UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

# Desarrollo de aplicaciones para móviles

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
**www.sceu.frba.utn.edu.ar/e-learning**

UTN.BA UNIVERSIDAD TECNOLÓGICA NACIONAL FACULTAD REGIONAL BUENOS AIRES | Centro de e-Learning

# Modulo 1: Programación y plataformas

## IONIC estructura de directorio

# Presentación:

En la presente unidad nos instalaremos IONIC framework.

Para realizar su instalación debemos instalar Java JDK, Node y la SDK de android.

A continuación te encontraras con un mini tutorial paso a paso sobre las aplicaciones antes mencionadas.

# Objetivos:

## Que los participantes*:

- Aprender como utilizar la estructura de directorio de IONIC 2
- Aprender que es vista y Controladores
- Como utilizar typescript

## Bloques temáticos*:

- Estructura de directorio en IONIC 2
- Modificando nuestro “hola mundo”
- Vista y controlador
- Controladores (.ts)
- Definir variables en TypeScript
- Mini juego para acertar números
- Ejercicio

# Consignas para el aprendizaje colaborativo

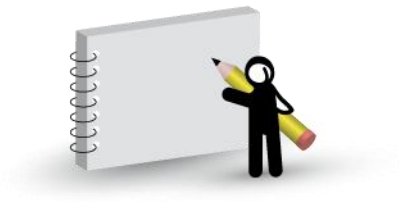
En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



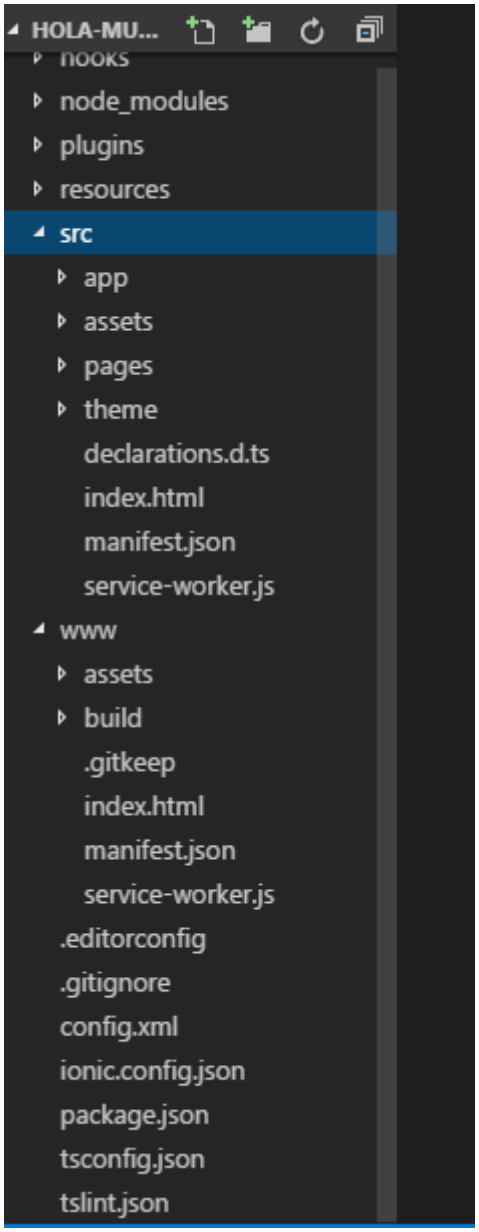
## Tomen nota*

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

# Estructura de directorio en IONIC 2

Al crear un proyecto con ionic 2 se crea una carpeta con el nombre del proyecto y dentro de ella una estructura de archivos y directorios que contiene todos los elementos del proyecto.



Veamos que contiene cada carpeta:

- **hooks**: Contiene scripts que se ejecutan en el proceso de construcción de la app y que pueden ser creados por el propio sistema de compilación para personalizar comandos de cordova, automatizar procesos, etc.
  *Normalmente no tendremos que modificar nada aquí.*

- **node_modules**: La carpeta node_modules se genera automáticamente al instalar las dependencias npm con “npm install”. Este comando explora el archivo package.json para todos los paquetes que necesitan ser instalados.
  *No necesitamos tocar nada en esta carpeta.*

- **platforms**: En esta carpeta se generará los proyectos nativos para cada plataforma que hayas añadido previamente. Si hemos añadido la plataforma IOS y Android se creará una carpeta llamada ios y otra llamada android y dentro tendrán los archivos y carpetas con la estructura de un proyecto nativo.
  Estas carpetas se generan al añadir la plataforma y se actualizan cada vez que compilas o ejecutas el proyecto en un emulador o en un dispositivo.
  *Salvo excepciones no tendremos que modificar nada en estas carpetas.*

- **plugins**: Contiene los plugins de Cordova que hayamos instalado. Se crean automáticamente en esta carpeta al instalar un plugin así que tampoco tendremos que modificar nada en esta carpeta a mano.

- **resources**: Contiene el icono y la “splash screen” (pantalla de presentación) de la aplicación con la que después podremos crear automáticamente todas las imágenes en todos los tamaños necesarios para cada plataforma, lo que nos ahorrará mucho tiempo al no tener que generar a mano todos los tamaños de imagen necesarios del icono y la pantalla de presentación.

- **src**: Esta es la carpeta más importante y donde realizaremos la mayor parte de nuestro trabajo. Aquí es donde están los archivos con el contenido de nuestra aplicación, donde definimos las pantallas, el estilo y el comportamiento que tendrá nuestra aplicación.

- **www**: Esta carpeta se genera automáticamente y contiene la versión actual del código cada vez que efectuamos un cambio. Si habías trabajado anteriormente con ionic 1 puede que tengas la tentación de editar el contenido de esta carpeta pero *NO debemos cambiar nada aquí* ya que todo lo que cambiemos en esta carpeta se machacará con cada cambio que realicemos en la carpeta src que es donde realmente debemos editar nuestra aplicación.

Veamos ahora varios archivos que se generan al crear un proyecto com ionic 2.

- **.editorconfig y .gitignore** son dos archivos ocultos, así que dependiendo de tu sistema puede que no los veas, están relacionados con la configuración del editor de código y Git, en principio no tenemos que preocuparnos por ellos.
- **config.xml**: El archivo config.xml contiene parámetros que se utilizan cuando se construye un proyecto nativo a partir de un proyecto ionic. Aquí deberemos indicar los permisos especiales que necesite la aplicación y otras configuraciones que puedan ser necesarias.
- **Ionic.config.json**: Contiene información básica sobre la configuración nuestro proyecto, se utiliza si vas a subir tu aplicación a la plataforma Ionic.io.
- **package.json**: Contiene paquetes y dependencias de nodeJS.
- **tsconfig.json y tslint.json**: Son archivos que contienen información necesaria a la hora de compilar TypeScript, no necesitamos editar estos archivos.

Aunque pueda parecer complicado en realidad la mayoría de los elementos los gestiona automáticamente Ionic y nosotros solo tenemos que preocuparnos de la carpeta src que es donde se va a situar nuestro código. Ocasionalmente puede que tengamos que editar algún archivo fuera del directorio src como el archivo config.xml.

# Modificando nuestro “hola mundo”

Si desplegamos el directorio src podemos ver la carpeta **pages**, en está carpeta es donde se van a alojar todas la páginas que contenga nuestra aplicación. Para que nos entendamos una página será como una vista o un pantalla de nuestra aplicación.

Al crear un proyecto con la plantilla blank ionic genera por defecto un página llamada **home**, que como su propio nombre indica es la pagina inicial que se va a mostrar al iniciar nuestra aplicación. Esta página la podemos mantener como página principal y modificarla, o podemos eliminarla y crear otra con el nombre que nosotros queramos. De momento vamos a mantener la que nos ha creado por defecto y vamos a modificar su contenido.

El archivo **home.html** que contiene la plantilla html de la página.

El archivo **home.scss** que contiene el archivo sass donde podremos modificar el estilo de los componentes de la página.

El archivo **home.ts** que es el archivos typescript que contiene el controlador de la página, donde definiremos el comportamiento de la misma, como por ejemplo la función con la lógica a ejecutarse cuando se pulse sobre un botón de la página etc.

## Nuestro home.html

Si abrimos el archivo home.html veremos que contiene algo como esto:

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Ionic Blank
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  The world is your oyster.
  <p>
    If you get lost, the <a
href="http://ionicframework.com/docs/v2">docs</a> will be your guide.
  </p>
</ion-content>
```

## Componentes de IONIC 2 y Angular 2

Las paginas se pueden crear utilizando html puro, sin embargo aquí podemos ver algunas etiquetas que no corresponden con las etiquetas html “estándar”. Lo que vemos aquí son componentes de ionic.

Ionic nos ofrece una amplia gama de componentes listos para utilizar y que nos facilitarán la labor de crear nuestra interfaz de usuario con un estilo atractivo y profesional.

Iremos viendo  diferentes componentes de ionic según los vayamos necesitando a lo largo de este curso. En este enlace podes consultar en la documentación oficial de ionic los componentes disponibles con pequeños ejemplos de cómo implementarlos: ***https://ionicframework.com/docs/v2/components/***

Todos los componentes de ionic comienzan con el prefijo “**ion-**“.

Como ionic está basado en Angular 2  si en algún caso no nos es suficiente con los componentes que nos ofrece ionic podríamos crear nuestros propios componentes personalizados de la misma manera que en angular 2, aunque en la mayoría de los casos no será necesario ya que ionic nos ofrece una amplia gama de componentes para poder desarrollar nuestras aplicaciones.

## Modifiquemos nuestro home.html

En la página principal (y única de momento) de nuestro proyecto hola-mundo vemos que tenemos los siguientes componentes:

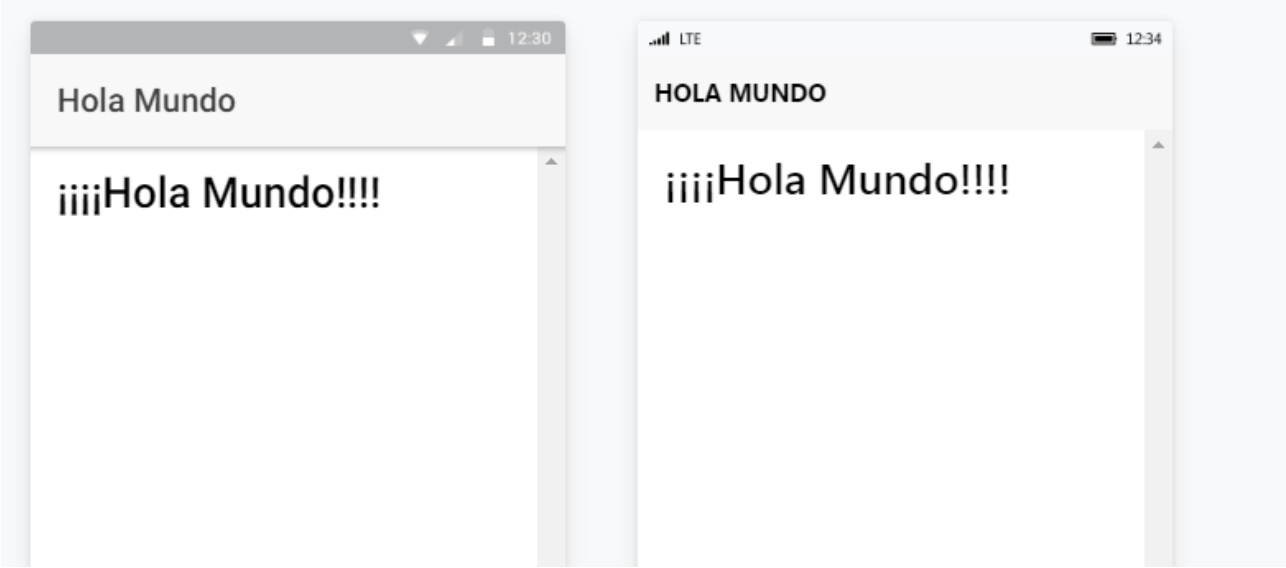
- **ion-header**: Cabecera.
- **ion-navbar**: Barra de navegación.
- **ion-title**: Título.
- **ion-content**: Contenido de la página.

Bien, vamos a cambiar el contenido de ion-title por “Hola Mundo”, también vamos a borrar todo lo que hay dentro de la etiqueta ion-content y vamos a poner “<h1>¡¡¡¡Hola mundo!!!!<h1>”, así el código de home.html debería quedar de la siguiente manera:

```
<ion-header>
  <ion-navbar>
    <ion-title>
      Hola Mundo
    </ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
 <h1>¡¡¡¡Hola Mundo!!!!</h1>
</ion-content>
```

Ejecutemos **ionic serve –l** para ver como quedo nuestra aplicación.



## Vista y controlador

- **Vista:** Dentro de la carpeta page ubicamos los archivos terminados en .html. La vista se refiere al HTML que el usuario vera en su aplicación.
- **Controlador:** Los controladores nos permiten mediante programación implementar la lógica de la presentación en AngularJS. En ellos podemos mantener el código necesario para inicializar una aplicación, gestionar los eventos, etc. Podemos decir que gestionan el flujo de la parte del cliente, lo que sería programación para implementar la funcionalidad asociada a la presentación. Podemos ubicar los controladores dentro de los archivos .ts en la carpeta page.

## Controladores (.ts)

Un controlador básico está organizado de la siguiente manera:

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

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

Import se utiliza para importar módulos que contienen librerías y clases para poder utilizarlas en nuestro proyecto. Podemos importar módulos propios de Ionic que ya se incorporan al crear un proyecto, librerías de AngularJS, librerías de terceros que podemos instalar o nuestras propias librerías.

En este caso se importa el elemento Component de Angular y el elemento NavController de ionic-angular.

Las páginas son componentes de Angular, por eso necesitamos importar Component.

Después vemos el siguiente código:

```
@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
```

**@Component** es un decorador de Angular. Angular 2 usa los decoradores para registrar un componente.

Los decoradores empiezan con @.

Los decoradores se colocan antes de la clase y definen cómo se va a comportar esta.

En el código vemos que **@Component** contiene un objeto con dos atributos, **selector:'page-home'**, que es el selector css que se va a aplicar a la página, y **templateUrl:'home.html'** que es la plantilla html que va a renderizar la página.

Por último se exporta una clase llamada HomePage:

```
export class HomePage {

  constructor(public navCtrl: NavController) {

  }

}
```

Tenemos que exportar la clase para luego poderla importar si queremos llamar a la página desde cualquier otro sitio de la aplicación, por ejemplo cuando navegamos entre páginas.

Vemos que la clase tiene un constructor, en TypeScript la creación de clases es muy similar a como sería en otros lenguajes de programación orientado a objetos como Java.

El constructor será la función que se ejecuta al instanciar la clase.

Todos los datos que se muestren en la plantilla y todas las funciones a las que se llame por ejemplo al hacer click en un botón deben formar parte de la clase HomePage que es el controlador de la página.

## Binding y Doble Binding (Angular)

El binding no es más que enlazar la información que tenemos en el controlador con lo que mostramos en el HTML. Esto se produce en dos sentidos:

**One-way binding:**

En este caso la información solamente fluye desde el controlador hacia la parte visual, o sea, desde el modelo hacia la representación de la información en el HTML. Lo conseguimos con la sintaxis "Mustache" de las dos llaves.

```
<p>El número secreto es {{ mayorMenor }}</p>
```

Ese dato estarías trayéndolo desde el scope y mostrándolo en la página. La información fluye desde el scope hacia la representación quiere decir que, si por lo que sea se actualiza el dato que hay almacenado en el modelo (scope) se actualizará automáticamente en la presentación (página).

**Two-way binding:**

En este segundo caso la información fluye desde el controlador hacia la parte visual (igual que en "one-way binding") y también desde la parte visual hacia el controlador. La implementas por medio de la directiva **[(ngModel)]**.

```
 <ion-input type="number" min="1" max="100" [(ngModel)]="num"
placeholder="Introduce un número del 1 al 100"></ion-input>
```

# Definir variables en TypeScript

Para definir una variable ponemos el nombre de la variable y seguido de dos puntos ":" el tipo de valor que va a contener. Si lo deseamos podemos inicializar la variable con un valor en el momento de declararla.

Ejemplo:

```
edad:number=1;
```

Los tipos primitivos de variable que podemos definir son:

- number (Numérico).
- string (cadenas de texto).
- boolean (Booleano: true o false).
- any (cualquier tipo).
- Array.

# Mini juego para acertar números

1. Crear un nuevo proyecto llamado “adivina”. Utilizar **ionic start con template blank**
2. Correr la nueva aplicación ejecutando **ionic serve**
3. Editar contenido de **home.html,** pegar el siguiente código:

```
4.   <ion-header>
5.     <ion-navbar>
6.       <ion-title>
7.         Adivina el número secreto
8.       </ion-title>
9.     </ion-navbar>
10.  </ion-header>
11.  <ion-content padding>
12.      <ion-input type="number" min="1" max="100" [(ngModel)]="num"
   placeholder="Introduce un número del 1 al 100"></ion-input>
13.      <p>El número secreto es {{ mayorMenor }}</p>
14.     <button ion-button block (click)="compruebaNumero()">Adivina</button>
15.  </ion-content>
```

- **Ion-input**: Es un componente de IONIC que veremos en las siguientes unidades, por ahora solo nos alcanza con saber que es un input con algunas funcionalidades extras. Como se ve tiene colocado **[(ngModel)]=”num”** lo cual nos indica que tendrá un doble binding con la variable **num** que estará definida en el controlador.
- **{{mayorMenor}}:** Indica un binding simple, el cual fluye de lo que tenga como contenido la variable **mayorMenor** en el controlador hacia la vista HTML. Ese contenido será visualizado por pantalla.
- **Button:** En posteriores unidades veremos de donde podemos obtener mas información acerca de componentes de IONIC, por el momento solo nos

alcanza con saber que este botón tiene definido que al hacerse clic sobre el mismo se ejecute la función **compruebaNumero** que está definida en el controlador.

4. Definir las variables **num** y **mayorMenor** dentro del controlador:

```
export class HomePage {

  num:number;
  mayorMenor: string = '...';

  constructor(public navCtrl: NavController) {

  }

}
```

5. También vamos a necesitar otra variable que contenga el número secreto que debemos adivinar, le vamos a llamar numSecret y como valor le vamos a asignar la respuesta a la llamada a una función llamada numAleatorio que crearemos a continuación.

```
numSecret: number = this.numAleatorio(0,100);
```

Hacemos referencia a la función **numAleatorio** con **this** porque va a ser un método de la propia clase.

6. Ahora vamos a crear la función **numAleatorio** que nos devolverá un número aleatorio.

Esta función recibe como parámetros dos valores que serán el rango mínimo y el máximo para el número aleatorio, es este caso le estamos pasando 0 y 100 por lo que obtendremos un número aleatorio entre 0 y 100.

```
7.   numAleatorio(a,b)

8.   {
```

```
9.          return Math.round(Math.random()*(b-a)+parseInt(a));

10.   }
```

7. Por último vamos a crear la función **compruebaNumero** que se llama al pulsar el botón Adivina que en la parte html lo hemos definido así:

```
compruebaNumero(){
    if(this.num)
    {
      if(this.numSecret < this.num)
      {
        this.mayorMenor = 'menor';
      }
      else if(this.numSecret > this.num)
      {
        this.mayorMenor = 'mayor';
      }
      else{
        this.mayorMenor = 'igual';
      }
    }
  }
```

Esta función compara el número secreto que esta contenido en la variable **numSecret** con el número introducido por el usuario que se aloja en la variable **num** y le asigna a la variable **mayorMenor** el texto “menor”, “mayor” o “igual” en función de si es menor, mayor o igual que esta.

Observa que debemos utilizar **this** para las variables ya que son atributos de la propia clase.

8. El código completo de home.ts quedará de esta manera:

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

import { NavController } from 'ionic-angular';

@Component({
  selector: 'page-home',
  templateUrl: 'home.html'
})
export class HomePage {

  num:number;
  mayorMenor: string = '...';
  numSecret: number = this.numAleatorio(0,100);

  constructor(public navCtrl: NavController) {

  }

  compruebaNumero(){
    if(this.num)
    {
      if(this.numSecret < this.num)
      {
        this.mayorMenor = 'menor';
      }
      else if(this.numSecret > this.num)
      {
        this.mayorMenor = 'mayor';
      }
      else{
        this.mayorMenor = 'igual';
      }
    }
  }

  numAleatorio(a,b)
  {
        return Math.round(Math.random()*(b-a)+parseInt(a));
  }

}
```

Ahora si lo ejecutamos en el navegador con ionic serve (si no lo tenias ya en marcha), podemos ver un campo para introducir un número, y debajo un texto que dice “el número secreto es … de”, en estos momentos  no hay introducido ningún número por lo que no aparece ningún número en el texto.

Si introducimos un número vemos que automáticamente en el texto aparece ese número el mismo tiempo que lo escribimos. Eso es porque estamos realizando un Data Binding con la variable num. Al utilizar [(ngModel)]=”num” en el campo input le estamos indicando que la variable coja el valor que introducimos en el input y lo refresque automáticamente en todos los sitios donde se utilice, por ejemplo en el texto de debajo. Lo mismo ocurre si cambiamos el valor de la variable desde el código.

Al pulsar el botón se ha ejecutado la función compruebaNumero(), dentro de la función se ha comprobado que el número secreto es mayor a 12 por lo que a la variable mayorMenor se le asigna el texto “mayor”.

Cuando acertemos el número secreto el texto dirá que el número secreto es igual al número introducido. Esto es un poco soso, vamos a hacer algunos cambios para que cuando acertemos el número nos muestre un mensaje felicitándonos por haber acertado.

Este mensaje lo ponemos en la plantilla html pero solo se debe mostrar cuando se cumpla una condición y es que la variable mayorMenor contenga el texto ‘igual’.

## Ejercicio

A nuestro juego de adivinar el numero debemos agregarle un contador de intentos, de esta forma el jugar tendrá hasta 5 intentos para adivinar el numero.

# Bibliografía utilizada y sugerida

***https://en.wikipedia.org/wiki/Ionic_(mobile_app_framework)***

***http://ionicframework.com/getting-started/***

***https://reviblog.net/2017/02/16/mini-juego-de-acertar-numeros-en-ionic-2-el-controlador-de-la-pagina-y-data-binding/***

# Lo que vimos:

En esta unidad hemos dejado IONIC listo para compilar.

Para ello tuvimos que instalar Java JDK, Node JS, Android SDK y Apache Cordova.

## Lo que viene:

En la siguiente unidad veremos cómo utilizar IONIC creator para crear nuestra primera aplicación de manera sencilla.

Mediante IONIC creator podremos generar una interfaz de usuario amigable en pocos pasos para luego aplicarle su funcionalidad.