Creación de un proyecto con Ionic

Contenido

Introducción:

Ionic es un SDK completo de código abierto para el desarrollo de aplicaciones móviles híbridas basado en HTML, CSS y JS.

En esta guía se plantea el paso a paso de instalación e integración de Ionic a un proyecto.

Instalación de Node JS

1-Descargar Node JS.

Ingresar a nodejs.org y descargar la versión estable (LTS) e instalarla en el equipo.

Es un proceso guiado que permite instalar node en el equipo, para ello solo es necesario aceptar las condiciones y seguir el curso que propone el paso a paso.

2-Verificar instalación.

Una vez instalado Node en el equipo debemos verificar la instalación, para ello abrir una consola de línea de comandos (CMD) e ingresar lo siguiente:

```
node --version
```

La línea de comandos anterior permite saber que versión de Node quedó instalada en el equipo.

3-Verificar NPM (Node package manager)

Además de verificar la versión que quedo instalada de Node, es necesario verificar la versión de npm instalada, para ello escribir en consola, la siguiente línea de comando

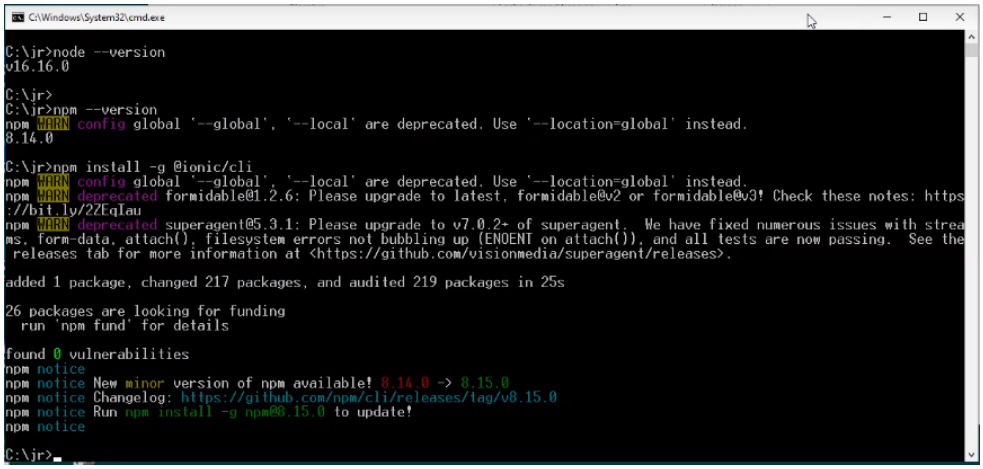
```
npm --version
```

4-Instalar IONIC

Mediante npm instalar Ionic de forma global, escribiendo lo siguiente en la consola de línea de comandos (CMD)

```
npm install -g @ionic/cli
```

Al finalizar la instalación, se visualiza en el CMD el estado del proceso en una ventana que se ve similar a esta:



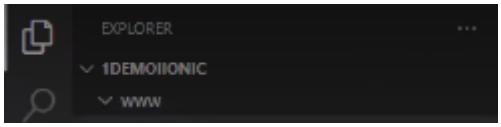
Luego de finalizada la instalación, comenzamos a trabajar con IONIC.

Crear una aplicación con IONIC

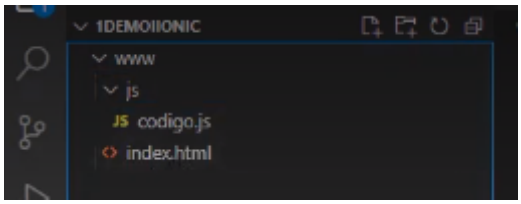
Crear proyecto

1-Crear una carpeta para el nuevo proyecto (para este tutorial le pondremos de nombre 1DEMOIONIC) y dentro de ella, una subcarpeta con el nombre “www”.

Abrir con VS Code la carpeta 1DEMOIONIC, la estructura que se visualiza debe ser la siguiente:



2-Crear el archivo index.html dentro de www y dentro de ella esta última, crear una nueva carpeta llamada “js”, la cual va a contener un archivo llamado código.js dentro.



3-Agregar librerías de IONIC.

Las librerías de IONIC se puede agregar de dos formas distintas:

Utilizando cdn. Ionic el cual ofrece un link que es útil para trabajar con IONIC y JS vanilla. En https://ionicframework.com/docs encontramos un link a los paquetes CDN. Copiar y pegar este código en el head de la página index.html.


```
<script type="module"
src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionic/ionic.esm.js"></script>
<script nomodule src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionic/ionic.js"></script>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@ionic/core/css/ionic.bundle.css" />
```

4-Comenzar a crear la aplicación.

A efectos de esta guía utilizaremos los fragmentos de código que se brindan en la documentación de Ionic https://ionicframework.com/docs/layout/structure, y los complementaremos con funcionalidades JavaScript.

Nota: Esta guía utiliza capturas de pantalla, el código para poder copiar se encuentra al final del documento.

Toda la aplicación Ionic debe estar envuelta en una etiqueta <ion-app></ion-app> dentro del body en el index.html
En el primer ejemplo colocaremos un header y un contenido.

5-Creación de un menú desplegable con dos secciones:

```
<ion-app>
//Aqui va toda la aplicación
</ion-app>
```

Dentro de las etiquetas de apertura y cierre de ion-app, vamos a crear nuestro menú que contendrá 2 secciones.

```
<!--Menu-->
<ion-menu side="start" menu-id="first" content-id="main" id="menu">
        <ion-header>
                <ion-toolbar color="primary">
                        <ion-title>Start Menu</ion-title>
                </ion-toolbar>
        </ion-header>
        <ion-content>
                <ion-list>
                        <ion-item href="/">Home</ion-item>
                        <ion-item href="/login">Login</ion-item>
                </ion-list>
        </ion-content>
</ion-menu>
<div id="main"></div>
```

El último div con id “main” es necesario para que el menú se despliegue en pantalla.

En este punto podemos probar la aplicación e intentar desplegar el menú arrastrando el mismo de izquierda a derecha en el navegador.

6-Agregar las pantallas para las secciones creadas en el punto anterior

Nuestra aplicación va a contar con varias secciones, pero todas están en el mismo documento html. Utilizando JS vamos a lograr mostrar y ocultar las distintas secciones de acuerdo con los ítems de menú seleccionados.

Para comenzar vamos a crear dos secciones iguales, una para el home, y otra para el login.

Cada una de ellas estará comprendida dentro de etiquetas page-home que darán nombre a cada sección.

Estas etiquetas deberán tener class ion-page.

```
<!-- Pantallas -->
<page-home class="ion-page" id="pantalla-home">
        <ion-header>
                <ion-toolbar>
                        <ion-buttons slot="start">
                                <ion-menu-button></ion-menu-button>
                        </ion-buttons>
                        <ion-title>Home</ion-title>
                </ion-toolbar>
        </ion-header>
        <ion-content>
                Pantalla de home
        </ion-content>
</page-home>
```

```
<page-login class="ion-page" id="pantalla-login">
        <ion-header>
                <ion-toolbar>
                        <ion-buttons slot="start">
                                <ion-menu-button></ion-menu-button>
                        </ion-buttons>
                        <ion-title>Login</ion-title>
                </ion-toolbar>
        </ion-header>
        <ion-content>
                Pantalla de login
        </ion-content>
</page-login>
```

En este ejemplo se agrega a cada página un pequeño header con un botón para mostrar el menú, y una sección para el contenido.

7-Ruteo

El componente de ruteo permite definir qué se desea mostrar cuando se accede a una url específica de la aplicación. Si prestamos atención a los ítems creados en el menú, observamos que cada uno tiene una url asignada. La url de la aplicación por defecto es la que se indica con solamente la barra y al asociarle el componente page-home, estamos indicando que al seleccionar esta opción queremos que se muestre lo que está definido en la sección home, lo mismo cuando seleccionas la opción login, queremos que se muestre lo que está en la sección definida como login.

Este componente de ruteo se agrega luego de la etiqueta de apertura de ion-app.

```
<!-- Componente de ruteo -->
<ion-router id="ruteo">
        <ion-route url="/" component="page-home"></ion-route>
        <ion-route url="/login" component="page-login"></ion-route>
</ion-router>
<ion-nav></ion-nav>
<!--Fin del Componente de ruteo -->
```

Para lograr la navegación entre los distintos componentes, tenemos que asignar el comportamiento a cada ítem del menú. Esto lo hacemos mediante JS, indicando qué función de JS se debe ejecutar.

En la imagen que se muestra a continuación se agregó a cada ítem la función cerrarMenu asociada al evento onclick. Esta función va a permitir que el menú se cierre automáticamente.

Esta función va a estar escrita en el archivo js que se encuentra en www/JS

```
<ion-item href="/" onclick="cerrarMenu()">Home</ion-item>
<ion-item href="/login" onclick="cerrarMenu()">Login</ion-item>
```

8-Comportamiento JavaScript.

Creación de variables globales

Desde JS llamaremos constantemente a algunos componentes que son estáticos, por lo cual dejaremos establecido su acceso de forma más rápida.

Estos componentes son: el menú, el componente de ruteo, y las dos páginas (home y login).

```
1   const MENU = document.querySelector("#menu");
2   const ROUTER = document.querySelector("#ruteo");
3   const HOME = document.querySelector("#pantalla-home");
4   const LOGIN = document.querySelector("#pantalla-login");
```

Cerrar menú automáticamente

Para poder responder al onlick definido en el html (punto 7), crear la función.

```
function cerrarMenu() {
        MENU.close();
}
```

9-Ion Router

El ion-route tiene eventos que nos permiten saber en qué momento el usuario desea acceder a una página dentro del nuestra aplicación. Ellos son

Name	Description
ionRouteDidChange	Emitted when the route had changed
ionRouteWillChange	Event emitted when the route is about to change

A efectos de esta aplicación tendremos en modo escucha al evento ionRouteDidChange del componente router (previamente capturado en const ROUTER), el cual al ejecutarse llamará a la función que se encargará de definir la navegación.

```
6   ROUTER.addEventListener('ionRouteDidChange', Navegar);
```

10-Navegación

La función Navegar recibe por parámetros el evento que la desencadenó. Este objeto posee mucha información, pero la más relevante es que nos indica hacia donde está queriendo navegar.

```
function Navegar(evt) {

}
```

La ruta hacia donde se desea navegar está definida en evt.detail.to, probar imprimir en consola esta propiedad.

```
function Navegar(evt) {

        console.log(evt.detail.to);
```

```
/                                                    codigo.js:17
/login                                               codigo.js:17
```

Ahora ya sabemos hacia donde desea navegar el usuario, y al tratarse de una aplicación de 1 página, la navegación se realiza ocultando y mostrando componentes.

El procedimiento será ocultar todas las pantallas y mostrar únicamente la pantalla solicitada.

Para ocultar todo podemos crear una función que llamaremos en la función de Navegar.

```
function OcultarPantallas(){
        HOME.style.display = "none";
        LOGIN.style.display = "none";
}
```

En la función de navegar, preguntamos por la ruta a la que se desea ir, y en función de su nombre, mostraremos la pantalla correspondiente.

```
function Navegar(evt) {
        const ruta = evt.detail.to;
        OcultarPantallas();
        if (ruta == "/") {
                HOME.style.display = "block";
        }else  if (ruta == "/login") {
                LOGIN.style.display = "block";
        }
}
```

Código

HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script type="module"
src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionic/ionic.esm.js"></script>
        <script nomodule
src="https://cdn.jsdelivr.net/npm/@ionic/core/dist/ionic/ionic.js"></script>
        <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/@ionic/core/css/ionic.bundle.css" />
        <title>Titulo</title>
</head>

<body>

        <ion-app>
                <!-- Componente de ruteo -->
                <ion-router id="ruteo">
                        <ion-route url="/" component="page-home"></ion-route>
                        <ion-route url="/login" component="page-login"></ion-route>
                </ion-router>
                <ion-nav></ion-nav>
                <!--Fin del Componente de ruteo -->

                <!--Menu-->
                <ion-menu side="start" menu-id="first" content-id="main" id="menu">
                        <ion-header>
```

```
                        <ion-toolbar color="primary">
                                <ion-title>Start Menu</ion-title>
                        </ion-toolbar>
                </ion-header>
                <ion-content>
                     <ion-list>
                       <ion-item href="/" onclick="cerrarMenu()">Home</ion-item>
                       <ion-item href="/login" onclick="cerrarMenu()">Login</ion-item>
                     </ion-list>
                </ion-content>
        </ion-menu>
        <div id="main"></div>

        <!-- Pantallas -->
        <page-home class="ion-page" id="pantalla-home">
                <ion-header>
                        <ion-toolbar>
                                <ion-buttons slot="start">
                                        <ion-menu-button></ion-menu-button>
                                </ion-buttons>
                                <ion-title>Home</ion-title>
                        </ion-toolbar>
                </ion-header>
                <ion-content>
                        Pantalla de home
                </ion-content>
        </page-home>
        <page-login class="ion-page" id="pantalla-login">
                <ion-header>
                        <ion-toolbar>
                                <ion-buttons slot="start">
                                        <ion-menu-button></ion-menu-button>
                                </ion-buttons>
                                <ion-title>Login</ion-title>
                        </ion-toolbar>
                </ion-header>
                <ion-content>
                        Pantalla de login
                </ion-content>
        </page-login>
    </ion-app>
    <script src="js/codigo.js"></script>
</body>
</html>
```

JS

```
const MENU = document.querySelector("#menu");
const ROUTER = document.querySelector("#ruteo");
const HOME = document.querySelector("#pantalla-home");
const LOGIN = document.querySelector("#pantalla-login");
ROUTER.addEventListener('ionRouteDidChange', Navegar);
function cerrarMenu() {
        MENU.close();
}

function Navegar(evt) {
        const ruta = evt.detail.to;
        OcultarPantallas();
        if (ruta == "/") {
                HOME.style.display = "block";
        }else  if (ruta == "/login") {
                LOGIN.style.display = "block";
        }
}

function OcultarPantallas(){
        HOME.style.display = "none";
        LOGIN.style.display = "none";
}
```

Incorporar IONIC a un proyecto web existente.

Para incorporar IONIC a un proyecto existente se debe armar la estructura de la aplicación, sus secciones, el ruteo y el comportamiento tal como se describe en esta guía.

Además, se debe trasladar todos los componentes visuales existentes a cada página de la aplicación.

Por último, debemos utilizar las etiquetas de IONIC que sustituyan a las del proyecto.

Recomendamos ver la documentación de los componentes de IONIC que debemos utilizar, en lugar de los de HTML.

https://ionicframework.com/docs/components