 = Math.Asin(Math.Sin(lat1)*Math.Cos(distRadians) +
Math.Cos(lat1)*Math.Sin(distRadians)*Math.Cos(rbearing));
    var lon2 = lon1 + Math.Atan2(Math.Sin(rbearing)*Math.Sin(distRadians)*Math.Cos(lat1),
                    Math.Cos(distRadians) - Math.Sin(lat1)*Math.Sin(lat2));

    return new CLLocationCoordinate2D(latitude: lat2*180/ Math.PI, longitude:
lon2*180/Math.PI);
}
```

5. Este fragmento de código final es el delegado del autocompletado, necesitamos que este delegado se encargue de todo lo que Google nos devolverá:

```
public  class AutoCompleteDelegate : AutocompleteViewControllerDelegate
{

    public override void DidFailAutocomplete(AutocompleteViewController viewController,
NSError error)
    {
```

```
        // TODO: handle the error.
        Debug.Print("Error: " + error.Description);
    }


    public override void DidAutocomplete(AutocompleteViewController viewController, Place
place)
    {
        Debug.Print(place.Name);
        Debug.Print(place.FormattedAddress);

        viewController.DismissViewController(true, null);
    }

    public override void DidRequestAutocompletePredictions(AutocompleteViewController
viewController)
    {
        UIApplication.SharedApplication.NetworkActivityIndicatorVisible = true;
    }

    public override void DidUpdateAutocompletePredictions(AutocompleteViewController
viewController)
    {
        UIApplication.SharedApplication.NetworkActivityIndicatorVisible = true;
    }

    public override void WasCancelled(AutocompleteViewController viewController)
    {
        viewController.DismissViewController(true, null);
    }

}
```

Ejecute su proyecto y debería funcionar a la perfección; este ejemplo está bastante centrado, pero esperamos que le brinde un ejemplo básico de cómo debería funcionar cualquiera de los controles de la interfaz de usuario de Google Autocompletar. ¡Gracias!

Lea Xamarin iOS Google Places Autocompletar en línea: https://riptutorial.com/es/xamarin-ios/topic/9041/xamarin-ios-google-places-autocompletar

Capítulo 23: Xamarin.iOS Navigation Drawer

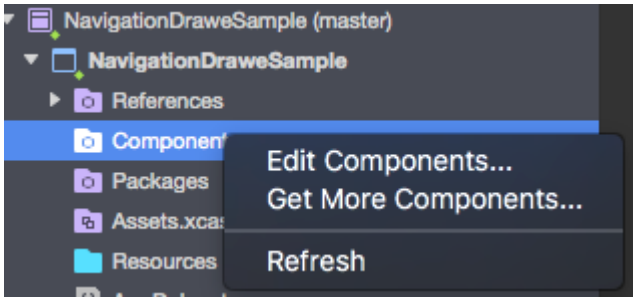
Sintaxis

1. Flayout Navigation Component: https://components.xamarin.com/view/flyoutnavigation

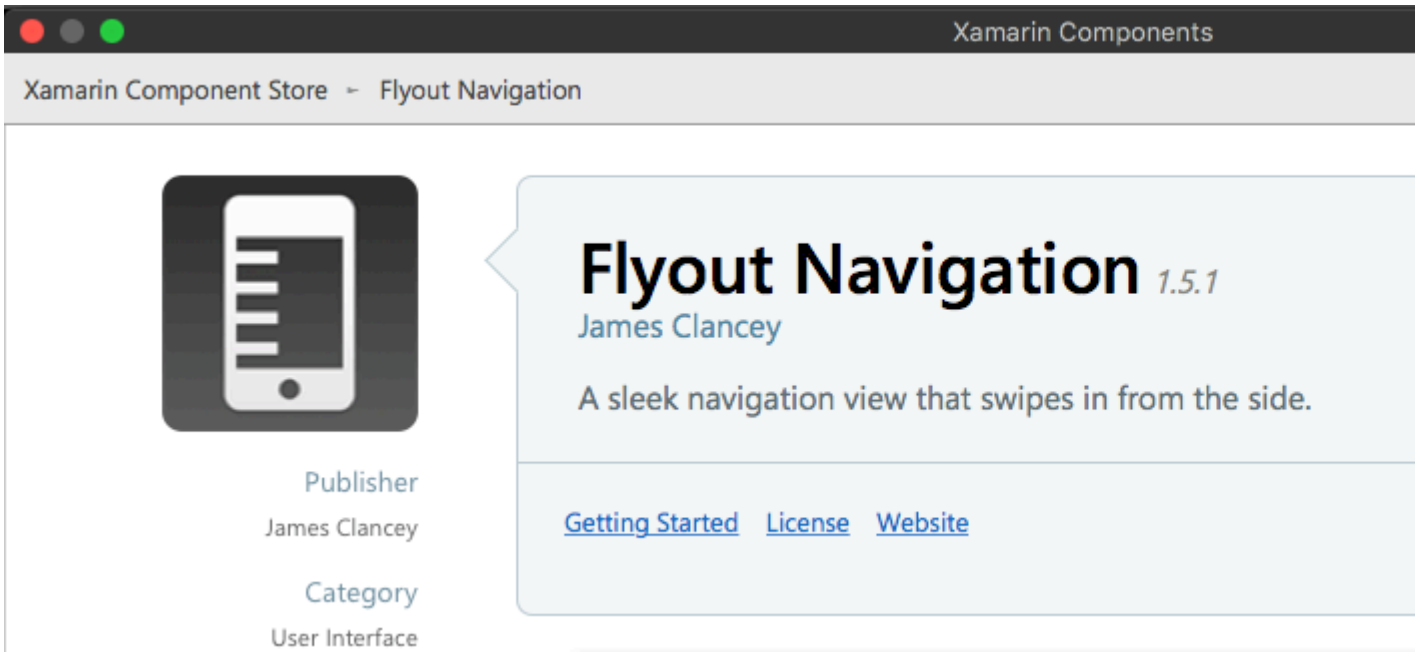
Examples

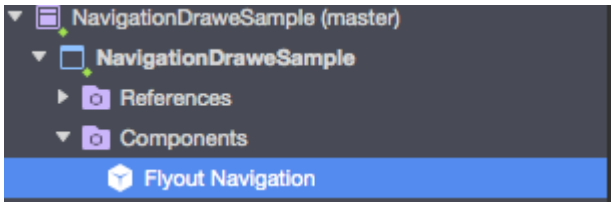
Xamarin.iOS Navigation Drawer

1. Crear un nuevo proyecto en blanco Xamarin.iOS (aplicación de vista única).
2. Haga clic derecho en la carpeta "Componentes" y seleccione "Obtener más componentes":

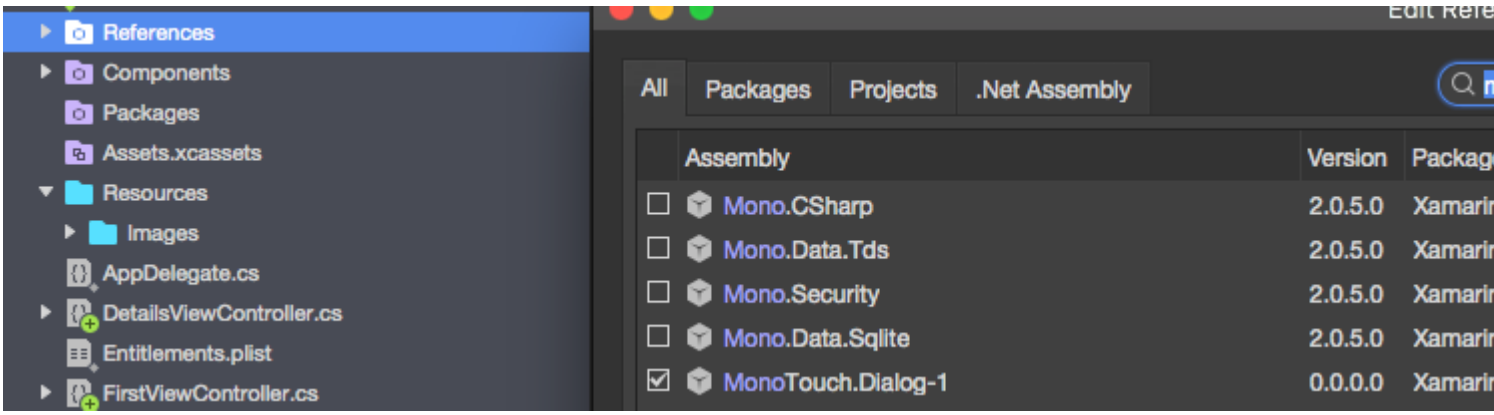


3. En el cuadro de búsqueda, escriba "Navegación flotante" y agregue el siguiente componente a su aplicación:

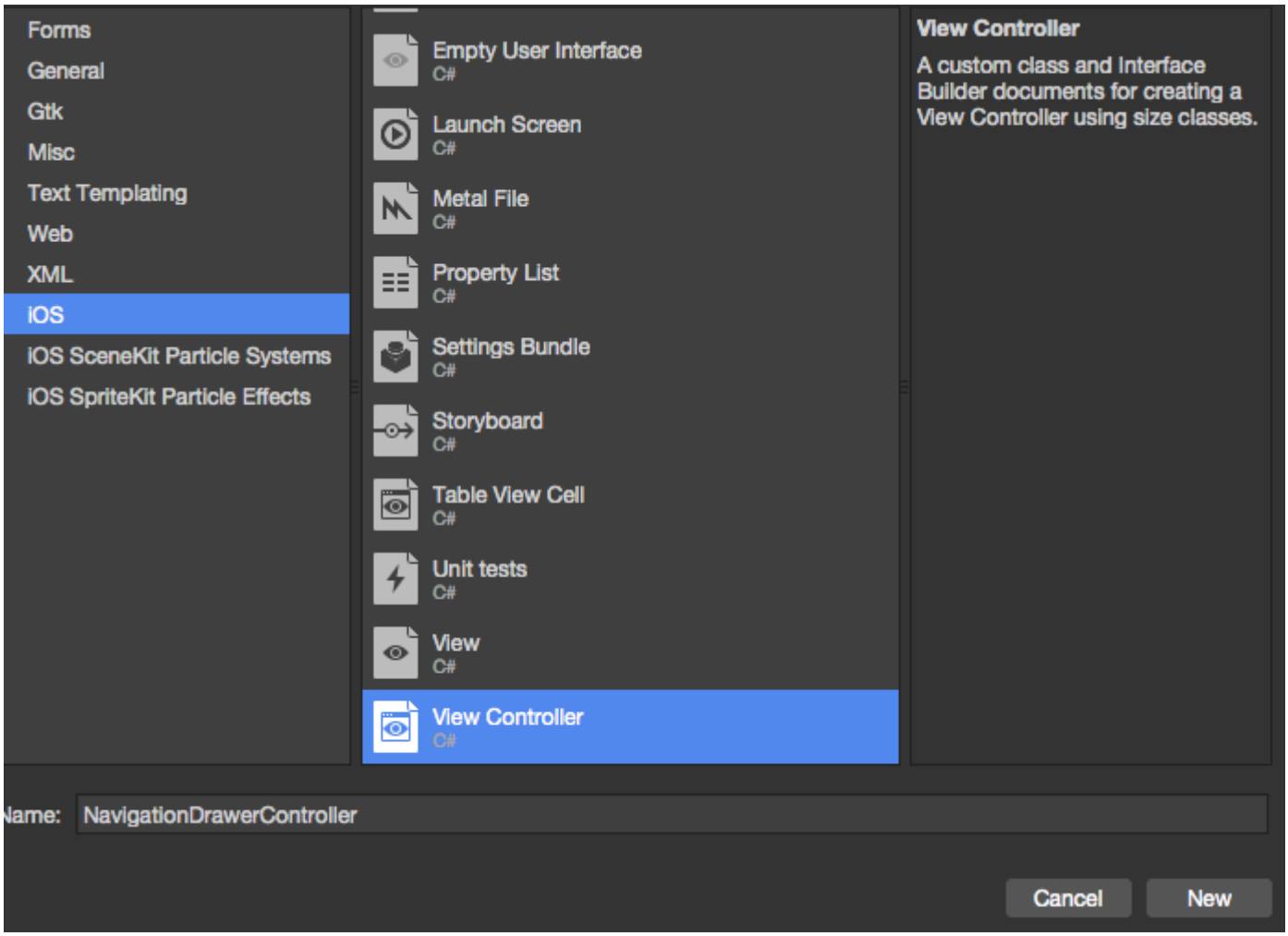




Recuerde también agregar la referencia "Mono.Touch.Dialog-1":



4. Ahora haga clic en el proyecto y agregue el nuevo UIViewController llamado "NavigationDrawerController":



5. Ahora el código para la clase "NavigationDrawerController" debería verse a continuación:

```
public partial class NavigationDrawerController : UIViewController
{
    public NavigationDrawerController(IntPtr handle) : base(handle)
    {
    }

    public override void ViewDidLoad()
    {
```

```
        base.ViewDidLoad();

        NavigationItem.LeftBarButtonItem = getMenuItem();
        NavigationItem.RightBarButtonItem = new UIBarButtonItem { Width = 40 };
    }

    UIBarButtonItem getMenuItem()
    {
        var item = new UIBarButtonItem();
        item.Width = 40;
        //Please provide your own icon or take mine from the GitHub sample:
        item.Image = UIImage.FromFile("Images/menu_button@2x.png");
        item.Clicked += (sender, e) =>
        {
            if (ParentViewController is MainNavigationController)
                (ParentViewController as MainNavigationController).ToggleMenu();
        };

        return item;
    }
}
```

No se preocupe si "MainNavigationController" está resaltado en rojo, lo agregaremos en el siguiente paso.

6. Ahora abra el archivo "Main.storyboard":

a) Agrega un UIViewController:

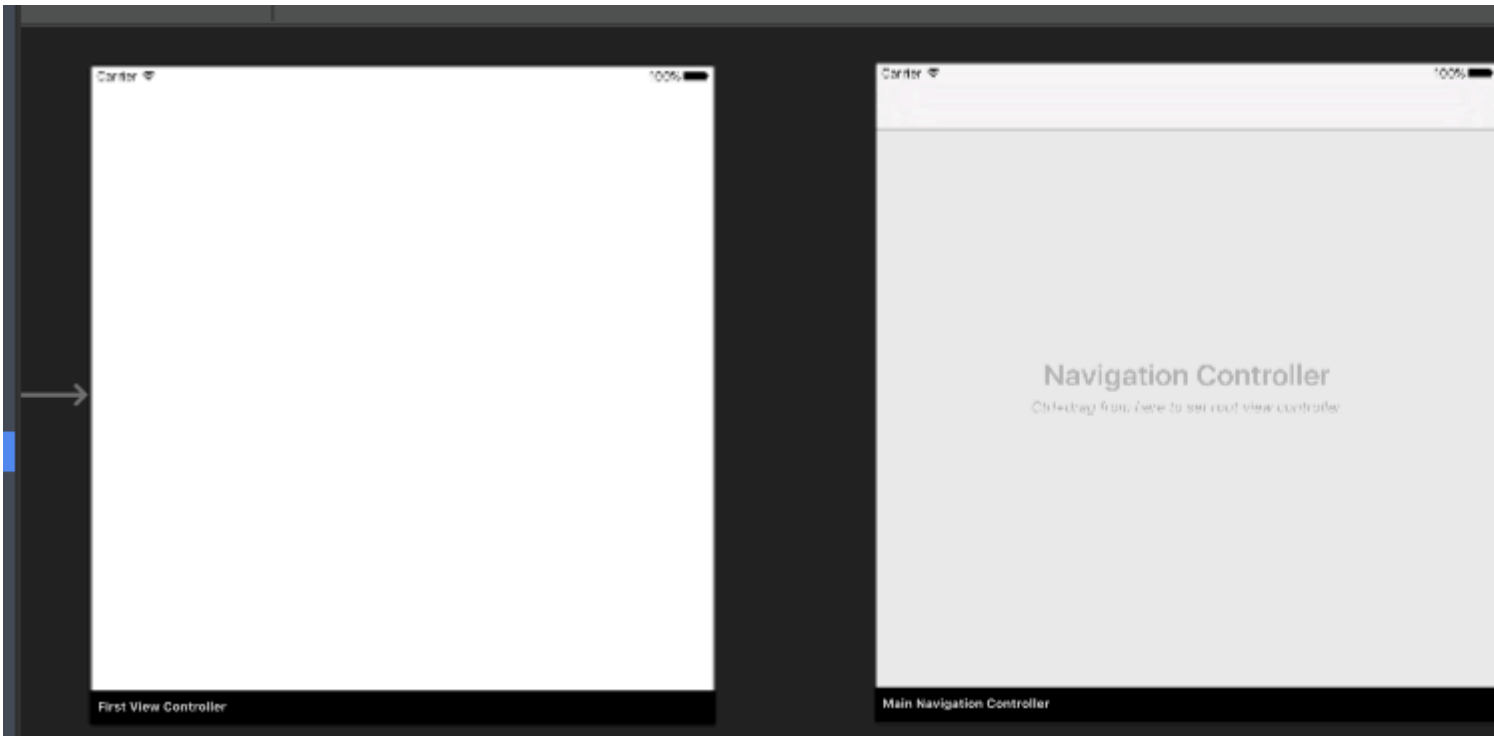
Rellene los campos "Class" y "StoryboardID" con este nombre: "FirstViewController"

b) Después de eso agregue Navigation Controller con root UIViewController:

Rellene los campos "Class" y "StoryboardID" con este nombre: "MainNavigationController" para el controlador de navegación

Rellene los campos "Class" y "StoryboardID" con este nombre: "DetailsViewController" para el controlador raíz

Xamarin (o Visual) Studio creará clases de código subyacente para los controladores anteriores.



7. Ahora abra la clase "FirstViewController" y pegue el siguiente código:

```
public partial class FirstViewController : UIViewController
{
    public FirstViewController (IntPtr handle) : base (handle)
    {
    }

    public override void ViewDidLoad()
    {
        base.ViewDidLoad();
        createNavigationFlyout();
    }

    void createNavigationFlyout()
    {
        var navigation = new FlyoutNavigationController
        {
            //Here are sections definied for the drawer:
            NavigationRoot = new RootElement("Navigation")
            {
                new Section ("Pages")
                {
                    new StringElement ("MainPage")
                }
            },

            //Here are controllers definied for the drawer (in this case navigation controller
with one root):
             ViewControllers = new[]
             {

(MainNavigationController)Storyboard.InstantiateViewController("MainNavigationController")
             }
        };

        View.AddSubview(navigation.View);
```

```
    }
}
```

8. Abra la clase "MainNavigationController" y pegue el siguiente código:

```
public partial class MainNavigationController : UINavigationController
{
     public MainNavigationController (IntPtr handle) : base (handle)
     {
     }
     //Responsible for opening/closing drawer:
     public void ToggleMenu()
     {
         if (ParentViewController is FlyoutNavigationController)
         (ParentViewController as FlyoutNavigationController).ToggleMenu();
     }
 }
```

9. La última clase llamada "DetailsViewController" debería verse así:

```
public partial class DetailsViewController : NavigationDrawerController
{
    public DetailsViewController (IntPtr handle) : base(handle)
    {
    }
}
```

Tenga en cuenta que "DetailsViewController" deriva de "NavigationDrawerController" que creamos al principio.

Eso es. Ahora puedes personalizar el cajón como quieras. Por favor, también encontrar la muestra lista en mi GitHub:

https://github.com/Daniel-Krzyczkowski/XamarinIOS/tree/master/Xamarin.iOS.NavigationDrawer

Lea Xamarin.iOS Navigation Drawer en línea: https://riptutorial.com/es/xamarin-ios/topic/6574/xamarin-ios-navigation-drawer

Creditos

S. No	Capítulos	Contributors
1	Empezando con Xamarin.iOS	Amy Burns, chrisntr, Community, Dominic, hankide, Sergey, valdetero
2	Agregar UIRefreshControl a una vista de tabla	manishKungwani, valdetero
3	Añadir barra de búsqueda a UITableView	Aditya Kumar, valdetero
4	Añadir PullToRefresh a UITableView	Aditya Kumar, valdetero
5	Cálculo de altura de fila variable en GetHeightForRow	Larry OBrien, valdetero
6	Cómo utilizar los catálogos de activos de activos	Aditya Kumar
7	Conectando con Microsoft Cognitive Services	Daniel Krzyczkowski, valdetero
8	Controlando la captura de pantalla en el conmutador multitarea de iOS	ben
9	Crea y usa celdas de la tabla de prototipos personalizados en xamarin.iOS usando el guión gráfico	Daniel Krzyczkowski, valdetero
10	Disposición automática en Xamarin.iOS	ben, patridge, Tom Hawkin, valdetero

11	Encuadernación de bibliotecas rápidas	Alex Sorokoletov, Elad Nava, Esam Sherif, J. Rahmati, James Mundy, Lucas Teixeira
12	identifición de toque	Amy Burns, ben, DannyC, Matthew, Peter Zhong, valdetero
13	Las alertas	chrisntr, Gil Sand, patridge, Pilatus, Prashant C, valdetero
14	Métodos de cambio de tamaño para UIImage	Frauke Nonnenmacher, raymondis, valdetero
15	Prácticas recomendadas para migrar de UILocalNotification al marco de notificaciones de usuario	Aditya Kumar
16	Programación concurrente en Xamarin.iOS	Ashan, Pilatus, Tom Gilder, valdetero
17	Trabajando con Xib y Storyboards en Xamarin.iOS	lukya
18	UIImageView zoom en combinación con UIScrollView	Citroenfris, valdetero
19	Uso de catálogos de activos	aniket.ghode, mnoronha
20	Uso de catálogos de activos de iOS para gestionar imágenes	dylansturg, valdetero
21	Xamarin iOS Google Places Autocompletar	Conrad
22	Xamarin.iOS Navigation Drawer	Daniel Krzyczkowski, valdetero