EBook Gratis

APRENDIZAJE ionic2

Free unaffiliated eBook created from
Stack Overflow contributors.

#ionic2

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ionic2

It is an unofficial and free ionic2 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 ionic2.

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 ionic2

Observaciones

Ionic 2 es una tecnología de desarrollo móvil multiplataforma. Este marco está diseñado para crear aplicaciones móviles híbridas y también se puede utilizar para aplicaciones de escritorio. Es una tecnología de escribir una vez, correr en todas partes. Utiliza tecnologías web como JavaScript / Typescript, Angular 2, HTML y CSS (SCSS / LESS). Las aplicaciones Ionic2 funcionan bien en `>=android 4.4` , pero quieres ejecutar en `android 4.1` a `android 4.3` tienes que usar cross-walk .

Examples

Instalación o configuración

Como Ionic 2 está mejorando cada día, siempre revise la **documentación oficial** para estar al tanto de los últimos cambios y mejoras.

Requisitos previos: Necesitará NodeJS para construir proyectos de Ionic 2. Puede descargar e instalar el nodo aquí y obtener más información sobre npm y los paquetes que usa Ionic 2 aquí .

---

1. Instalar Ionic 2

Al igual que Ionic 1, puede utilizar la CLI o la GUI de Ionic para crear y probar rápidamente aplicaciones en el navegador. Incluso tiene todas las funciones para trabajar con sus aplicaciones Ionic 1, por lo que no tendrá que cambiar nada.

Para usar Ionic 2, simplemente instale ionic desde npm:

```
$ npm install -g ionic
```

Si recibe un error EACCES, siga las instrucciones aquí para otorgar al nodo los permisos que necesita.

2. Creando tu primera aplicación

Una vez que se instale la CLI, ejecute el siguiente comando para iniciar su primera aplicación:

```
$ ionic start MyIonic2Project
```

La plantilla de pestañas se usa de forma predeterminada, pero puede elegir otra plantilla pasando una bandera. Por ejemplo:

```
$ ionic start MyIonic2Project tutorial
$ cd MyIonic2Project
$ npm install
```

Esto utilizará la plantilla tutorial .

Para ejecutar su aplicación, cambie a su directorio de proyectos y ejecute `ionic serve -lc` :

```
$ ionic serve -lc
```

El -l activa la recarga en vivo de la página, el -c muestra los registros de la consola. Si tiene problemas para crear su aplicación, asegúrese de que package.json coincida con el de ionic2-app-base

¡Puedes jugar con tu nueva aplicación allí mismo en el navegador!

3. Construyendo a un dispositivo

También puede crear su nueva aplicación en un dispositivo físico o en un emulador de dispositivo. Necesitarás a Cordova para proceder.

Para instalar Cordova, ejecute:

```
$ npm install -g cordova
```

Consulte los documentos del simulador de iOS para crear aplicaciones de iOS (NOTA: no puede compilar dispositivos o emuladores de iOS en ningún sistema operativo que no sea OSX), ni los documentos de Genymotion para crear una aplicación de Android.

Ejecutando en dispositivo iOS:

Para crear una aplicación iOS, es necesario que trabaje en una computadora OSX, ya que necesitará la infraestructura de cacao para poder compilar para ios, si es el caso, primero deberá agregar la plataforma a cordova ejecutando siguiente comando:

```
$ ionic cordova platform add ios
```

Necesitará Xcode para compilar en un dispositivo iOS.

Finalmente, ejecute su aplicación con el siguiente comando:

```
$ ionic cordova run ios
```

Ejecutando en un dispositivo Android:

Los pasos para Android son casi idénticos. Primero, agregue la plataforma:

```
$ ionic cordova platform add android
```

Luego instale el SDK de Android que le permite compilar en un dispositivo Android. Aunque el SDK de Android viene con un emulador, es muy lento. Genymotion es mucho más rápido. Una vez instalado, simplemente ejecute el siguiente comando:

```
$ ionic cordova run android
```

¡Y eso es! ¡Felicidades por crear tu primera aplicación Ionic 2!

Ionic tiene recarga en vivo también. Entonces, si desea desarrollar su aplicación y ver los cambios que se están produciendo en vivo en el emulador / dispositivo, puede hacerlo ejecutando los siguientes comandos:

Para iOS:

```
$ ionic cordova emulate ios -lcs
```

Ten cuidado, en iOS 9.2.2 la carga de datos no funciona. Si desea trabajar con livereload, edite el archivo config.xml agregando lo siguiente:

```
<allow-navigation href="*"/>
```

Luego en el `<platform name="ios">` :

```
<config-file parent="NSAppTransportSecurity" platform="ios" target="*-Info.plist">
  <dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
  </dict>
</config-file>
```

Para Android:

```
$ ionic cordova run android -lcs
```

La `l` significa live-recargar, `c` para los registros de la consola y `s` para los registros del servidor. Esto le permitirá ver si hay errores / advertencias durante la ejecución.

Construyendo para Windows

Si desea construir su proyecto para Windows, necesita trabajar en una computadora con Windows. Para comenzar, instale la plataforma de Windows en su proyecto ionic2 ejecutando el siguiente comando:

```
$ionic cordova platform add windows
```

Luego simplemente ejecuta el siguiente comando:

```
$ionic cordova run windows
```

Para ejecutar en el navegador

```
$ionic serve
```

para el dispositivo de inspección del navegador Chrome (escriba en la barra de direcciones del navegador Chrome)

```
chrome://inspect/#devices
```

Lea Empezando con ionic2 en línea: https://riptutorial.com/es/ionic2/topic/3632/empezando-con-ionic2

Capítulo 2: Añadir aplicación iónica a la vista iónica

Introducción

ionic view es una aplicación móvil que debes instalar en tu móvil, para que puedas ver tu aplicación sin crear archivos .apk. Al compartir su ID de aplicación, otros también pueden ver su aplicación en su dispositivo móvil utilizando la vista iónica.

Sitio: https://view.ionic.io/

Examples

Pasos para agregar su aplicación a la vista iónica

Los siguientes pasos deben realizarse en app.ionic.io

1. Crea una cuenta o inicia sesión en tu cuenta iónica
2. Haga clic en "Nueva aplicación" en el Panel de control y asigne un nombre a su aplicación.

   ```
   I named my app as 'MyIonicApp'
   ```

3. En la sección de descripción general de esta aplicación recién creada, habrá un ID debajo del nombre de la aplicación.

   ```
   MyIonicApp ID is 4c5051c1
   ```

Los siguientes pasos se realizan en el símbolo del sistema de **Node.js**

1. Inicie sesión en su cuenta iónica ejecutando

   ```
   $ ionic login
   ```

2. Rootea la carpeta de tu aplicación.
3. Para cargar su aplicación en la vista iónica, primero debe vincular su aplicación con la ID que creó en el sitio iónico. Ejecuta el siguiente comando para enlazar,

   ```
   $ ionic link [your-app-id]
   ```

   Para MyIoincApp, el comando será,

   ```
   $ ionic link 4c5051c1
   ```

El comando anterior actualizará el ID de la aplicación en el archivo de configuración de MyIonicApp.

4. Una vez hecho el enlace, carga la aplicación ejecutando

```
$ ionic upload
```

Nota

Una vez que la carga sea exitosa, abra la vista iónica en su móvil para ver la aplicación.

Otros pueden ver su aplicación, enviando el ID de la aplicación en la sección 'Vista previa de una aplicación' en la vista iónica.

Lea Añadir aplicación iónica a la vista iónica en línea:
https://riptutorial.com/es/ionic2/topic/10542/anadir-aplicacion-ionica-a-la-vista-ionica

Capítulo 3: Angularfire2 con Ionic2

Introducción

Aquí le mostraré cómo integrar AngularFire2 y usar esta base de datos en tiempo real en nuestra aplicación Ionic.

Examples

Inicialización de AngularFire

En primer lugar, necesita inicializar los módulos de angularfire en su módulo de aplicación de la siguiente manera:

```
  const firebaseConfig = {
  apiKey: 'XXXXXXXXXX',
  authDomain: 'XXXXXXXXXX',
  databaseURL: 'XXXXXXXXXX',
  storageBucket: 'XXXXXXXXXX',
  messagingSenderId: 'XXXXXXXXXX'
};
```

Puede obtener estas claves iniciando sesión en firebase y creando un nuevo proyecto.

```
imports: [
    AngularFireModule.initializeApp(firebaseConfig),
    AngularFireDatabaseModule,
    AngularFireAuthModule
  ],
```

Utilizando AngularFire2

Una vez que lo tengas en tu aplicación, simplemente impórtalo:

```
import { AngularFireDatabase } from 'angularfire2/database';
constructor (private _af: AngularFireDatabase) {}
```

Con esta lista observable, puede acceder a una lista de artículos en una ruta, por ejemplo, si tiene raíz / artículos / alimentos, puede obtener alimentos como este:

```
this._af.list('root/items/food');
```

Y puede simplemente poner un nuevo elemento aquí y aparecerá en su base de datos de base de fuego, o puede actualizar un elemento y lo verá actualizado en su base de datos. Puedes empujar y actualizar así:

```
this._af.list('root/items/food').push(myItemData);
```

```
this._af.list('root/items/food').update(myItem.$key, myNewItemData);
```

O incluso puede remover artículos de su lista de alimentos:

```
this._af.list('root/items/food').remove(myItem.$key);
```

Lea Angularfire2 con Ionic2 en línea: https://riptutorial.com/es/ionic2/topic/10918/angularfire2-con-ionic2

Capítulo 4: Componentes CSS de Ionic2

Examples

Cuadrícula

El sistema de cuadrícula de Ionic se basa en flexbox, una característica de CSS compatible con todos los dispositivos compatibles con Ionic. La cuadrícula se compone de tres unidades: cuadrícula, filas y columnas. Las columnas se expandirán para llenar su fila y se cambiarán de tamaño para que se ajusten a columnas adicionales.

Clase	Anchura
ancho-10	10%
ancho-20	20%
ancho-25	25%
ancho-33	33.3333%
ancho-50	50%
ancho-67	66.6666%
ancho-75	75%
ancho-80	80%
ancho-90	90%

Ejemplo.

```
    <ion-grid>
      <ion-row>
        <ion-col width-10>This column will take 10% of space</ion-col>
      </ion-row>
    </ion-grid>
```

Tarjetas

Las tarjetas son una excelente manera de mostrar piezas importantes de contenido y están emergiendo rápidamente como un patrón de diseño central para las aplicaciones. Son una excelente manera de contener y organizar la información, a la vez que configuran expectativas predecibles para el usuario. Con tanto contenido para mostrar a la vez y, a menudo, tan poco espacio de pantalla, las tarjetas se han convertido rápidamente en el patrón de diseño elegido por muchas compañías.

Ejemplo.

```
<ion-card>
  <ion-card-header>
        Header
  </ion-card-header>
  <ion-card-content>
        The British use the term "header", but the American term "head-shot" the English
simply refuse to adopt.
  </ion-card-content>
</ion-card>
```

Lea Componentes CSS de Ionic2 en línea:
https://riptutorial.com/es/ionic2/topic/8011/componentes-css-de-ionic2

Capítulo 5: Configuración y depuración de Ionic 2 en Visual Studio Code

Introducción

Visual Studio es un IDE de código abierto que proporciona una función inteligente y de edición para el código. Este IDE es compatible con muchos idiomas como (Ionic, C, C #, AngularJs, TypeScript, Android, etc.). Estos lenguajes pueden ejecutar código allí agregando sus Extensiones en VSCode. Al utilizar VSCode, podemos ejecutar y depurar el código de diferentes idiomas.

Examples

Instalación de VSCode

Primero necesitas descargar e instalar el VSCode. Esta última versión de VSCode está disponible para descargar en su sitio web oficial . Después de descargar el VSCode debes instalarlo y abrirlo.

```
Introduction of Extensions in VSCode
```

VSCode es un editor abierto, por lo que proporciona un editor para todos los idiomas, pero para ejecutar el código que necesita para agregar la Extensión para ese idioma en particular. Para ejecutar y editar su código iónico, debe agregar la Extensión **ionic2-vscode** en suVSCode. En el lado izquierdo de VSCode Editor hay 5 iconos en los que el icono más bajo se usa para la Extensión. Las Extensiones que puede obtener usando la **tecla de acceso directo (ctrl + shift + X)** .

```
Add Extension for Ionic2 in VsCode
```

Al presionar **ctrl + shift + X,** se muestra la parte de la extensión donde se muestran los **tres puntos** superiores **...** estos puntos se conocen como más ícono. Al hacer clic, se abre un cuadro de diálogo que muestra el número de opciones a elegir. elija la opción según su necesidad, pero para obtener toda la extensión, debe seleccionar la **Extensión recomendada** mostrada. En la lista de todas las eXtensiones puede instalar su Extensión `(ionic2-vscode),npm`

Crea y agrega tu proyecto jónico en VSCode

VsCode no puede crear el proyecto iónico porque es un editor de código. Por lo tanto, puede crear su proyecto iónico mediante **CLI** o **cmd** . Crea tu proyecto por debajo del comando

```
$ ionic start appName blank
```

El comando anterior se usa para crear una aplicación iónica de plantilla en blanco. Ionic2 proporciona tres tipos de plantillas en **blanco, pestañas y sidemenu** . Asi que. puede reemplazar la plantilla en blanco por cualquiera de las otras dos plantillas según su necesidad.

Ahora, tu proyecto jónico ha sido creado. Por lo tanto, puedes agregar tu proyecto en VSCode para editarlo. Para agregar su proyecto siga los puntos a continuación.

1. Ir al menú **Archivo** en VScode.
2. Haga clic en el menú **Abrir carpeta** dentro de archivo.
3. Encuentra y abre la carpeta de tu proyecto.

   Puede abrir la carpeta directamente usando la tecla de acceso directo **Ctrl + O** o **Ctrl + K**

`

Ejecuta y depura tu proyecto jónico

> Ejecutar y depurar en Chrome

Para ejecutar el proyecto iónico, utilice el comando siguiente en **terminal o cmd o CLI**

```
$ ionic serve
```

Para depurar el proyecto iónico, primero debe agregar la extensión **(Debugger for chrome)** y luego configurar el archivo launch.json como este.

```
{
        "version": "0.2.0",
        "configurations": [

            {
          "name": "Launch in Chrome",
          "type": "chrome",
          "request": "launch",
          "url": "http://localhost:8100",
          "sourceMaps": true,
          "webRoot": "${workspaceRoot}/src"
        }
]
}
```

> ***Ejecutar y depurar en Android***

Para ejecutar el proyecto iónico en Android, debe agregar la plataforma Android mediante el comando siguiente en terminal o cmd o CLI:

```
$ ionic cordova platform add android
```

Construye Android con este comando

```
$ ionic cordova build android
```

Ejecutar el comando para la plataforma Android

```
$ ionic cordova run android
```

Ahora, su aplicación se ejecuta en un dispositivo Android real.

Para la depuración en el dispositivo Android, debe agregar **Cordova o la extensión de Android** en VSCode. y configurar el archivo launch.json como este.

```
{
    "version": "0.2.0",
    "configurations": [
    {
                "name": "Run Android on device",
                "type": "cordova",
                "request": "launch",
                "platform": "android",
                "target": "device",
                "port": 9222,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}",
                "ionicLiveReload": false
            },
            {
                "name": "Run iOS on device",
                "type": "cordova",
                "request": "launch",
                "platform": "ios",
                "target": "device",
                "port": 9220,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}",
                "ionicLiveReload": false
            },
            {
                "name": "Attach to running android on device",
                "type": "cordova",
                "request": "attach",
                "platform": "android",
                "target": "device",
                "port": 9222,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}"
            },
            {
                "name": "Attach to running iOS on device",
                "type": "cordova",
                "request": "attach",
                "platform": "ios",
                "target": "device",
                "port": 9220,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}"
            },
            {
                "name": "Run Android on emulator",
```

```
            "type": "cordova",
            "request": "launch",
            "platform": "android",
            "target": "emulator",
            "port": 9222,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}",
            "ionicLiveReload": false
        },
        {
            "name": "Run iOS on simulator",
            "type": "cordova",
            "request": "launch",
            "platform": "ios",
            "target": "emulator",
            "port": 9220,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}",
            "ionicLiveReload": false
        },
        {
            "name": "Attach to running android on emulator",
            "type": "cordova",
            "request": "attach",
            "platform": "android",
            "target": "emulator",
            "port": 9222,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}"
        },
        {
            "name": "Attach to running iOS on simulator",
            "type": "cordova",
            "request": "attach",
            "platform": "ios",
            "target": "emulator",
            "port": 9220,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}"
        },
        {
            "name": "Serve to the browser (ionic serve)",
            "type": "cordova",
            "request": "launch",
            "platform": "serve",
            "cwd": "${workspaceRoot}",
            "devServerAddress": "localhost",
            "sourceMaps": true,
            "ionicLiveReload": true
        },
        {
            "name": "Simulate Android in browser",
            "type": "cordova",
            "request": "launch",
            "platform": "android",
            "target": "chrome",
            "simulatePort": 8000,
            "livereload": true,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}"
        },
```

```
            {
                "name": "Simulate iOS in browser",
                "type": "cordova",
                "request": "launch",
                "platform": "ios",
                "target": "chrome",
                "simulatePort": 8000,
                "livereload": true,
                "sourceMaps": true,
                "cwd": "${workspaceRoot}"
            }
]
}
```

Después de la configuración, siga los siguientes pasos o claves breves para la depuración:

1. Ir al menú de **depuración** .
2. Haga clic en **iniciar depuración** .

   o

```
Short keys
```

- Depuración - F5
- Paso a paso - F10
- Entrar y salir - F11
- Detener depuración - Shift + F5
- Reinicie la depuración -ctrl + shift_F5

Lea Configuración y depuración de Ionic 2 en Visual Studio Code en línea: https://riptutorial.com/es/ionic2/topic/10559/configuracion-y-depuracion-de-ionic-2-en-visual-studio-code

Capítulo 6: Constructor y OnInit

Introducción

Con respecto a ionic2, el `constructor` : en términos simples, lo usamos para crear una instancia de nuestros complementos, servicios, etc., por ejemplo: tiene una página (vista) donde desea mostrar la lista de todos los estudiantes y tiene un archivo json que contiene a todos los estudiantes (este archivo es su archivo de datos) lo que debe hacer es crear un servicio en este servicio, creará un método y realizará una solicitud http.get para obtener los datos de json, así que, ¿aquí necesita qué? http simplemente hacerlo de esta manera:

Examples

Ejemplo de Método de Servicio al Estudiante para usar Http en el constructor

```
import {Http} from '@angular/http';
@Injectable()
export class StudentService{
    constructor(public http: Http){}
    getAllStudents(): Observable<Students[]>{
        return this.http.get('assets/students.json')
        .map(res => res.json().data)
        }
    }
```

note el constructor ahora nuevamente, si queremos usar este método de servicio, iremos a nuestra vista / página y:

```
import {StudentService} from './student.service';
import { SocialSharing } from '@ionic-native/social-sharing';
export class HomePage implements OnInit {

  constructor(public _studentService: StudentService, public socialSharing: SocialSharing) {
   }
```

nuevamente, note el constructor aquí, estamos creando una instancia de StudentService en el constructor y una cosa más, estamos usando el complemento socialSharing para usar que también estamos creando una instancia de eso en el constructor.

Método ngOnInit para obtener la lista de estudiantes en la vista de carga

`OnInit` : esto es algo realmente asombroso en ionic2 o podemos decir en AngularJs2. Con el mismo ejemplo anterior podemos ver lo que es ngOnInit. Así que ya está listo con el método de servicio, ahora en su vista / página desea que los datos de la lista de estudiantes estén disponibles tan pronto como aparezca su vista, esta debería ser la primera operación que se produzca automáticamente en la carga, porque a medida que la vista carga al estudiante La lista debe ser visible. Entonces la clase implementa OnInit y usted define ngOnInit. Ejemplo:

Ejemplo de ngOnInit para obtener la lista de estudiantes en la página / vista

```
export class HomePage implements OnInit {
...
....
constructor(....){}

ngOnInit(){
    this._studentService.getAllStudents().subscribe(
     (students: Students[]) => this.students = students,
    )
```

Lea Constructor y OnInit en línea: https://riptutorial.com/es/ionic2/topic/9907/constructor-y-oninit

Capítulo 7: Del código a la tienda de aplicaciones - Android

Introducción

Encontrará instrucciones paso a paso sobre cómo preparar y cargar la aplicación ionic de producción en Google Play.

Examples

Listo para producción

Creación de proyecto de aplicación

Al crear una aplicación de Android lista para la tienda de aplicaciones, es importante que al utilizar el `ionic start` , agregue las `--appname|-a` y `--id|-i` que se usa para Google Play para identificar su aplicación desde otras aplicaciones.

Si está comenzando un nuevo proyecto de aplicación móvil, puede usar el siguiente ejemplo de cli.

```
$ ionic start --v2 -a "App Example" -i "com.example.app" -t "tabs"
```

1. Archivo de configuración de la aplicación

Si desea configurar esta información dentro de una aplicación existente, puede modificar `config.xml` . Recomiendo a los que usaron el comando anterior para modificar `config.xml` también.

Confirmar / editar la `widget id` , `name` , `description` y atributos de `author` .

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<widget id="com.example.app" version="1.0.0" xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0">
  <name>Example App</name>
  <description>Example app for stackoverflow users</description>
  <author email="admin@example.com" href="http://example.com/">Your name or team</author>
  ...
</widget>
```

2. Icono y pantalla de bienvenida.

Los tipos de archivos compatibles con el icono y la imagen de bienvenida son png, psd o ai y deben tener un nombre de archivo que corresponda a lo que es el `icon` o la `splash` y se coloca debajo del directorio de recursos en la raíz de su proyecto. Las dimensiones mínimas de la

imagen del icono deben ser 192x192 px y no deben tener esquinas redondeadas. y la pantalla de inicio es mucho más complicada, así que haga clic aquí para leer más. No obstante, las dimensiones mínimas deben ser de 2208x2208 px.

Si tiene un archivo de íconos para generar, use este comando. `ionic resources --icon` Si tiene un archivo de bienvenida para generar, use este comando. `ionic resources --splash`

3. Aplicación de producción de edificios.

Antes de crear su aplicación de producción, elimine los datos de registro confidenciales.

Para crear una versión de lanzamiento con todas las optimizaciones predeterminadas en su lugar, use la etiqueta --release & --prod

```
ionic build android --release --prod
```

Para obtener una lista completa de las optimizaciones disponibles, puede visitar el repositorio @ionic / app-scripts

4. Crear clave privada

Ahora, debemos firmar el APK sin firmar (`android-release-unsigned.apk`) y ejecutar una utilidad de alineación para optimizarlo y prepararlo para la tienda de aplicaciones. Si ya tiene una clave de firma, omita estos pasos y use esa en su lugar.

A continuación, localice su archivo APK sin firmar `android-release-unsigned.apk` dentro de dir `/platforms/android/build/outputs/apk/` y use el comando `keytools` que se usará para firmar nuestro archivo apk. Puedes usar el siguiente ejemplo:

```
$ keytool -genkey -v -keystore my-release-key.keystore -alias androidKey -keyalg RSA -keysize
2048 -validity 10000
```

puede encontrar `my-release-key.keystore` en su directorio actual.

Generemos nuestra clave privada usando el comando keytool que viene con el JDK. Si no se encuentra esta herramienta, consulte la guía de instalación:

Primero se le pedirá que cree una contraseña para el almacén de claves. Luego, responda al resto de las preguntas de las herramientas útiles y, cuando todo haya terminado, debe tener un archivo llamado my-release-key.keystore creado en el directorio actual.

Nota: asegúrese de guardar este archivo en un lugar seguro, si lo pierde, ¡no podrá enviar actualizaciones a su aplicación!

5. Firmar APK

Para firmar el APK sin firmar, ejecute la herramienta jarsigner que también se incluye en el JDK:

```
$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.keystore
HelloWorld-release-unsigned.apk alias_name
```

Esto firma el apk en su lugar. Finalmente, necesitamos ejecutar la herramienta de alineación zip para optimizar el APK. La herramienta zipalign se puede encontrar en / path / to / Android / sdk / build-tools / VERSION / zipalign.

```
$ zipalign -v 4 HelloWorld-release-unsigned.apk HelloWorld.apk
```

¡Ahora tenemos nuestro binario de lanzamiento final llamado HelloWorld.apk y podemos lanzarlo en Google Play Store para que lo disfrute todo el mundo!

Publica tu aplicación en Google Play Store. Ahora que tenemos nuestro APK de lanzamiento listo para Google Play Store, podemos crear una lista de Play Store y cargar nuestro APK. Para comenzar, deberá visitar la Consola de desarrollador de Google Play Store y crear una nueva cuenta de desarrollador. Costará $ 25 una tarifa única.

Una vez que tenga una cuenta de desarrollador, puede seguir adelante y hacer clic en "Publicar una aplicación de Android en Google Play" y seguir las instrucciones en pantalla.

Lea Del código a la tienda de aplicaciones - Android en línea:
https://riptutorial.com/es/ionic2/topic/9659/del-codigo-a-la-tienda-de-aplicaciones---android

Capítulo 8: Examen de la unidad

Introducción

La prueba de unidades en general le da seguridad adicional a un producto para evitar problemas al modificar / agregar funciones. Una red de seguridad que dice "TODO TODO FUNCIONA". Las pruebas unitarias no reemplazan de ninguna manera las pruebas reales que puede realizar un control de calidad adecuado.

En este documento basaremos los ejemplos en este repositorio: https://github.com/driftyco/ionic-unit-testing-example

Examples

Pruebas unitarias con karma / jazmín

La prueba de unidad en iónico es la misma que en cualquier aplicación angular.

Usaremos algunos marcos para hacer esto.

Karma - un marco para ejecutar pruebas

Jasmine - un marco para escribir pruebas

PhantomJS - una aplicación que ejecuta javascript sin un navegador

En primer lugar, instalemos todo, así que asegúrese de que su package.json incluya estas líneas en las dependencias de desarrollo. Creo que es importante tener en cuenta que las dependencias de desarrollo no afectan en absoluto a tu aplicación y solo están ahí para ayudar al desarrollador.

```
"@ionic/app-scripts": "1.1.4",
"@ionic/cli-build-ionic-angular": "0.0.3",
"@ionic/cli-plugin-cordova": "0.0.9",
"@types/jasmine": "^2.5.41",
"@types/node": "^7.0.8",
"angular2-template-loader": "^0.6.2",
"html-loader": "^0.4.5",
"jasmine": "^2.5.3",
"karma": "^1.5.0",
"karma-chrome-launcher": "^2.0.0",
"karma-jasmine": "^1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"karma-sourcemap-loader": "^0.3.7",
"karma-webpack": "^2.0.3",
"null-loader": "^0.1.1",
"ts-loader": "^2.0.3",
"typescript": "2.0.9"
```

Repasar un poco los paquetes.

```
"angular2-template-loader": "^0.6.2", - will load and compile the angular2 html files.

"ts-loader": "^2.0.3", - will compile the actual typescript files

"null-loader": "^0.1.1", - will not load the assets that will be missing, such as fonts and
images. We are testing, not image lurking.
```

También deberíamos agregar este script a nuestros scripts package.json:

```
"test": "karma start ./test-config/karma.conf.js"
```

También tome nota en tsconfig que está excluyendo los archivos spec.ts de la compilación:

```
 "exclude": [
    "node_modules",
    "src/**/*.spec.ts"
  ],
```

Ok, ahora vamos a tomar la configuración de prueba real. Cree una carpeta de `test-config` en su carpeta de proyecto. (Tal como se mencionó en el script package.json) Dentro de la carpeta, cree 3 archivos:

`webpack.test.js` - que le dirá al paquete web qué archivos cargar para el proceso de prueba

```
var webpack = require('webpack');
var path = require('path');

module.exports = {
  devtool: 'inline-source-map',

  resolve: {
    extensions: ['.ts', '.js']
  },

  module: {
    rules: [
      {
        test: /\.ts$/,
        loaders: [
          {
            loader: 'ts-loader'
          } , 'angular2-template-loader'
        ]
      },
      {
        test: /\.html$/,
        loader: 'html-loader'
      },
      {
        test: /\.(png|jpe?g|gif|svg|woff|woff2|ttf|eot|ico)$/,
        loader: 'null-loader'
      }
    ]
  },

  plugins: [
```

```
    new webpack.ContextReplacementPlugin(
      // The (\\|\/) piece accounts for path separators in *nix and Windows
      /angular(\\|\/)core(\\|\/)(esm(\\|\/)src|src)(\\|\/)linker/,
      root('./src'), // location of your src
      {} // a map of your routes
    )
  ]
};

function root(localPath) {
  return path.resolve(__dirname, localPath);
}
```

`karma-test-shim.js` - que cargará las bibliotecas relacionadas angulares, como las bibliotecas de zona y de prueba, además de configurar el módulo para la prueba.

```
Error.stackTraceLimit = Infinity;

require('core-js/es6');
require('core-js/es7/reflect');

require('zone.js/dist/zone');
require('zone.js/dist/long-stack-trace-zone');
require('zone.js/dist/proxy');
require('zone.js/dist/sync-test');
require('zone.js/dist/jasmine-patch');
require('zone.js/dist/async-test');
require('zone.js/dist/fake-async-test');

var appContext = require.context('../src', true, /\.spec\.ts/);

appContext.keys().forEach(appContext);

var testing = require('@angular/core/testing');
var browser = require('@angular/platform-browser-dynamic/testing');

testing.TestBed.initTestEnvironment(browser.BrowserDynamicTestingModule,
browser.platformBrowserDynamicTesting());
```

`karma.conf.js` - define la configuración de cómo probar con karma. Aquí puede cambiar de Chrome a PhantomJS para que este proceso sea invisible y más rápido, entre otras cosas.

```
var webpackConfig = require('./webpack.test.js');

module.exports = function (config) {
  var _config = {
    basePath: '',

    frameworks: ['jasmine'],

    files: [
      {pattern: './karma-test-shim.js', watched: true}
    ],

    preprocessors: {
      './karma-test-shim.js': ['webpack', 'sourcemap']
    },
```

```
    webpack: webpackConfig,

    webpackMiddleware: {
      stats: 'errors-only'
    },

    webpackServer: {
      noInfo: true
    },

    browserConsoleLogOptions: {
      level: 'log',
      format: '%b %T: %m',
      terminal: true
    },

    reporters: ['kjhtml', 'dots'],
    port: 9876,
    colors: true,
    logLevel: config.LOG_INFO,
    autoWatch: true,
    browsers: ['Chrome'],
    singleRun: false
  };

  config.set(_config);
};
```

Ahora que hemos configurado todo, vamos a escribir alguna prueba real. Para este ejemplo escribiremos un archivo de especificaciones de app.component. Si desea ver las pruebas de una página y no el componente principal, puede consultar aquí: https://github.com/driftyco/ionic-unit-testing-example/blob/master/src/pages/page1/page1. espec.ts

Lo que debemos hacer primero es probar nuestro constructor. Esto creará y ejecutará el constructor de nuestra app.component

```
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [MyApp],
      imports: [
        IonicModule.forRoot(MyApp)
      ],
      providers: [
        StatusBar,
        SplashScreen
      ]
    })
  }));
```

La declaración incluirá nuestra principal aplicación iónica. Las importaciones serán las importaciones necesarias para esta prueba. No todo.

Los proveedores incluirán las cosas que se inyectan en el constructor, pero que no forman parte de la importación. Por ejemplo, app.component inyecta el servicio de la Plataforma, pero como es parte de IonicModule, no es necesario mencionarlo en los proveedores.

Para las próximas pruebas necesitaremos obtener una instancia de nuestro componente:

```
  beforeEach(() => {
    fixture = TestBed.createComponent(MyApp);
    component = fixture.componentInstance;
  });
```

A continuación algunas pruebas para ver que todo está en orden:

```
  it ('should be created', () => {
    expect(component instanceof MyApp).toBe(true);
  });

  it ('should have two pages', () => {
    expect(component.pages.length).toBe(2);
  });
```

Así que al final tendremos algo como esto:

```
import { async, TestBed } from '@angular/core/testing';
import { IonicModule } from 'ionic-angular';

import { StatusBar } from '@ionic-native/status-bar';
import { SplashScreen } from '@ionic-native/splash-screen';

import { MyApp } from './app.component';

describe('MyApp Component', () => {
  let fixture;
  let component;

  beforeEach(async(() => {
    TestBed.configureTestingModule({
      declarations: [MyApp],
      imports: [
        IonicModule.forRoot(MyApp)
      ],
      providers: [
        StatusBar,
        SplashScreen
      ]
    })
  }));

  beforeEach(() => {
    fixture = TestBed.createComponent(MyApp);
    component = fixture.componentInstance;
  });

  it ('should be created', () => {
    expect(component instanceof MyApp).toBe(true);
  });

  it ('should have two pages', () => {
    expect(component.pages.length).toBe(2);
  });

});
```

Ejecutar las pruebas por

```
npm run test
```

Y eso es todo para las pruebas básicas. Hay algunas formas de atajar la escritura de prueba, como escribir su propio TestBed y tener herencia en las pruebas que podrían ayudarlo a largo plazo.

Lea Examen de la unidad en línea: https://riptutorial.com/es/ionic2/topic/9561/examen-de-la-unidad

Capítulo 9: Geolocalización

Examples

Uso simple

En su `package.json` asegúrese de incluir las dependencias:

```
{
    ...
    "dependencies": {
        ...
        "ionic-native": "^1.3.10",
        ...
    },
    ...
}
```

Para utilizar la geolocalización:

```
// custom-component.ts

import {Geolocation} from 'ionic-native';
import template from './custom-component.html';

@Component({
    selector: 'custom-component',
    template: template
})
export class CustomComponent {

    constructor() {

        // get the geolocation through a promise
        Geolocation.getCurrentPosition().then((position:Geoposition)=> {
            console.log(
                position.coords.latitude,
                position.coords.longitude);
        });
    }
}
```

Viendo la posición

Para una solución más en tiempo real, puede usar la función watchPosition en Geolocation que notifica cada vez que ocurre un error o un cambio de posición. A diferencia de getCurrentPosition, watchPosition devuelve un Observable

```
import {Geolocation} from 'ionic-native';
import template from './custom-component.html';

@Component({
```

```
selector: 'custom-component',
template: template
})
export class CustomComponent {
constructor() {

    // get the geolocation through an observable
        Geolocation.watchPosition(<GeolocationOptions>{
            maximumAge: 5000, // a maximum age of cache is 5 seconds
            timeout: 10000, // time out after 10 seconds
            enableHighAccuracy: true // high accuracy
        }).subscribe((position) => {
            console.log('Time:' + position.timestamp);
            console.log(
                'Position:' + position.coords.latitude + ',' +
                position.coords.longitude);
            console.log('Direction:' position.coords.heading);
            console.log('Speed:' position.coords.speed);

        });
}
```

Lea Geolocalización en línea: https://riptutorial.com/es/ionic2/topic/5840/geolocalizacion

Capítulo 10: InAppBrowser

Introducción

A veces, el cliente solo necesita abrir una aplicación web en la aplicación móvil, para esto podemos usar InAppBrowser de tal manera que se parece a una aplicación en lugar de eso, estamos abriendo un sitio web / aplicación web en el dispositivo móvil, tan pronto como el usuario toque el ícono de la aplicación En lugar de abrir la primera vista de la aplicación, podemos abrir InAppBrowser directamente.

Examples

Un ejemplo vivo de este uso es esta aplicación:

En esta aplicación, abro directamente InAppBrowser cuando el usuario toca el ícono de la aplicación en lugar de cargar la primera página de la aplicación. Por lo tanto, a los usuarios les parecería que están viendo la aplicación del mismo sitio web / aplicación web.

Ejemplo de código para usar InAppBrowser

```
  platform.ready().then(() => {
      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
      var url= "https://blog.knoldus.com/";
      var browserRef = window.cordova.InAppBrowser.open(url, "_self", "location=no",
"toolbar=no");
      browserRef.addEventListener("exit", (event) => {
          return navigator["app"].exitApp();
        }
      );
```

Lea InAppBrowser en línea: https://riptutorial.com/es/ionic2/topic/9801/inappbrowser

Capítulo 11: Inicio de sesión social con Angularfire2 / Firebase

Examples

Inicio de sesión nativo en Facebook con Angularfire2 / Firebase

app.ts

```
import {Component} from '@angular/core';
import {Platform, ionicBootstrap} from 'ionic-angular';
import {StatusBar} from 'ionic-native';
import {LoginPage} from './pages/login/login';
import {FIREBASE_PROVIDERS, defaultFirebase, AuthMethods, AuthProviders, firebaseAuthConfig}
from 'angularfire2';

@Component({
  template: '<ion-nav [root]="rootPage"></ion-nav>'
})

export class MyApp {

  private rootPage: any;

  constructor(private platform: Platform) {
    this.rootPage = LoginPage;

    platform.ready().then(() => {
      // Okay, so the platform is ready and our plugins are available.
      // Here you can do any higher level native things you might need.
      StatusBar.styleDefault();
    });
  }
}

ionicBootstrap(MyApp, [
  FIREBASE_PROVIDERS,
  defaultFirebase({
    apiKey: myAppKey,
    authDomain: 'myapp.firebaseapp.com',
    databaseURL: 'https://myapp.firebaseio.com',
    storageBucket: 'myapp.appspot.com',
  }),
  firebaseAuthConfig({})
]);
```

login.html

```
<ion-header>
  <ion-navbar>
    <ion-title>Home</ion-title>
  </ion-navbar>
</ion-header>
```

```
<ion-content padding class="login">
  <button (click)="facebookLogin()">Login With Facebook</button>
</ion-content>
```

login.ts

```
import {Component} from '@angular/core';
import {Platform} from 'ionic-angular';
import {AngularFire, AuthMethods, AuthProviders} from 'angularfire2';
import {Facebook} from 'ionic-native';

declare let firebase: any; // There is currently an error with the Firebase files, this will
fix it.

@Component({
  templateUrl: 'build/pages/login/login.html'
})
export class LoginPage {

  constructor(private platform: Platform, public af: AngularFire) {

  }

  facebookLogin() {
    Facebook.login(['public_profile', 'email', 'user_friends'])
      .then(success => {
        console.log('Facebook success: ' + JSON.stringify(success));
        let creds =
firebase.auth.FacebookAuthProvider.credential(success.authResponse.accessToken);
        this.af.auth.login(creds, {
          provider: AuthProviders.Facebook,
          method: AuthMethods.OAuthToken,
          remember: 'default',
          scope: ['email']
        }).then(success => {
          console.log('Firebase success: ' + JSON.stringify(success));
        }).catch(error => {
          console.log('Firebase failure: ' + JSON.stringify(error));
        });
      }).catch(error => {
        console.log('Facebook failure: ' + JSON.stringify(error));
      });
  }
}
```

Lea Inicio de sesión social con Angularfire2 / Firebase en línea:
https://riptutorial.com/es/ionic2/topic/5518/inicio-de-sesion-social-con-angularfire2---firebase

Capítulo 12: Modales

Examples

Usando Modales

Los modales se deslizan fuera de la pantalla para mostrar una IU temporal, que a menudo se usa para páginas de inicio de sesión o registro, composición de mensajes y selección de opciones.

```
import { ModalController } from 'ionic-angular';
import { ModalPage } from './modal-page';

export class MyPage {
  constructor(public modalCtrl: ModalController) {
  }

  presentModal() {
    let modal = this.modalCtrl.create(ModalPage);
    modal.present();
  }
}
```

NOTA: Un modal es un panel de contenido que recorre la página actual del usuario.

Pasando datos a través de un modal

Los datos se pueden pasar a un nuevo modal a través de `Modal.create()` como segundo argumento. Se puede acceder a los datos desde la página abierta inyectando `NavParams` . Tenga en cuenta que la página, que se abrió como modal, no tiene una lógica "modal" especial dentro de ella, pero utiliza `NavParams` diferente a una página estándar.

Primera página:

```
import { ModalController, NavParams } from 'ionic-angular';

export class HomePage {

 constructor(public modalCtrl: ModalController) {

 }

 presentProfileModal() {
   let profileModal = this.modalCtrl.create(Profile, { userId: 8675309 });
   profileModal.present();
 }

}
```

Segunda pagina:

```
import { NavParams } from 'ionic-angular';
```

```
export class Profile {

 constructor(params: NavParams) {
   console.log('UserId', params.get('userId'));
 }

}
```

Lea Modales en línea: https://riptutorial.com/es/ionic2/topic/6415/modales

Capítulo 13: Modales

Examples

Modal simple

Modal es una interfaz de usuario temporal que se muestra en la parte superior de la página actual. Esto se usa a menudo para iniciar sesión, registrarse, editar opciones existentes y seleccionar opciones.

Veamos un ejemplo simple con los modales utilizados. Para empezar estamos creando un proyecto en blanco iónico. Creamos un modo simple que muestre un mensaje y salgamos al hacer clic en el botón. Para hacer eso primero estamos creando vista para nuestro modal.

Mensaje.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Modal
    </ion-title>
    <ion-buttons start>
      <button (click)="dismiss()">
        <span primary showWhen="ios">Cancel</span>
        <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <h1>Modal Without Params is created successfully.</h1>
  <button full (click)="dismiss()"> Exit </button>
</ion-content>
```

Mensaje.ts

```
import { Component } from '@angular/core';
import { ViewController } from 'ionic-angular';
@Component({
  templateUrl: 'build/pages/message/message.html',
})
export class MessagePage {
  viewCtrl;
  constructor(viewCtrl: ViewController) {
    this.viewCtrl = viewCtrl;
  }
  dismiss(){
    this.viewCtrl.dismiss();
  }
}
```

Este modal muestra un mensaje. El modal se puede cerrar o "descartar" utilizando el método de

desconexión de los controladores de vista.

Inicio.html

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Modal Example
    </ion-title>
  </ion-navbar>
</ion-header>
<ion-content padding>
  <button full (click)="openModal()">ModalWithoutParams-Message</button>
</ion-content>
```

Inicio.ts

```
import { Component } from '@angular/core';
import { ModalController } from 'ionic-angular';
import {MessagePage} from '../message/message';
@Component({
  templateUrl: 'build/pages/home/home.html'
})
export class HomePage {
  modalCtrl;
  data;
  constructor(modalCtrl: ModalController) {
    this.modalCtrl = modalCtrl;
    this.data = [{name: "aaa", email: "aaa.a@som.com", mobile: "1234567890", nickname: "zzz"},
      {name: "bbb", email: "bbb.a@som.com", mobile: "1234567890", nickname: "yyy"},
      {name: "ccc", email: "ccc.a@som.com", mobile: "1234567890", nickname: "xxx"}]
  }
  openModal() {
    let myModal = this.modalCtrl.create(MessagePage);
    myModal.present();
  }
}
```

Ahora estamos creando nuestra página de inicio importando el **ModalController** y nuestro modelo de datos MessagePage. El método de **creación** de ModalController crea modal para nuestro modelo de datos MessagePage que se guarda para controlar la variable myModal. **El** método **presente** abre el modal en la parte superior de nuestra página actual.

Modal con Parámetros en despedir:

Ahora sabemos cómo crear un modal. Pero qué pasa si queremos pasar algunos datos de modal a nuestra página de inicio. Para hacerlo, veamos un ejemplo con modal como Registro de la página que pasa parámetros a la página principal.

Registrarse.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>
```

```
      Login
    </ion-title>
    <ion-buttons start>
      <button (click)="dismiss()">
        <span primary showWhen="ios">Cancel</span>
        <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <ion-list>
    <ion-item>
      <ion-label>Name</ion-label>
      <ion-input type="text" [(ngModel)]="name"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Email</ion-label>
      <ion-input type="text" [(ngModel)]="email"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Mobile</ion-label>
      <ion-input type="number" [(ngModel)]="mobile"></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Nickname</ion-label>
      <ion-input type="text" [(ngModel)]="nickname"></ion-input>
    </ion-item>
  </ion-list>
  <button full (click)="add()">Add</button>
</ion-content>
```

Registrarse.ts

```
import { Component } from '@angular/core';
import { ViewController } from 'ionic-angular';
@Component({
  templateUrl: 'build/pages/register/register.html',
})
export class ResisterPage {
  viewCtrl;
  name;
  email;
  mobile;
  nickname;
  constructor(viewCtrl: ViewController) {
    this.viewCtrl = viewCtrl;
    this.name = "";
    this.email = "";
    this.mobile = "";
    this.nickname = "";
  }
  dismiss(){
    this.viewCtrl.dismiss();
  }
  add(){
    let data = {"name": this.name, "email": this.email, "mobile": this.mobile, "nickname":
this.nickname};
    this.viewCtrl.dismiss(data);
  }
```

```
}
```

El registro modal obtiene el objeto de datos con valores ingresados por el usuario y los parámetros se pasan a nuestra página actual al descartar con el método de desconexión de viewControllers. Ahora se envían los parámetros.

Entonces, ¿cómo vamos a recuperar los parámetros en la página de inicio? Para hacerlo, estamos creando un botón en la página de inicio y llamamos Registrar modal al hacer clic. Para mostrar el usuario, estamos mostrando una lista.

Inicio.html

```
<ion-list>
  <ion-item *ngFor="let datum of data">
    <h1>{{datum.name}}</h1>
  </ion-item>
</ion-list>
<button full secondary (click)="openModalParams()">ModalWithParams-Register</button>
```

Inicio.ts

```
import {ResisterPage} from '../register/register';

  openModalParams(){
    let modalWithParams = this.modalCtrl.create(ResisterPage);
    modalWithParams.present();

    modalWithParams.onDidDismiss((result) =>{
      if(result){
        this.data.unshift(result);
      }
    });
  }
```

El método ViewController **onDidDismiss** se ejecuta cada vez que se cierra un modal. Si los datos se pasan como parámetros desde el modal, podemos recuperarlos utilizando el método onDidDismiss. Aquí los datos introducidos por el usuario se anexan a los datos existentes. Si no se pasan datos como parámetro, el valor devuelto será nulo.

Modal con parámetros en crear:

Pasar parámetros a un modal es similar a cómo pasamos valores a un NavController. Para hacerlo, estamos modificando nuestra lista en home.html para abrir un modal al hacer clic en un elemento de la lista y pasar los parámetros necesarios como un segundo argumento al método de **creación** .

Inicio.html

```
<ion-list>
   <ion-item *ngFor="let datum of data" (click)="openModalwithNavParams(datum)">
     <h1>{{datum.name}}</h1>
   </ion-item>
```

```
  </ion-list>
```

Inicio.ts

```
import {EditProfilePage} from '../edit-profile/edit-profile';

  openModalwithNavParams(data){
    let modalWithNavParams = this.modalCtrl.create(EditProfilePage,{Data: data});
    modalWithNavParams.present();
  }
```

Similar a otras vistas, usamos NavParams para recuperar los datos enviados desde la vista anterior.

Edit-Profile.html

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      Login
    </ion-title>
    <ion-buttons start>
      <button (click)="dismiss()">
        <span primary showWhen="ios">Cancel</span>
        <ion-icon name="md-close" showWhen="android,windows"></ion-icon>
      </button>
    </ion-buttons>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <h2>Welcome {{name}}</h2>
  <ion-list>
    <ion-item>
      <ion-label>Email</ion-label>
      <ion-input type="text" value={{email}}></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Mobile</ion-label>
      <ion-input type="number" value={{mobile}}></ion-input>
    </ion-item>
    <ion-item>
      <ion-label>Nickname</ion-label>
      <ion-input type="text" value={{nickname}}></ion-input>
    </ion-item>
  </ion-list>
  <button full (click)="dismiss()">Close</button>
</ion-content>
```

Editar-perfil.ts

```
import { Component } from '@angular/core';
import { ViewController, NavParams } from 'ionic-angular';
@Component({
  templateUrl: 'build/pages/edit-profile/edit-profile.html',
})
export class EditProfilePage {
  viewCtrl;
```

```
  navParams;
  data;
  name;
  email;
  mobile;
  nickname;
  constructor(viewCtrl: ViewController, navParams: NavParams) {
    this.viewCtrl = viewCtrl;
    this.navParams = navParams;
    this.data = this.navParams.get('Data');
    this.name = this.data.name;
    this.email = this.data.email;
    this.mobile = this.data.mobile;
    this.nickname = this.data.nickname;

  }
  dismiss(){
    this.viewCtrl.dismiss();
  }
}
```

Lea Modales en línea: https://riptutorial.com/es/ionic2/topic/6612/modales

Capítulo 14: Notificaciones push enviadas y recibidas

Observaciones

El SenderID que está presente en el ejemplo de inicialización es un ID de remitente gcm que le proporciona Google. También debería estar presente al instalar el complemento.

```
ionic plugin add phonegap-plugin-push --variable SENDER_ID="XXXXXXX"
```

Si desea agregar datos adicionales a sus notificaciones push, consulte este enlace que explica cómo agregar más tipografías https://github.com/phonegap/phonegap-plugin-push/blob/master/docs/TYPESCRIPT.md

Examples

Inicialización

El complemento de notificación de inserción requiere un inicio y una inicialización que le indica al complemento que comience a ejecutarse con la identificación del remitente proporcionada.

```
 let push = Push.init({
     android: {
       senderID: "------------",
     },
     ios: {
       alert: "true",
       badge: true,
       sound: "false",
     },
     windows: {},
   });
```

Registro

El paso de registro registra la aplicación con el sistema del dispositivo y devuelve un ID de registro

```
import { Push, RegistrationEventResponse} from "ionic-native";

       //the push element is created in the initialization example
       push.on("registration", async (response: RegistrationEventResponse) => {
                   //The registration returns an id of the registration on your device
                   RegisterWithWebApi(response.registrationId);

       });
```

Recibir una notificación de inserción

Para recibir notificaciones push, se supone que debemos decirle al complemento que escuche las notificaciones push entrantes. Este paso se realiza después de la inicialización y registro.

```
import { Push, NotificationEventResponse} from "ionic-native";

        //the push element is created in the initialization example
        push.on("notification", (response: NotificationEventResponse) => {
            let chatMessage: ChatMessage = <ChatMessage>{
              title: response.title,
              message: response.message,
              receiver: response.additionalData.replyTo,
              image: response.image
            };
            DoStuff(chatMessage));
       });
```

Lea Notificaciones push enviadas y recibidas en línea:
https://riptutorial.com/es/ionic2/topic/5874/notificaciones-push-enviadas-y-recibidas

Capítulo 15: Solución para 'show-delete' en deprecación

Examples

Solución

Estoy desarrollando una aplicación móvil que usa ionic 2 con Angular 2.

Tengo una lista de iones llena de iones artículos. Quiero que esos iones ión tengan la capacidad de ser eliminados si es necesario, tal como se presenta aquí en el sitio web iónico.

Sin embargo, muchas cosas han cambiado en el **ionic 2** desde que la primera versión y el estilo anterior de un botón que abre todo el **elemento de ión** a la vez ya no es posible ya que ya no se admiten la **eliminación de programas y la reordenación de programas** . La única opción disponible es tener el **ión-elemento deslizante** como ión-elemento, lo que nos da la posibilidad de deslizar cada elemento uno a la vez para revelar el botón Eliminar.

Eso no es lo que quería. Quería un botón que abra todos los iones al mismo tiempo.

Después de dedicar un tiempo a eso, se me ocurrió una solución de trabajo y logré el resultado deseado con ionic 2, y lo compartiré con usted.

Aquí está mi solución:

En el archivo .html:

```
<ion-header>
  <ion-navbar>
    <ion-buttons start (click)="manageSlide()">
      <button>
        <ion-icon name="ios-remove"></ion-icon>
      </button>
    </ion-buttons>
    <ion-title>PageName</ion-title>
  </ion-navbar>
</ion-header>
```

y para la lista:

```
<ion-list #list1>
  <ion-item-sliding #slidingItem *ngFor="let contact of contacts | sortOrder">
    <button #item ion-item>
      <p>{{ item.details }}</p>
      <ion-icon id="listIcon" name="arrow-forward" item-right></ion-icon>
    </button>
    <ion-item-options side="left">
      <button danger (click)="doConfirm(contact, slidingItem)">
        <ion-icon name="ios-remove-circle-outline"></ion-icon>
```

```
      Remove
      </button>
    </ion-item-options>
  </ion-item-sliding>
</ion-list>
```

En el archivo **.ts** , primero haga sus importaciones:

```
import { ViewChild } from '@angular/core';
import { Item } from 'ionic-angular';
import { ItemSliding, List } from 'ionic-angular';
```

luego refiérase al elemento html declarando un ViewChild:

```
@ViewChild(List) list: List;
```

Finalmente, agrega tus clases para manejar el trabajo:

```
public manageSlide() {

    //loop through the list by the number retreived of the number of ion-item-sliding in the
list
    for (let i = 0; i < this.list.getElementRef().nativeElement.children.length; i++) {

        // retreive the current ion-item-sliding
        let itemSlide = this.list.getElementRef().nativeElement.children[i].$ionComponent;

        // retreive the button to slide within the ion-item-sliding
        let item = itemSlide.item;

        // retreive the icon
        let ic = item._elementRef.nativeElement.children[0].children[1];

        if (this.deleteOpened) {
            this.closeSlide(itemSlide);
        } else {
            this.openSlide(itemSlide, item, ic);
        }
    }

    if (this.deleteOpened) {
        this.deleteOpened = false;
    } else {
        this.deleteOpened = true;
    }
}
```

Luego la clase de apertura:

```
private openSlide(itemSlide: ItemSliding, item: Item, inIcon) {
  itemSlide.setCssClass("active-sliding", true);
  itemSlide.setCssClass("active-slide", true);
  itemSlide.setCssClass("active-options-left", true);
  item.setCssStyle("transform", "translate3d(72px, 0px, 0px)")
}
```

Y la clase de clausura:

```
private closeSlide(itemSlide: ItemSliding) {
  itemSlide.close();
  itemSlide.setCssClass("active-sliding", false);
  itemSlide.setCssClass("active-slide", false);
  itemSlide.setCssClass("active-options-left", false);
```

}

Espero que te sirva de ayuda.

Disfruta y buena codificación ...

Lea Solución para 'show-delete' en deprecación en línea:
https://riptutorial.com/es/ionic2/topic/6620/solucion-para--show-delete--en--ion-list--deprecacion

Capítulo 16: Usando Servicios

Observaciones

Una cosa muy importante sobre el uso de servicios compartidos, es que deben incluirse en la matriz de `providers` del componente superior donde deben compartirse.

¿Porqué es eso? Bueno, supongamos que incluimos la referencia `MyService` en la matriz de `providers` de cada `Component` . Algo como:

```
@Component({
  templateUrl:"page1.html",
  providers: [MyService]
})
```

Y

```
@Component({
  templateUrl:"page2.html",
  providers: [MyService]
})
```

De esa forma **, se creará una nueva instancia del servicio para cada componente, de** modo que la instancia en la que una página guardará los datos será diferente de la instancia utilizada para obtener los datos. Así que eso no funcionará.

Para hacer que toda la aplicación use la misma instancia (hacer que el servicio funcione como un servicio *único*), podemos agregar su referencia en el `App Component` la `App Component` siguiente manera:

```
@Component({
  template: '<ion-nav [root]="rootPage"></ion-nav>',
  providers: [MyService]
})
```

También puede agregar la referencia de `MyService` en `ionicBootstrap(MyApp, [MyService]);` Pero de acuerdo con las guías de estilo Angular2.

> Proporcione servicios al inyector Angular 2 en el componente más alto donde se compartirán.
>
> ¿Por qué? El inyector Angular 2 es jerárquico.
>
> ¿Por qué? Al proporcionar el servicio a un componente de nivel superior, esa instancia se comparte y está disponible para todos los componentes secundarios de ese componente de nivel superior.
>
> ¿Por qué? Esto es ideal cuando un servicio es compartir métodos o estado.

> ¿Por qué? Esto no es ideal cuando dos componentes diferentes necesitan instancias diferentes de un servicio. En este escenario, sería mejor proporcionar el servicio al nivel de componente que necesita la instancia nueva e independiente.

Y

> Funcionará. Simplemente no es una buena práctica. **La opción del proveedor bootstrap está diseñada para configurar y anular los servicios prerregistrados propios de Angular** , como su soporte de enrutamiento.

... el `App Component` la `App Component` sería la mejor opción.

Examples

Compartir información entre diferentes páginas.

Uno de los ejemplos más fáciles de usar *servicios compartidos* es cuando queremos almacenar algunos datos de una página determinada de nuestra aplicación, y luego obtener esos datos nuevamente, pero de otra página.

Una opción podría ser enviar esos datos como un parámetro (por ejemplo, si una página llama a la otra), pero si queremos usar esos datos desde una parte completamente diferente de la aplicación, esa no es la mejor manera de hacerlo. eso. Ahí es cuando *los servicios compartidos* entran en juego.

En este ejemplo, usaremos un servicio simple llamado `MyService` que solo tiene dos métodos simples: `saveMessage()` para almacenar una cadena y `getMessage()` para volver a obtenerla. Este código es parte de este desplumador de trabajo donde puede verlo en acción.

```
import {Injectable} from '@angular/core';

@Injectable()
export class MyService {

  private message: string;

  constructor(){ }

  public saveMessage(theMessage: string): void {
    this.message = theMessage;
  }

  public getMessage(): string {
    return this.message;
  }
}
```

Luego, cuando queremos almacenar un nuevo mensaje, solo podemos usar `saveMessage(theMessageWeWantToSave);` Método de la instancia de `MyService` (llamado simplemente `service`).

```
import { Component } from "@angular/core";
import { MyService } from 'service.ts';

@Component({
  templateUrl:"page1.html"
})
export class Page1 {

  message: string;

  // ...

  public saveSecretMessage(): void {
    this.service.saveMessage(this.message);
  }
}
```

De la misma manera, cuando queremos obtener esos datos, podemos usar el método `getMessage()` de la instancia de servicio como esta:

```
import { Component } from "@angular/core";
import { MyService } from 'service.ts';

@Component({
  templateUrl:"page2.html"
})
export class Page2 {

  enteredMessage: string;

  constructor(private service: MyService) {
    this.enteredMessage = this.service.getMessage();
  }

  // ...
}
```

Recuerde revisar la sección de *Comentarios* para ver dónde se debe `MyService` la referencia del servicio `MyService` y por qué.

Lea Usando Servicios en línea: https://riptutorial.com/es/ionic2/topic/4407/usando-servicios

Capítulo 17: Uso de pestañas

Observaciones

Siempre recuerde revisar los documentos de Ionic 2 Tab para estar al tanto de los últimos cambios y actualizaciones.

Examples

Cambiar la pestaña seleccionada programáticamente desde la página infantil

Puede echar un vistazo al código completo en este Plunker de trabajo .

En este ejemplo, uso un servicio compartido para manejar la comunicación entre las páginas dentro de la pestaña (páginas secundarias) y el contenedor de pestañas (el componente que contiene las pestañas). Aunque probablemente pueda hacerlo con Eventos, me gusta el enfoque de servicio compartido porque es más fácil de entender y también para mantener cuando la aplicación comienza a crecer.

TabService

```
import {Injectable} from '@angular/core';
import {Platform} from 'ionic-angular/index';
import {Observable} from 'rxjs/Observable';

@Injectable()
export class TabService {

  private tabChangeObserver: any;
  public tabChange: any;

  constructor(private platform: Platform){
    this.tabChangeObserver = null;
    this.tabChange = Observable.create(observer => {
        this.tabChangeObserver = observer;
    });
  }

  public changeTabInContainerPage(index: number) {
    this.tabChangeObserver.next(index);
  }
}
```

Básicamente, el `TabService` solo crea un `Observable` para permitir que el contenedor de pestañas se suscriba a él, y también declara el método `changeTabInContainerPage()` que se llamará desde las páginas secundarias.

Luego, en cada página secundaria (las que están dentro de las pestañas) solo agregamos un botón y vinculamos el evento `click` a un método que llama al servicio:

Página1.html

```
<ion-content class="has-header">
    <h1>Page 1</h1>
    <button secondary (click)="changeTab()">Select next tab</button>
</ion-content>
```

Página1.ts

```
import { Component } from '@angular/core';
import { Observable } from 'rxjs/Observable';
import { TabService } from 'tabService.ts';

@Component({
  templateUrl:"page1.html"
})
export class Page1 {

  constructor(private tabService: TabService) { }

  public changeTab() {
    this.tabService.changeTabInContainerPage(1);
  }
}
```

Y finalmente, en `TabsPage` , solo nos suscribimos al servicio, y luego cambiamos la pestaña seleccionada con `this.tabRef.select(index);`

```
import { Component, ViewChild } from "@angular/core";
import { Page1 } from './page1.ts';
import { Page2 } from './page2.ts';
import { TabService } from 'tabService.ts';

@Component({
  templateUrl: 'tabs.html'
})
export class TabsPage {
  @ViewChild('myTabs') tabRef: Tabs;

  tab1Root: any = Page1;
  tab2Root: any = Page2;

  constructor(private tabService: TabService){
    this.tabService.tabChange.subscribe((index) => {
      this.tabRef.select(index);
    });
  }
}
```

Tenga en cuenta que estamos obteniendo una referencia a la instancia de Tabs agregando `#myTabs` en el elemento `ion-tabs` , y lo obtenemos del componente con `@ViewChild('myTabs') tabRef: Tabs;`

```
<ion-tabs #myTabs>
  <ion-tab [root]="tab1Root" tabTitle="Tab 1"></ion-tab>
  <ion-tab [root]="tab2Root" tabTitle="Tab 2"></ion-tab>
```

```
</ion-tabs>
```

Cambiar pestaña con selectedIndex

En lugar de obtener una referencia al DOM, simplemente puede cambiar el índice de la pestaña usando el atributo selectedIndex en las iones de las pestañas

HTML:

```
<ion-tabs [selectedIndex]="tabIndex" class="tabs-icon-text" primary >
     <ion-tab tabIcon="list-box"   [root]="tabOne"></ion-tab>
     <ion-tab tabIcon="contacts"  [root]="tabTwo"></ion-tab>
     <ion-tab tabIcon="chatboxes"  [tabBadge]="messagesReceived" [root]="tabFive"></ion-tab>
</ion-tabs>
```

TS:

```
import { Events} from "ionic-angular";

export class tabs {
 public tabIndex: number;
 constructor(e: Events) {
   tabs.mySelectedIndex = navParams.data.tabIndex || 0;
   e.subscribe("tab:change", (newIndex) => this.tabIndex = newIndex);
 }
}
```

Si desea cambiarlo desde otro servicio de controlador, puede enviar un evento:

```
e.publish("tab:change",2);
```

Lea Uso de pestañas en línea: https://riptutorial.com/es/ionic2/topic/5569/uso-de-pestanas

Creditos

S. No	Capítulos	Contributors
1	Empezando con ionic2	Akilan Arasu, Cameron637, carstenbaumhoegger, Community, FreeBird72, Guillaume Le Mière, Ian Pinto, Ketan Akbari, misha130, Raymond Ativie, sebaferreras, tymspy, Will.Harris
2	Añadir aplicación iónica a la vista iónica	Saravanan Sachi
3	Angularfire2 con Ionic2	Fernando Del Olmo
4	Componentes CSS de Ionic2	Ketan Akbari
5	Configuración y depuración de Ionic 2 en Visual Studio Code	misha130, PRIYA PARASHAR
6	Constructor y OnInit	niks
7	Del código a la tienda de aplicaciones - Android	Luis Estevez, misha130
8	Examen de la unidad	misha130
9	Geolocalización	Matyas, misha130
10	InAppBrowser	niks
11	Inicio de sesión social con Angularfire2 / Firebase	Cameron637, Gianfranco P.
12	Modales	Raymond Ativie
13	Notificaciones push enviadas y recibidas	misha130
14	Solución para 'show-	Amr ElAdawy, Roman Lee

	delete' en deprecación	
15	Usando Servicios	sebaferreras
16	Uso de pestañas	misha130, sebaferreras