1. Instalamos los snippets para VS Code de Angular.

2. Creación del proyecto: **ng new** appHeroes

3. Copio las imágenes del proyecto y favicon.ico en la ruta **src/assets/img**

4. Creamos el componente navbar dentro una carpeta shared y copiamos código desde bootstrap.

   En **navbar.component.html** copiamos el código de navbar.

5. El contenido de **app.component.html**:

   <app-navbar></app-navbar>

6. Creamos un componente home. El contenido de **home.component.html** será el componente Jumbotron copiado de bootstrap.

7. Creamos el sistema de rutas: **app.routes.ts**.

8. Agregamos a **app.module.ts** la referencia a este fichero recién creado.

9. Para poder utilizar las rutas en **app.component.html**:

   <router-oulet> </router-outlet>

10. Utilizamos las rutas de cada componente agregadas a **app.route.ts** del sitio web en **navbar.component.html**:

    <a class = “nav-link” [routerLink] =”[‘home’]”>home</a>

11. En **navbar.component.html** para que se marque la ruta activa, le ponemos la propiedad routerLinkActive = “active”:

12. Creamos un componente about. El contenido de **about.component.html** será el generado por lorem ipsum.

13. Creamos un componente heroes (**heroes.component.ts** y **heroes.component.html**) y utilizaremos tarjetas de colores de bootstrap. Trabajamos el diseño y lo repetimos hasta formar una página con seis tarjetas. A cada tarjeta le agregamos un botón copiado de bootstrap.

14. Creamos el servicio a través del snippet: ag-service. Introducimos un console.log en el constructor del servicio.

15. Incluimos el servicio en **app.module.ts**. NOTA: los servicios se incorporan SIEMPRE en los *providers*.

16. Copiamos en el servicio el contenido del fichero **heroes.txt** y lo declaramos como un objeto JSON **heroes** dentro de la clase de tipo *any* y privado a la clase. En el servicio también, creamos una función **getHeroes()** que retorne el JSON this.heroes.

17. En el componente de **heroes** creado en el punto 13 inyectamos el servicio en el constructor y cargamos el objeto **heroe**s declarado en la clase invocando a la función **getHeroes()** del servicio.

18. Creamos una interfaz para definir un tipo de datos personalizado en el componente de **heroes**. Así para definir el nuevo objeto de datos **heroes**:

```
export interface Heroe {
        nombre: string;
        bio: string;
        img: string;
        aparicion: string;
        casa: string;
}
```

    Utilizamos esta interface en **heroes.component.ts** y el objeto JSON dejará de ser de tipo any a Heroe así como la función getHeroe del servicio.

19. Diseñamos la página **heroes.component.html** a través de la clase card-columns de bootstrap y rellenamos las tarjetas mediante un ngFor que recorra nuestra objeto JSON heroes.

20. Creamos un componente heroe (**heroe.component.ts** y **heroe.component.html**) para mostrar la información de cada héroe individual.

21. Para poder acceder a este componente **heroe** desde **heroes**, agregamos también la ruta de **heroe** a **app.route.ts**. Para acceder aun héroe específico la ruta debe ser:

```
{ path: 'heroe/:id', component: HeroeComponent }
```

22. En **heroes.component.html** agregamos un índice al **ngFor** y que será el utilicemos para la redirección a la página de **heroe** mediante la modificación de la propiedad **routerLink** del enlace:

```
< a [routerLink] = "[ '/heroe' , i]" class="..."> Ver más ... </a>
```

23. Para la redirección de la página en **heroes.component.ts** inyectamos en el contructor el router y la función verHeroe(id: number) redirecciona a través del método navigate del objeto _router:

```
this._router.navigate(['/heroe',id]);
```

24. Para recibir los parámetros por url debemos utilizar el objeto de Angular ActivatedRoute. Para ello lo importamos en el componente typescript de héroes y lo utilizamos de la siguente forma dentro del constructor tras inyectarlo:

```
this.activatedRoute.params.subscribe ( params => {
        console.log(params)
});
```

25. Para obtener un héroe en particular creamos una función **en el servicio** parecida a getHeroe pero que devuelva sólo el héroe asociado al id pasado como parámetro:

```
getHeroe( idx: Number) {

        return this.heroes[idx];
}
```

26. En el componente de héroe **heroe.component.ts** usamos esta función para acceder a cada una e las páginas de los héroes. Para ello en el constructor la invocamos de esta forma:

```
this.activatedRoute.params.subscribe( params =>{

        this.heroe = this._heroesService.getHeroe(params['id']);
});
```

27. Creamos el componente HTML del héroe **heroe.component.html** siguiendo el diseño propuesto.

28. A través de un ngIf especificamos el tipo de icono de la casa (Marvel ó DC), atendiendo al contenido del campo del objeto JSON 'casa'.

29. Para pasar a mayúsculas una string debemos utilizar pipes.

```
{{ heroe.nombre | uppercase }}
```

30. Para el buscador de Héroes, le asignamos un id a la caja de texto y un evento al botón (keyup) invocando una función a la que le pasaremos como parámetro el contenido de la caja de texto.

```
(keyup.enter) = "buscarHeroe(buscarTexto.value)"
```

31. Esta función **bucarHero**e la desarrollaremos en el servicio y lo único que hace es crear un array auxiliar con los elementos del objeto JSON que coincidan con el parámetro de búsqueda.

32. Creamos un nuevos componente **buscador.component** para mostrar los resultados de la búsqueda que, por supuesto agregaremos a app.routes.ts.

33. Inyectamos activatedRoute en el constructor de **buscador.component** para devolver el objeto relacionado con el término pasado como parámetro al igual que en el punto 26.

```
ngOnInit() {
        this.activatedRoute.params.subscribe ( params => {

        console.log(params['termino'])
        });
}
```

34. Para poder invocar este **buscador.component** debemos redireccionar la petición desde el navbar.component.ts e inyectar el objeto Router en el constructor:

```
buscarHeroe( termino: string) {

        this.router.navigate(['/buscar', termino ]);
}
```

35. En **buscador.component.ts** inyectamos el servicio en el constructor para invocarlo en la función ngOnInit() para obtener un array de los héroes que cumplen el criterio de búsqueda:

```
ngOnInit() {

        this.activatedRoute.params.subscribe( params => {

                console.log(params['termino']);

                this.heroes= this._heroesService.buscarHeroes(params['termino']);

                console.log(this.heroes);

        });
}
```

    Debemos agregar el array **heroes** y la variable **termino** a la clase del componente.

36. En **buscador.component.html** creamos una clase de bootstrap debajo del nombre del héroe para cuando la búsqueda nos retorne un array vacío que se controlará a través de un ngIf que controle que el objeto heroes que ha leído el component no está vacío.