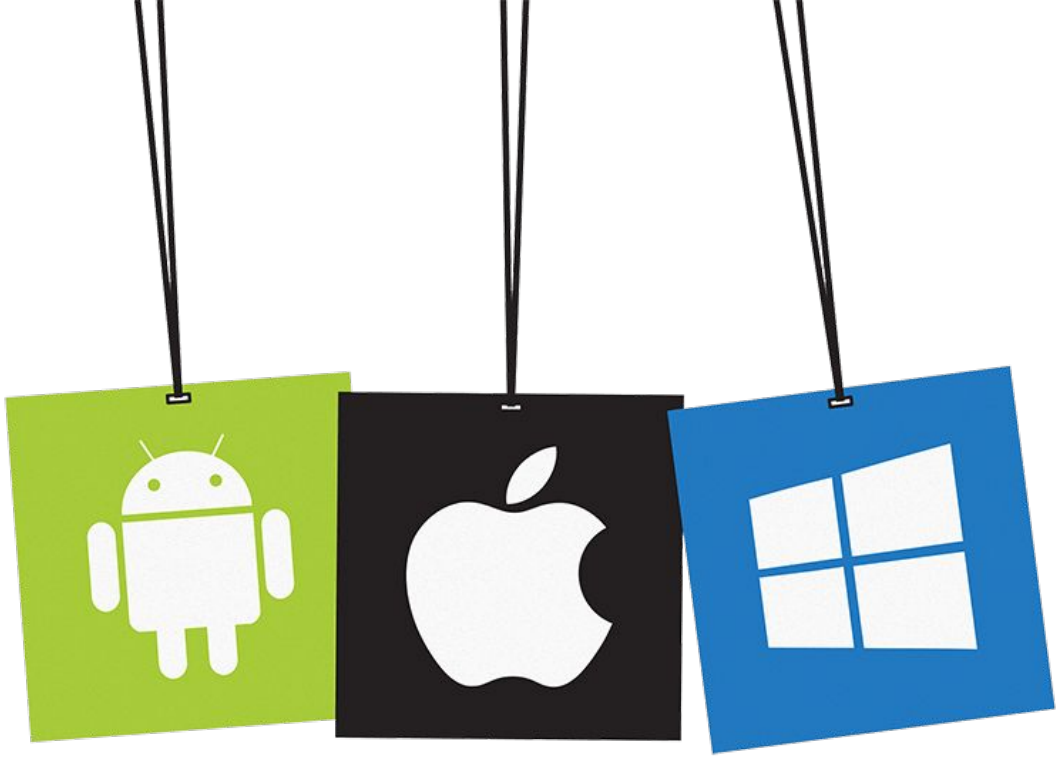
Enfoques de Desarrollo de Aplicaciones Móviles Multiplataforma

Enfoque Híbrido con Ionic

Clase 1

Esp. Juan Fernández Sosa



Java **Objective-C** **C#**

Kotlin **Swift**

Ionic Framework is an open source UI toolkit for building performant, high-quality mobile and desktop apps using web technologies — HTML, CSS, and JavaScript — with integrations for popular frameworks like Angular, React and Vue.

https://ionicframework.com/docs

Utilizando Ionic + Angular + Cordova

Ionic Framework

Provee de componentes UI como modals, alerts, estilos, controles de navegación, listas, etc. (similar a Bootstrap)

Se encarga de dar un *look and feel* similar a una app nativa

Angular

Motor de la aplicación.

Interactúa con servicios, peticiones HTTPs, procesamiento de datos, etc.

Utiliza directivas, pipes, HTTPs, data binding, servicios, módulos, etc.

Desarrollado en TypeScript.

Sigue el patrón MVC

Cordova Framework / CapacitorJS

Permite compilar el proyecto para distintas plataformas.

Permite acceder a características del dispositivo móvil mediante el uso de plugins (cámara, giroscopio, GPS, calendario, archivos, etc)

Guía de instalación

Instalamos el entorno para poder desarrollar aplicaciones multiplataforma

Qué necesitamos?

- Node, npm → los descargamos desde https://nodejs.org/en/
  - Capacitor necesita una versión de NodeJS >=20
- Capacitor, Ionic
  - npm install -g @ionic/cli
  - npm install @capacitor/core @capacitor/cli
- SDKs necesarios para desarrollar aplicaciones para Android e iOS → leer guía en
  - Descargar Android Studio
  - npm i @capacitor/android
- Editor de textos (Visual Studio Code, Atom, WebStorm, ALM, etc)

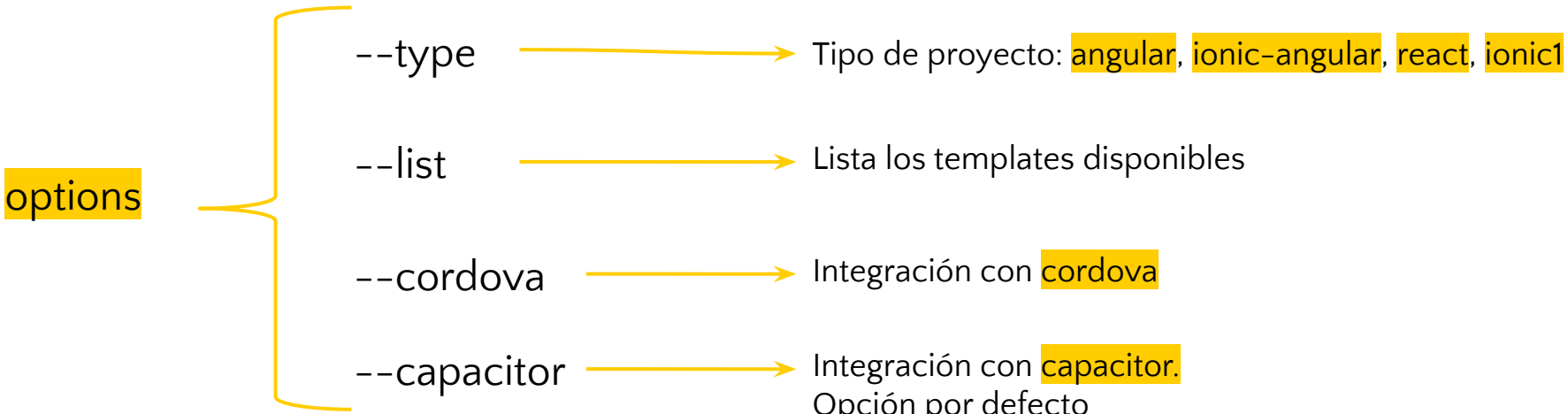
Primer App en Ionic

Vamos a crear una aplicación usando Ionic

Crear nuestra primer app

https://ionicframework.com/docs/cli/commands/start

$ ionic start <name> <template> [options]



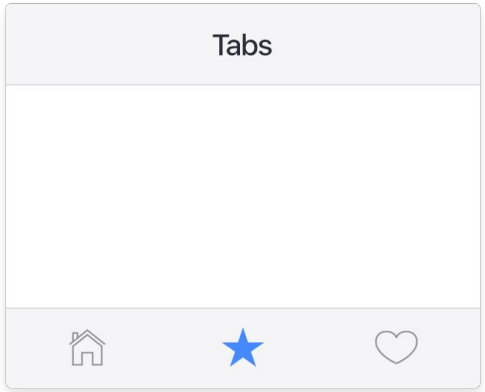
Crear nuestra primer app

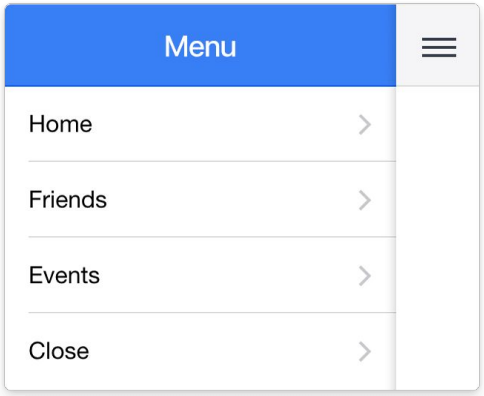
$ ionic start <name> <template> [options]

$ ionic start --list

```
name       | project type    | description
-----------------------------------------------------------------------------------------------------------------
_
blank      | angular         | A blank starter project
sidemenu   | angular         | A starting project with a side menu with navigation in the content area
tabs       | angular         | A starting project with a simple tabbed interface
tabs       | ionic-angular   | A starting project with a simple tabbed interface
blank      | ionic-angular   | A blank starter project
sidemenu   | ionic-angular   | A starting project with a side menu with navigation in the content area
super      | ionic-angular   | A starting project complete with pre-built pages, providers and best practices for Ionic development
.
tutorial   | ionic-angular   | A tutorial based project that goes along with the Ionic documentation
aws        | ionic-angular   | AWS Mobile Hub Starter
tabs       | ionic1          | A starting project for Ionic using a simple tabbed interface
blank      | ionic1          | A blank starter project for Ionic
sidemenu   | ionic1          | A starting project for Ionic using a side menu with navigation in the content area
maps       | ionic1          | An_Ionic starter project using Google Maps and a side menu
```

Crear nuestra primer app





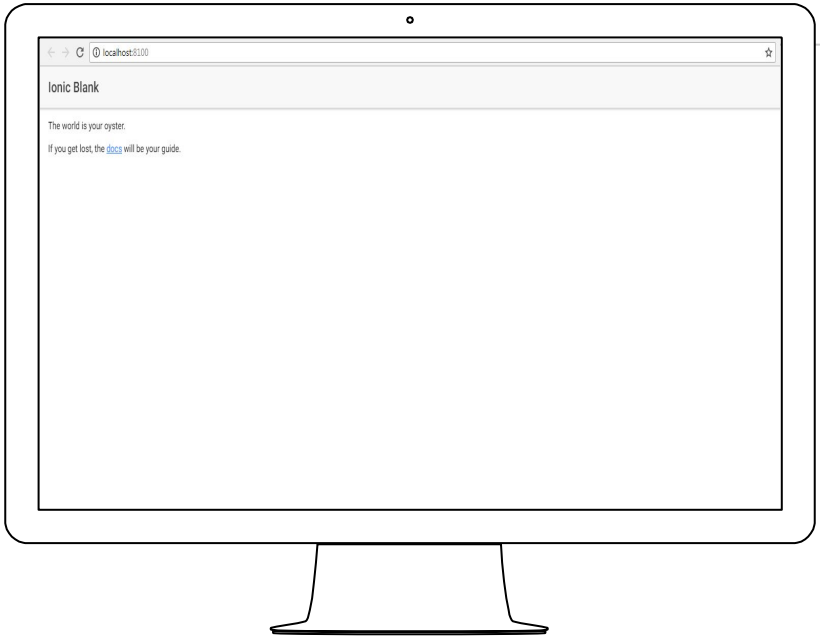
$ ionic start myApp blank

$ ionic start myApp tabs

$ ionic start myApp sidemenu

$ ionic serve

```
$ cd ./myApp
$ ionic serve
```



Inicia un servidor de desarrollo para probar la app. Se ejecuta en el browser.

Los cambios en el código fuente son actualizados automáticamente.

$ ionic serve

Al ejecutar ionic serve, el código dentro de “/src” es **transpilado** en la correcta versión de JavaScript que el browser entiende.

Se puede trabajar en un nivel más alto usando TypeScript

$ ionic serve

Ionic serve **NO** ejecuta ni Cordova ni Capacitor, porque la compilación se hace para un browser.

↓

No se pueden utilizar los plugins!!!

↓

Cordova/Capacitor entran en acción cuando se hace la compilación para alguna de las plataformas (Android o iOS)

Estructura de nuestro proyecto

./myApp

myApp

/node_modules → Dependencias a ser instaladas en nuestro proyecto

/src → Código fuente de la App (Donde escribimos nuestro código)

capacitor.config.ts → Archivo de configuración para Capacitor

angular.json → Configuración del proyecto Angular

/android → Código específico para cada plataforma

/ios → Código específico para cada plataforma

/www → Bundle compilado

Se generan cuando vamos Trabajando en el proyecto

Estructura de nuestro proyecto

./myApp/src

myApp

/app → Código de la App (módulos, componentes y servicios)

/pages → Páginas o “vistas” de la app. Similar al concepto Activity

/assets → Imágenes y recursos estáticos

/theme → Estilos generales de diseño/ variable sass

index.html → Punto de entrada de la App. Primer archivo en ser cargado por el navegador

main.ts → se carga Angular al proyecto.

myApp/src/app

app.component.ts

Define al componente root de la app.

Es el primer componente que es cargado en la app

app.component.html

Es el template principal de la aplicación.

Los estilos se definen en **app.component.scss**

app.module.ts

Punto de entrada de la aplicación.

Contiene al *root Module* que controla el resto de la app.

Se definen las páginas, servicios, plugins a usar por la App.

app-routing.module.ts

Información de ruteo de la app (ruteo de Angular)

myApp/src/app/page

Las páginas las creamos desde el CLI de ionic con el comando ionic g page pages/myPage

myPage/myPage.page.ts

Lógica/comportamiento de la página

myPage/myPage.page.html

HTML de la página

myPage/myPage.page.scss

Se definen los estilos para esa página

myPage/myPage.page.module.ts

Configuración del componente/pagina.

myPage/myPage.page.spect.ts

Test en nuestra página

Cada vez que se crea una página, se agrega automáticamente al archivo de ruteo.

src/app/home

- Ingresamos a http://bit.ly/enfoques-A0 copiamos el contenido en *home.pages.html*
- Levantamos el servidor de desarrollo -> ionic serve

Desde el browser accedemos al Inspector y apretamos

Con eso vemos la versión “mobile”

Data binding Angular

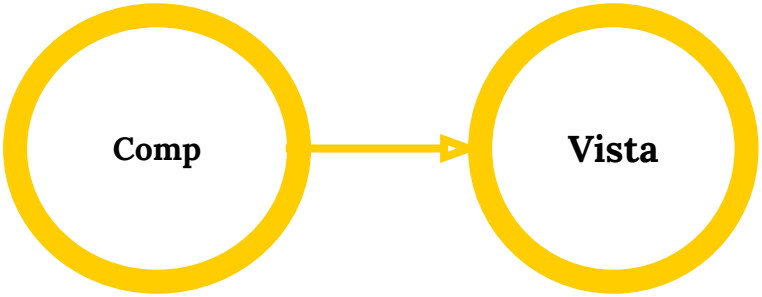
Data Binding o enlace de datos, es la sincronización automática de datos entre el modelo y un componente de la vista.
Cuando el modelo cambia, las vistas reflejan ese cambio y viceversa.

Tipos de Data binding en Angular: Interpolation

Permite enlazar el valor de un dato del componente a la vista. Cuando este valor cambia, cambia también en la vista.

Se utilizan {{ }}

Enlace de datos en una dirección



../pages/image/image.html

```
<ion-content>
    <p>{{mensaje}}</p>
</ion-content>
```

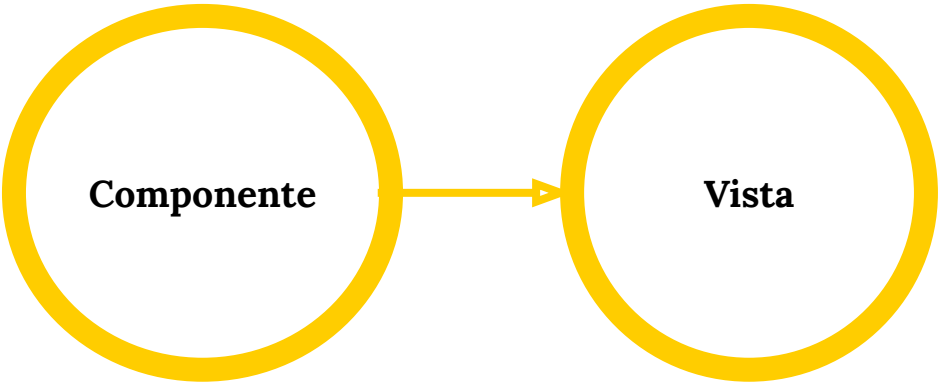
../pages/image/image.ts

```
export class ImagePage {
mensaje:string=”Hola Mundo!”
....
}
```

Tipos de Data binding en Angular: Property binding

Modifica el valor una propiedad de un elemento de la vista desde el modelo.

Enlace de datos en una dirección



../pages/image/image.html

```
<ion-content>
    <img src={{logo_path}} />
    <img [src]=”logo_path”/>
</ion-content>
```

../pages/image/image.ts

```
export class ImagePage {
logo_path:string=”direccion_de_ima
gen.png”
....
}
```

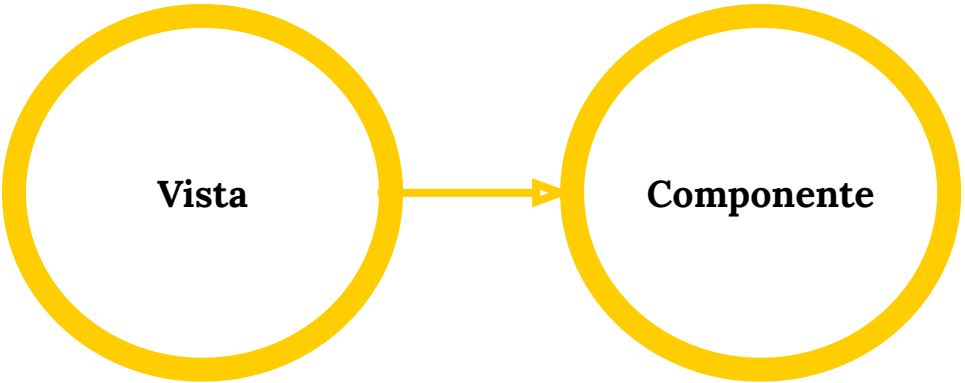
Actividad 1

- Modificar nuestra aplicación de prueba para que la información que muestra en la vista sea enviada desde el componente a la misma.

Tipos de Data binding: Event binding

Captura los eventos iniciados por el usuario desde la vista y los envía a la lógica del componente

Enlace de datos en una dirección



../pages/image/image.html

```
<ion-content>
  <button(click)=”doSomething()”>
  </button>
</ion-content>
```

../pages/image/image.ts

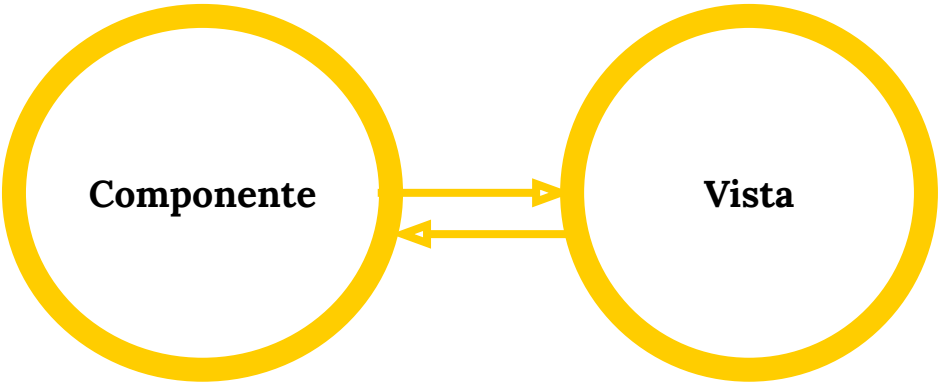
```
export class ImagePage {
...
  doSomething(){
    alert(“hello world”);
  }
}
```

Actividad 2

- Modificar la aplicación para que cuando el usuario haga *click* en alguna de las imágenes, se muestre, en consola (*console.log()*) o por medio de un *alert()*, un mensaje.

Tipos de Data binding: Two-way binding

Es una combinación de las dos anteriores. Sincronización continua entre la vista y el modelo/componente.



../pages/image/image.html

```
<ion-content>
     <input [(ngModel)]=”nombre”
/>
     <button type=”submit”
(click)=”submit()”/>
</ion-content>
```

../pages/image/image.ts

```
export class ImagePage {
nombre:string=”juan”
...
submit(){
alert(this.nombre);
}}
```

Actividad 3

- Vamos a agregar en nuestra app un input y a medida que el usuario ingresa un valor, se debe mostrar en la pantalla.

Actividad 4.

- Vamos a cambiar las *alerts* y los *inputs* usados en las actividades anteriores por aquellos que nos provee ionic.
- Investigarlos desde https://ionicframework.com/docs/components/

Compilando nuestra app para Android

Agregando plataformas al proyecto

Todas las plataformas que queramos agregar a nuestro proyecto se agregan desde línea de comandos

```
$ npm install @capacitor/android @capacitor/ios
```

```
$ npx cap add android
```

```
$ npx cap add ios
```

Android project algunos comandos

Para compilar nuestra app para Android, ejecutamos el siguiente comando

$ npx cap **build android**

Al finalizar mostrará la ubicación del ejecutable apk que podremos instalar en el dispositivo móvil

Para compilar y ejecutar la aplicación en el dispositivo móvil que tengamos conectado (o en el emulador)

$ npx cap **run android**