1. **Creando y haciendo deploy del proyecto (10 mins)**

   Para crear nuestro proyecto ejecutamos el siguiente comando en nuestra consola (GitBash si estás en Windows)

   ```
   ng new HandsOnLab
   ```

   Ahora para poder hacer deploy del proyecto que acabamos de crear es necesario estar dentro del directorio del proyecto que acabamos de crear, para ello ejecutamos el siguiente comando:

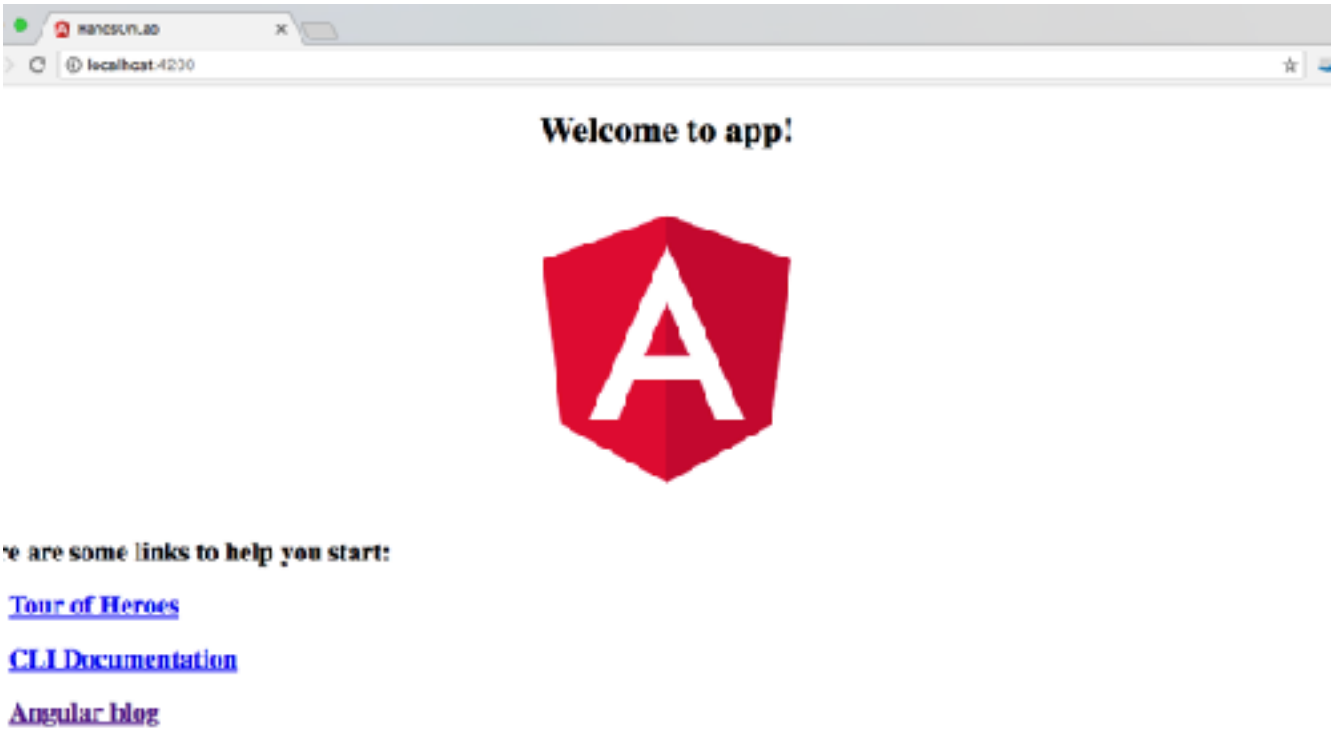
   ```
   cd HandsOnLab/
   ```

   Ya que estamos dentro ejecutamos el siguiente comando para realizar el deploy:

   ```
   ng serve
   ```

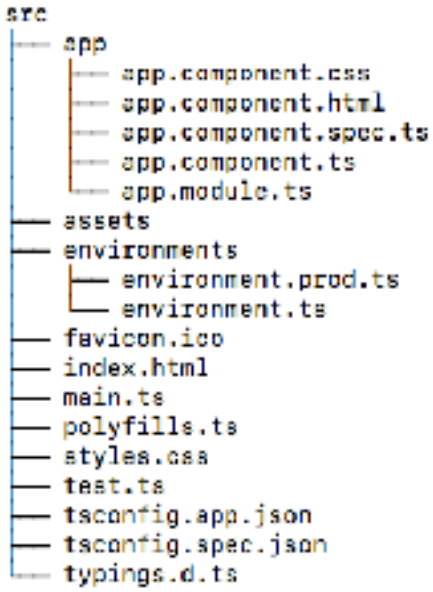
   Para finalizar mandamos a llamar a la aplicación desde el navegador:

   http://localhost:4200/



2. **Editando nuestro primer componente (10 mins)**

   Para empezar entendamos un poco la estructura de un proyecto de Angular. La aplicación vive en el directorio `src/`. Es aquí donde se encuentran todos los componentes, templates, styles (css), imágenes, etc para que nuestra aplicación funcione. El directorio `src/` está estructurado de la siguiente manera:



El primer lugar que visitaremos dentro del directorio `src/` será el directorio `app/`. Para ponernos en contexto debemos recordar que Angular es modular, lo que significa que se basa en ***NgModules***. Toda aplicación Angular tiene al menos un NgModule, el Módulo Raíz, convencionalmente llamado *AppModule*.

El *AppModule* se encuentra alojado en el directorio `app/`, dentro encontraremos los archivos:
• ***app.component.ts***: Define el AppComponent como component.
• ***app.component.html***: Define el template HTML de AppComponent.
• ***app.component.css***: Define los css styles del AppComponent.
• ***app.component.spec.ts***: Archivo que define la prueba unitaria.
• ***app.module.ts***: Define el NgModule raíz. Recordar que cualquier NgModule, incluyendo el raíz, pueden tener más de un componente.

Empecemos por darle un vistazo al archivo *app.module.ts*, el cual define el NgModule Raíz.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Lo primero que vemos es el import del *BrowserModule* y del *NgModule (core)*, estos son imports esenciales para poder acceder a la metadata y correr nuestra aplicación en el Browser. Lo siguiente que vemos es el import del *AppComponent* que tenemos definido en nuestro módulo.

A continuación tenemos el ***@NgModule***, en terminología Angular es un decorator function, que es un objeto de metadatos cuyas propiedades describen el módulo (AppModule, el Módulo Raíz). Sin este decorator function para el framework *app.module.ts* solamente será una clase de TypeScript, con el establecemos la metadata del módulo e indicamos al framework que esta clase es un módulo.
• ***declarations***: las *view classes* en el módulo. En este caso solo tenemos el componente AppComponent.
• ***imports***: otros módulos que necesitamos exportar para ser utilizados en el nuestro. En este caso estamos importando el BrowserModule.
• ***providers***: definición de los servicios del módulo:
• ***bootstrap***: La vista principal de la aplicación, que será llamada el componente raíz. Esta propiedad es exclusiva del Módulo Raíz.

Ahora que hemos analizado la definición de nuestro Módulo Raíz pasemos a ver la definición de nuestro AppComponent. Este está compuesto por cuatro archivos, el componente, el template html, el archivo de estilos css y la prueba unitaria. Nosotros nos enfocaremos únicamente el los tres primeros, en otra ocasión hablaremos sobre las pruebas unitarias.

Empecemos viendo el archivo *app.component.ts*, el cual define el componente:

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'app';
}
```

Lo primero es importar el componente *@angular/core* para poder utilizar el decorator function ***@Component*** y establecer la metadata del componente e indicar al framework que es un componente. Los elementos que podemos ver en ***@Component*** son:
• ***selector***: indica el nombre del tag HTML en el cual será embebido el componente. En este caso será en el <app-root></app-root>
• ***templateUrl***: indicamos el template HTML que está a cargo de este componente.
• ***styleUrls***: indicamos el archivo de estilos css.
• ***providers***: declaramos los Services que utilizará nuestro componente.

Para finalizar debemos definir la clase AppComponente, lo primero que notarán es que empieza con la palabra *export*, esto es esencial si deseamos importar el componente en otros lugares de nuestra aplicación, luego la palabra reservada *class* que indica que es una Clase, y finalmente el nombre AppComponente.

Definimos una propiedad llamada *title* que es inicializada, dado que TypeScript infiere el tipo de dato no es necesario indicarlo.

Ahora podemos pasar y explorar el archivo *app.component.html*, el template del componente:

```
<!--The content below is only a placeholder and can be replaced.-->
<div style="text-align:center">
  <h1>
    Welcome to {{ title }}!
  </h1>
  <img width="300" alt="Angular Logo" src="
9IjAgMCAyNTAgMjUwIj4KICAgIDxwYXRoIGZpbGw9IiNERDAwMzEiIGQ9Ik0xMjUgMzBMMzEuOSA2M
y4ybDE0LjIgMTIzLjFMMTI1IDIzMGw3OC45LTQzLjcgMTQuMi0xMjMuMXoiIC8+CiAgICA8cGF0aCB
maWxsPSIjQzMwMDJGIiBkPSJNMTI1IDMwdjIyLjItLjFWMjMwbDc4LjktNDMuNyAxNC4yLTEyMy4xT
DEyNSAzMHoiIC8+CiAgICA8cGF0aCAgZmlsbD0iI0ZGRkZGRiIgZD0iTTEyNSA1Mi4xTDY2LjggMTg
yLjZoMjEuN2wxMS43LTI5LjJoNDkuNGwxMS43IDI5LjJIMTgzTDEyNSA1Mi4xem0xNyA4My4zaC0zN
GwxNy00MC45IDE3IDQwLjl6IiAvPgogIDwvc3ZnPg==">
</div>
<h2>Here are some links to help you start: </h2>
<ul>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://angular.io/
tutorial">Tour of Heroes</a></h2>
  </li>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://github.com/angular/
angular-cli/wiki">CLI Documentation</a></h2>
  </li>
  <li>
    <h2><a target="_blank" rel="noopener" href="https://
blog.angular.io/">Angular blog</a></h2>
  </li>
</ul>
```

Este es el HTML que manejará el componente, y que será embebido en el tag <app-root></app-root> que especificamos en la metadata *selector* del componente. Como podrán ver no hay nada fuera de lo normal a la sintaxis HTML, con excepción de `{{ title }}!`, con esta sintaxis estamos indicando que aquí será seteado el valor de la propiedad *title* definida en el componente.

Para finalizar veremos el archivo *app.component.css*

```

```

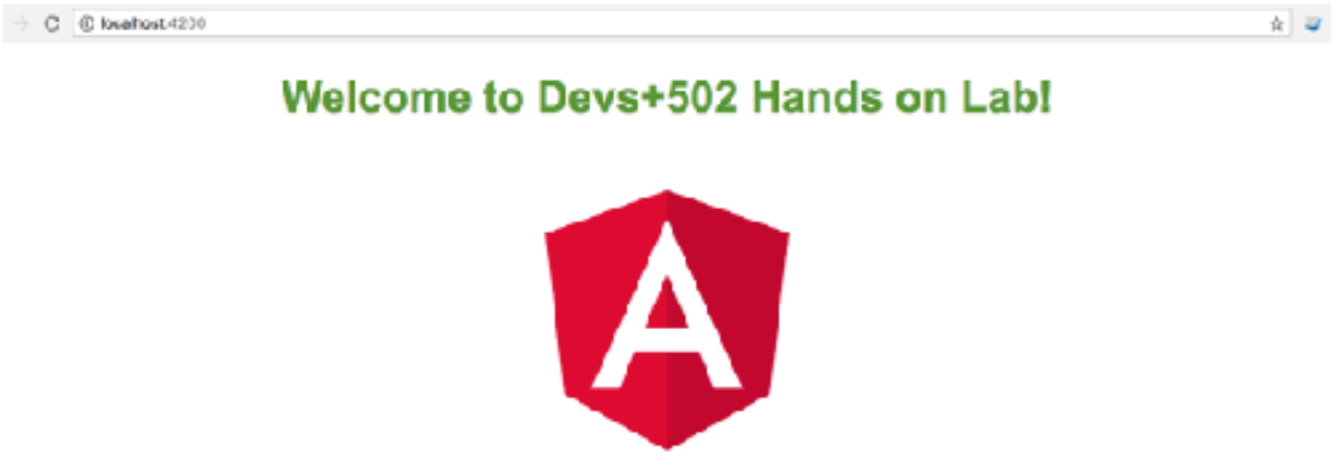
No, esto no es un error, el archivo está vacío. Se debe a que al principio no hemos puesto ningún estilo a nuestro proyecto. La primera parte de nuestro proyecto será agregar un poco de estilos y cambiar la información que desplegamos.

Ahora que ya comprendemos en profundidad como está estructurada esta sección, procederemos a hacer algunos cambios. El primero será en el componente, donde cambiaremos el valor de la propiedad *title*.

```
export class AppComponent {
  title = 'Devs+502 Hands on Lab';
}
```

Ahora pongamos algo de estilo, si recuerdas el mensaje "Welcome …" está dentro del tag <h1>, así que agregaremos algo de estilo para este tag en el archivo *app.component.css*

```
h1 {
    color: rgb(88, 153, 51);
    font-family: Arial, Helvetica, sans-serif;
    font-size: 250%;
}
```



Para finalizar con esta sección vamos a explorar otros dos archivos, *index.html* y *main.ts*, ambos se encuentran al mismo nivel que el directorio `app/`.

Empecemos por *index.html*, este es el archivo html padre, donde se embeberá el componente raíz (AppComponente)

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>HandsOnLab</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Es en este archivo donde se encuentra el tag <app-root></app-root> que especificamos en la metadata ***selector*** del componente. El siguiente archivo es el *main.ts*

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.log(err));
```

En este archivo es donde indicamos si nuestro ambiente será ejecutado en modo de producción o desarrollo.

3. Creando un nuevo Componente (5 mins)

Para crear un nuevo componente ejecutaremos el siguiente comando dentro del directorio del proyecto: ***selector***

```
ng generate component heroes
```

Esto generará una nuevo directorio `heroes/` dentro de `app/` con los archivos
• ***heroes.component.css***: archivo de estilos css del componente
• ***heroes.component.html***: template html del componente heroes
• ***heroes.component.spec.ts***: prueba unitaria para componente heroes
• ***heroes.component.ts***: componente heroes

Empecemos por explorar *heroes.component.ts*

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

@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css']
})
export class HeroesComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}
```

Podemos ver varios de los elementos que ya hemos definido en *app.component.ts*, pero ahora podemos ver que también se import ***OnInit*** y clase *HeroesComponent* lo implementa. En la estructura de la clase podemos ver un constructor definido y el método *ngOnInit()*, este último es un buen lugar para inicializar la lógica de nuestro componente.

Ahora veamos *heroes.component.html*, en el cual tenemos el mínimo necesario de html para desplegar un párrafo.

```
<p>
  heroes works!
</p>
```

Como podrán imaginar el archivo *heroes.component.css* se encuentra vacío.

El primer paso de esta misión es integrar el *HeroesComponent* a nuestro html principal. A modo de experimento intenta agregar el <app-heroes> tag al index.html al mismo nivel que <app-root>. Te darás cuenta que no aparece nuestra el texto **heroes works!**, esto se debe a que en el index.html solo se puede embeber el template del componente principal.

Para poder ver reflejado nuestro nuevo componente, lo vamos a agregar en el templete de app component, en el archivo *app.component.html*

```
<h1>
    Welcome to {{ title }}!
</h1>
<app-heroes></app-heroes>
```

El resultado de está inyección de código se verá de la siguiente forma:

Nuestra misión será definir alguna propiedad a nuestra clase *HeroesComponent*

```
hero = 'Codetron';
```

Para continuar mostremos el valor de hero en nuestro html

```
<p>
  heroes works!
  << {{ hero }} >>
</p>
```

El resultado se verá de la siguiente manera:

4. **Manejando Objetos (5 mins)**

Para agregar más complejidad a nuestra aplicación vamos a trabajar con objetos. Para ello es

necesario que definamos la clase Hero. Para ello crearemos el archivo Hero.ts en el directorio app/, en ella definiremos dos propiedades (id, name). El resultado final se vería de la siguiente manera:

```
export class Hero {
    id: number;
    name: String;
}
```

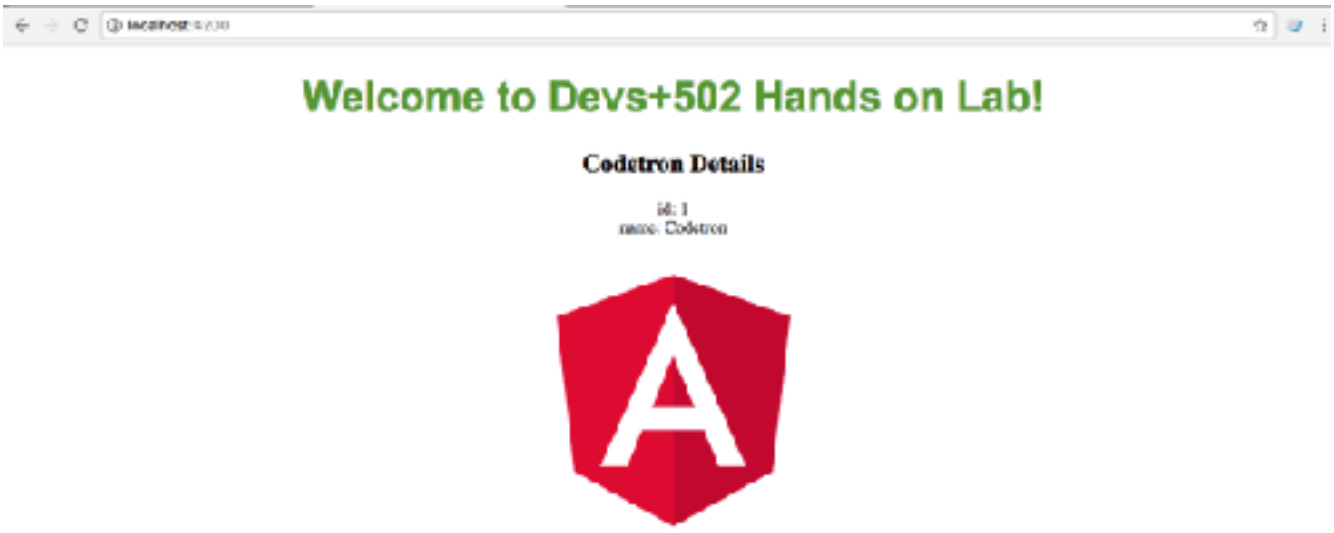
Para utilizar nuestra nueva clase ***Hero*** regresaremos a *heroes.component.ts*, y modificaremos la propiedad *hero* que es un ***String***, y utilizaremos nuestro objeto Hero.

```
//importamos Hero class
import {Hero} from '../Hero'

...

  hero: Hero = {
    id: 1,
    name: 'Codetron'
  };

...
```

Ahora modificamos *heroes.component.html* para desplegar la nueva información:

```
<h2>{{ hero.name }} Details</h2>
<div><span>id: </span>{{hero.id}}</div>
<div><span>name: </span>{{hero.name}}</div>
```

Nuestro resultado se verá así:



5. Editando, Binding de Doble Vía (mins)

Lo que haremos ahora será crear un formulario super sencillo, en el cual modificaremos el nombre de nuestro Héroe. Para ello, necesitamos importar *FormsModule*, agregaremos el siguiente import en nuestro archivo *app.module.ts*.

```
import { FormsModule } from '@angular/forms';
```

Agregar a la metadata el import de *FormsModule*

```
imports: [
    BrowserModule,
    FormsModule
  ],
```

Ahora agregaremos un pequeño input que nos servirá para poder modificar el valor de name de nuestro Héroe en *heroes.component.html*.

```
<div>
    <label>name:
      <input [(ngModel)]="hero.name" placeholder="name">
    </label>
</div>
```