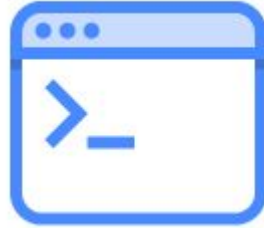


ionic





Introducción
a Ionic



Entorno de
desarrollo



Ionic UI



Ionic API



Ionic native



Estética



Introducción

Typescript



Entorno de desarrollo

Ionic básico

<u>start</u>	Crea un proyecto (tabs, blank, sidemenu, super, conference, tutorial, aws)
<u>serve</u>	Muestra la aplicación en un servidor local con el navegador
<u>docs</u>	Abre la documentación oficial de Ionic
<u>generate</u>	Genera componentes de Ionic: directive, page, pipe, provider, tabs, page Login, pipe
<u>info</u>	Muestra información del sistema y del entorno

Ionic Dashboard

login	Login a la plataforma Ionic
signup	Crea una cuenta
upload	Sube una aplicación a tu cuenta

Ionic cordova

<u>cordova build</u>	Build (prepare + compile)
<u>cordova compile</u>	Compila en la plataforma nativa
<u>cordova emulate</u>	Lanza el emulador de la plataforma nativa
<u>cordova platform</u>	Añade o quita plataformas
<u>cordova plugin</u>	Maneja los plugins
<u>cordova prepare</u>	Prepara la compilación del proyecto
<u>cordova resources</u>	Crea icono y splash screen de forma automática
<u>cordova run</u>	Ejecuta un proyecto en un dispositivo conectado al PC (USB/WiFi)



Ionic UI

Ionic UI



Ionic - Enlaces de interés

[Ionic framework](#)

[Ionic cloud services](#)

[Awesome Ionic](#)

Ionic

- ❏ UI components
- ❏ API calls
- ❏ Ionic Native
- ❏ Theming
- ❏ Testing

Ionic UI

action-sheets	icons	ranges
alerts	inputs	searchbars
badges	lists	segments
buttons	loading	selects
cards	menus	slides
checkboxboxes	modals	tabs
datetime	navigation	toast
fabs	popovers	toggles
gestures	radios	toolbar
grid		

Ionic UI - Pages

Representan pantallas de una aplicación, desde el punto de vista de Angular 2 son componentes, con un Template y Controller. Internamente consta de dos partes: header y content

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

<ion-content>
  <button (click)="goToOtherPage()">
    Go to OtherPage
  </button>
</ion-content>
```

Ionic UI - Buttons

Los botones contienen un texto y/o un icono y pueden ser modificado con atributos. Se trata de una etiqueta standard `<button>` pero tiene un modificador `ion-button` que es una directiva. El objeto Button dispara un evento (click) `(click)=someFn($event)`

```
<button ion-button color="light">Light</button>
<button ion-button>Default</button>
<button ion-button color="secondary">Secondary</button>
<button ion-button color="danger">Danger</button>
<button ion-button color="dark">Dark</button>
```

- ❑ [Default Style](#)
- ❑ [Outline Style](#)
- ❑ [Clear Style](#)
- ❑ [Round Buttons](#)
- ❑ [Block Buttons](#)
- ❑ [Full Buttons](#)
- ❑ [Button Sizes](#)
- ❑ [Icon Buttons](#)
- ❑ [Buttons In Components](#)

Ionic UI - Input

Son entradas de distinto tipo de datos. Con 2-way databinding [(ngModel)] actualizamos el valor de la variable de entrada.

```
<ion-list>

  <ion-item>
    <ion-label>Username</ion-label>
    <ion-input type="text" value=""></ion-input>
  </ion-item>

</ion-list>
```

- [Fixed Inline Labels](#)
- [Floating Labels](#)
- [Inline Labels](#)
- [Inset Labels](#)
- [Placeholder Labels](#)
- [Stacked Labels](#)

clearInput	boolean	Un botón para limpiar la entrada aparecerá cuando exista una
clearOnEdit	boolean	Se limpiará el campo cuando capture el foco
disabled	boolean	Campo deshabilitado
max	any	Valor máximo
min	any	Valor mínimo
placeholder	string	Texto que se muestra antes de introducir datos en el campo
readonly	boolean	Sólo lectura
step	any	Valor incremental del campo
type	string	Tipo de entrada de datos: "text", "password", "email", "number", "search", "tel", or "url".
value	string	Valor de entrada del campo

Ionic UI - Lists

Las listas se usan de muchas maneras y de forma masiva en las aplicaciones móviles

```
<ion-list>
  <ion-list-header>
    Action
  </ion-list-header>
  <ion-item>Terminator II</ion-item>
  <ion-item>The Empire Strikes Back</ion-item>
  <ion-item>Blade Runner</ion-item>
</ion-list>
```

- ❑ [Basic Lists](#)
- ❑ [Inset List](#)
- ❑ [List Dividers](#)
- ❑ [List Headers](#)
- ❑ [Icon List](#)
- ❑ [Avatar List](#)
- ❑ [Multi-line List](#)
- ❑ [Sliding List](#)
- ❑ [Thumbnail List](#)

Ionic UI - Cards

Son una extensión del concepto de lista, se usan para mostrar contenidos de información relevantes para el usuario y se han convertido en un standard de las aplicaciones móviles.

```
<ion-card>

  <ion-card-header>
    Card Header
  </ion-card-header>

  <ion-card-content>
    <!-- Add card content here! -->
  </ion-card-content>

</ion-card>
```

- ❏ [Basic Cards](#)
- ❏ [Card Headers](#)
- ❏ [Card Lists](#)
- ❏ [Card Images](#)
- ❏ [Background Images](#)
- ❏ [Advanced Cards](#)

Ionic UI - Navigation

La navegación entre páginas de una aplicación Ionic se realiza mediante un objeto NavController, el concepto es una pila en el que se van añadiendo o quitando páginas mediante push y pop.

Los objetos más usados para la navegación en las aplicaciones son Tabs y SideMenu, se pueden ver ejemplos de éstos generando proyectos desde la herramienta de Ionic.

Navegación - NavController

El Servicio NavController se encarga de gestionar las funciones de navegación por las páginas de nuestra aplicación mediante el patrón de diseño de pila, cuando vamos a una nueva página, se hace push sobre NavController y cuando volvemos a la página anterior, se hace pop. NavController tiene bastantes opciones para variar el modo de navegación, puedes ver más en la [documentación oficial](#)

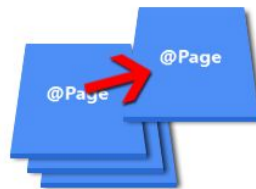
Navigation Stack



Push



Pop



Ionic 3 - ionView: ciclo de vida de páginas Ionic

Evento	Cuándo se llama
<code>ionViewDidLoad</code>	Se llama cuando la página ha sido cargada
<code>ionViewWillEnter</code>	Se llama antes de que se visualice la página
<code>ionViewDidEnter</code>	Se llama cuando ya se ha visualizado la página
<code>ionViewWillLeave</code>	Se llama antes de desaparecer del primer plano la página
<code>ionViewDidLeave</code>	Se llama después de desaparecer del primer plano la página
<code>ionViewWillUnload</code>	Se llama antes de destruir la página
<code>ionViewCanEnter</code>	Devuelve un boolean, y bloquea el acceso a esa página
<code>ionViewCanLeave</code>	Devuelve un boolean, y bloquea la transición a otra página



Ionic API

Ionic API- Infinite Scroll


Se usa para realizar nuevas peticiones a la fuente de información para aumentar los contenidos de una colección

```
<ion-content>

<ion-list>
  <ion-item *ngFor="let i of items">{{i}}</ion-item>
</ion-list>

<ion-infinite-scroll (ionInfinite)="doInfinite($event)">
  <ion-infinite-scroll-content></ion-infinite-scroll-content>
</ion-infinite-scroll>

</ion-content>
```



```
doInfinite(infiniteScroll) {
  console.log('Begin async operation');

  setTimeout(() => {
    for (let i = 0; i < 30; i++) {
      this.items.push( this.items.length );
    }

    console.log('Async operation has ended');
    infiniteScroll.complete();
  }, 500);
}
```

Ionic API - Peticiones Http con patrón Observable

Para peticiones asíncronas usamos el objeto Observable que devuelve 3 casos:

- Success
- Error
- Completed

```
this.http.get(url).subscribe(  
  (data) => {  
    console.log(data);  
  },  
  (err) => {  
    console.log(err);  
  },  
  () => {  
    console.log("completed");  
  }  
);
```




Ionic native

Ionic 2 + Cordova

Qué hay que tener instalado

Hace falta tener Ionic y Cordova instalado mediante npm:

```
> npm install -g ionic cordova
```

Hay que instalar el JDK de Java y Android SDK (Google ahora proporciona el sdk de Android aparte). Puedes encontrar los enlaces [aquí](#)

Para trabajar con las plataformas nativas

Primero tenemos que añadir las plataformas que vamos a usar en el proyecto:

```
> ionic cordova platform add android
```

Para lanzar en el emulador la aplicación:

```
> ionic cordova emulate android
```

Para generar un apk sin firmar:

```
> ionic cordova build android
```

Firmar apks

En si queremos instalar en un dispositivo físico, es recomendable firmar el apk, hay que usar la herramienta jarsigner dentro de Java. Un ejemplo en Windows:

```
>"D:\Program Files\Java\jdk1.8.0_131\bin\jarsigner.exe" -verbose -sigalg  
SHA1withRSA -digestalg SHA1 -keystore <tu archivo de clave> <path del apk>  
<usuario>
```

Enviar al dispositivo el apk

Ya tenemos el apk firmado y sólo queda enviarlo al dispositivo, recuerda habilitar las opciones de desarrollador e instalar apps desde el USB. hay que conectar el dispositivo mediante USB al equipo, para comprobar que está conectado y puedes comunicarte con él, pon en el terminal:

```
>adb devices  
List of devices attached  
GC000120    device
```

Si tienes una salida como la anterior, ya sólo tienes que instalar el apk:

```
> adb install -r <path de tu apk>
```

Añadiendo un plugin de Cordova (1)

En [la página oficial de cordova](#) tenemos la lista de plugins disponibles para añadir a nuestra app.

Para añadir un plugin, primero hay que instalar ionic-native en el proyecto:

```
> npm install @ionic-native/core --save
```

Añadiendo un plugin de Cordova (2)

Ahora queda instalar el plugin de Ionic 2 que vamos a usar en nuestro proyecto:

```
> npm install @ionic-native/camera --save
```

Tenemos que importar el plugin de cordova correspondiente

```
> ionic cordova plugin add cordova-plugin-camera
```


Añadiendo un plugin de Cordova (3)

En el módulo de la app hay que importar el plugin que hemos instalado:

```
import { Camera } from '@ionic-native/camera';
```

```
...
```

```
@NgModule({
```

```
...
```

```
  providers: [
```

```
    ...
```

```
    Camera
```

```
    ...
```

```
  ]
```

```
...  
})
```

```
export class AppModule {}
```

Añadiendo un plugin de Cordova (4)

Tenemos que usar la llamada a platform para comprobar que el sistema está respondiendo a las peticiones que hacemos desde la app, para ello hay que importar platform e inicializarlo en los parámetros del constructor

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

Usamos la estructura de promesa para hacer las peticiones:

```
platform.ready().then(() => {  
  geolocation.getCurrentPosition().then(pos => {  
    console.log('lat: ' + pos.coords.latitude + ', lon: ' + pos.coords.longitude);  
  });  
});
```

Añadiendo un plugin de Cordova (5)

También podemos suscribirnos a los eventos que genere nuestra petición mediante un Observable:

```
platform.ready().then(() => {  
    const watch = geolocation.watchPosition().subscribe(pos => {  
        console.log('lat: ' + pos.coords.latitude + ', lon: ' + pos.coords.longitude);  
    });  
    // to stop watching  
    watch.unsubscribe();  
});
```



Estética

Sass