

Trabajo teórico grupal

Introducción a Angular

Carlos Poveda Cañizares

Adil Akhazzan



Índice

1. Introducción

Antes de comenzar con nuestro pequeño proyecto introductorio, conviene mencionar algunos aspectos de interés relacionados con Angular.

¿Qué es Angular?

Angular es un framework opensource que nos ayuda con la creación de aplicaciones web SPA (Single Page Application), es decir, aplicaciones web de una sola página.

Angular nos permite separar completamente el frontend del backend y mantiene nuestro proyecto mucho más ordenado gracias al patrón MVC (Modelo-Vista-Controlador). Además, Angular es modular y escalable.

También hay que nombrar algunos pre requisitos que debemos tener en cuenta antes de comenzar a utilizar Angular.

Pre requisitos:

- Conocimientos en JavaScript.
- Conocimientos en HTML.
- Conocimientos en CSS.

Cabe destacar que es bastante recomendable que se tengan también algunos conocimientos básicos de TypeScript.

Por último, hemos de tener instalado node.js en nuestro ordenador para poder seguir el tutorial.

2. Instalación

Para comenzar, debemos instalar Angular en nuestro ordenador, para esto ejecutaremos el siguiente comando:

```
npm install -g @angular/cli
```

Si nuestra instalación se ha completado correctamente, se debería mostrar algo parecido a la siguiente imagen:

```
PS C:\Users\Carlo\Desktop\Angular> npm install -g @angular/cli
added 183 packages, and audited 184 packages in 16s

22 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Una vez hemos instalado Angular en nuestro ordenador, el siguiente paso es crear nuestro proyecto, para ello debemos ejecutar el siguiente comando:

```
ng new tutorial-angular
```

Donde “tutorial-angular” es el nombre que tendrá nuestro proyecto, por lo que puede ser sustituido a conveniencia del usuario.

Se nos ofrecerán algunas opciones a la hora de crear nuestro proyecto, en las siguientes imágenes veremos las seleccionadas por nosotros:

```
PS C:\Users\Carlo\Desktop\Angular> ng new tutorial-angular
? Would you like to add Angular routing? (y/N) y
```

```
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
SCSS   [ https://sass-lang.com/documentation/syntax#scss ]
Sass   [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
Less   [ http://lesscss.org ]
```

Tras esto, el instalación comenzará y tras unos minutos, si la instalación se ha realizado correctamente, debería aparecernos algo como la siguiente captura de pantalla:

```
CREATE tutorial-angular/src/environments/environment.prod.ts (51 bytes)
CREATE tutorial-angular/src/environments/environment.ts (658 bytes)
CREATE tutorial-angular/src/app/app-routing.module.ts (245 bytes)
CREATE tutorial-angular/src/app/app.module.ts (393 bytes)
CREATE tutorial-angular/src/app/app.component.html (23364 bytes)
CREATE tutorial-angular/src/app/app.component.spec.ts (1103 bytes)
CREATE tutorial-angular/src/app/app.component.ts (220 bytes)
CREATE tutorial-angular/src/app/app.component.css (0 bytes)
✓ Packages installed successfully.
```

Una vez creado nuestro proyecto de Angular, simplemente tendremos que entrar en la carpeta del proyecto e iniciarlo con el siguiente comando:

`ng serve --open`

La primera vez que iniciamos el proyecto tardará unos instantes, pero si todo ha ido según lo previsto, debería aparecer algo como esto en la consola:

```
PS C:\Users\Carlo\Desktop\Angular\tutorial-angular> ng serve --open
✓ Browser application bundle generation complete.

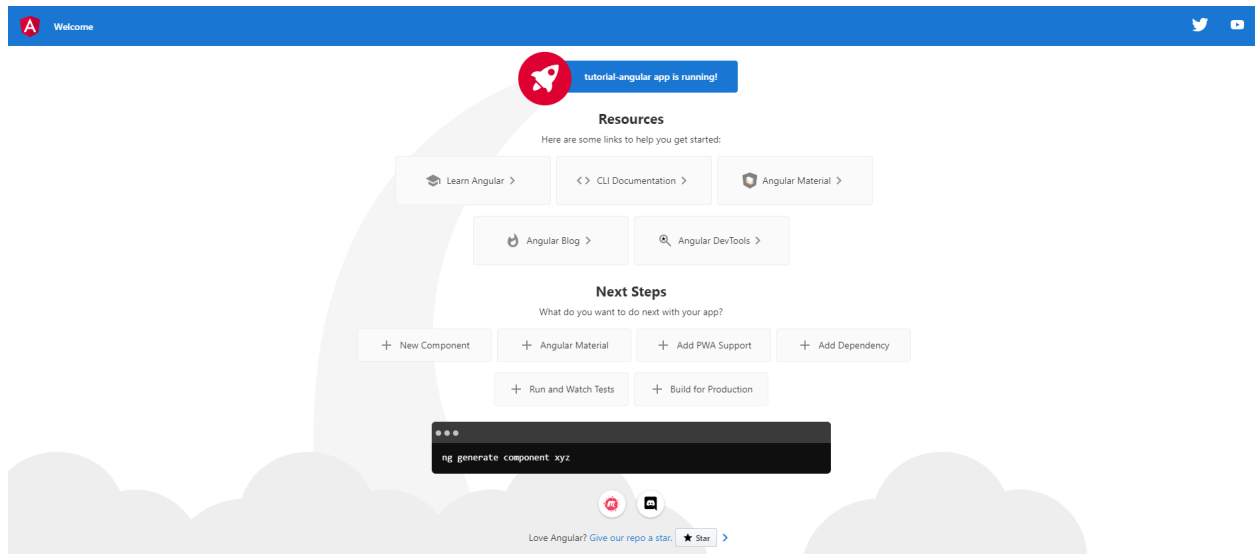
Initial Chunk Files | Names          | Raw Size
vendor.js           | vendor        | 2.00 MB |
polyfills.js        | polyfills     | 339.25 kB |
styles.css, styles.js | styles       | 212.52 kB |
main.js             | main         | 53.60 kB |
runtime.js          | runtime      | 6.87 kB |
                    | Initial Total | 2.60 MB

Build at: 2022-01-07T12:15:35.049Z - Hash: 33b80d1377fbf5de - Time: 11572ms

** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
```

Y se nos abrirá una ventana en nuestro navegador predeterminado, en el puerto 4200 por defecto:



Ya tenemos creado nuestro primer proyecto en Angular.

3. Conceptos Básicos Angular

Componente

Un componente en Angular es un elemento compuesto:

- Un archivo .html que será el cuerpo de nuestro componente, es decir la interfaz que los usuarios van a ver.
- Un archivo .ts que contiene toda la lógica de nuestro componente. Los distintos métodos y variables de nuestro componente.
- Un archivo .css que almacena el estilo de nuestro componente

A diferencia que en Vue , los componentes en angular se almacenan en carpetas separadas y sus distintos elementos se almacenan en distintos archivos. En vue tanto la lógica, el html y el css se almacenan dentro de un mismo archivo **.vue** que hace referencia a nuestro componente.

El comando para genera un componente es **ng generate componente “nombre”**. Este comando te genera todos los archivos del componente antes mencionados en una carpeta , además también importa ese componente en el componente principal app para poder utilizarlo, en vue estos pasos se deben de crear manualmente.

Módulos

Los módulos en angular son como unos contenedores donde se declaran una serie de componentes relacionados entre sí . Por ejemplo , si tenemos productos y usuarios, podríamos dividir nuestro código en dos módulos .Un módulo usuario donde se guardan los diferentes componentes y lógica del usuario y también sus rutas, y otro módulo productos.

Los módulos nos permiten crear nuestra aplicación de una manera estructurada que nos permite tener tantos módulos como necesitemos de una manera clara. Angular tiene un módulo principal llamado **app.module** , allí es donde se indican los nuevos módulos creados.

El comando **ng generate module “nombre carpeta/nombre” –module app –flat** nos crea el módulo automáticamente y agrega las dependencias necesarias al app.module. Si queremos que también nos cree el routing de ese módulo añadimos **–routing**

Servicios

Un servicio se encarga de proporcionar los datos necesarios para nuestra aplicación. Para ello se suele conectar a la base de datos o a una Api Rest.

No solo nos proporciona dichos datos , también define qué podemos hacer con esos datos. La utilización de servicios es una buena práctica de baidoa que deja el código muy limpio y se puede entender rápidamente qué hace cada servicio . Este concepto quedará más claro tras observar su uso en la aplicación de ejemplo.

Routing

El routing es el sistema que se encarga de redirigir al usuario al componente adecuado según la url ingresado . Para ello las rutas se crean en un archivo llamado routing.modules.ts.

Ejemplo:

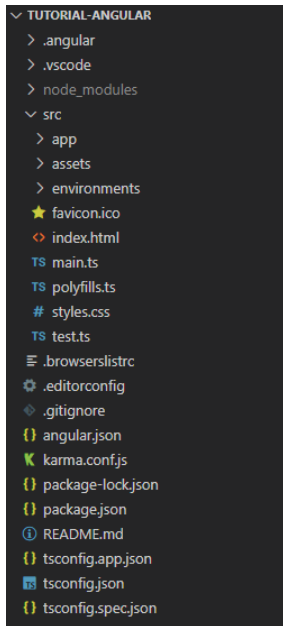
```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { PokemonComponent } from '../pokemon/pokemon.component';
import { ListPokemonComponent } from '../list-pokemon/list-pokemon.component';
const routes: Routes = [
  {
    path: '',
    component: ListPokemonComponent
  },
  {
    path: 'pokemon/:id',
    component: PokemonComponent
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class PokedexRoutingModule { }
```

Como se puede apreciar cuando se introduce una url se redirige al componente adecuado. Este archivo se genera al crear un módulo si se introducen los comandos mostrados anteriormente , sino, se deberá de declarar de manera manual , o añadir las rutas en el app-routing. En este archivo se guardan las rutas de manera general , la forma más adecuada es que cada módulo(ex usuario) tenga sus propias rutas .

4. Explicación del proyecto base

Una vez hemos creado nuestro proyecto, podemos entrar a su carpeta para ver los archivos que se han creado (nosotros lo hacemos desde Visual Studio Code):



En esta sección vamos a explicar brevemente, los archivos más importantes del proyecto base.

En primer lugar, vamos a abrir “index.html” que se trata del html principal de nuestro proyecto:


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

Como podemos ver, se trata de un archivo html básico, pero dentro del body, encontramos una etiqueta `<app-root></app-root>`, esta etiqueta es la llamada a un componente. Este componente se encuentra dentro de la carpeta `src`:

```

v app
  TS app-routing.module.ts
  # app.component.css
  <> app.component.html
  TS app.component.spec.ts
  TS app.component.ts
  TS app.module.ts
```

En esta carpeta encontramos un archivo “.css”, que se trata de la hoja de estilos del componente. También, vemos un archivo “.html”, que se trata del archivo que contiene el html de nuestro componente y un archivo `app-routing` donde se almacenan las distintas rutas de nuestra aplicación. Un archivo “.spec.ts” donde podremos realizar pruebas de Angular. El siguiente archivo es “`app.components.ts`”, aquí es donde agregaremos el código de nuestro componente (la extensión `.ts`, hace referencia a TypeScript), vamos a explicar con más detalle este archivo:

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

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

Con “@Component” se está definiendo un componente de Angular. Dentro de esto encontramos el “selector”, esto da el nombre (‘app-root’) a la etiqueta html que llamara a nuestro componente, podemos comprobar que en la captura de “index.html” anterior, se llamaba a este componente mediante un etiqueta con el mismo nombre. También encontramos “templateUrl” que asigna el archivo que contiene el html del componente y de igual forma con “styleUrls” se asigna el archivo que contiene la hoja de estilos del componente. Y al final del archivo vemos que la clase se exporta como “AppComponent”.

Por último, para que nuestro componente funcione correctamente, debemos dirigirnos al archivo “app.module.ts”:

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

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

Como vemos aquí se importa el archivo “app.component” que habíamos exportado en el archivo anteriormente mencionado, además encontramos un decorador “@NgModule” que se encarga de dar atributos extra a nuestro componente. El atributo más importante

es “bootstrap” que se encarga de levantar nuestro componente. En último lugar, se exporta la clase como “AppModule”.

Por último, nos dirigimos de nuevo a la carpeta “src” y al archivo “main.ts”, donde se importará nuestro componente, ya que si no importamos nuestro componente aquí, este nunca funcionará.

```
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.error(err));
```

5. Pequeño ejercicio de iniciación

Para comprender un poco mejor el funcionamiento de Angular, vamos a realizar un ejercicio extremadamente sencillo que nos servirá como ejemplo.

En primer lugar, vamos a crear un nuevo componente, esto se puede realizar de manera manual, añadiendo los archivos uno a uno manualmente o mediante un comando que nos ofrece Angular, nosotros vamos a optar por la segunda opción, ya que es más rápida. El comando que tenemos que ejecutar es el siguiente:

```
ng generate component games //podemos reducirlo a “ng g c games”
```

Si todo ha ido bien debería aparecer algo como esto:

```
PS C:\Users\Carlo\Desktop\Angular\tutorial-angular> ng generate component games
CREATE src/app/games/games.component.html (20 bytes)
CREATE src/app/games/games.component.spec.ts (619 bytes)
CREATE src/app/games/games.component.ts (271 bytes)
CREATE src/app/games/games.component.css (0 bytes)
UPDATE src/app/app.module.ts (471 bytes)
```

Y podremos comprobar que se ha creado un nuevo componente dentro de la carpeta “app” de nuestro proyecto. También se ha importado de manera automática en todos los archivos necesarios para que este componente funcione, es decir donde hemos mostrado en el apartado anterior con el componente base.

Para el ejercicio vamos a crear otro componente de la misma manera, pero en este caso se llamará “game”:

```
PS C:\Users\Carlo\Desktop\Angular\tutorial-angular> ng generate component game
CREATE src/app/game/game.component.html (19 bytes)
CREATE src/app/game/game.component.spec.ts (612 bytes)
CREATE src/app/game/game.component.ts (267 bytes)
CREATE src/app/game/game.component.css (0 bytes)
UPDATE src/app/app.module.ts (545 bytes)
```

Ahora que ya tenemos los dos componentes, vamos a comenzar con el pequeño ejercicio.

Como hemos mostrado anteriormente, en nuestro “index.html” llamábamos al componente base mediante la etiqueta <app-root>, vamos a aprovechar esto para realizar nuestro ejercicio. En primer lugar, borramos todo el contenido de “app.component.html”. Una vez borrado todo el contenido, introducimos lo siguiente:

```
<h1 align="center">{{title}}</h1>
```

Con las llaves “{{title}}” estamos haciendo referencia a la variable “title” que se encuentra en el archivo “app.component.ts”:

```
import { Component } from '@angular/c

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

Si ahora iniciamos nuestro proyecto, en nuestro navegador debería aparecer lo siguiente:

Tutorial-angular

Ahora nos dirigiremos al html del componente “games”, borramos el contenido que viene por defecto e introduciremos lo siguiente:

```
<h2 align="center">Listado de juegos</h2>
```

Y volvemos a “app.component.html” e introducimos la etiqueta que invoca al componente games:

```
<h1 align="center">{{title}}</h1>

<app-games></app-games>
```

Tras realizar esto, si nos dirigimos a nuestro navegador, veremos que la aplicación ha cambiado y el resultado es el siguiente:

Tutorial-angular

Listado de juegos

Por último, nos dirigimos al html del componente “game” e introducimos lo siguiente:

```
<h3 align="center">Juego</h3>
```

Y por último, llamamos a la etiqueta que invoca al componente desde el html del componente “games”:

```
<h2 align="center">Listado de juegos</h2>  
<app-game></app-game>
```

Si volvemos al navegador, veremos que nuestra aplicación ha vuelto a cambiar:

Tutorial-angular

Listado de juegos

Juego







Con este pequeño ejercicio hemos aprendido cómo crear nuevos componentes y cómo llamar a unos componentes desde otros, conociendo un poco mejor cómo funciona el flujo de Angular.

A partir de ahora realizaremos una aplicación un poco más elaborada, que nos servirá para entender mejor el funcionamiento de Angular.

6. Aplicación Básica en Angular

En esta sección vamos a realizar una aplicación básica que consistirá en una Pokédex básica haciendo llamadas a la famosa Poké API. Para Ello utilizaremos los conceptos básicos explicados con anterioridad. Cabe destacar que se explicara un poco por encima debido a que abordaremos una explicación más profunda en el video.

Vista de la página :


PokéDex			
ID pokemon	Imagen	Nombre	
1		bulbasaur	Ver Pokemon
2		ivysaur	Ver Pokemon
3		venusaur	Ver Pokemon
4		charmander	Ver Pokemon
5		chameleon	Ver Pokemon
6		charizard	Ver Pokemon

Información de cada pokemon:

PokéDex

bulbasaur

ID: 1



Tipo:
grasa

Altura:
0.7 cm

Peso:
6.9 kg

Habilidad:
overgrow

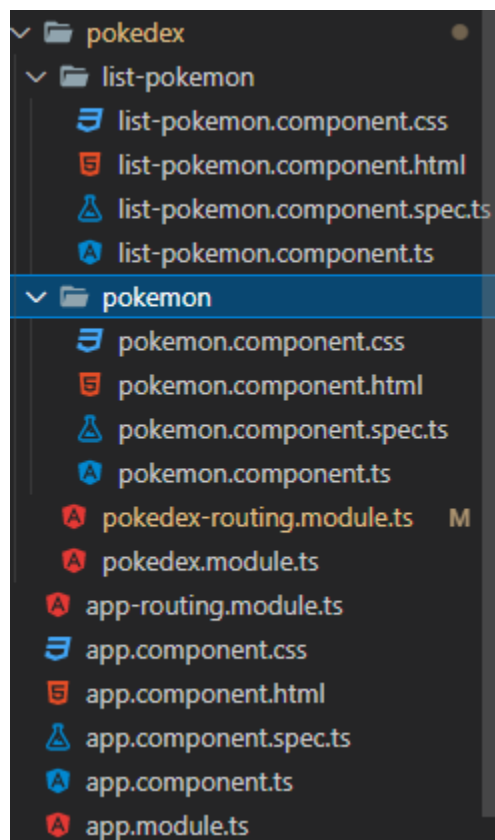
[Volver a Lista](#)

Tras iniciar el proyecto de angular , hemos creado una módulo servicio llamado pokédex. En este módulo guardaremos todos los componentes y servicios que utilizaremos en este proyecto.

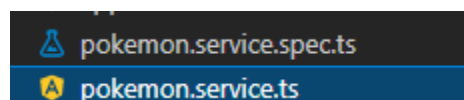
Para ello hemos utilizado el comando :

ng generate module “nombre carpeta/nombre” –module app –flat –routing

De esta manera se nos crea el módulo y se añade al módulo principal del proyecto app-module.. Después pasamos a crear cada componente. Creamos un componente para listar los pokemon y otro para la información detallada de cada uno. Tras esto se nos queda esto en nuestro proyecto base :



Ahora creamos el servicio para obtener los datos de la api Poké Api y creamos sus distintos métodos:



PokemonService:

```
@Injectable({
  providedIn: 'root'
})
export class PokemonService {

  constructor() {
  }

  getAllPokemon(){
    return axios.get("https://pokeapi.co/api/v2/pokemon?offset=0&limit=251")
  }

  getPokemon(index: any){
    return axios.get(`https://pokeapi.co/api/v2/pokemon/${index}`)
  }

  async getpokemonImages(listimages: any){
    for(let i = 1; i <= 251; i++){
      await this.getPokemon(i).then(resp => {
        listimages.push(resp.data.sprites.front_default)
        console.log(resp.data.sprites.front_default)
      })
    }
  }
}
```

Estas son las únicas peticiones que necesitaremos. Tras esto modificamos el html de cada componente para darle la estructura deseada y modificamos su .ts para darle la lógica necesaria.

ListPokemonComponent:

```
import { Component, OnInit } from '@angular/core';
import { PokemonService } from '../..//pokemon.service';
import { Router } from '@angular/router';
@Component({
  selector: 'app-list-pokemon',
  templateUrl: './list-pokemon.component.html',
  styleUrls: ['./list-pokemon.component.css']
})
export class ListPokemonComponent implements OnInit {

  listpokemon:Array<any>=[];
  listimages:Array<any>=[];

  constructor(private pokemon_service:PokemonService,private router: Router) {
    this.pokemon_service
  }

  ngOnInit(): void {
    this.pokemon_service.getAllPokemon().then(resp=>{
      this.listpokemon=resp.data.results;
    })
    this.pokemon_service.getpokemonImages(this.listimages)
  }
  pokemonInfo(index:number){
    this.router.navigateByUrl('pokemon/'+index);
  }
}
```

Aquí declaramos las variables que necesitaremos . Una para guardar todos los pokemon y la otra para guardar la imagen de los pokemon. Podemos observar que la información de cada variable la obtenemos de los métodos del servicio Pokemon Service creado con anterioridad. El constructor inicializa las variables de angular para poder utilizarlas . El método ngOnInit() es el primer método que se ejecuta cuando se carga el componente, por eso hemos inicializado las variables que vamos a utilizar allí.

Finalmente el método pokemon Info simplemente redirige a la ruta del pokémon seleccionado.

Ahora creamos las rutas de nuestra página para poder cargar cada componente . Para ello modificamos **pokedex-routing.module.ts**

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { PokemonComponent } from '../pokemon/pokemon.component';
import { ListPokemonComponent } from '../list-pokemon/list-pokemon.component';
const routes: Routes = [
  {
    path: '',
    component: ListPokemonComponent
  },
  {
    path: 'pokemon/:id',
    component: PokemonComponent
  }
];
@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class PokedexRoutingModule { }
```

En nuestro caso solo hemos necesitado establecer la ruta principal y la ruta del componente. Y con esto ya tendremos nuestro proyecto funcionando. Como se puede observar el funcionamiento de angular es bastante parecido a vue, puesto que ambos trabajan con el sistema de componentes.



Bibliografía

[**https://docs.angular.lat/**](https://docs.angular.lat/)

[**https://desarrolloweb.com/articulos/servicios-angular.html**](https://desarrolloweb.com/articulos/servicios-angular.html)

[**https://unprogramador.com/services-en-angular-que-son-y-para-que-funcionan/**](https://unprogramador.com/services-en-angular-que-son-y-para-que-funcionan/)

[**https://unprogramador.com/services-en-angular-que-son-y-para-que-funcionan/**](https://unprogramador.com/services-en-angular-que-son-y-para-que-funcionan/)

[**https://ngchallenges.gitbook.io/project/componentes**](https://ngchallenges.gitbook.io/project/componentes)