Práctica Angular – Parte 1

Películas

Creación del proyecto

Abrimos un terminal y desde nuestra carpeta de proyectos Angular ejecutamos lo siguiente para crear la estructura base del proyecto:

```
ng new movies
```

Nos preguntará si queremos sistema de enrutado y de qué manera queremos trabajar con CSS. Respondemos a la primera cuestión que no y para la segunda indicamos CSS (ambas son las opciones por defecto).

Mientras Angular CLI crea el proyecto podemos descomprimir el archivo de recursos que se adjunta (recursos.zip). Este archivo contiene lo siguiente:

- **img**: carpeta que contiene las portadas de las películas
- **animate.css**: Fichero de hoja de estilos CSS. Contiene animaciones CSS3 que nos van a ayudar a que la aplicación proporcione una mejor experiencia de usuario.ng v
- **datos.txt**: Fichero que contiene un array de objetos con información de películas. Parece formato JSON pero no lo es. Más adelante utilizaremos este fichero en la aplicación para mostrar información sobre las películas.
- **favicon.ico**: Fichero con el icono del proyecto.
- **A-64.png**: Icono para mostrar en el *navbar*.

Copia de las imágenes

- Vamos a copiar las portadas de las películas. Sencillamente arrastramos la carpeta `img` que contiene los ficheros JPG a la carpeta `src → assets` del proyecto.
- También tenemos que copiar el fichero `A-64.png` a la carpeta `src → assets`.
- En fichero `favicon.ico` lo copiamos en la raíz de la carpeta `src`.

Instalación de Bootstrap

Hay varias formas de instalar Bootstrap, una de ella es hacerlo mediante `npm`. Por otro lado, hay que tener en cuenta que Bootstrap tiene otras dependencias, por lo que también hay que instalarlas:

```
npm install jquery
```

```
npm install popper.js
```

```
npm install bootstrap
```

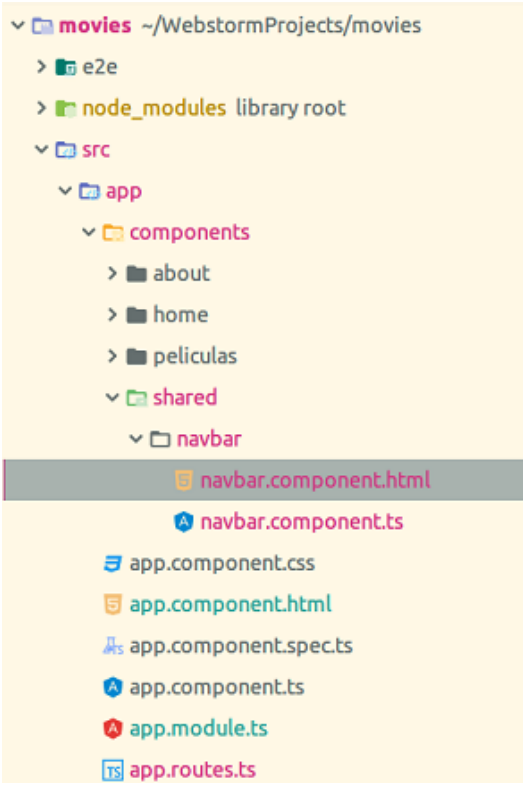
Una vez instalado lo anterior, tendremos las librerías correspondientes en la carpeta `node_modules`.

Por último, tenemos que hacer cambios en el fichero `angular.json`. Lo dejamos igual que en la siguiente figura:

```
"styles": [
  "src/styles.css",
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": [
  "node_modules/jquery/dist/jquery.slim.min.js",
  "node_modules/popper.js/dist/umd/popper.min.js",
  "node_modules/bootstrap/dist/js/bootstrap.min.js"
]
,
```

Estructura del proyecto

En esta primera parte de la práctica crearemos una estructura de carpetas y ficheros como se muestra en la siguiente figura:



En la carpeta `src → app` vamos a crear una carpeta llamada `components` y dentro de ésta crearemos otra llamada `shared`. En `shared` pondremos los componentes que serán usados de manera transversal, como es el caso del *navbar*, *header*, *footer*, *sidebar*, etc.

Creación del componente navbar

Comenzamos por crear la barra de navegación:

```
ng g c components/shared/navbar -is --spec=false
```

Explicación de los modificadores:

`-is` Indicamos que los estilos CSS serán *inline*, es decir que no queremos un fichero a parte para los estilos.

`--spec=false` Indicamos que no queremos fichero de pruebas unitarias para este componente.

Una vez finalizado el proceso anterior podremos comprobar que:

- Solamente ha generado 2 archivos: el de clase (.ts) y la plantilla (.html).
- Que el fichero `app.module.ts` declara este nuevo componente.

Si miramos el contenido del fichero de clase generado veremos que la etiqueta HTML asociada es `app-navbar`:

navbar.component.ts

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

@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styles: []
})
export class NavbarComponent implements OnInit {
  constructor() { }
  ngOnInit() { }
}
```

Por tanto, dado que queremos que el *navbar* aparezca en nuestra aplicación, tenemos que incluirlo en el componente raíz de la aplicación, esto es, el fichero `app.component.ts`. Para ello, borramos todo su contenido que viene por defecto y copiamos el siguiente:

app.component.ts

```
<app-navbar></app-navbar>
```

Contenido del *navbar*

Vamos a usar el *navbar* de Bootstrap. Borramos el contenido del fichero de plantilla que viene por defecto y pegamos el siguiente:

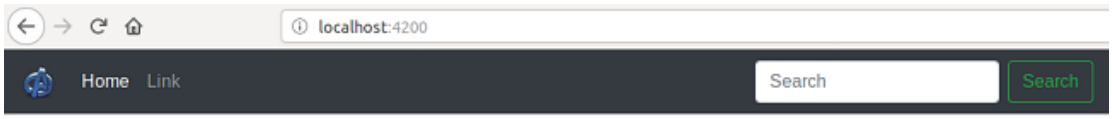
navbar.component.html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <a class="navbar-brand" href="#">
    <img src="assets/img/A-64.png" width="30" height="30" alt="">
  </a>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>
    </ul>
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">
```

```
      <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
    </form>
  </div>
</nav>
```

Levantamos el servidor: `ng serve`

Una vez que esté arrancado el servidor, accedemos a `localhost:4200`. Deberíamos ver lo siguiente:



Creación del componente home

La home no tendrá lógica de programación, sencillamente será la página predeterminada de la aplicación y se mostrará al usuario cuando el acceso se produzca en la URL `localhost:4200`. Sin embargo, también haremos más adelante que desde el *navbar* haya un enlace desde el que poder ir a la *home*.

Este componente lo creamos directamente en la carpeta src → app → components:

```
ng g c components/home -is --spec=false
```

Si miramos el contenido del fichero de clase generado veremos que la etiqueta HTML asociada es `app-home`:

home.component.ts

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styles: []
```

```
})
export class HomeComponent implements OnInit {
  constructor() { }
  ngOnInit() { }
}
```

Contenido del *home*

Vamos a usar el *jumbotron* de Bootstrap. Borramos el contenido del fichero de plantilla que viene por defecto y pegamos el siguiente:

home.component.html

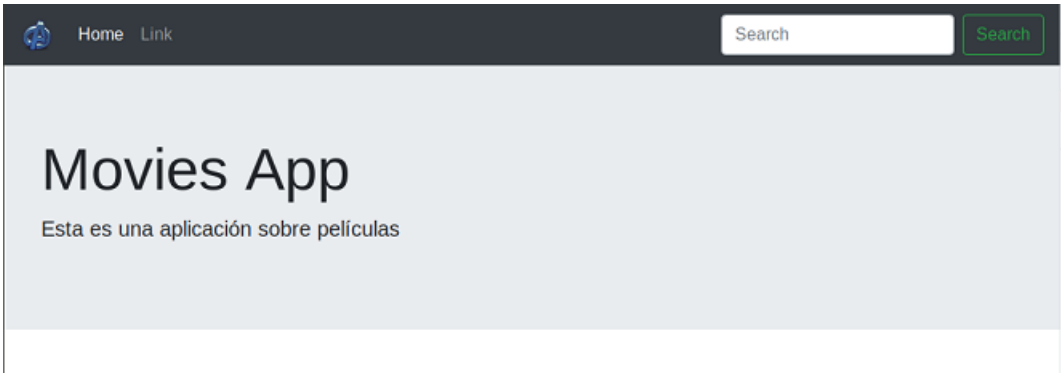
```
<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h1 class="display-4">Movies App</h1>
    <p class="lead">Esta es una aplicación sobre películas</p>
  </div>
</div>
```

Ahora, para ver cómo queda la página home, añadimos la etiqueta asociada a este componente al fichero `app.component.html`:

app.component.html

```
<app-navbar></app-navbar>
<app-home></app-home>
```

Si accedemos a `localhost:4200` deberíamos ver lo siguiente:



Creación del componente *about*

La página *about* no tendrá lógica de programación, simplemente proporcionará al usuario información sobre la aplicación.

Este componente lo creamos directamente en la carpeta src → app → components:

```
ng g c components/about -is --spec=false
```

Si miramos el contenido del fichero de clase generado veremos que la etiqueta HTML asociada es `app-about`:

about.component.ts

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

@Component({
  selector: 'app-about',
  templateUrl: './about.component.html',
  styles: []
})
export class AboutComponent implements OnInit {
  constructor() { }
  ngOnInit() { }
}
```

Contenido de *about*

Borramos el contenido del fichero de plantilla que viene por defecto y pegamos el siguiente:

about.component.html

```
<h1>Acerca de Movies App</h1>
<hr>
<p>Esta es una aplicación realizada con Angular 8 y Bootstrap cuyo propósito es conocer diversos aspectos del framework, principalmente enrutamiento y servicios.</p>
```

Haremos más adelante que desde el *navbar* haya un enlace desde el que poder ir a esta página.

Creación del componente *peliculas*

Este componente tendrá cierta lógica de programación. Su papel consistirá en mostrar al usuario la información sobre todas las películas de la aplicación.

Este componente lo creamos directamente en la carpeta src → app → components:

```
ng g c components/peliculas -is --spec=false
```

Si miramos el contenido del fichero de clase generado veremos que la etiqueta HTML asociada es `app-peliculas`:

peliculas.component.ts

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

@Component({
  selector: 'app-peliculas',
  templateUrl: './peliculas.component.html',
  styles: []
})
export class peliculasComponent implements OnInit {
  constructor() { }
  ngOnInit() { }
}
```

Contenido de *peliculas*

De momento dejamos el contenido que viene por defecto.

peliculas.component.html

```
<p>
  peliculas works!
</p>
```

Sistema de navegación

Las rutas nos permiten navegar a diferentes componentes o diferentes páginas sin tener que refrescar todo el contenido, sino solamente cargar aquella parte que nos interese dentro de una página.

Creación de rutas

Creamos un nuevo fichero llamado app.routes.ts en la raíz de `src → app`:

El contenido para el fichero es el que sigue:

app.routes.ts

```
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './components/home/home.component';
import { AboutComponent } from './components/about/about.component';
import { PeliculasComponent } from
'./components/peliculas/peliculas.component';

const APP_ROUTES: Routes = [
  { path: 'home', component: HomeComponent},
  { path: 'about', component: AboutComponent},
  { path: 'peliculas', component: PeliculasComponent},
  { path: '**', pathMatch: 'full', redirectTo: 'home' }
];
export const APP_ROUTING = RouterModule.forRoot(APP_ROUTES);
```

A continuación tenemos que indicar a Angular qué dispone de las rutas indicadas anteriormente. Esto se hace en el módulo raíz de la aplicación, `app.module.ts`. Dado que el fichero de rutas es una especie de módulo, tenemos que incluir una referencia a `APP_ROUTING` en la propiedad `imports`:

app.module.ts

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

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

// Rutas
import { APP_ROUTING } from './app.routes';

// Componentes
import { AppComponent } from './app.component';
import { NavbarComponent } from
'./components/shared/navbar/navbar.component';
import { HomeComponent } from './components/home/home.component';
import { AboutComponent } from './components/about/about.component';
import { PeliculasComponent } from
'./components/peliculas/peliculas.component';

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

Ahora tenemos que definir la etiqueta `<router-outlet>` en `app.component.html`, ya que de esta manera es como el sistema carga de forma dinámica las páginas. Por tanto, modificamos el fichero para que quede como sigue (hemos aprovechado de paso para incluir la clase `containter` de Bootstrap para que quede centrado:

app.component.html

```
<app-navbar></app-navbar>
<div class="container">
  <router-outlet></router-outlet>
</div>
```

Finalmente, tenemos que modificar la plantilla del *navbar* para indicar adecuadamente los títulos del menú y los enlaces. Los cambios realizados en el *navbar* respecto a la anterior versión son los siguientes:

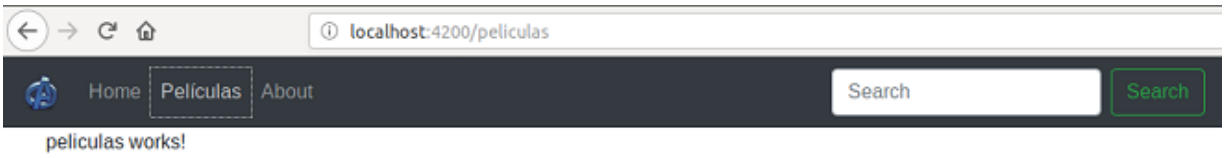
- Se ha eliminado del primer `<li>` la clase CSS de Bootstrap `active`, cuyo objetivo era mostrar de un color destacado la primera opción del menú del *navbar*. La eliminación de esta clase se ha debido a que ahora el propio Angular activará de forma dinámica la opción seleccionada. Esto es gracias a la directiva de atributo `routerLinkActive`.
- Se ha añadido un tercer `<li>` para reflejar las tres opciones que tenemos: home, películas y about.
- Por comodidad, se utilizan las directivas de atributo `routerLink`, en lugar del atributo estándar `href`. Entre corchetes va una cadena que ha de coincidir con alguna de las rutas definidas en el fichero `app.routes.ts`.

navbar.component.html

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>

  <a class="navbar-brand" href="#">
    <img src="assets/img/A-64.png" width="30" height="30" alt="">
  </a>

  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item" routerLinkActive="active">
        <a class="nav-link" [routerLink]="['home']">Home</a>
      </li>
      <li class="nav-item" routerLinkActive="active">
        <a class="nav-link" [routerLink]="['peliculas']">Películas</a>
      </li>
      <li class="nav-item" routerLinkActive="active">
        <a class="nav-link" [routerLink]="['about']">About</a>
      </li>
    </ul>
```

```
    <form class="form-inline my-2 my-lg-0">
      <input class="form-control mr-sm-2" type="search" placeholder="Search"
aria-label="Search">
      <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
    </form>
  </div>
</nav>
```

En estos momentos deberíamos poder navegar por las tres opciones del menú:



Una pequeña mejora estética para que el contenido no quede tan pegado en la parte superior con el *navbar*: añadimos una clase CSS en el div del componente raíz de la aplicación:

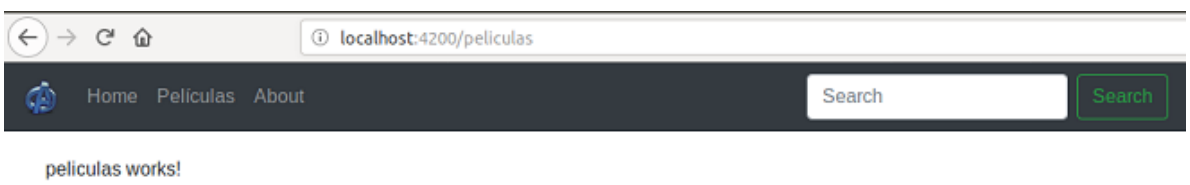
app.component.html

```
<app-navbar></app-navbar>
<div class="container main-container">
  <router-outlet></router-outlet>
</div>
```

Y ahora en el fichero src → styles.css añadimos el siguiente código:

styles.css

```
.main-container {
  margin-top: 20px;
}
```



Utilizando animaciones

Vamos a hacer ahora que cuando se cargue una página se produzca una animación que mejore la visualización de la misma. Para ello, tenemos que copiar el contenido del fichero `animate.css`, el cual se proporciona como un recurso para esta práctica, y tenemos que pegarlo al final del fichero `src → styles.css`:

styles.css

```
.main-container {
  margin-top: 20px;
}

.animated {
  -webkit-animation-duration: 1s;
  animation-duration: 1s;
  -webkit-animation-fill-mode: both;
  animation-fill-mode: both;
}
.fast {
  -webkit-animation-duration: 0.4s;
  animation-duration: 0.4s;
  -webkit-animation-fill-mode: both;
  animation-fill-mode: both;
}
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}
.fadeIn {
  animation-name: fadeIn;
}
```

Ahora tenemos que aplicar estas clases CSS allá donde consideremos. Por ejemplo en ‘about’ y en ‘home’ (mejor todavía no hacemos nada de esto para `peliculas.component.html`):

about.component.html

```
<h1 class="animated fadeIn fast">Acerca de Movies App</h1>
<hr>
<p class="animated fadeIn">Esta es una aplicación realizada con Angular 8 y Bootstrap cuyo propósito es conocer diversos aspectos del framework, principalmente enrutamiento y servicios.</p>
```

home.component.html

```
<div class="jumbotron jumbotron-fluid animated fadeIn fast">
  <div class="container">
    <h1 class="display-4">Movies App</h1>
    <p class="lead">Esta es una aplicación sobre películas</p>
  </div>
</div>
```

Con estas modificaciones deberíamos notar una pequeña animación al cargarse ambas páginas, sobretodo para el caso de *about*.

Conclusión

En esta primera parte de la práctica hemos hecho bastante trabajo:

- Creación del proyecto
- Adición de *assets*, principalmente las imágenes que necesitaremos
- Componentes iniciales de la aplicación: home, about, peliculas, navbar
- Sistema de navegación y definición de las rutas
- Amimaciones para mejorar visualmente la carga de las páginas

En la siguiente parte de esta práctica comenzamos a dar contenido a la página de las películas.