***Tutorial de temas***

Angular JS, lonicJS, Cordova, Android, IOS y Firebase

**Indice**

Contenido

[Pruébelo usted mismo Ejemplos en cada capítulo 2](#_Toc511292223)

[**Ejemplo AngularJS** 3](#_Toc511292224)

[Lo que ya deberías saber 3](#_Toc511292225)

[Ejemplos de AngularJS 4](#_Toc511292226)

[Referencia AngularJS 4](#_Toc511292227)

[**AngularJS Incluye** 4](#_Toc511292228)

[Con AngularJS, puede incluir HTML desde un archivo externo. 4](#_Toc511292229)

[**Ejemplo** 4](#_Toc511292230)

[Incluir código AngularJS 4](#_Toc511292231)

[**myTable.htm:** 5](#_Toc511292232)

[**Ejemplo** 5](#_Toc511292233)

[Incluir dominios cruzados 5](#_Toc511292234)

[**Ejemplo:** 6](#_Toc511292235)

[**Tutorial Iónico** 7](#_Toc511292236)

[Ahora que tiene Ionic y sus dependencias instaladas, ¡puede crear su primera aplicación! Esta sección lo guiará a través del proceso de iniciar una nueva aplicación, agregar páginas, navegar entre esas páginas, y más. ¡Empecemos! 7](#_Toc511292237)

[**Estructura del proyecto** 9](#_Toc511292238)

[Analicemos la anatomía de una aplicación iónica. Dentro de la carpeta que se creó, tenemos una estructura de proyecto típica de Cordova donde podemos instalar complementos nativos y crear archivos de proyecto específicos de la plataforma. 9](#_Toc511292239)

[**./src/index.html** 9](#_Toc511292240)

[Es el principal punto de entrada para la aplicación, aunque su propósito es configurar secuencias de comandos y CSS incluye y arrancar, o comenzar a ejecutar, nuestra aplicación. No pasaremos mucho de nuestro tiempo en este archivo. 9](#_Toc511292241)

[**./src/** 10](#_Toc511292242)

[**./src/app/app.html** 11](#_Toc511292243)

[**Agregar páginas** 13](#_Toc511292244)

[Ahora que tenemos una comprensión básica del diseño de una aplicación Ionic, analicemos el proceso de creación y navegación por las páginas de nuestra aplicación. 13](#_Toc511292245)

[**Creando una página** 14](#_Toc511292246)

[**Crear páginas adicionales** 17](#_Toc511292247)

[**Navegando hacia las páginas** 19](#_Toc511292248)

[Recordemos de la sección anterior que teníamos una función dentro de nuestra clase que se parecía a esto: 19](#_Toc511292249)

[**Cómo funciona?** 20](#_Toc511292250)

[**Próximos pasos** 21](#_Toc511292251)

[**Componentes** 21](#_Toc511292252)

[Hojas de acción 22](#_Toc511292253)

[Alertas 24](#_Toc511292254)

[**Uso Básico** 25](#_Toc511292255)

[**Alertas rápidas** 26](#_Toc511292256)

[**Alertas de confirmación** 27](#_Toc511292257)

[**Radio** 29](#_Toc511292258)

[Las alertas de radio son un tipo de alerta de confirmación, pero usan el componente de radio para ofrecer varias opciones. Se proporciona un conjunto de opciones para el usuario, pero solo se puede elegir una opción. 29](#_Toc511292259)

[**Caja** 30](#_Toc511292260)

[Insignias 32](#_Toc511292261)

[**Uso Básico** 32](#_Toc511292262)

[Botones 33](#_Toc511292263)

[Lo que vas a aprender: 34](#_Toc511292264)

[Requisitos 34](#_Toc511292265)

[Cuestiones 35](#_Toc511292266)

[**Módulo 1: Creando un Proyecto Cordova** 35](#_Toc511292267)

[Pasos 35](#_Toc511292268)

[**Módulo 2: Construyendo un Proyecto Cordova** 37](#_Toc511292269)

[Construyendo para iOS 37](#_Toc511292270)

[Construyendo para Android 37](#_Toc511292271)

[**Módulo 3: Configuración de los archivos del taller** 38](#_Toc511292272)

[Pasos 38](#_Toc511292273)

[**Módulo 4: Elegir una estrategia de almacenamiento de datos** 38](#_Toc511292274)

[Paso 1: Explora diferentes mecanismos de persistencia 38](#_Toc511292275)

[Paso 2: prueba la aplicación con diferentes mecanismos de persistencia 39](#_Toc511292276)

[**Módulo 5: Uso de notificaciones nativas** 40](#_Toc511292277)

[Pasos 40](#_Toc511292278)

[**Módulo 6: evitar el retardo de clic de 300 ms** 41](#_Toc511292279)

[Pasos 41](#_Toc511292280)

[**Módulo 7: Configuración de una aplicación de una sola página** 41](#_Toc511292281)

[Pasos 42](#_Toc511292282)

[**Módulo 8: Uso de plantillas de manubrios** 43](#_Toc511292283)

[Paso 1: defina las plantillas 43](#_Toc511292284)

[Paso 2: usa las plantillas 44](#_Toc511292285)

[Paso 3: reparación del problema de la barra de estado en iOS7 45](#_Toc511292286)

[**Módulo 9: Creando Vistas** 46](#_Toc511292287)

[Paso 1: crea la vista de inicio 46](#_Toc511292288)

[Paso 2: crea la vista EmployeeList 47](#_Toc511292289)

[Paso 3: Uso de la Vista de inicio 48](#_Toc511292290)

[**Módulo 10: implementación del enrutamiento de vista** 49](#_Toc511292291)

[Paso 1: crea la plantilla de empleado 49](#_Toc511292292)

[Paso 2: crea la clase EmployeeView 51](#_Toc511292293)

[Paso 3: Implementar el enrutamiento de vista 52](#_Toc511292294)

[**Audiencia** 53](#_Toc511292295)

[**Requisitos previos** 53](#_Toc511292296)

[Tutoriales de iPhone 55](#_Toc511292297)

[Idiomas 55](#_Toc511292298)

[Desarrollo de aplicaciones 55](#_Toc511292299)

[Desarrollo de aplicaciones 55](#_Toc511292300)

[Otro 55](#_Toc511292301)

[**Audiencia** 56](#_Toc511292302)

[**Requisitos previos** 56](#_Toc511292303)

[Hacer un Hola Mundo en Firebase 57](#_Toc511292304)

[Instalar Firebase en un proyecto web 58](#_Toc511292305)

[Acceso a los datos por referencia 59](#_Toc511292306)

[Evento "value" para sincronizarme con el dato 60](#_Toc511292307)

[Conexiones por Sockets 61](#_Toc511292308)

[Vídeo de introducción a Firebase 62](#_Toc511292309)

**Angular JS**



AngularJS extiende HTML con nuevos atributos.

AngularJS es perfecto para aplicaciones de una sola página (SPA).

AngularJS es fácil de aprender.

[¡Comienza a aprender AngularJS ahora!](https://www.w3schools.com/angular/angular_intro.asp)

Este tutorial está especialmente diseñado para ayudarlo a aprender AngularJS de la manera más rápida y eficiente posible.

Primero, aprenderá los conceptos básicos de AngularJS: directivas, expresiones, filtros, módulos y controladores.

Luego aprenderá todo lo demás que necesita saber sobre AngularJS:

Eventos, DOM, Formularios, Entrada, Validación, Http y más.

Pruébelo usted mismo Ejemplos en cada capítulo

En cada capítulo, puede editar los ejemplos en línea y hacer clic en un botón para ver el resultado.

### **Ejemplo AngularJS**

<!DOCTYPE html>

<html lang="en-US">

<Script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>

<body>

<div ng-app="">

<p>Name : <input type="text" ng-model="name"></p>

<h1>Hello {{name}}</h1>

</div>

</body>

</html>

## Lo que ya deberías saber

Antes de estudiar AngularJS, debe tener una comprensión básica de:

* HTML
* CSS
* JavaScript

**Historia de AngularJS**

AngularJS versión 1.0 fue lanzada en 2012.

Miško Hevery, una empleada de Google, comenzó a trabajar con AngularJS en 2009.

La idea resultó muy bien, y el proyecto ahora cuenta con el respaldo oficial de Google.

## Ejemplos de AngularJS

El tutorial AngularJS de W3Schools contiene muchos ejemplos de AngularJS.

[Ejemplos de AngularJS](https://www.w3schools.com/angular/angular_examples.asp)

## Referencia AngularJS

La referencia AngularJS contiene todas las directivas y filtros utilizados en este tutorial.

[Referencia AngularJS](https://www.w3schools.com/angular/angular_ref_directives.asp)

# **AngularJS Incluye**

# Con AngularJS, puede incluir HTML desde un archivo externo.

Con AngularJS, puede incluir contenido HTML utilizando la directiva **ng-include**:

### **Ejemplo**

<body ng-app="">  
<div ng-include="'myFile.htm'"></div>  
</body>

Pruebalo tu mismo:

<https://www.w3schools.com/angular/tryit.asp?filename=try_ng_include>

## Incluir código AngularJS

Los archivos HTML que incluye con la directiva ng-include, también pueden contener código AngularJS:

### **myTable.htm:**

<table>  
<tr ng-repeat="x in names">  
<td>{{ x.Name }}</td>  
<td>{{ x.Country }}</td>  
</tr>  
</table>

Incluya el archivo "myTable.htm" en su página web, y se ejecutará todo el código AngularJS, incluso el código dentro del archivo incluido:

### **Ejemplo**

<body>  
  
<div ng-app="myApp" ng-controller="customersCtrl">   
  <div ng-include="'myTable.htm'"></div>  
</div>  
  
<script>

var app = angular.module('myApp', []);  
app.controller('customersCtrl', function($scope, $http) {  
    $http.get("customers.php").then(function (response) {  
        $scope.names = response.data.records;  
    });  
});

</script>}

*Pruébalo tu mismo:*

<https://www.w3schools.com/angular/tryit.asp?filename=try_ng_include_table>

## Incluir dominios cruzados

De forma predeterminada, la directiva ng-include no le permite incluir archivos de otros dominios.

Para incluir archivos de otro dominio, puede agregar una lista blanca de archivos legales y / o dominios en la función de configuración de su aplicación:

### **Ejemplo:**

<body ng-app="myApp">  
  
<div ng-include="'https://tryit.w3schools.com/angular\_include.php'"></div>  
  
<script>

var app = angular.module('myApp', [])  
app.config(function($sceDelegateProvider) {  
    $sceDelegateProvider.resourceUrlWhitelist([  
        'https://tryit.w3schools.com/\*\*'  
    ]);  
});

</script>  
</body>

***Pruébalo tu mismo:***

<https://www.w3schools.com/angular/tryit.asp?filename=try_ng_include_crossdomain>

** Ionic Framework**

# **Tutorial Iónico**

# Ahora que tiene Ionic y sus dependencias instaladas, ¡puede crear su primera aplicación! Esta sección lo guiará a través del proceso de iniciar una nueva aplicación, agregar páginas, navegar entre esas páginas, y más. ¡Empecemos!

Ionic usa TypeScript para su código. Si no está familiarizado con TypeScript, no se preocupe, es muy similar al JavaScript normal. Pero si desea leer sobre él, eche un vistazo a [esta página de recursos](https://ionicframework.com/docs/resources/what-is/#typescript) .

***Comenzar una nueva aplicación Ionic***

Para iniciar una nueva aplicación, abra su terminal / símbolo del sistema y ejecute:

$ ionic start MyIonicProject tutorial

* start le dirá a la CLI que cree una nueva aplicación.
* MyIonicProject será el nombre del directorio y el nombre de la aplicación de su proyecto.
* tutorial será la plantilla inicial para su proyecto.

Junto con la creación de su proyecto, esto también instalará [módulos de nodo](https://ionicframework.com/docs/resources/what-is/#npm) para la aplicación, y le preguntará si desea configurar [Cordova](https://ionicframework.com/docs/resources/what-is/#cordova) .

Junto con la plantilla de tutorial, Ionic también proporciona las siguientes plantillas oficiales:

* tabs : un diseño simple de 3 pestañas
* sidemenu: un diseño con un menú deslizable en el lateral
* blank: un arranque simple con una sola página
* super: proyecto inicial con más de 14 diseños de página listos para usar
* tutorial: un proyecto de inicio guiado

Si no especifica una plantilla al inicio, se le pedirá que elija una.

**Ver la aplicación en un navegador**

Ahora puede ingresar a la carpeta que se creó. Para obtener una vista previa rápida de su aplicación en el navegador, use el comando.

$ cd MyIonicProject/

$ ionic serve

# **Estructura del proyecto**

# Analicemos la anatomía de una aplicación iónica. Dentro de la carpeta que se creó, tenemos una estructura de proyecto típica de Cordova donde podemos instalar complementos nativos y crear archivos de proyecto específicos de la plataforma.

### **./src/index.html**

### Es el principal punto de entrada para la aplicación, aunque su propósito es configurar secuencias de comandos y CSS incluye y arrancar, o comenzar a ejecutar, nuestra aplicación. No pasaremos mucho de nuestro tiempo en este archivo.

Para que funcione tu aplicación, Ionic busca la <ion-app> etiqueta en tu HTML. En este ejemplo tenemos:

<**ion-app**></**ion-app**>

Y los siguientes scripts cerca de la parte inferior:

*<!-- Ionic's root component and where the app will load -->*

<**ion-app**></**ion-app**>

*<!-- The polyfills js is generated during the build process -->*

<**script** src="build/polyfills.js"></**script**>

*<!-- The vendor js is generated during the build process*

*It contains all of the dependencies in node\_modules -->*

<**script** src="build/vendor.js"></**script**>

*<!-- The main bundle js is generated during the build process -->*

<**script** src="build/main.js"></**script**>

Todos estos scripts son generados por el sistema de compilación, por lo que no debe preocuparse por ellos.

### **./src/**

Dentro del srcdirectorio encontramos nuestro código. Aquí es donde tendrá lugar la mayor parte del trabajo para una aplicación Ionic. Cuando corremos ionic serve, nuestro código dentro de src/se [transporta](https://ionicframework.com/docs/resources/what-is/#transpiler) a la versión correcta de Javascript que el navegador entiende (actualmente, [ES5](https://ionicframework.com/docs/resources/what-is/#es5) ). Eso significa que podemos trabajar a un nivel superior utilizando TypeScript, pero compilar hasta la forma más antigua de Javascript que necesita el navegador.

src/app/app.module.ts es el punto de entrada para nuestra aplicación.

Cerca de la parte superior del archivo, deberíamos ver esto:

@**NgModule({**

declarations: [MyApp, HelloIonicPage, ItemDetailsPage, ListPage],

imports: [BrowserModule, IonicModule.forRoot(MyApp)],

bootstrap: [IonicApp],

entryComponents: [MyApp, HelloIonicPage, ItemDetailsPage, ListPage],

providers: []

})

**export** **class** AppModule {}

Cada aplicación tiene un módulo raíz que básicamente controla el resto de la aplicación. Esto es muy similar a ng-appIonic 1 y AngularJS. Aquí también es donde usamos nuestra aplicación para iniciar sesión ionicBootstrap.

En este módulo, estamos configurando el componente raíz para MyApp, en src/app/app.component.ts. Este es el primer componente que se carga en nuestra aplicación, y generalmente es un shell vacío para que se carguen otros componentes. En app.component.ts, estamos configurando nuestra plantilla src/app/app.html, así que veamos allí.

### **./src/app/app.html**

Aquí está la plantilla principal para la aplicación en src/app/app.html:

<**ion-nav** id="nav" [root]="rootPage" #nav swipeBackEnabled="false"></**ion-nav**>

<**ion-menu** [content]="nav">

<**ion-header**>

<**ion-toolbar**>

<**ion-title**>Pages</**ion-title**>

</**ion-toolbar**>

</**ion-header**>

<**ion-content**>

<**ion-list**>

<**button** ion-item \*ngFor="let p of pages" (click)="openPage(p)">

{{p.title}}

</**button**>

</**ion-list**>

</**ion-content**>

</**ion-menu**>

En esta plantilla, configuramos un [ion-menu](https://ionicframework.com/docs/components/#menus)para que funcione como un menú lateral, y luego un [ion-nav](https://ionicframework.com/docs/api/components/nav/Nav/) componente para actuar como el área de contenido principal. La propiedad de [ion-menu](https://ionicframework.com/docs/components/#menus)la [content]propiedad está ligada a la variable local navde nuestro [ion-nav](https://ionicframework.com/docs/api/components/nav/Nav/), por lo que sabe dónde debe animarse.

A continuación, veamos cómo crear más páginas y realizar navegación básica.

# **Agregar páginas**

# Ahora que tenemos una comprensión básica del diseño de una aplicación Ionic, analicemos el proceso de creación y navegación por las páginas de nuestra aplicación.

Echando un vistazo a src/app/app.html, vemos esta línea cerca de la parte inferior:

<**ion-nav** [root]="rootPage" #content swipeBackEnabled="false"></**ion-nav**>

Preste atención al [root]enlace de la propiedad. Esto establece lo que es esencialmente la primera página o "raíz" para el ion-navcomponente. Cuando se ion-navcarga, el componente al que hace referencia la variable rootPageserá la página raíz.

En src/app/app.component.ts, el MyAppcomponente especifica esto en su constructor:

...

**import** { HelloIonicPage } **from** '../pages/hello-ionic/hello-ionic';

...

**export** **class** MyApp {

...

*// make HelloIonicPage the root (or first) page*

rootPage: **any** **=** HelloIonicPage;

pages: Array**<**{ title: **string**, component: **any** }**>**;

**constructor**(**private** platform: Platform, **private** menu: MenuController, ...) {

...

}

...

}

Vemos que rootPageestá configurado como HelloIonicPage, por HelloIonicPagelo que será la primera página cargada en el controlador de navegación. Echemos un vistazo.

### **Creando una página**

A continuación, veamos el HelloIonicPageque estamos importando. Dentro de la src/pages/hello-ionic/carpeta, ve y abre hello-ionic.ts.

Puede haber notado que cada página tiene su propia carpeta que lleva el nombre de la página. Dentro de cada carpeta, también vemos un .htmly un .scssarchivo con el mismo nombre. Por ejemplo, en el interior de hello-ionic/nos encontraremos hello-ionic.ts, hello-ionic.htmly hello-ionic.scss. Aunque no se requiere el uso de este patrón, puede ser útil mantener las cosas organizadas.

A continuación, vemos la HelloIonicPageclase. Esto crea una Página - un componente angular con todas las directivas iónicas ya provistas, que se cargarán utilizando el sistema de navegación de Ionic. Tenga en cuenta que debido a que las páginas deben cargarse dinámicamente, no es necesario que tengan un selector. Sin embargo, el selector es útil para anular los estilos predeterminados en una página específica (ver hello-ionic.scss):

**import** { Component } **from** '@angular/core';

**@Component**({

selector: 'page-hello-ionic',

templateUrl: 'hello-ionic.html'

})

**export** **class** HelloIonicPage {

**constructor**() {

}

}

Todas las páginas tienen una clase y una plantilla asociada que también se está compilando. Veamos src/pages/hello-ionic/hello-ionic.html- el archivo de plantilla para esta página:

<**ion-header**>

<**ion-navbar**>

<**button** ion-button menuToggle>

<**ion-icon** name="menu"></**ion-icon**>

</**button**>

<**ion-title**>Hello Ionic</**ion-title**>

</**ion-navbar**>

</**ion-header**>

<**ion-content** padding>

<**h3**>Welcome to your first Ionic app!</**h3**>

<**p**>

This starter project is our way of helping you get a functional app running in record time.

</**p**>

<**p**>

Follow along on the tutorial section of the Ionic docs!

</**p**>

<**p**>

<**button** ion-button color="primary" menuToggle>Toggle Menu</**button**>

</**p**>

</**ion-content**>

El <ion-navbar>es una plantilla para la [barra de navegación](https://ionicframework.com/docs/api/components/navbar/Navbar/) en esta página. A medida que navegamos hacia esta página, el botón y el título de la barra de navegación cambian como parte de la transición de la página.

El resto de la plantilla es un código iónico estándar que configura nuestra área de contenido e imprime nuestro mensaje de bienvenida.

### **Crear páginas adicionales**

Para crear una página adicional, no necesitamos hacer mucho más que asegurarnos de configurar correctamente el título y cualquier otra cosa que deseemos que muestre la barra de navegación.

Vamos a ver el contenido de src/pages/list/list.ts. En el interior, verá que se define una nueva página:

**import** {Component} **from** "@angular/core";

**import** {NavController, NavParams} **from** 'ionic-angular';

**import** {ItemDetailsPage} **from** '../item-details/item-details';

**@Component**({

templateUrl: 'list.html'

})

**export** **class** ListPage {

selectedItem: **any**;

icons: **string**[];

items: Array**<**{ title: **string**, note: **string**, icon: **string** }**>**;

**constructor**(**public** navCtrl: NavController, **public** navParams: NavParams) {

*// If we navigated to this page, we will have an item available as a nav param*

**this**.selectedItem **=** navParams.get('item');

**this**.icons **=** ['flask', 'wifi', 'beer', 'football', 'basketball', 'paper-plane',

'american-football', 'boat', 'bluetooth', 'build'];

**this**.items **=** [];

**for** (**let** i **=** 1; i **<** 11; i**++**) {

**this**.items.push({

title: 'Item ' **+** i,

note: 'This is item #' **+** i,

icon: **this**.icons[Math.floor(Math.random() **\*** **this**.icons.length)]

});

}

}

itemTapped(event, item) {

**this**.navCtrl.push(ItemDetailsPage, {

item: item

});

}

}

Esta página creará una página de lista básica que contiene una cantidad de artículos.

En general, esta página es muy similar a la que vimos anteriormente. En la siguiente sección, ¡aprenderemos cómo navegar a una página nueva!

# **Navegando hacia las páginas**

# Recordemos de la sección anterior que teníamos una función dentro de nuestra clase que se parecía a esto:

itemTapped(event, item) {

**this**.navCtrl.push(ItemDetailsPage, {

item: item

});

}

Es posible que haya notado que estamos haciendo referencia ItemDetailsPage. Esta es una página incluida en el tutorial de inicio. Vamos a importarlo app/pages/list/list.tspara que podamos usarlo:

...

**import** { ItemDetailsPage } **from** '../pages/item-details/item-details';

Después de guardar el archivo, notará que el ionic serveproceso recompilará su aplicación con los nuevos cambios y volverá a cargar el navegador. ¡Revisemos nuestra aplicación en el navegador, y cuando toquemos un elemento, navegará a la página de detalles del elemento! Observe que el menú-alternar se reemplaza por un botón de retroceso en su lugar. Este es un estilo nativo que Ionic sigue, pero que se puede configurar.

### **Cómo funciona?**

La navegación en Ionic funciona como una pila simple, donde colocamos pushnuevas páginas en la parte superior de la pila, lo que nos lleva hacia adelante en la aplicación y muestra un botón Atrás. Para ir hacia atrás, tenemos popla página superior desactivada. Como configuramos this.navCtrlel constructor, podemos llamar this.navCtrl.push()y pasarle la página a la que queremos acceder. También podemos pasarle un objeto que contiene datos que nos gustaría pasar a la página a la que se está navegando. Usar pushpara navegar a una nueva página es simple, pero el [sistema de navegación](https://ionicframework.com/docs/components/#navigation) de Ionic es muy flexible. Consulte los [documentos de navegación](https://ionicframework.com/docs/components/#navigation)para ver ejemplos de navegación más avanzados.

Cuando se trata de URL, el último Ionic funciona de forma un poco diferente a Ionic v1.x. En lugar de utilizar URL para navegar, las usamos para asegurarnos de que siempre podamos volver a una página (por ejemplo, al iniciar la aplicación). Esto significa que no estamos limitados a usar hrefpara navegar. Sin embargo, todavía tenemos la opción de usar una URL para navegar a una página cuando sea necesario.

### **Próximos pasos**

¡Buen trabajo ! ¡Has superado el tutorial y estás en camino hacia el dominio iónico ! Si está buscando información general sobre qué más se incluye con Ionic, consulte los [documentos de componentes](https://ionicframework.com/docs/components) . Para aprender sobre el uso de las API del dispositivo, diríjase a la [sección Native](https://ionicframework.com/docs/native/) . Si en algún momento necesita ayuda, consulte nuestra [sección de recursos para desarrolladores](https://ionicframework.com/docs/developer-resources/) o haga una pregunta en [los foros](http://forum.ionicframework.com/)

# **Componentes**

Las aplicaciones iónicas están hechas de bloques de construcción de alto nivel llamados componentes. Los componentes le permiten construir rápidamente una interfaz para su aplicación. Ionic viene con una serie de componentes, incluidos modales, ventanas emergentes y tarjetas. Consulte los ejemplos a continuación para ver cómo se ve cada componente y para aprender a usarlos. Una vez que esté familiarizado con los conceptos básicos, diríjase a los [documentos](https://ionicframework.com/docs/api) de la [API](https://ionicframework.com/docs/api) para obtener ideas sobre cómo personalizar cada componente.

## Hojas de acción

Las Hojas de acción se deslizan hacia arriba desde el borde inferior de la pantalla del dispositivo y muestran un conjunto de opciones con la capacidad de confirmar o cancelar una acción. Las Hojas de acción a veces se pueden usar como una alternativa a los menús, sin embargo, no se **deben** usar para la navegación.

La Hoja de acción siempre aparece encima de cualquier otro componente en la página y debe descartarse para interactuar con el contenido subyacente. Cuando se activa, el resto de la página se oscurece para dar más enfoque a las opciones de la Hoja de acción.

Para obtener más información, consulte los [*documentos de*](https://ionicframework.com/docs/api/components/action-sheet/ActionSheetController) la [*API*](https://ionicframework.com/docs/api/components/action-sheet/ActionSheetController) .

**Uso Básico**

**import** { ActionSheetController } **from** 'ionic-angular';

**export** **class** MyPage {

**constructor**(**public** actionSheetCtrl: ActionSheetController) {

}

presentActionSheet() {

**let** actionSheet **=** **this**.actionSheetCtrl.create({

title: 'Modify your album',

buttons: [

{

text: 'Destructive',

role: 'destructive',

handler: () **=>** {

console.log('Destructive clicked');

}

},{

text: 'Archive',

handler: () **=>** {

console.log('Archive clicked');

}

},{

text: 'Cancel',

role: 'cancel',

handler: () **=>** {

console.log('Cancel clicked');

}

}

]

});

actionSheet.present();

}

}

## Alertas

Las alertas son una gran manera de ofrecer al usuario la posibilidad de elegir una acción o lista de acciones específica. También pueden proporcionar al usuario información importante o requerirles que tomen una decisión (o decisiones múltiples).

Desde la perspectiva de la interfaz de usuario, Alerts puede considerarse como un tipo de modal "flotante" que cubre solo una parte de la pantalla. Esto significa que las Alertas solo deben usarse para acciones rápidas como la verificación de contraseñas, notificaciones de aplicaciones pequeñas u opciones rápidas. Más en los flujos de usuario de profundidad debería reservarse para pantalla completa [Modals](https://ionicframework.com/docs/components/#modals) .

Las alertas son bastante flexibles y se pueden personalizar fácilmente.

Para obtener más información, consulte los [*documentos de*](https://ionicframework.com/docs/api/components/alert/AlertController) la [*API*](https://ionicframework.com/docs/api/components/alert/AlertController) .

* Contenido
* [Alertas básicas](https://ionicframework.com/docs/components/#alerts)
* [Alertas rápidas](https://ionicframework.com/docs/components/#alert-prompt)
* [Alertas de confirmación](https://ionicframework.com/docs/components/#alert-confirm)
* [Alertas de radio](https://ionicframework.com/docs/components/#alert-radio)
* [Alertas de Checkbox](https://ionicframework.com/docs/components/#alert-checkbox)

### **Uso Básico**

Las Alertas Básicas generalmente se usan para notificar al usuario sobre nueva información (un cambio en la aplicación, una nueva característica, etc.), una situación urgente que requiere reconocimiento o como una confirmación para el usuario de que una acción fue exitosa o no.

**import** { AlertController } **from** 'ionic-angular';

**export** **class** MyPage {

**constructor**(**public** alertCtrl: AlertController) {

}

showAlert() {

**let** alert **=** **this**.alertCtrl.create({

title: 'New Friend!',

subTitle: 'Your friend, Obi wan Kenobi, just accepted your friend request!',

buttons: ['OK']

});

alert.present();

}

}

### **Alertas rápidas**

Los avisos ofrecen una forma de ingresar datos o información. A menudo, las solicitudes se usarán para pedirle al usuario una contraseña antes de avanzar en el flujo de una aplicación.

**import** { AlertController } from 'ionic-angular';

**export** **class** MyPage {

**constructor**(public alertCtrl: AlertController) {

}

showPrompt() {

**let** prompt = **this**.alertCtrl.create({

title: 'Login',

message: "Enter a name for this new album you're so keen on adding",

inputs: [

{

name: 'title',

placeholder: 'Title'

},

],

buttons: [

{

text: 'Cancel',

handler: data => {

console.log('Cancel clicked');

}

},

{

text: 'Save',

handler: data => {

console.log('Saved clicked');

}

}

]

});

prompt.present();

}

}

### **Alertas de confirmación**

Las Alertas de Confirmación se usan cuando se requiere que el usuario confirme explícitamente una opción en particular antes de avanzar en la aplicación. Un ejemplo común de Confirmation Alert es verificar para asegurarse de que un usuario desea eliminar o eliminar un contacto de su libreta de direcciones.

**import** { AlertController } from 'ionic-angular';

**export** **class** MyPage {

**constructor**(public alertCtrl: AlertController) {

}

showConfirm() {

**let** confirm = **this**.alertCtrl.create({

title: 'Use this lightsaber?',

message: 'Do you agree to use this lightsaber to do good across the intergalactic galaxy?',

buttons: [

{

text: 'Disagree',

handler: () => {

console.log('Disagree clicked');

}

},

{

text: 'Agree',

handler: () => {

console.log('Agree clicked');

}

}

]

});

confirm.present();

}

}

### **Radio**

### Las alertas de radio son un tipo de alerta de confirmación, pero usan el componente de [radio](https://ionicframework.com/docs/components/#radio) para ofrecer varias opciones. Se proporciona un conjunto de opciones para el usuario, pero solo se puede elegir una opción.

**import** { AlertController } from 'ionic-angular';

**export** **class** MyPage {

**constructor**(public alertCtrl: AlertController) {

}

showRadio() {

**let** alert = **this**.alertCtrl.create();

alert.setTitle('Lightsaber color');

alert.addInput({

**type**: 'radio',

label: 'Blue',

value: 'blue',

checked: true

});

alert.addButton('Cancel');

alert.addButton({

text: 'OK',

handler: data => {

**this**.testRadioOpen = false;

**this**.testRadioResult = data;

}

});

alert.present();

}

}

### **Caja**

Las alertas de casilla de verificación son un tipo de alerta de confirmación, pero usan el componente [casilla de verificación](https://ionicframework.com/docs/components/#checkbox) para ofrecer varias opciones. Ofrecen al usuario un conjunto de opciones para elegir.

**import** { AlertController } from 'ionic-angular';

**export** **class** MyPage {

**constructor**(public alertCtrl: AlertController) {

}

showCheckbox() {

**let** alert = **this**.alertCtrl.create();

alert.setTitle('Which planets have you visited?');

alert.addInput({

**type**: 'checkbox',

label: 'Alderaan',

value: 'value1',

checked: true

});

alert.addInput({

**type**: 'checkbox',

label: 'Bespin',

value: 'value2'

});

alert.addButton('Cancel');

alert.addButton({

text: 'Okay',

handler: data => {

console.log('Checkbox data:', data);

**this**.testCheckboxOpen = false;

**this**.testCheckboxResult = data;

}

});

alert.present();

}

}

## Insignias

Las insignias son pequeños componentes que normalmente comunican un valor numérico al usuario. Se usan típicamente dentro de un artículo.

Para obtener más información, consulte los [*documentos de*](https://ionicframework.com/docs/api/components/badge/Badge) la [*API*](https://ionicframework.com/docs/api/components/badge/Badge) .

### **Uso Básico**

<**ion-item**>

<**ion-icon** name="logo-twitter" item-start></**ion-icon**>

Followers

<**ion-badge** item-end>260k</**ion-badge**>

</**ion-item**>

Las insignias también pueden recibir cualquier atributo de color:

<**ion-badge** color="secondary"></**ion-badge**>

## Botones

Los botones son una forma esencial de interactuar y navegar a través de una aplicación, y deben comunicar claramente qué acción ocurrirá después de que el usuario los toque. Los botones pueden consistir en texto y / o un ícono, y se pueden mejorar con una amplia variedad de atributos.

Por razones de accesibilidad, los botones usan un <button>elemento estándar , pero se mejoran con una ion-buttondirectiva.

** Cordova**

En este tutorial, aprenderá estrategias y mejores prácticas para construir aplicaciones móviles nativas con HTML, JavaScript y CSS. Desarrollará desde cero una aplicación de directorio de empleados Apache Cordova (también conocida como PhoneGap) utilizando la arquitectura de una sola página, plantillas HTML, eventos táctiles y técnicas de optimización del rendimiento.

## Lo que vas a aprender:

* Cómo crear y compilar (compilar y empaquetar) un proyecto de Cordova utilizando la CLI (interfaz de línea de comando).
* Cómo usar diferentes estrategias de almacenamiento de datos.
* Cómo usar varias API de Cordova, como Geolocalización, Contactos y Cámara.
* Cómo manejar desafíos específicos de dispositivos móviles como eventos táctiles, desplazamiento, diseño, transiciones de página, etc.
* Cómo crear una aplicación usando una arquitectura de una sola página y plantillas HTML.

## Requisitos

* Para completar este taller, todo lo que necesita es un editor de código, un navegador moderno y una conexión a Internet.
* Se asume un conocimiento práctico de HTML y JavaScript, pero no es necesario ser un gurú de JavaScript.

Un dispositivo móvil o un Mobile SDK **no** es un requisito para este tutorial. Sin embargo, si desea ejecutar y probar la aplicación en un dispositivo móvil o en un emulador para una plataforma móvil compatible con Cordova, necesita el SDK móvil para esa plataforma (iOS SDK, Android SDK, etc.) instalado en su sistema. Si no desea instalar un Mobile SDK, podrá probar su aplicación en un navegador de su computadora.

## Cuestiones

* Por favor, cree un problema [aquí](https://github.com/ccoenraets/apache-cordova-tutorial/issues) si tiene algún problema o si tiene una sugerencia para mejorar este taller.
* También puede usar la sección Comentarios en la parte inferior de cada módulo para hacer una pregunta o informar un problema.

# **Módulo 1: Creando un Proyecto Cordova**

## Pasos

1. Asegúrese de tener una versión actualizada de [Node.js](http://nodejs.org/) instalada en su sistema.
2. Abra Terminal (Mac) o una ventana de comando (Windows) y escriba el siguiente comando para instalar la CLI de Cordova:
3. npm install -g cordova

o en una Mac:

sudo npm install -g cordova

Si ya tiene Cordova instalado en su computadora, asegúrese de actualizar a la última versión:

npm update -g cordova

o

sudo npm update -g cordova

1. Navegue (cd) a un directorio donde almacena proyectos en su sistema de archivos.
2. Utilizando la CLI de Cordova, cree un proyecto de Cordova llamado **Workshop** en un directorio llamado **workshop** :
3. cordova create workshop com.yourname.workshop Workshop
4. Navega al directorio del proyecto:
5. cd workshop
6. Añadir soporte para la plataforma iOS (opcional)

Para poder compilar para la plataforma iOS, el SDK de iOS debe estar instalado en su sistema. Si no es así, puede omitir este paso y agregar soporte para otra plataforma, o simplemente ejecutar la aplicación tutorial en su navegador.

* 1. Asegúrese de que iOS SDK esté disponible en su sistema.
  2. En la línea de comando, asegúrese de estar en el directorio del **taller** y escriba:
  3. cordova platforms add ios

1. Agregue soporte para la plataforma Android (opcional)

Para poder compilar para la plataforma Android, el SDK de Android debe estar instalado en su sistema. Si no es así, puede omitir este paso y agregar soporte para otra plataforma, o simplemente ejecutar la aplicación tutorial en su navegador.

* 1. Asegúrese de que Android SDK y la herramienta de compilación de ant están disponibles en su sistema. El SDK de Android está disponible [aquí](http://developer.android.com/sdk) . Las herramientas **android** y **ant** deben estar disponibles en su camino.
  2. En la línea de comando, asegúrese de estar en el directorio del **taller** y escriba:
  3. cordova platforms add android

1. Asegúrese de estar en el directorio del **taller** y agregue complementos básicos a sus proyectos:
2. cordova plugin add org.apache.cordova.device
3. cordova plugin add org.apache.cordova.console
4. Examine la estructura del directorio bajo el taller.
   1. La carpeta **www** es donde codificará su aplicación HTML / JavaScript. Abra el archivo index.html en un navegador para ver la aplicación predeterminada creada por la CLI de Cordova.
   2. La carpeta de **plataformas** es donde Cordova construirá su aplicación para diferentes plataformas (iOS, Android, etc.). El contenido de esta carpeta se generará automáticamente por la CLI de Cordova, y nunca debe editar el código en ese directorio.
   3. Los complementos se instalan en el directorio de **complementos** .
   4. Los parámetros de la aplicación (nombre, autor, etc.) se almacenan en **config.xml** .

# **Módulo 2: Construyendo un Proyecto Cordova**

Este módulo es opcional. Si no agregó soporte para ninguna plataforma en el módulo anterior, puede omitir este módulo y acceder al [módulo 3](https://ccoenraets.github.io/cordova-tutorial/setup-files.html) .

## Construyendo para iOS

Necesita el iOS SDK instalado en su computadora para crear una versión iOS de su aplicación siguiendo los pasos a continuación.

En la línea de comando, asegúrese de estar en el directorio del **taller** y escriba:

cordova build ios

El proyecto está construido en la carpeta **workshop / platforms / ios** . Haga doble clic en **Workshop.xcodeproj** para abrir el proyecto en Xcode y ejecútelo en el emulador o en su dispositivo.

También puede ejecutar la aplicación en el emulador de iOS directamente desde la línea de comandos. Primera instalación de [ios-sim](https://github.com/phonegap/ios-sim) :

npm install -g ios-sim

o

sudo npm install -g ios-sim

Luego ejecute la aplicación en el emulador de iOS:

cordova emulate ios

## Construyendo para Android

Necesita el SDK de Android instalado en su computadora para crear una versión de Android de su aplicación siguiendo los pasos a continuación.

Para construir el proyecto en la carpeta **workshop / platforms / android** y ejecutarlo en un dispositivo Android conectado a su computadora con un cable USB, escriba:

cordova run android

Para compilar el proyecto en la carpeta **taller / plataformas / android** y ejecutarlo en el emulador de Android, escriba:

cordova emulate android

# **Módulo 3: Configuración de los archivos del taller**

## Pasos

1. Borre los contenidos del **taller /** carpeta de su proyecto .
2. Descargue los archivos de soporte para este tutorial [aquí](https://github.com/ccoenraets/cordova-tutorial/archive/master.zip) , o clone el repositorio:
3. git clone <https://github.com/ccoenraets/cordova-tutorial>
4. Si descargó el archivo comprimido, descomprímalo en cualquier lugar de su sistema de archivos.
5. Copie los contenidos de **starter-www** en la carpeta del **taller / www de** su proyecto .
6. Crea y prueba tu aplicación:
   * Si tiene un SDK móvil instalado en su sistema, repita los pasos en el Módulo 2.
   * Si no lo hace, simplemente abra **www / index.html** en un navegador en su computadora.
7. Escriba algunos caracteres en el cuadro de búsqueda para buscar empleados por nombre. Hacer clic en un enlace de empleado no produce ningún resultado en este momento.

Las soluciones de módulo por módulo están disponibles en el directorio de **soluciones** en apache-cordova-tutorial.

# **Módulo 4: Elegir una estrategia de almacenamiento de datos**

## Paso 1: Explora diferentes mecanismos de persistencia

Abra los siguientes archivos y explore los diferentes servicios de persistencia:

1. www / js / services / memory / EmployeeService.js
2. www / js / services / json / EmployeeService.js
3. www / js / services / localstorage / EmployeeService.js
4. www / js / services / websql / EmployeeService.js

## Paso 2: prueba la aplicación con diferentes mecanismos de persistencia

La aplicación está configurada inicialmente para trabajar con el almacén de datos en memoria. Para cambiar el mecanismo de persistencia local para la aplicación:

1. En **index.html** : en lugar de js / services / memory / EmployeeService.js, importe el archivo .js para el servicio de su elección, por ejemplo: js / services / websql / EmployeeService.js.
2. Pruebe la aplicación.

Para probar el servicio JSON, asegúrese de que el servidor Node.js proporcionado como parte de los materiales se esté ejecutando:

1. Abra una ventana de terminal o comando, y navegue al directorio del servidor bajo cordova-tutorial
2. Instale las dependencias del servidor:
3. npm install
4. Comience el servidor
5. node server

El servidor implementa CORS (Intercambio de recursos de origen cruzado) para admitir solicitudes HTTP entre sitios. Por lo tanto, puede invocar los servicios desde un archivo cargado desde otro dominio o desde el sistema de archivos.

Dado que services / json / EmployeeService.js apunta a **localhost** , esto solo funcionará cuando se ejecuta la aplicación en el navegador de su computadora, y no en su dispositivo porque no conoce su computadora como "localhost". Para que el servicio JSON funcione al ejecutar la aplicación en su dispositivo, asegúrese de que su computadora y dispositivo estén en la misma subred, identifique la dirección IP de su computadora y reemplace localhost con esa dirección IP en services / json / EmployeeService.js. Como alternativa, también puede implementar el servicio en un servidor disponible públicamente. En una aplicación de la vida real, normalmente se externalizaría el nombre de host en algún tipo de archivo de configuración.

Todos los demás servicios de almacenamiento de datos proporcionados en www / js / services funcionan de fábrica cuando se ejecuta la aplicación en el navegador y en el dispositivo.

# **Módulo 5: Uso de notificaciones nativas**

Una alerta de JavaScript predeterminada revela el hecho de que su aplicación no es nativa. En esta sección, configuramos la infraestructura básica para mostrar las alertas nativas cuando la aplicación se está ejecutando en un dispositivo, y recurrimos a las alertas de JavaScript predeterminadas cuando se ejecuta en el navegador.

## Pasos

1. Asegúrese de estar en el directorio del **taller** y agregue el complemento de diálogo nativo a su proyecto:
2. cordova plugin add org.apache.cordova.dialogs
3. En index.html, agregue la siguiente etiqueta de script (como la primera etiqueta de script en la parte inferior del cuerpo):
4. <script src="cordova.js"></script>

Esto instruye a la CLI de Cordova para que inyecte una versión específica de la plataforma de cordova.js en tiempo de compilación. En otras palabras, cordova.js no necesita estar (ni debería estar) presente en su carpeta proyecto / www.

1. Cuando se ejecuta en un dispositivo con el objeto navigator.notification disponible (el complemento de diálogo está instalado), anula la función window.alert () y reemplaza su implementación predeterminada con una llamada a navigator.notification.alert ().

Abra **js / app.js** y agregue este código al bloque "Registro de eventos":

document.addEventListener('deviceready', function () {

if (navigator.notification) { // Override default HTML alert with native dialog

window.alert = function (message) {

navigator.notification.alert(

message, // message

null, // callback

"Workshop", // title

'OK' // buttonName

);

};

}

}, false);

1. Pruebe la aplicación: haga clic en el botón Ayuda.
   * Cuando ejecuta la aplicación en el navegador, debería ver una alerta estándar del navegador.
   * Cuando ejecuta la aplicación en su dispositivo, debería ver una alerta nativa.

# **Módulo 6: evitar el retardo de clic de 300 ms**

## Pasos

1. Pruebe la aplicación en su dispositivo iOS o en el emulador de iOS: pulse el botón Ayuda y observe el retraso antes de que aparezca el cuadro de diálogo.

Este retraso ocurre porque el sistema operativo está esperando aproximadamente 300 ms para ver si el usuario volverá a tocar el objetivo (y, por lo tanto, realizará un doble toque).

1. En index.html, agregue la siguiente etiqueta de script:
2. <script src="lib/fastclick.js"></script>

FastClick es una biblioteca de código abierto creada por Financial Times. Más información aquí .

1. En app.js, registre FastClick dentro del controlador de eventos deviceready.
2. FastClick.attach(document.body);
3. Pruebe la aplicación: haga clic en el botón Ayuda. El mensaje debería aparecer ahora sin demora.

# **Módulo 7: Configuración de una aplicación de una sola página**

Una "Aplicación de una sola página" es una aplicación web que vive en una sola página HTML. Las "vistas" de la aplicación se inyectan y eliminan del DOM según sea necesario a medida que el usuario navega a través de la aplicación. Una arquitectura de aplicación de una sola página es especialmente adecuada para aplicaciones móviles:

* La ausencia de actualizaciones de página proporciona una experiencia más fluida y más cercana a la nativa.
* La interfaz de usuario se crea completamente en el lado del cliente sin depender de un servidor para crear la interfaz de usuario, lo que la convierte en una arquitectura ideal para aplicaciones que funcionan sin conexión.

En este módulo, configuramos la infraestructura básica para convertir el Directorio de empleados en una aplicación de una sola página.

## Pasos

1. En index.html: elimine el marcado HTML dentro de la etiqueta del cuerpo ( **con la excepción de las etiquetas del script** ).
2. Dentro de la función inmediata en app.js, defina una función llamada **renderHomeView ()** (justo después de la función findByName). Implemente la función para agregar programáticamente la marca de Vista de inicio al elemento de cuerpo.
3. function renderHomeView() {
4. var html =
5. "<h1>Directory</h1>" +
6. "<input class='search-key' type='search' placeholder='Enter name'/>" +
7. "<ul class='employee-list'></ul>";
8. $('body').html(html);
9. $('.search-key').on('keyup', findByName);
10. }
11. Modifique la lógica de inicialización del servicio de datos: cuando el servicio se haya inicializado con éxito, llame a la función renderHomeView () para mostrar programáticamente la Vista de inicio.
12. var service = new EmployeeService();
13. service.initialize().done(function () {
14. renderHomeView();
15. });
16. Dado que movió el registro del evento de teclado dentro de la función renderHomeView (), asegúrese de eliminar el registro de evento de teclado original en la sección Registro de evento.
17. Como el botón Ayuda ya no existe, elimine el controlador de eventos de clic para el botón de ayuda (en la sección Registro de eventos).
18. Pruebe la aplicación.

Asegúrese de no haber pasado por alto los pasos 4 y 5 anteriores, de lo contrario, puede encontrar problemas difíciles de seguir relacionados con los eventos que se activarán dos veces más adelante en este tutorial.

# **Módulo 8: Uso de plantillas de manubrios**

Escribir fragmentos HTML en JavaScript e insertarlos mediante programación en el DOM es tedioso. Hace que su aplicación sea más difícil de escribir y más difícil de mantener. Las plantillas HTML solucionan este problema al desacoplar la definición de la interfaz de usuario (marcado HTML) de su código. Hay una serie de excelentes soluciones de plantillas HTML, incluidas Mustache.js, Handlebars.js y Underscore.js, por nombrar algunas.

En este módulo, creamos dos plantillas para simplificar el código de la aplicación del Directorio de empleados. Usamos[Handlebars.js](http://handlebarsjs.com/) pero el mismo resultado se puede lograr usando otras soluciones de plantilla HTML.

## Paso 1: defina las plantillas

Modifique index.html de la siguiente manera:

1. Agregue una etiqueta de script para incluir la biblioteca handlebars.js:
2. <script src="lib/handlebars.js"></script>
3. Agregue ratchet.css y styles.css a la cabecera de index.html

[Ratchet](http://goratchet.com/) es un simple kit de herramientas de CSS que proporciona estilos para aplicaciones móviles.

<link href="assets/ratchet/css/ratchet.css" rel="stylesheet">

<link href="assets/css/styles.css" rel="stylesheet">

1. Crea una plantilla HTML para renderizar la Vista de inicio. Agregue esta etiqueta de script como el primer elemento secundario de la etiqueta de cuerpo:
2. <script id="home-tpl" type="text/template">
3. <header class="bar bar-nav">
4. <h1 class="title">Directory</h1>
5. </header>
6. <div class="bar bar-standard bar-header-secondary">
7. <input class='search-key' type="search"/>
8. </div>
9. <div class="content"></div>
10. </script>
11. Crea una plantilla HTML para representar la lista de empleados. Agregue esta etiqueta de script inmediatamente después de la anterior:
12. <script id="employee-list-tpl" type="text/template">
13. <ul class="table-view">
14. {{#each this}}
15. <li class="table-view-cell media">
16. <a href="#employees/{{ id }}">
17. <img class="media-object pull-left" src="assets/pics/{{pic}}">
18. <div class="media-body">
19. {{firstName}} {{lastName}}
20. <p>{{title}}</p>
21. </div>
22. </a>
23. </li>
24. {{/each}}
25. </ul>
26. </script>

## Paso 2: usa las plantillas

Modifique la función inmediata en app.js de la siguiente manera:

1. Inmediatamente antes de la declaración de la variable de servicio , declare dos variables que contienen la versión compilada de las plantillas definidas anteriormente:
2. var homeTpl = Handlebars.compile($("#home-tpl").html());
3. var employeeListTpl = Handlebars.compile($("#employee-list-tpl").html());
4. Modifique renderHomeView () para usar la plantilla homeTpl en lugar del HTML en línea:
5. function renderHomeView() {
6. $('body').html(homeTpl());
7. $('.search-key').on('keyup', findByName);
8. }
9. Modifique findByName () para usar la plantilla employeeListTpl en lugar del HTML en línea:
10. function findByName() {
11. service.findByName($('.search-key').val()).done(function (employees) {
12. $('.content').html(employeeListTpl(employees));
13. });
14. }
15. Pruebe la aplicación.

## Paso 3: reparación del problema de la barra de estado en iOS7

En iOS7, la barra de estado se superpone a las vistas de la aplicación. Como resultado, el texto de la barra de estado puede colisionar con el texto del encabezado de la aplicación como se muestra en la captura de pantalla anterior. Puede solucionar este problema utilizando el complemento de la barra de estado.

1. Agregue el complemento de la barra de estado:
2. cordova plugins add org.apache.cordova.statusbar
3. En app.js, agregue el siguiente código en la parte superior del controlador deviceready:
4. StatusBar.overlaysWebView( false );
5. StatusBar.backgroundColorByHexString('#ffffff');
6. StatusBar.styleDefault();
7. Cree nuevamente la aplicación y pruebe su aplicación en el emulador iOS o en un dispositivo iOS.

# **Módulo 9: Creando Vistas**

Es hora de proporcionar a la aplicación cierta estructura. Si seguimos agregando todas las funciones principales de la aplicación a la función inmediata que inicia la aplicación, muy rápidamente crecerá fuera de control. En esta sección, creamos un objeto HomeView que encapsula la lógica para crear y representar la vista de Inicio.

## Paso 1: crea la vista de inicio

1. Cree un archivo llamado HomeView.js en el directorio js y defina un constructor HomeView implementado de la siguiente manera:
2. var HomeView = function (service) {
3. }

La función constructora toma el servicio de datos del empleado como argumento

1. HomeView usa una vista anidada para mostrar la lista de empleados. Definir la lista de empleados como una vista separada la hace reutilizable en otros contextos. Definiremos EmployeeListView en el Paso 2 a continuación. Por ahora, defina una variable local para realizar un seguimiento de la vista anidada.
2. var HomeView = function (service) {
3. var employeeListView;
4. }
5. Defina una función initialize () dentro del constructor HomeView.
   * Defina un envoltorio div para la vista. El envoltorio div se usa para adjuntar los eventos relacionados con la vista.
   * Cree una instancia de la vista anidada (definirá EmployeeListView en el paso 2).
   * Finalmente, invoque la función initialize () dentro de la función del constructor HomeView.
6. var HomeView = function (service) {
7. var employeeListView;
8. this.initialize = function () {
9. // Define a div wrapper for the view (used to attach events)
10. this.$el = $('<div/>');
11. this.$el.on('keyup', '.search-key', this.findByName);
12. employeeListView = new EmployeeListView();
13. this.render();
14. };
15. this.initialize();
16. }
17. Mueva la función renderHomeView () de app.js a la clase HomeView. Para mantener la vista reutilizable, adjunte el HTML al envoltorio div (this.el) en lugar del cuerpo del documento. Como la función ahora está encapsulada en HomeView, también puede cambiarle el nombre desde renderHomeView () para simplemente renderizar ().
18. this.render = function() {
19. this.$el.html(this.template());
20. $('.content', this.$el).html(employeeListView.$el);
21. return this;
22. };
23. Mueva la función findByName () de app.js a HomeView y ajústela para que funcione con la vista anidada.
24. this.findByName = function() {
25. service.findByName($('.search-key').val()).done(function(employees) {
26. employeeListView.setEmployees(employees);
27. });
28. };

## Paso 2: crea la vista EmployeeList

1. Crea un archivo llamado EnployeeListView.js en el directorio js
2. Implemente EmployeeListView de la siguiente manera:
3. var EmployeeListView = function () {
4. var employees;
5. this.initialize = function() {
6. this.$el = $('<div/>');
7. this.render();
8. };
9. this.setEmployees = function(list) {
10. employees = list;
11. this.render();
12. }
13. this.render = function() {
14. this.$el.html(this.template(employees));
15. return this;
16. };
17. this.initialize();
18. }

## Paso 3: Uso de la Vista de inicio

1. En index.html, agregue etiquetas de script para incluir EmployeeListView.js y HomeView.js (justo **antes de** la etiqueta de script para app.js):
2. <script src="js/EmployeeListView.js"></script>
3. <script src="js/HomeView.js"></script>
4. En app.js, elimine la función renderHomeView ().
5. Elimine la función findByName ().
6. Modificar la inicialización de las plantillas. En lugar de declararlos como variables locales, agréguelos al prototipo de sus respectivas clases:
7. HomeView.prototype.template = Handlebars.compile($("#home-tpl").html());
8. EmployeeListView.prototype.template =
9. Handlebars.compile($("#employee-list-tpl").html());
10. Modifique la lógica de inicialización del servicio para mostrar la Vista inicial cuando el servicio se haya inicializado correctamente. Pase el servicio como argumento al constructor de Home View.
11. service.initialize().done(function () {
12. $('body').html(new HomeView(service).render().$el);
13. });
14. Pruebe la aplicación

# **Módulo 10: implementación del enrutamiento de vista**

En esta sección, agregamos una vista de detalles del empleado. Dado que la aplicación ahora tiene más de una vista, también agregamos un mecanismo de enrutamiento de vista simple.

## Paso 1: crea la plantilla de empleado

Abra index.html y agregue una plantilla para representar una vista detallada de los empleados:

<script id="employee-tpl" type="text/template">

<header class="bar bar-nav">

<a class="btn btn-link btn-nav pull-left" href="#">

<span class="icon icon-left-nav"></span>

</a>

<h1 class="title">Employee</h1>

</header>

<div class="content">

<div class="card">

<ul class="table-view">

<li class="table-view-cell media">

<img class="media-object pull-left emp-pic" src="assets/pics/{{pic}}">

<div class="media-body">

{{ firstName }} {{ lastName }}

<p>{{ title }}</p>

</div>

</li>

<li class="table-view-cell media">

<a href="tel:{{ officePhone }}" class="push-right">

<span class="media-object pull-left icon icon-call"></span>

<div class="media-body">

Call Office

<p>{{ officePhone }}</p>

</div>

</a>

</li>

<li class="table-view-cell media">

<a href="tel:{{ cellPhone }}" class="push-right">

<span class="media-object pull-left icon icon-call"></span>

<div class="media-body">

Call Cell

<p>{{ cellPhone }}</p>

</div>

</a>

</li>

<li class="table-view-cell media">

<a href="sms:{{ cellPhone }}" class="push-right">

<span class="media-object pull-left icon icon-sms"></span>

<div class="media-body">

SMS

<p>{{ cellPhone }}</p>

</div>

</a>

</li>

<li class="table-view-cell media">

<a href="mailto:{{ email }}" class="push-right">

<span class="media-object pull-left icon icon-mail"></span>

<div class="media-body">

Email

<p>{{ email }}</p>

</div>

</a>

</li>

</ul>

</div>

</div>

</script>

## Paso 2: crea la clase EmployeeView

1. Cree un archivo llamado EmployeeView.js en el directorio js y defínalo de la siguiente manera:
2. var EmployeeView = function(employee) {
3. this.initialize = function() {
4. this.$el = $('<div/>');
5. };
6. this.render = function() {
7. this.$el.html(this.template(employee));
8. return this;
9. };
10. this.initialize();
11. }
12. En index.html, agregue una etiqueta de script para incluir EmployeeView.js (justo **antes de** la etiqueta de script para app.js):
13. <script src="js/EmployeeView.js"></script>

## Paso 3: Implementar el enrutamiento de vista

1. En index.html, agregue una etiqueta de script para incluir router.js (justo **después de** la etiqueta de script para jquery.js):
2. <script src="lib/router.js"></script>
3. Abra app.js. Agregue la plantilla de empleado compilada al prototipo de EmployeeView:
4. EmployeeView.prototype.template = Handlebars.compile($("#employee-tpl").html());
5. Define las dos rutas utilizadas en la aplicación:
6. service.initialize().done(function () {
7. router.addRoute('', function() {
8. $('body').html(new HomeView(service).render().$el);
9. });
10. router.addRoute('employees/:id', function(id) {
11. service.findById(parseInt(id)).done(function(employee) {
12. $('body').html(new EmployeeView(employee).render().$el);
13. });
14. });
15. router.start();
16. })Pruebe la aplicación.

 Android

Android es un sistema operativo de código abierto y basado en Linux para dispositivos móviles como teléfonos inteligentes y tabletas. Android fue desarrollado por Open Handset Alliance, liderado por Google y otras compañías. Este tutorial le enseñará la programación básica de Android y también le mostrará algunos conceptos avanzados relacionados con el desarrollo de aplicaciones de Android.

# **Audiencia**

Este tutorial ha sido preparado para los principiantes para ayudarlos a comprender la programación básica de Android. Después de completar este tutorial, se encontrará en un nivel moderado de experiencia en programación de Android desde donde puede llevarlo a los siguientes niveles.

# **Requisitos previos**

La programación de Android se basa en el lenguaje de programación Java, por lo que si tiene una comprensión básica de la programación Java, será divertido aprender el desarrollo de aplicaciones de Android.

#### **Las apps proporcionan varios puntos de entrada**

Las apps de Android se compilan como una combinación de diferentes componentes que se pueden invocar de manera individual. Por ejemplo, una actividad individual proporciona una sola pantalla para una interfaz de usuario y un servicio realiza tareas de manera independiente en segundo plano.

Desde un componente puedes iniciar otro componente con una intent. Incluso puedes iniciar un componente en otra app, como una actividad en una app de mapas para mostrar una dirección. Este modelo proporciona varios puntos de entrada para una app y permite que cualquier app se comporte como “predeterminada” de un usuario para una acción que otras apps pueden invocar.

**Obtén más información:**

* [Aspectos fundamentales de apps](https://developer.android.com/guide/components/fundamentals.html?hl=es-419)
* [Intents y filtros de intents](https://developer.android.com/guide/components/intents-filters.html?hl=es-419)
* [Actividades](https://developer.android.com/guide/components/activities.html?hl=es-419)

#### **Las apps se adaptan a diferentes dispositivos**

Android proporciona un framework de apps adaptable que te permite ofrecer recursos exclusivos para diferentes configuraciones de dispositivos. Por ejemplo, puedes crear diferentes archivos de diseño XML para diferentes tamaños de pantalla y el sistema determina qué diseño aplicar en función del tamaño de pantalla del dispositivo actual.

Puedes consultar la disponibilidad de funciones del dispositivo en tiempo de ejecución para averiguar si alguna función de la app requiere hardware específico, como una cámara. Si fuera necesario, también puedes declarar funciones que requiera tu app para que los mercados de apps como Google Play Store no permitan su instalación en dispositivos que no admitan esas funciones.

**Obtén más información:**

* [Compatibilidad con dispositivos](https://developer.android.com/guide/practices/compatibility.html?hl=es-419)
* [Información general de recursos](https://developer.android.com/guide/topics/resources/overview.html?hl=es-419)
* [Información general acerca de la interfaz de usuario](https://developer.android.com/guide/topics/ui/overview.html?hl=es-419)

*** IOS***

## Tutoriales de iPhone

Este sitio contiene una tonelada de divertidos tutoriales escritos, ¡tantos que cada vez son más difíciles de encontrar! Así que armé esta pequeña página para ayudar a todos a encontrar rápidamente el tutorial que están buscando. ¡Espero que lo disfrutes! :]

## Idiomas

* [Tutoriales de lenguaje Swift](http://www.raywenderlich.com/swift-language-tutorials)

## Desarrollo de aplicaciones

* [Tutoriales de iOS](http://www.raywenderlich.com/ios-tutorials)
* [Tutoriales de Android](http://www.raywenderlich.com/android-tutorials)
* [Tutoriales de OS X](https://www.raywenderlich.com/category/macos)

## Desarrollo de aplicaciones

* [Tutoriales de Sprite Kit](http://www.raywenderlich.com/sprite-kit-tutorials)
* [Tutoriales de Unity](http://www.raywenderlich.com/unity-tutorials)
* [Tutoriales de Unreal Engine](http://www.raywenderlich.com/unreal-engine-tutorials)

## Otro

* [Artículos](http://www.raywenderlich.com/articles)
* [Tutoriales no admitidos](http://www.raywenderlich.com/unsupported-tutorials)

*** FIREBASE***

Firebase es una plataforma de back-end para construir aplicaciones Web, Android e IOS. Ofrece una base de datos en tiempo real, diferentes API, múltiples tipos de autenticación y plataforma de alojamiento. Este es un tutorial introductorio que cubre los conceptos básicos de la plataforma Firebase y explica cómo tratar con sus diversos componentes y subcomponentes.

# **Audiencia**

Este tutorial está dirigido a desarrolladores que necesitan una plataforma back-end simple y fácil de usar. Después de terminar este tutorial, estará familiarizado con la plataforma web de Firebase. También puede usar esto como referencia en su desarrollo futuro.

Este tutorial está destinado a hacerte sentir cómodo para comenzar a utilizar la plataforma back-end de Firebase y sus diversas funciones.

# **Requisitos previos**

Necesitará conocimientos de JavaScript para poder seguir este tutorial. El conocimiento sobre alguna plataforma de backend no es necesario, pero podría ayudarte a comprender los diversos conceptos de Firebase.

Una vez creada nuestra aplicación podemos administrarla desde un panel de control que nos ofrece las opciones que resumimos a continuación:

* Data: nos ofrece el acceso para visualizar y editar nuestro modelo de datos. Básicamente lo que editamos es un JSON, ya que el sistema de Firebase es de base de datos documental.
* Security & Rules: Nos permite el acceso a una sección donde podemos escribir las reglas de acceso a los recursos dentro de Data, así como la validación que se aplicará para decidir si se permite o no una escritura de información.
* Simulator: desde donde podemos simular accesos en diversas situaciones para verificar si tendrían o no permisos para realizar ciertas operaciones.
* Analytics: Una serie de reportes estadísticos con los que podemos monitorar el estado de la aplicación y la cantidad de recursos que están siendo destinados para su funcionamiento (usuarios concurrentes, transferencia, etc).
* Login & Auth: desde aquí se puede controlar cuáles son los mecanismos que estarán disponibles para la autenticación de usuarios, tanto email y clave como por redes sociales, junto con los datos de configuración.
* Hosting: Inicialmente con una ayuda para aprovechar lo que sería un espacio de alojamiento donde desplegar la app realizada. Una vez activado también se pueden controlar aquí las configuraciones del hosting.
* Secrets: por último una página desde donde se administran claves del API de nuestra App que serán utilizadas para accesos del lado del servidor y autenticación por medio de JSON Web Tokens.

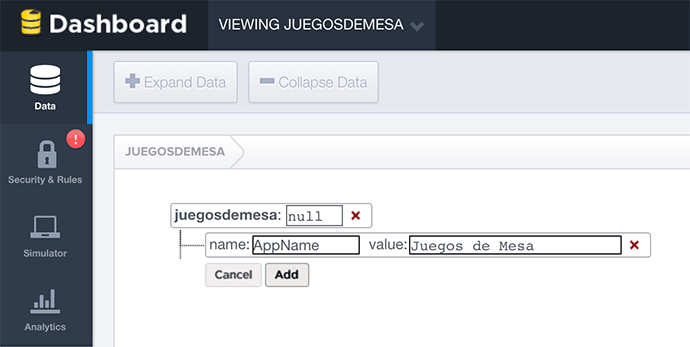
## Hacer un Hola Mundo en Firebase

Haremos a continuación lo que sería equivalente a un "Hola Mundo" en Firebase. Básicamente vamos a acceder a un dato que hay en nuestra base de datos y experimentar como las modificaciones de ese dato repercuten en mi aplicación que lo está mostrando al usuario. De hecho, comprobaremos que ese dato, una vez modificado, se actualiza automáticamente en todos los clientes conectados. Quizás parezca mucha operativa pero Firebase ya te ofrece la mayoría del trabajo listo sin que necesites preocuparte.

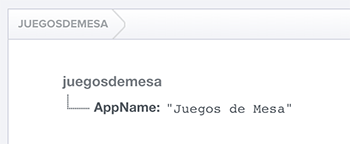
Comenzamos editando información de nuestra base de datos, creando un dato que podamos leer. En próximos artículos mostraremos cómo hacer esto mediante el API de Firebase, así que de momento y para facilitarnos las cosas, lo haremos desde la sección "Data" del gestor de nuestra aplicación. Te valdrá para hacer estas pruebas cualquier aplicación que dispongas. Si la acabas de crear comprobarás que la situación inicial de nuestros datos está con un valor "null", osea, vacío.

Vamos a agregar un nuevo dato con el icono "+" que aparece al lado de "null" cuando acercas el ratón a esa área de la pantalla.

Verás en seguida un formulario para asignar un nuevo juego de datos en tu aplicación. Aquí debemos rellenar los pares clave/valor que necesitemos introducir dentro del modelo de esta aplicación. De momento simplemente vamos a escribir una clave y un valor, pulsando luego el botón "Add".



Ahora tenemos un primer dato en nuestra aplicación. Es simplemente una propiedad con un valor pero será suficiente para este ejemplo.



## Instalar Firebase en un proyecto web

Para hacer funcionar a Firebase se tiene que cargar una librería Javascript que contiene el código del API que este sistema nos proporciona para acceder a los datos.

Lo más cómodo es que la instales usando Bower con el correspondiente comando:

bower install firebase

Una vez descargadas las dependencias, verás el archivo firebase.js dentro de bower\_components/firebase.

Nota: Seguramente estés familiarizado con esta operativa de gestión de dependencias, pero si no es así accede al [artículo de Bower](http://www.desarrolloweb.com/articulos/uso-bower-gestor-dependencias.html).

Ahora en la página .html que quieras usar Firebase debes colocar el script firebase.js en la cabecera. Eso nos permitirá usar el API de Firebase.

<script src="bower\_components/firebase/firebase.js"></script>

## Acceso a los datos por referencia

Para acceder a información de nuestra aplicación Firebase usa lo que se conoce como referencias. Son enlaces con tu modelo de datos que apuntan a un nodo en concreto del JSON. A través de esas referencias podemos acceder a un documento completo o navegar por los hijos para acceder a otros nodos dependientes. Para hacer todas estas operaciones de recorido de los datos Firebase nos expone una nutrida API que iremos conociendo en diversos artículos.

Estas referencias se crean a través de una llamada a la función Firebase(), indicándole como parámetro el nodo de nuestro modelo que queremos referenciar. Los nodos están compuestos por una ruta definida con el nombre de tu aplicación, algo como https://ejemplo.firebaseio.com y luego una ruta. Si no indicas ninguna ruta simplemente accederás a la raíz de tus datos y si especificas una ruta podrás acceder a un nodo más interior.

Para nuestro primer ejemplo vamos a indicar como referencia la raíz del modelo de datos. Creamos esa referencia con el siguiente código Javascript:

var ref = new Firebase("https://juegosdemesa.firebaseio.com/");

Nota: Lógicamente, sustituirás "juegosdemesa" por el nombre de tu aplicación en la cadena "https://juegosdemesa.firebaseio.com/".

Sin embargo, aunque nuestra referencia apunte a la raíz, puedo dirigirme a uno de sus hijos con el método child().

ref.child("appName")

## Evento "value" para sincronizarme con el dato

En Firebase cuando accedes a una información puede tener un estado determinado, pero sin embargo al ser una aplicación en tiempo real, ese estado puede cambiar en cualquier instante. Firebase me avisará en el mismo momento en el que se ha producido el cambio facilitándome el nuevo estado de manera automática.

Por tanto, mientras que en otro tipo de sistemas consultarías un dato una vez, en Firebase debemos estar preparados para los cambios posteriores.

Por eso para leer un dato lo que vamos a hacer es suscribirnos a un evento. Nada más suscribirnos obtendremos el valor actual y cada vez que el dato cambie se disparará ese mismo evento indicando el nuevo valor alojado en ese dato. Igual que en librerías como jQuery, nos suscribimos a eventos con el método on(). El evento al que nos vamos a suscribir es "value".

ref.child("appName").on("value", function(snapshot){

console.log(snapshot.val());

document.getElementById("titular").textContent = snapshot.val();

});

El método on() me permite suscribirme indicando el nombre del evento que me interesa, en este caso "value" y la función que queremos registrar como manejadora del evento. En esa función recibiremos el "snapshot" que es una instantánea del dato que tenemos en Firebase. Cuando cambie se disparará de nuevo el método recibiendo la siguiente instantánea.

Para acceder al valor de la instantánea usamos el método val().

En este caso dentro del manejador de evento mostramos el valor por console.log() y luego usamos ese mismo valor para cargarlo como contenido en un elemento del DOM que tiene el id "titular".

## Conexiones por Sockets

En Firebase las conexiones se realizan mediante sockets, por eso es tan rápido en recibir los cambios que hayan realizado en el modelo. Cuando un cliente abre una página donde se usa Firebase se mantiene abierto un canal de comunicación con el servidor bidireccional, que permite comunicarse desde el cliente al servidor y desde el servidor a todos los clientes.

Para experimentarlo abre la página que acabas de programar. Por si no ha quedado claro lo que deberías tener, te mostramos el código completo.

Nota: Recuerda que debes abrir, siempre como norma habitual, los ejemplos de Firebase desde un servidor web, igual que ocurre con los ejemplos de Javascript en general donde se usa Ajax o Web Components.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Hola Firebase</title>

<script src="bower\_components/firebase/firebase.js"></script>

<script>

var ref = new Firebase("https://juegosdemesa.firebaseio.com/");

ref.child("appName").on("value", function(snapshot){

console.log(snapshot.val());

document.getElementById("titular").textContent = snapshot.val();

});

</script>

</head>

<body>

<h1 id="titular"></h1>

</body>

</html>

Una vez abierto en el navegador, desde el sistema de "Admin" de tu app Firebase en la sección "Data", simplemente cambia el dato que hay en el campo "appName" que habíamos creado al principio. El cambio debería trasladarse casi al instante hacia todos los navegadores que tuvieses visualizando en esa página. Si no es así revisa los pasos anteriores y consulta el vídeo que encontrarás un poco más abajo.

Nota: Firebase también responde a las operaciones por medio de una interfaz "REST", como cualquier API REST que puedas conocer. De hecho las conexiones mediante REST son las que utilizarás desde lenguajes de servidor como PHP o Python. Sin embargo, las comunicaciones por sockets son mucho más rápidas debido a que no tienen que hacer toda la operativa de una request al servidor con HTTP, con todas las etapas del protocolo. El canal de comunicación permanece abierto y los datos fluyen de un lado a otro sin más. Además los sockets permiten una comunicación que ahorra llamadas en aplicaciones al servidor en tiempo real, porque el cliente no tiene necesidad de hacer constantes request para recuperar los datos que hayan cambiado, sino que simplemente tiene que escuchar el socket, a la espera que Firebase le mande los cambios cuando se estén produciendo.

## Vídeo de introducción a Firebase

Para facilitarte las cosas te contamos en vídeo todo este proceso de trabajo con Firebase que te hemos relatado en el artículo, agregando consideraciones y consejos diversos.

En el vídeo podrás ver paso por paso todas las acciones para crear tu primer ejemplo con Firebase. Esperamos que te parezca útil y si te gusta no olvides darle al "me gusta".