UA . MASTER MOVILES

MÁSTER UNIVERSITARIO EN DESARROLLO DE SOFTWARE PARA DISPOSITIVOS MÓVILES

PROGRAMACIÓN HIPERMEDIA PARA DISPOSITIVOS MÓVILES

Ionic v4 - Introducción

CONTENIDO

1. Introducción
2. Instalación
3. Crear y probar un proyecto
4. Contenido de un proyecto
5. Páginas

INTRODUCCIÓN

- **Ionic** es un *framework* open source construido usando HTML5, CSS3 y Javascript para el desarrollo de **aplicaciones híbridas** para dispositivos móviles.
- Incluye una completa **librería de componentes**, estilos y animaciones que simulan el aspecto nativo de las distintas plataformas.
- Utiliza **Angular** para el desarrollo del código dinámico de la aplicación.
- Está perfectamente integrado con **Cordova**, por lo que podremos compilar nuestros proyectos directamente.

INSTALACIÓN

- Para instalar Ionic primero necesitamos Node.Js
- Lo podemos instalar desde su Web:

  https://nodejs.org/en/

- Descomprimir el zip y acceder al directorio que se genera.
- Compilar la librería, según el sistema operativo:
  - Mac: ejecutar el “pkg” que se descarga.
  - Windows: ejecutar el “smi” que se descarga.
  - Linux, ejecutar los siguientes comandos:

```
$ ./configure
$ make
$ make install
```

INSTALACIÓN

- A continuación instalamos Ionic con:

```
$ sudo npm install -g cordova ionic
```

* En Windows ejecutaremos el mismo comando pero sin sudo.

* Para instalar la última versión de Ionic: `ionic@latest`

- Para comprobar que se ha instalado correctamente ejecutamos:

```
$ ionic
```

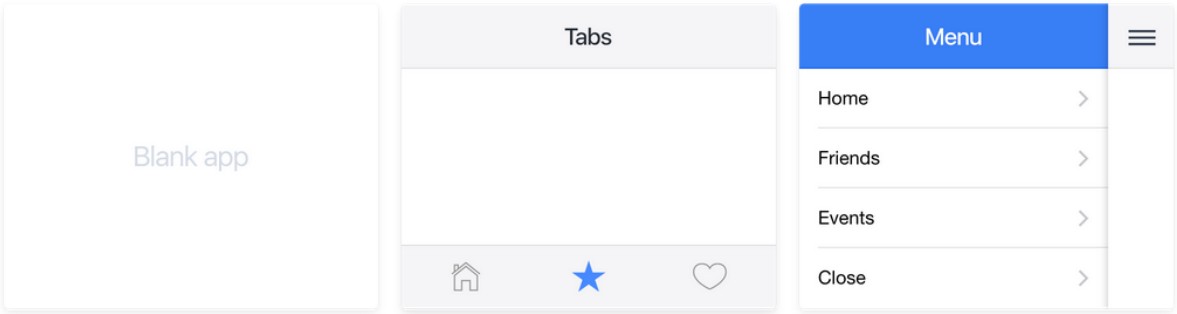
CREAR UN PROYECTO

- Crear un nuevo proyecto en blanco:

```
$ ionic start myApp blank
```

- Crear proyectos con plantillas predefinidas:

```
$ ionic start myApp tabs
$ ionic start myApp sidemenu
```



- Si no indicamos el tipo nos lo preguntará.

ABRIR EN EL NAVEGADOR

- Podemos ver y depurar un proyecto en el navegador ejecutando:

```
$ ionic serve
```

- Tenemos que dejar la consola abierta mientras estemos trabajando para que funcione el servidor.
- Para finalizarlo tenemos que pulsar Ctrl+C.
- Como opción podemos lanzar el servidor con "**`--lab`**"
  - Esto nos permitirá ver el resultado obtenido para cada plataforma.
  - La primera vez tendremos que confirmar la instalación de "`@ionic/lab`"
- *NOTA:* Al guardar cambios se recargará automáticamente la aplicación en el navegador. Si no funciona en las últimas versiones de Linux:

```
echo fs.inotify.max_user_watches=524288 | sudo tee /etc/sysctl.d/40-max-user-watches.conf &&
sudo sysctl --system
```

AÑADIR PLATAFORMAS

- Para ver las plataformas disponibles ejecutamos:

```
$ ionic cordova platform
```

- Para añadir una plataforma:

```
$ ionic cordova platform add <plataforma>
```

- Por ejemplo:

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

- Podemos utilizar los mismos comandos que con Cordova.
- La primera vez que ejecutemos "`ionic cordova`" instalará las dependencias de este *plugin* para Ionic.

AÑADIR PLATAFORMAS

	Mac	Linux	Windows
Android	x	x	x
iOS	x		
OS X	x		
Windows 8.1 Phone 8.1, 10			x
Browser	x	x	x

EMULAR O INSTALAR

- También podemos emular o instalar un proyecto con:

```
$ ionic cordova emulate <platform>
$ ionic cordova run <platform>
```

- Por ejemplo:

```
$ ionic cordova emulate android
```

- Opcionalmente podemos añadir las opciones:

```
--livereload o -l
--consolelogs o -c
```

- También podemos compilar un proyecto:

```
$ ionic cordova build <platform>
```

- **NOTA**: La primera vez que ejecutemos estos comandos nos pedirá que instalemos "`npm i -g native-run`"

ESTRUCTURA DE UN PROYECTO

Al crear un nuevo proyecto se generará la siguiente estructura:

- `e2e/` → (*End-to-end*) Test automatizados.
- `node_modules/` → Módulos y dependencias instaladas de Ionic.
- **`platforms`**`/` → Código del proyecto para las distintas plataformas.
- **`plugins`**`/` → Módulos para acceso a características nativas.
- `resources/` → Iconos y *splashscreen* específicos de las plataformas.
- **`src`**`/` → Código fuente principal de nuestra aplicación.
- **`www`**`/` → Código web compilado.
- **`config.xml`** → Contiene la configuración de Cordova.
- `ionic.config.json` → Configuración del proyecto de Ionic.
- `package.json` → Dependencias y paquetes de Node.Js.

ESTRUCTURA DE UN PROYECTO: CARPETA “SRC”

Contenido de la carpeta “**src**”:

- **app**/ → Contiene el código de la aplicación:
  - Páginas
  - Servicios (o proveedores de contenido)
  - Componentes
  - Directivas
  - Módulo y componente principal de la app
- **assets**/ → Recursos de la aplicación: imágenes, vídeos, etc.
- **theme**/ → Temas y estilos de la aplicación.
- **manifest.json** → Configuración del proyecto.
- **index.html** → Fichero principal que iniciará la aplicación.
- **main.ts** → Fichero que inicia la carga de la aplicación.

ESTRUCTURA DE UN PROYECTO: CARPETA “APP”

- Contenido de la carpeta “**`src/app`**”:
  - **`home/`** → Página principal incluida de ejemplo.
  - **`app.component.ts`** → Componente principal.
  - **`app.component.html`** → Plantilla del componente principal.
  - **`app.component.scss`** → Hoja de estilo del componente principal.
  - **`app.module.ts`** → Módulo principal de la aplicación.
- Dentro de esta carpeta se añadirán:
  - Las nuevas páginas y componentes dentro de sus propias carpetas.
  - Las nuevas directivas y servicios en la carpeta raíz.

CÓDIGO BÁSICO. FICHERO `INDEX.HTML`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Ionic App</title>
  <base href="/" />
  <meta name="viewport" content="width=device-width, ..."/>
  <meta name="..." content="..."/>
  <meta name="..." content="..."/>
  <link rel="icon" type="image/png" href="assets/icon/favicon.png" />
</head>
<body>
  <!-- Componente principal de Ionic que cargará la aplicación -->
  <app-root></app-root>
</body>
</html>
```

PÁGINAS

- Las pantallas de una aplicación en Ionic se crean mediante "`pages`" o páginas.
- Las páginas son lo mismo que los componentes en Angular.
- Se almacenarán dentro de la carpeta "`src/app`" dentro de sus propias carpetas.
- Y contendrán tres elementos principales:
  - La plantilla o vista asociada (`.html`).
  - La hoja de estilo en SASS (`.scss`).
  - Fichero en TypeScript con la clase que define la página (`.ts`).

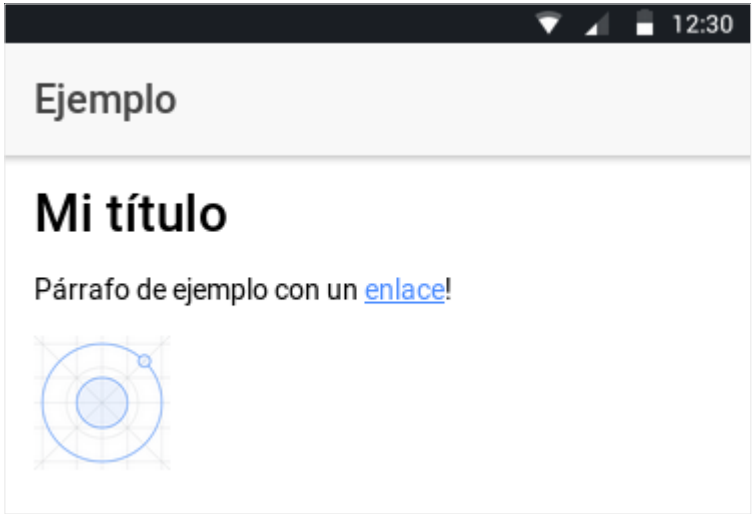
PLANTILLA DE UNA PÁGINA

- La plantilla (`.html`) asociada con una página contendrá el siguiente código por defecto:

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

<ion-content>

</ion-content>
```

PLANTILLA DE UNA PÁGINA

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

<ion-content class="ion-padding">
  <h1>Mi título</h1>
  <p>Párrafo de ejemplo con un <a href="#">enlace</a>!</p>
  <img src="assets/icon/favicon.png" width="20%" />
</ion-content>
```



CREAR NUEVAS PÁGINAS

- Para crear nuevas páginas usamos el siguiente comando:

```
$ ionic generate page pagina2

$ ionic g page pagina2
```

- Este comando creará la carpeta "`/src/app/pagina2`" con los siguientes ficheros:
  - `pagina2.module.ts` → Definición de módulo
  - `pagina2.page.html` → Plantilla o vista
  - `pagina2.page.scss` → Hoja de estilo
  - `pagina2.page.ts` → Clase TypeScript
  - `Pagina2.page.spec.ts` → Unit test

RUTAS

- Al crear una página automáticamente se añade una entrada al fichero de rutas "`app-routing.module.ts`"
- Dentro de este fichero podremos ver que hay un array con las páginas de la aplicación:

```
const routes: Routes = [
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  {
    path: 'home',
    loadChildren: () => import('./home/home.module').then(
                                m => m.HomePageModule)},
  {
    path: 'autor',
    loadChildren: () => import('./autor/autor.module').then(
                                m => m.AutorPageModule)
  },
];
```

¿PREGUNTAS?