

Desarrollo web en entorno cliente



Tema 3: Angular

Índice

¿Qué es Angular?	3
Archivos de configuración para el espacio de trabajo:.....	4
Archivos de proyecto aplicación:	4
Archivos de origen de la aplicación:	5
Archivos de configuración de la aplicación:	6
Tutorial Tour of Heroes	6
¿Qué objetivos has conseguido?	13

¿Qué es Angular?

Es un framework opensource desarrollado por Google para desarrollar aplicaciones web de una sola página. Como por ejemplo las webs SPA(Single Page Application) que cargan de golpe toda la página, esto supone que la primera vez tardan más en cargar pero luego es instantáneo.

Es modular y escalable lo que permite que se adapte a lo que necesitemos en nuestro proyecto. Al basarse en el estándar de componentes web, junto al conjunto de interfaz de programación de aplicaciones API, permite crear etiquetas HTML personalizadas.

El lenguaje de programación que se utiliza es TypeScript y separa en plantillas la parte de front de la parte del back pudiendo utilizar otras herramientas más específicas para sacar mayor rendimiento.

Se integra bien con herramientas de testing y con Ionic lo que facilita la creación de web responsive.

La arquitectura de una aplicación Angular se basa en determinados conceptos fundamentales. Los bloques de construcción básicos son componentes que están organizados en **NgModules**.

Una aplicación en Angular, al menos tiene que tener un componente raíz que permita el arranque.

Los componentes definen las vistas, que son conjuntos de elementos de pantalla entre los que Angular puede elegir y modificar de acuerdo con la lógica y los datos de su programa.

Los componentes utilizan los servicios que proporcionan funciones específicas y pueden inyectarse en componentes como dependencias consiguiendo que el código sea modular, reutilizable y eficiente.

Tanto los componentes como los servicios son clases que utilizan decoradores que marca su tipo y nos proporciona metadatos que le indica a Angular como usarlos.

Los metadatos para una clase de servicio proporcionan la información que Angular necesita para ponerla a disposición de los componentes a través de la inyección de dependencia.

La clase servicio permiten asociar los datos o la lógica con una vista específica y compartirlos entre los componentes.

El inyector de dependencia está conectado a angular y se utiliza para proporcionar nuevos componentes con servicios.

Una plantilla combina HTML con las etiquetas personalizadas de angular que modifican los elementos HTML antes que se muestren. Las directivas de plantilla proporcionan la lógica del programa y el marcado de enlace conecta los datos de su aplicación y el DOM.

Servicios e inyección de dependencias, permiten mantener sus clases de componentes de forma ágil y eficiente. No obtienen datos del servidor, no validan la entrada del usuario ni inician sesión directamente en la consola; delegan esas tareas a los servicios.

Enrutamiento, sirve para definir las rutas de navegación entre los diferentes estados de la aplicación y jerarquías. Se basa en las convenciones familiares del navegador que interpreta los

enlaces cuando un usuario realiza una acción. Por ejemplo, el ingresar una url, el clic en los enlaces de una página a otra, clic en botones, etc..

El enrutador asigna rutas similares a una url a vistas en lugar de páginas, interceptando el comportamiento del navegador ante una acción del usuario y muestra u oculta las jerarquías de vista.

Estructura de archivos

Archivos de configuración para el espacio de trabajo:

Todos los proyectos dentro de un espacio de trabajo comparten un conjunto de ficheros de configuración. En el nivel superior de la jerarquía se encuentran todos los ficheros de configuración del espacio de trabajo, los archivos de configuración para la aplicación raíz y las subcarpetas para los archivos fuente y de prueba de la aplicación a nivel raíz.

.editorconfig → Configuración para editores de código.

.gitignore → Especifica archivos sin seguimiento intencional que Git debe ignorar.

README.md → Documentación introductoria para la aplicación raíz.

angular.json → Los valores predeterminados de configuración de CLI para todos los proyectos en el espacio de trabajo, incluidas las opciones de configuración para compilar, servir y probar herramientas que utiliza la CLI, como TSLint , Karma y Transportador.

package.json → Configura las dependencias del paquete npm que están disponibles para todos los proyectos en el espacio de trabajo.

package-lock.json → Proporciona información sobre la versión de todos los paquetes instalados node_modules por el cliente npm.

src/ → Archivos de origen para el proyecto de aplicación de nivel raíz.

node_modules/ → Proporciona paquetes npm a todo el espacio de trabajo. Las node_modules dependencias de todo el espacio de trabajo son visibles para todos los proyectos.

tsconfig.json → La configuración básica de TypeScript para proyectos en el espacio de trabajo. Todos los demás archivos de configuración heredan de este archivo base.

tslint.json → Configuración de TSLint predeterminada para proyectos en el espacio de trabajo.

Archivos de proyecto aplicación:

El comando "CLI ng new ejemplo-app" crea una carpeta de espacio de trabajo llamada "ejemplo-app" y genera un nuevo esqueleto en src/carpeta al nivel superior del espacio de trabajo. Una nueva aplicación contiene los archivos fuente para un módulo raíz, con un componente raíz y una plantilla.

Archivos de origen de la aplicación:

En el nivel superior de `src/` soporte para probar y ejecutar la aplicación. Las subcarpetas contienen el origen de la aplicación y la configuración específica de la aplicación.

app/ → Contiene los archivos de componentes en los que se definen la lógica y los datos de su aplicación.

assets/ → Contiene imágenes y otros archivos de activos que se copiarán tal cual cuando compile su aplicación.

environments/ → Contiene opciones de configuración de compilación para entornos de destino particulares. De forma predeterminada, hay un entorno de desarrollo estándar sin nombre y un entorno de producción ("prod"). Puede definir configuraciones de entorno de destino adicionales.

favicon.ico → Un icono que se utilizará para esta aplicación en la barra de marcadores.

index.html → La página HTML principal que se muestra cuando alguien visita su sitio. La CLI agrega automáticamente todos los archivos JavaScript y CSS al crear su aplicación, por lo que normalmente no necesita agregar ninguna etiqueta `<script>` o `<link>` aquí manualmente.

main.ts → El principal punto de entrada para su aplicación. Compila la aplicación con el compilador JIT y arranca el módulo raíz de la aplicación (AppModule) para que se ejecute en el navegador. También puede usar el compilador AOT sin cambiar ningún código agregando la `--aot` bandera a la CLI buildy los servecomandos.

polyfills.ts → Proporciona scripts de polyfill para compatibilidad con el navegador.

styles.sass → Muestra una lista de archivos CSS que proporcionan estilos para un proyecto. La extensión refleja el preprocesador de estilo que ha configurado para el proyecto.

test.ts → El principal punto de entrada para sus pruebas unitarias, con alguna configuración específica de Angular. Normalmente no es necesario editar este archivo.

Dentro de la `src/` carpeta, la `app/` carpeta contiene la lógica y los datos del proyecto. Los componentes, las plantillas y los estilos.

app/app.component.ts → Define la lógica del componente raíz de la aplicación, denominado AppComponent. La vista asociada con este componente raíz se convierte en la raíz de la jerarquía de vistas a medida que agrega componentes y servicios a su aplicación.

app/app.component.html → Define la plantilla HTML asociada a la raíz AppComponent.

app/app.component.css → Define la hoja de estilo CSS base para la raíz AppComponent.

app/app.component.spec.ts → Define una prueba unitaria para la raíz AppComponent.

app/app.module.ts → Define el módulo raíz, llamado AppModule, que le dice a Angular cómo ensamblar la aplicación. Inicialmente declara solo el AppComponent. A medida que agrega más componentes a la aplicación, deben declararse aquí.

Archivos de configuración de la aplicación:

Son los archivos de configuración específicos de la aplicación que se encuentran en `projects/project-name/`.

.browserslistrc → Configura el uso compartido de los navegadores de destino y las versiones de Node.js entre varias herramientas de front-end.

karma.conf.js → Configuración de Karma específica de la aplicación .

tsconfig.app.json → Configuración de TypeScript específica de la aplicación , incluidas las opciones del compilador de plantillas de TypeScript y Angular.

tsconfig.spec.json → Configuración de TypeScript para las pruebas de la aplicación. Consulte Configuración de TypeScript .

tslint.json → Configuración de TSLint específica de la aplicación .

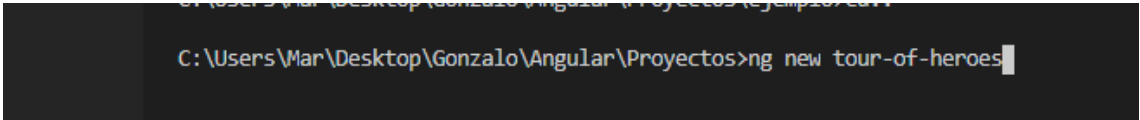
Tutorial Tour of Heroes

Con este tutorial, vamos aprender los conceptos básicos de Angular. Una vez completado aprenderemos a usar las directivas para mostrar y ocultar elementos y mostrar una lista de los datos del héroe. Crearemos componentes para mostrar los detalles de cada héroe.

Utilizaremos el enlace de datos unidireccional para los datos solo de lectura, agregaremos campos que podremos modificar, vincularemos métodos de componentes a eventos del usuario como clics a botones o teclas. El usuario podrá elegir un héroe y lo pondrá editar.

Crear el proyecto

Lo primero que vamos a crear un nuevo espacio de trabajo y un proyecto inicial de la aplicación. Para ello ejecutaremos el siguiente comando “`ng new tour-of-heroes`” en el terminal de nuestro editor, en este caso Visual Studio Code



```
C:\Users\Mar\Desktop\Gonzalo\Angular\Proyectos\ejemplo>ng new tour-of-heroes
```

```
TERMINAL  PROBLEMS  OUTPUT  CONSOLA DE DEPURACIÓN

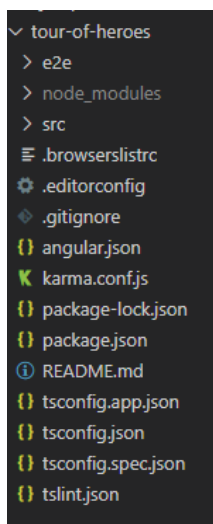
Microsoft Windows [Versión 10.0.19041.630]
(c) 2020 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Mar\Desktop\Gonzalo\Angular\Proyectos\ejemplo>cd..

C:\Users\Mar\Desktop\Gonzalo\Angular\Proyectos>ng new tour-of-heroes
? Do you want to enforce stricter type checking and stricter bundle budgets in the workspace?
  This setting helps improve maintainability and catch bugs ahead of time.
  For more information, see https://angular.io/strict No
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use?
  CSS
> SCSS [ https://sass-lang.com/documentation/syntax#scss ]
  Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less  [ http://lesscss.org ]
  Stylus [ https://stylus-lang.com ]
```

A las preguntas contestaremos No a mandar información, Yes a Angular routing y elegimos SCSS.

Cuando termina de instalar nos crea el espacio de trabajo con una carpeta raíz llamada tour-of-heroes. También nos crea el esqueleto inicial de la aplicación.

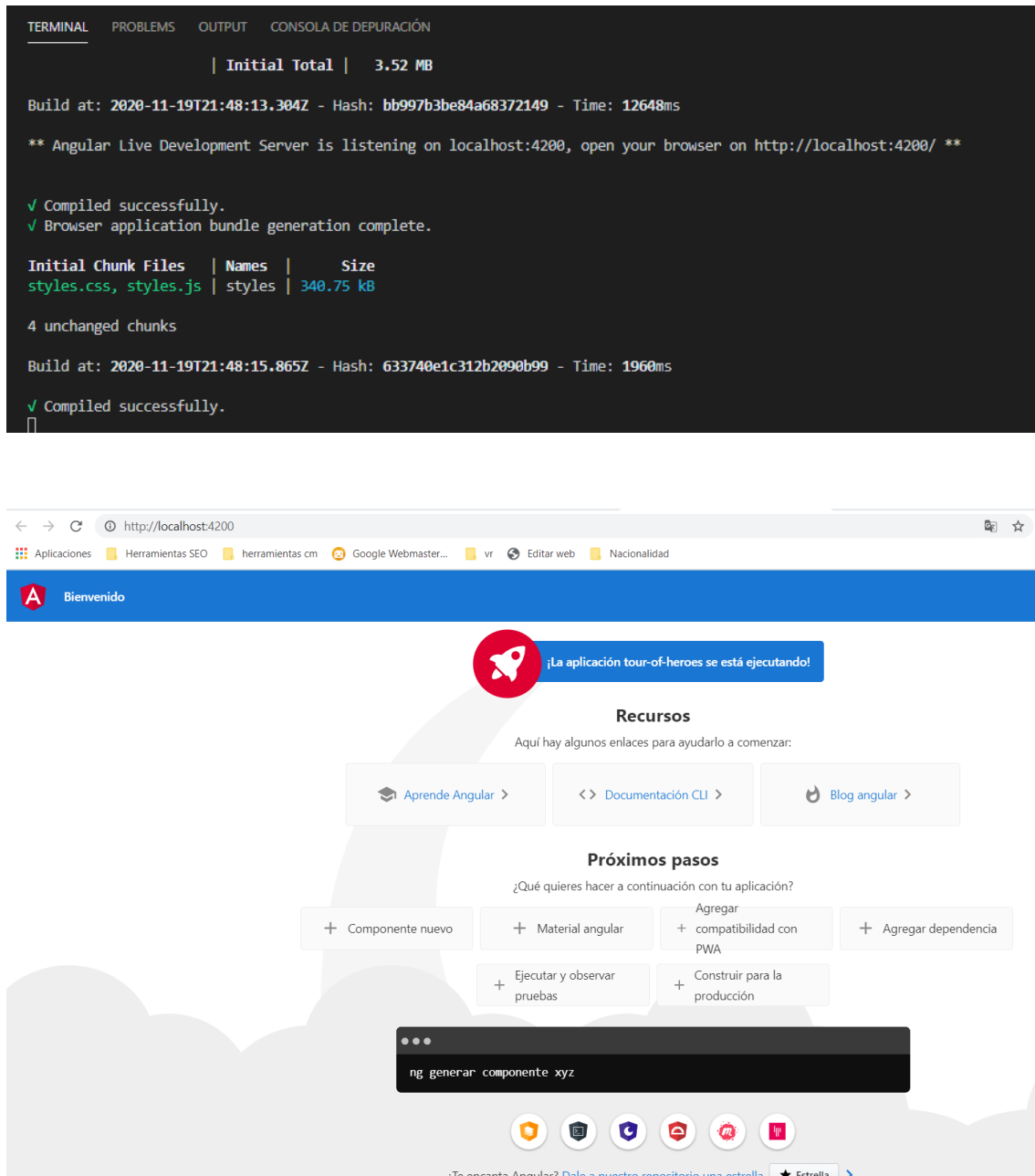


Nos dirigimos a la carpeta del proyecto cd tour-of-heroes en el terminal y ejecutamos

```
C:\Users\Mar\Desktop\Gonzalo\Angular\Proyectos>cd tour-of-heroes

C:\Users\Mar\Desktop\Gonzalo\Angular\Proyectos\tour-of-heroes>ng serve --open
```

Iniciamos el servidor y abre el navegador con la url: <http://localhost:4200/> para cargar nuestra aplicación.



La página que abre es el Shell de la aplicación que está controlado por el componente AppComponent.

Los componentes son bloques de construcción para las aplicaciones. Nos muestran datos en pantalla, capturan la entrada del usuario y en función a esas acciones toma ciertas medidas.

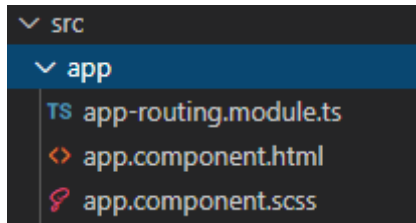
Vamos hacer algunos cambios en la página de inicio de la aplicación, nos dirigimos a `src/app`.

Nos encontramos el Shell AppComponent dividido en tres extensiones de fichero:

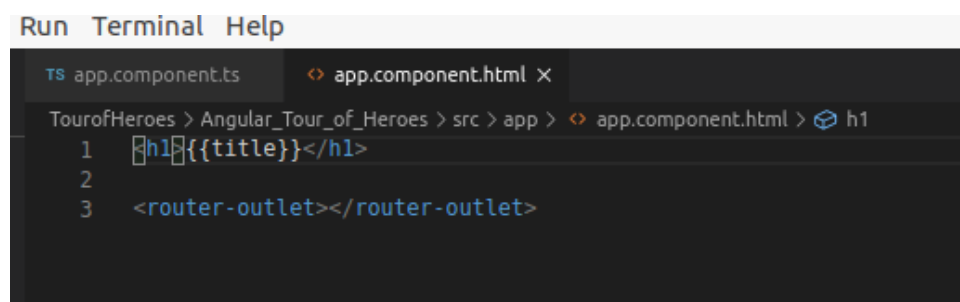
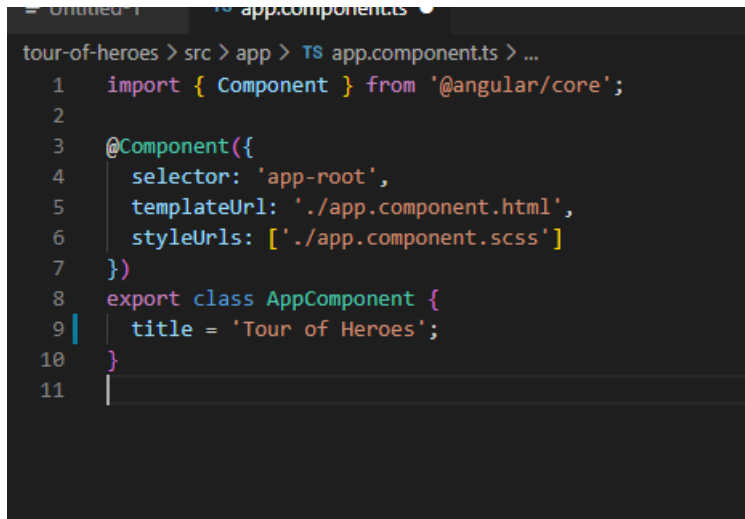
.ts para la clase de componente escrito en TypeScript.

.html para la plantilla del componente escrita en HTML.

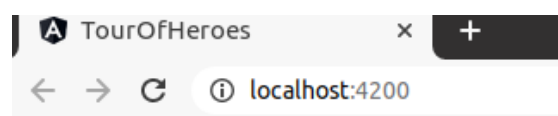
.css para los estilos CSS.



Vamos a cambiar el título de la aplicación, para eso nos dirigimos a la clase componente “app.component.ts” y cambiamos el valor del title por Tour of Heroes

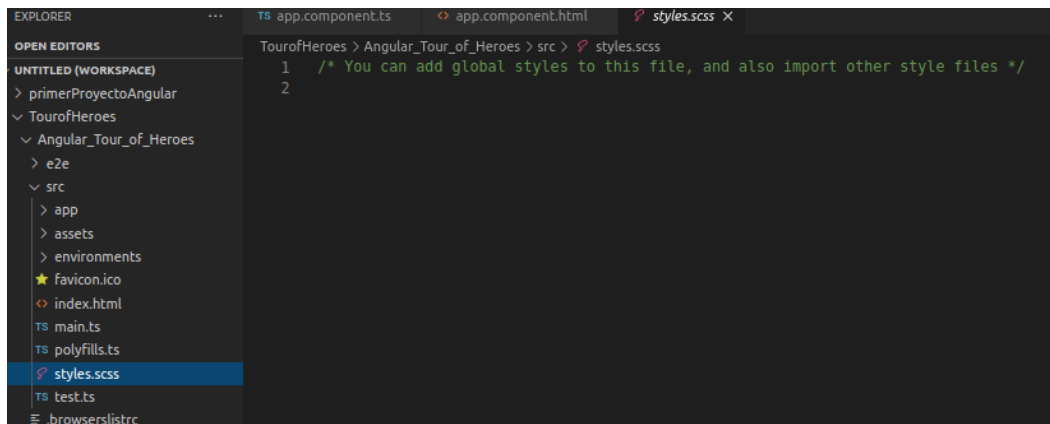


Obtenemos como resultado en el navegador lo siguiente:



Tour of Heroes

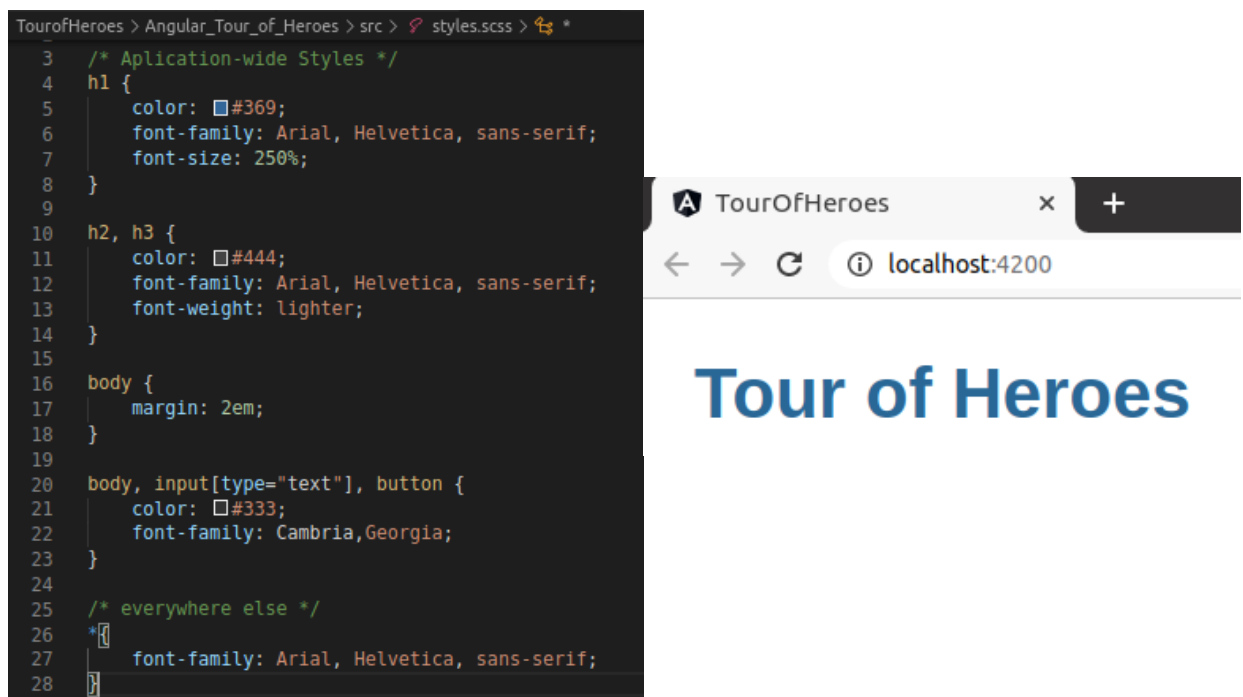
Ahora vamos aplicar estilo al h1, para ello abrimos el fichero de configuración de estilos css.



src/styles.css

Cuando creamos un proyecto, este archivo de configuración global de estilos viene vacío.

Le aplicamos algunos estilos.



Ahora vamos a crear un nuevo componente para mostrar información del héroe

Utilizando Angular Cli generamos el nuevo componente introduciendo el siguiente comando en el terminal “ **ng generate componet heroes** “

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

estudiante@Gonzalo:~/Desktop/TourofHeroes/Angular_Tour_of_Heroes$ ng generate component heroes
CREATE src/app/heroes/heroes.component.scss (0 bytes)
CREATE src/app/heroes/heroes.component.html (21 bytes)
CREATE src/app/heroes/heroes.component.spec.ts (626 bytes)
CREATE src/app/heroes/heroes.component.ts (276 bytes)
UPDATE src/app/app.module.ts (475 bytes)
estudiante@Gonzalo:~/Desktop/TourofHeroes/Angular_Tour_of_Heroes$
```

En src/app nos crea el nuevo componente heroes con los siguientes archivos de configuración: component.scss para los estilos, component.html para el html, component.ts para el TypeScript

```
TS heroes.component.spec.ts  TS heroes.component.ts x
TourofHeroes > Angular_Tour_of_Heroes > src > app > heroes > TS heroes.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-heroes',
5    templateUrl: './heroes.component.html',
6    styleUrls: ['./heroes.component.scss']
7  })
8  export class HeroesComponent implements OnInit {
9
10     constructor() { }
11
12     ngOnInit(): void {
13     }
14
15 }
```

Es importante fijarse en la anotación @Component y en @angular/core, con estas anotaciones angular puede identificar que se utiliza una librería y que nos referimos a un componente.

@Component es una función que indica a angular que se trata de un componente y especifica los metadatos de un componente. Tiene tres propiedades de metadatos y es la siguiente:

- 1.- **selector**, es el selector de componente de elementos CSS.
- 2.- **templateUrl**, ubicación de la plantilla del componente.
- 3.- **styleUrl**, ubicación de los estilos CSS.

Vamos añadir una propiedad a un héroe y lo vamos a llamar “WindStorm”, abrimos el fichero **heroes.component.ts**

En la clase **HeroesComponent** creamos una variable heroee con el nombre WindStorm

```
ts heroes.component.ts •
TourofHeroes > Angular_Tour_of_Heroes > src > app > heroes > ts heroes.component.ts
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-heroes',
5    templateUrl: './heroes.component.html',
6    styleUrls: ['./heroes.component.scss']
7  })
8  export class HeroesComponent implements OnInit {
9
10     hero = 'windStorm';
11
12     constructor() { }
13
14     ngOnInit(): void {
15     }
16
17 }
```

Vamos a mostrar a nuestro heroe, para eso nos dirigimos al componente html y escribimos {{heroe}}

```
TourofHeroes > Angular_Tour_of_Heroes > src > app > heroes > heroes.component.html
1  {{heroe}}
2
```

Ahora hay que añadirlo a la aplicación general app.component.html

```
TourofHeroes > Angular_Tour_of_Heroes > src > app > app.component.html > ...
1  <h1>{{title}}</h1>
2  <app-heroes></app-heroes>
3
```

Vamos a crear la interface de nuestro héroe, nos dirigimos a la siguiente ruta src/app y creamos un fichero hero.ts para nuestra interfaz. Abrimos el fichero y creamos la interface con dos parámetros (id,name) para la creación de nuestra plantilla de héroe con esos dos atributos.

```
TourofHeroes > Angular_Tour_of_Heroes > src > app > ts hero.ts • Hero > name
1  export interface Hero {
2    id: number;
3    name: string;
4  }
```

¿Qué objetivos has conseguido?

Como crear un proyecto de angular desde cero, ampliar la gramática de TypeScript, crear un componente, crear una interfaz. Iniciar el ejemplo de tour de héroes.

Bibliografía: <https://angular.io/start>

Github: [https://github.com/Kurtgon/Angular Tour of Heroes](https://github.com/Kurtgon/Angular_Tour_of_Heroes)