

## Taller 7

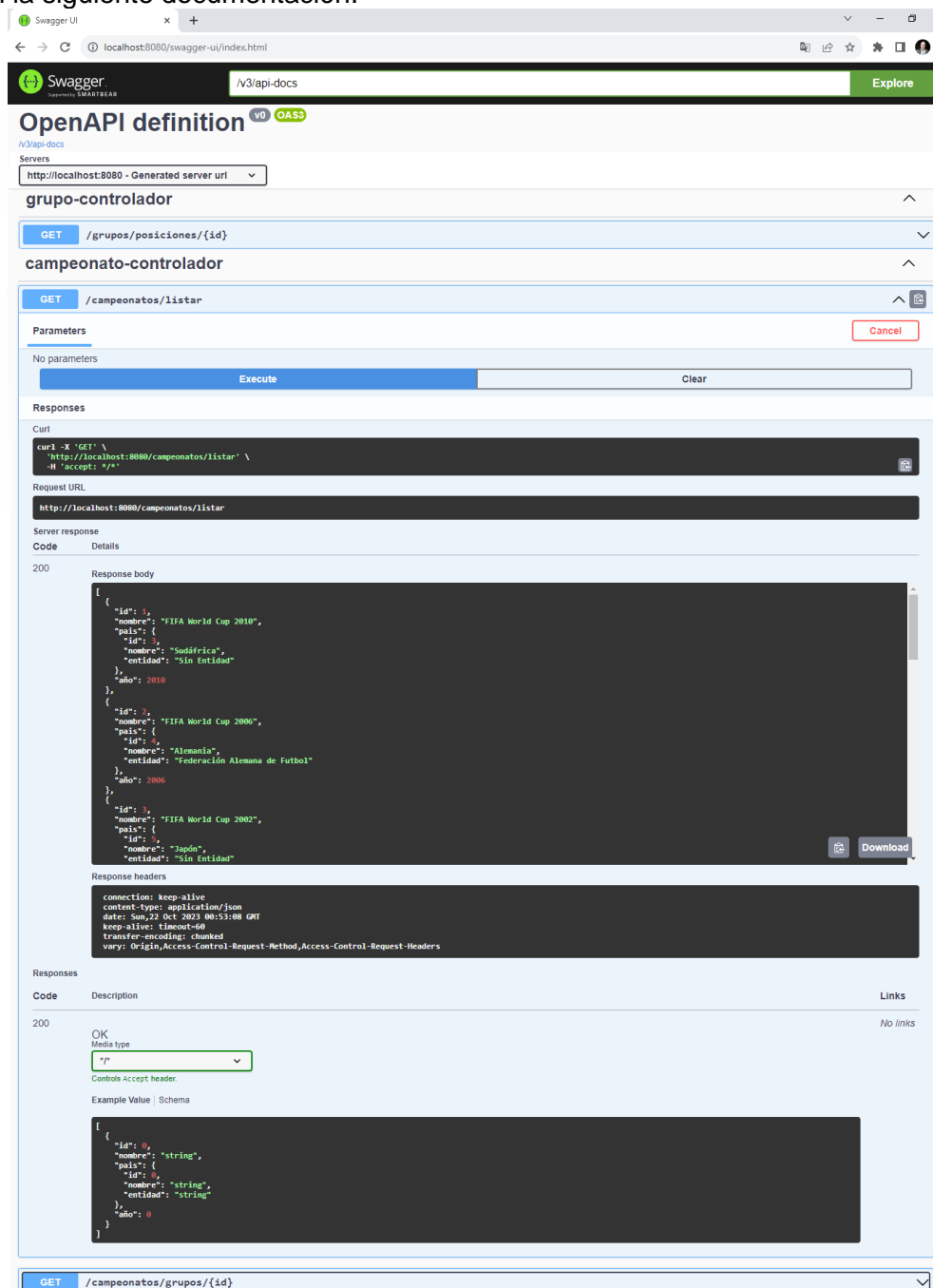
Fecha: Abril de 2024

**Indicador de logro a medir:** Aplicar los conceptos de la lógica de programación, de la orientación a objetos, de la orientación a servicios y de los patrones arquitectónicos en el desarrollo de una aplicación para Internet que acceda una base de datos relacional.

### NOTAS:

- Este taller se debe hacer con carácter evaluativo (punto No 2). Representa en total una calificación de 15%.
- Los primeros ejercicios se entregan resueltos como ejemplo para el desarrollo de los demás

1. Elaborar un aplicativo cliente Web en *Angular CLI* para una API RESTfull en *Spring Boot* que cumple con la siguiente documentación:



The screenshot displays the Swagger UI for an API. The top navigation bar shows the Swagger logo and the API path `/v3/api-docs`. Below this, the "OpenAPI definition" section is visible, showing the API's servers and endpoints. The endpoint `GET /campeonatos/listar` is selected, and its details are shown. The response body is a JSON array of tournament data, including fields like `id`, `nombre`, `pais`, `entidad`, and `año`. The response headers and status code (200) are also visible.

```
{
  "id": 1,
  "nombre": "FIFA World Cup 2010",
  "pais": {
    "id": 1,
    "nombre": "Sudáfrica",
    "entidad": "Sin Entidad"
  },
  "año": 2010
},
{
  "id": 2,
  "nombre": "FIFA World Cup 2006",
  "pais": {
    "id": 2,
    "nombre": "Alemania",
    "entidad": "Federación Alemana de Fútbol"
  },
  "año": 2006
},
{
  "id": 3,
  "nombre": "FIFA World Cup 2002",
  "pais": {
    "id": 3,
    "nombre": "Japón",
    "entidad": "Sin Entidad"
  },
  "año": 2002
}
```

- Un método que devuelve la lista de campeonatos FIFA que están registrados en la base de datos

Swagger v3/api-docs Explore

**OpenAPI definition** v0 QA53

V3/api-docs

Servers  
http://localhost:8080 - Generated server url

**grupo-controlador**

**GET** /grupos/posiciones/{id}

**campeonato-controlador**

**GET** /campeonatos/listar

**GET** /campeonatos/grupos/{id}

**Parameters** Cancel

Name	Description
<b>id</b> * required integer (int64) (path)	10

Execute Clear

**Responses**

**Curl**

```
curl -X 'GET' \
  'http://localhost:8080/campeonatos/grupos/10' \
  -H 'accept: */*'
```

**Request URL**

```
http://localhost:8080/campeonatos/grupos/10
```

**Server response**

Code	Details
200	<p><b>Response body</b></p> <pre>{   "id": 19,   "campeonato": {     "id": 10,     "nombre": "FIFA World Cup 2018",     "pais": {       "id": 49,       "nombre": "Rusia",       "entidad": "Sin Entidad"     },     "año": 2018   },   "nombre": "A" }, {   "id": 20,   "campeonato": {     "id": 49,     "nombre": "FIFA World Cup 2018",     "pais": {       "id": 69,       "nombre": "Rusia",       "entidad": "Sin Entidad"     },     "año": 2018   },   "nombre": "B" } }</pre> <p><span>Download</span></p> <p><b>Response headers</b></p> <pre>connection: keep-alive content-type: application/json date: Sun, 22 Oct 2023 01:16:07 GMT keep-alives: timeout=60 transfer-encoding: chunked vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers</pre>

**Responses**

Code	Description	Links
200	OK	No links

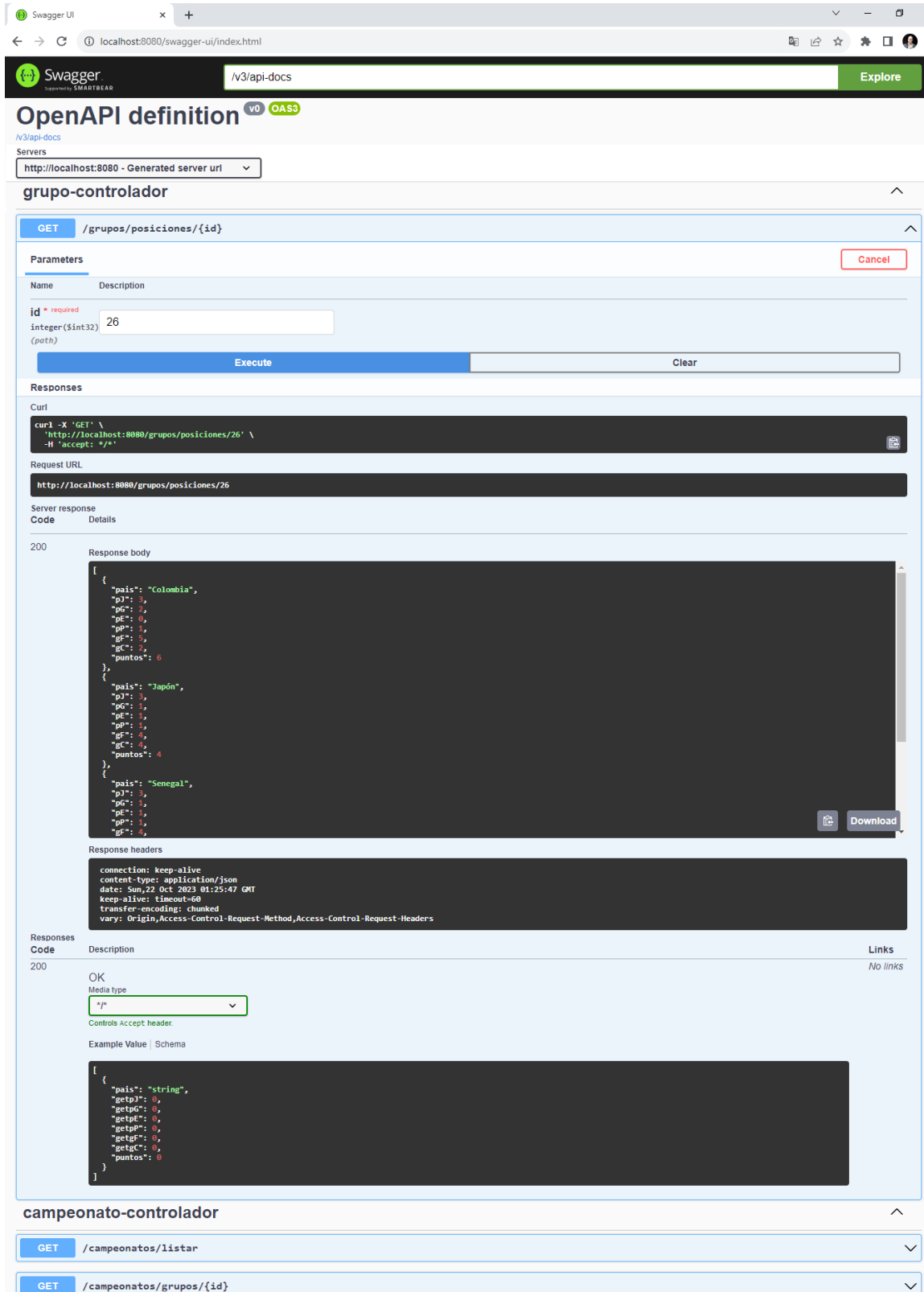
**Media type**  
\*/\*

Controls Accept header.

Example Value | Schema

```
{
  "id": 0,
  "campeonato": {
    "id": 0,
    "nombre": "string",
    "pais": {
      "id": 0,
      "nombre": "string",
      "entidad": "string"
    },
    "año": 0
  },
  "nombre": "string"
}
```

- Un método que, dado un campeonato, devuelve la lista de los grupos registrados



The screenshot displays the Swagger UI interface for an API. The top navigation bar shows the Swagger logo and the API version 'v3/api-docs'. The main heading is 'OpenAPI definition' with version 'v0' and 'OAS3' tags. Below this, the 'Servers' section lists 'http://localhost:8080 - Generated server url'. The selected API is 'grupo-controlador'. The 'GET /grupos/posiciones/{id}' endpoint is highlighted. The 'Parameters' section shows a required integer parameter 'id' with a value of '26'. The 'Responses' section shows a '200' status code with a JSON response body containing an array of group data. The response body is displayed in a dark-themed editor with a 'Download' button. The 'Response headers' section shows headers like 'connection: keep-alive', 'content-type: application/json', and 'date: Sun, 22 Oct 2023 01:25:47 GMT'. The 'Responses' table at the bottom shows a '200' status code with a description 'OK' and a media type of '\*//\*'. The 'Example Value' section shows a JSON object with group details.

Swagger UI interface showing the API definition for 'grupo-controlador'.

The API endpoint is **GET /grupos/posiciones/{id}**.

The parameter **id** is required and is an integer (int32).

The response is a 200 status code with a JSON body containing an array of group data.

The response body is displayed in a dark-themed editor with a 'Download' button.

The response headers are:

- connection: keep-alive
- content-type: application/json
- date: Sun, 22 Oct 2023 01:25:47 GMT
- keep-alive: timeout=60
- transfer-encoding: chunked
- vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers

The response is a 200 status code with a description 'OK' and a media type of '\*//\*'.

The response body is displayed in a dark-themed editor with a 'Download' button.

The response body is:

```
{
  "pais": "Colombia",
  "pJ": 3,
  "pG": 0,
  "pE": 0,
  "pP": 1,
  "pS": 0,
  "pC": 0,
  "puntos": 0
},
{
  "pais": "Japón",
  "pJ": 3,
  "pG": 1,
  "pE": 1,
  "pP": 1,
  "pS": 0,
  "pC": 0,
  "puntos": 4
},
{
  "pais": "Senegal",
  "pJ": 3,
  "pG": 1,
  "pE": 1,
  "pP": 1,
  "pS": 0,
  "pC": 0,
  "puntos": 0
}
```

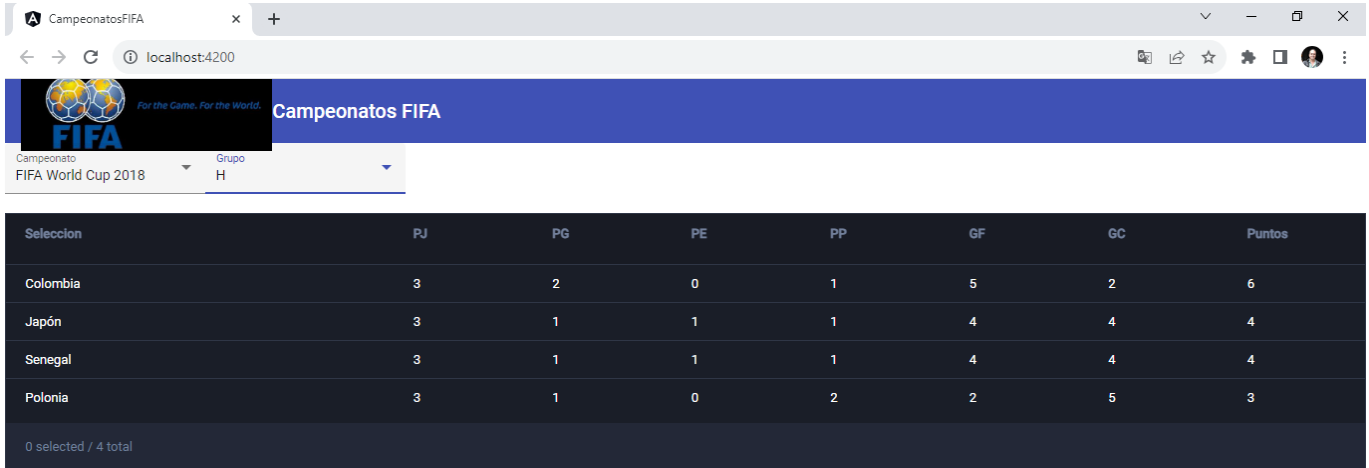
The API endpoint is **GET /campeonatos/listar**.

The API endpoint is **GET /campeonatos/grupos/{id}**.

- Un método que, dado un grupo, devuelve la tabla de posiciones

El aplicativo web debe permitir:

- Listar los campeonatos en una lista desplegable y permitir seleccionar uno
- Cuando se seleccione un campeonato, listar los grupos del campeonato seleccionado
- Permitir seleccionar uno de los grupos listados y mostrar la respectiva tabla de posiciones



Selección	PJ	PG	PE	PP	GF	GC	Puntos
Colombia	3	2	0	1	5	2	6
Japón	3	1	1	1	4	4	4
Senegal	3	1	1	1	4	4	4
Polonia	3	1	0	2	2	5	3

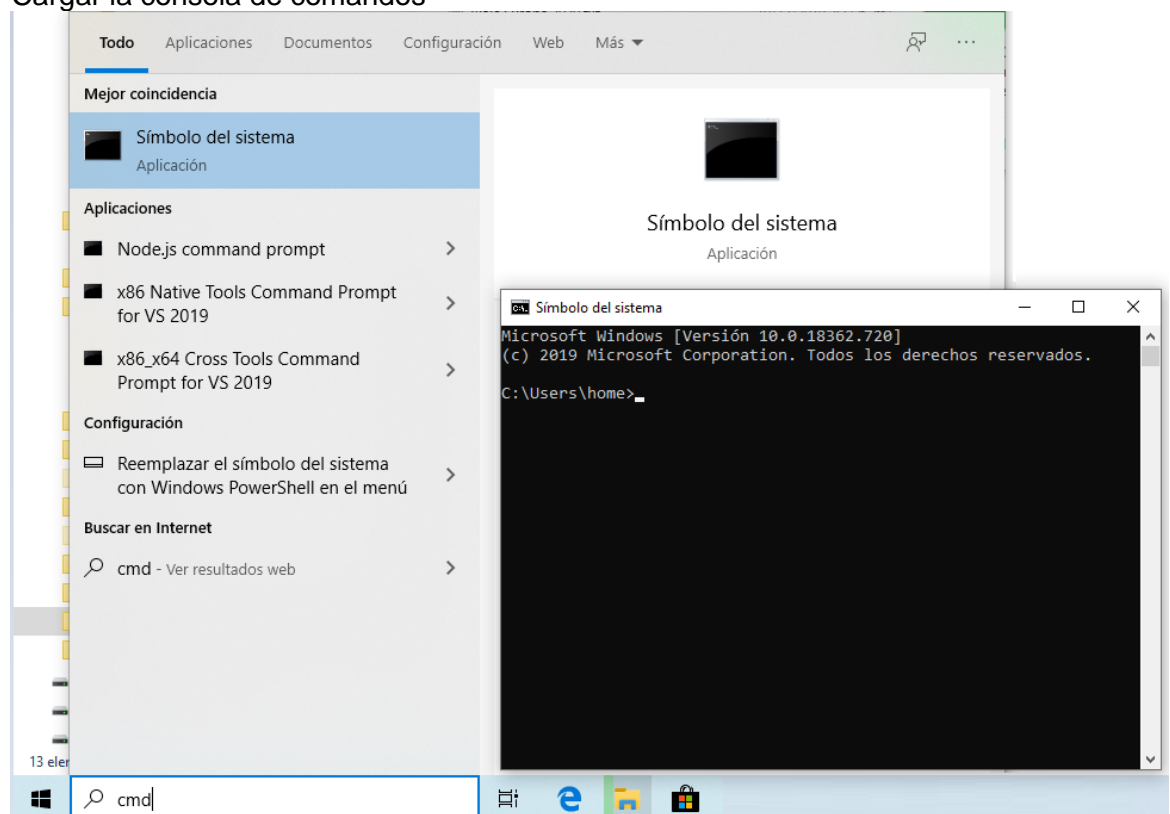
0 selected / 4 total


R/

Para este aplicativo web se tendrán las siguientes consideraciones:

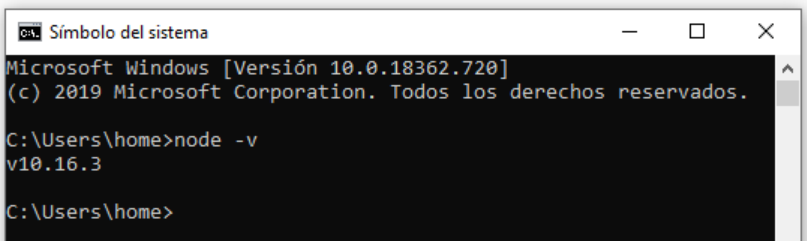
## 1. Instalación del ambiente de desarrollo para implementar aplicaciones *Angular*

### a) Cargar la consola de comandos



- b) Verificar que esté instalado **Node.JS** ( es un entorno de ejecución de *JavaScript* del lado del servidor que utiliza un modelo asíncrono y dirigido por eventos. Es una Máquina Virtual muy rápida). En caso contrario, se debe descargar e instalar primero:


```
node -v
```



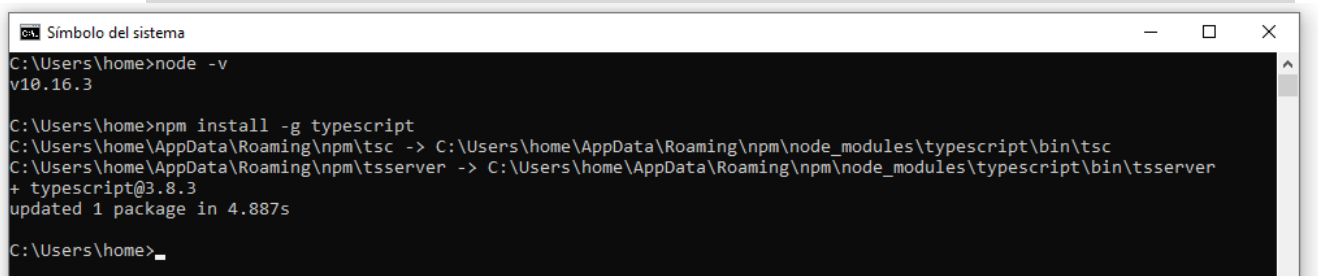
```
Microsoft Windows [Versión 10.0.18362.720]
(c) 2019 Microsoft Corporation. Todos los derechos reservados.

C:\Users\home>node -v
v10.16.3

C:\Users\home>
```

- c) Instalar **TypeScript** ( es un lenguaje de programación de código abierto que convierte lo codificado en *JavaScript*, por lo que es un Superset de *JavaScript*)


```
npm install -g typescript
```



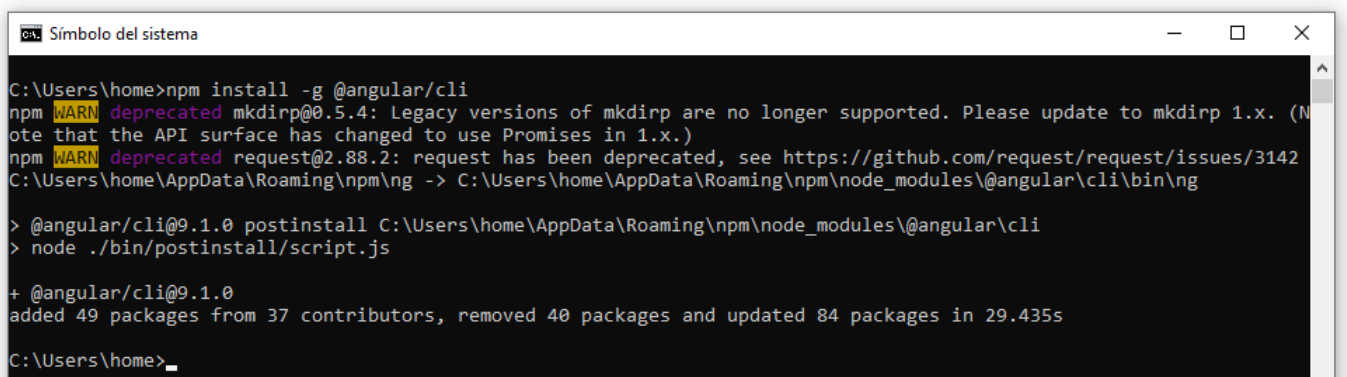
```
C:\Users\home>node -v
v10.16.3

C:\Users\home>npm install -g typescript
C:\Users\home\AppData\Roaming\npm\tsc -> C:\Users\home\AppData\Roaming\npm\node_modules\typescript\bin\tsc
C:\Users\home\AppData\Roaming\npm\tsserver -> C:\Users\home\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
+ typescript@3.8.3
updated 1 package in 4.887s

C:\Users\home>
```

- d) Instalar **Angular CLI** ( Command Line Interface *Angular CLI* es una herramienta *Node.JS* que facilita el inicio de proyectos y la creación del esqueleto - **Scaffolding** -de la mayoría de los componentes de una aplicación *Angular*)

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



```
C:\Users\home>npm install -g @angular/cli
npm WARN deprecated mkdirp@0.5.4: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note that the API surface has changed to use Promises in 1.x.)
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142
C:\Users\home\AppData\Roaming\npm\ng -> C:\Users\home\AppData\Roaming\npm\node_modules\@angular\cli\bin\ng

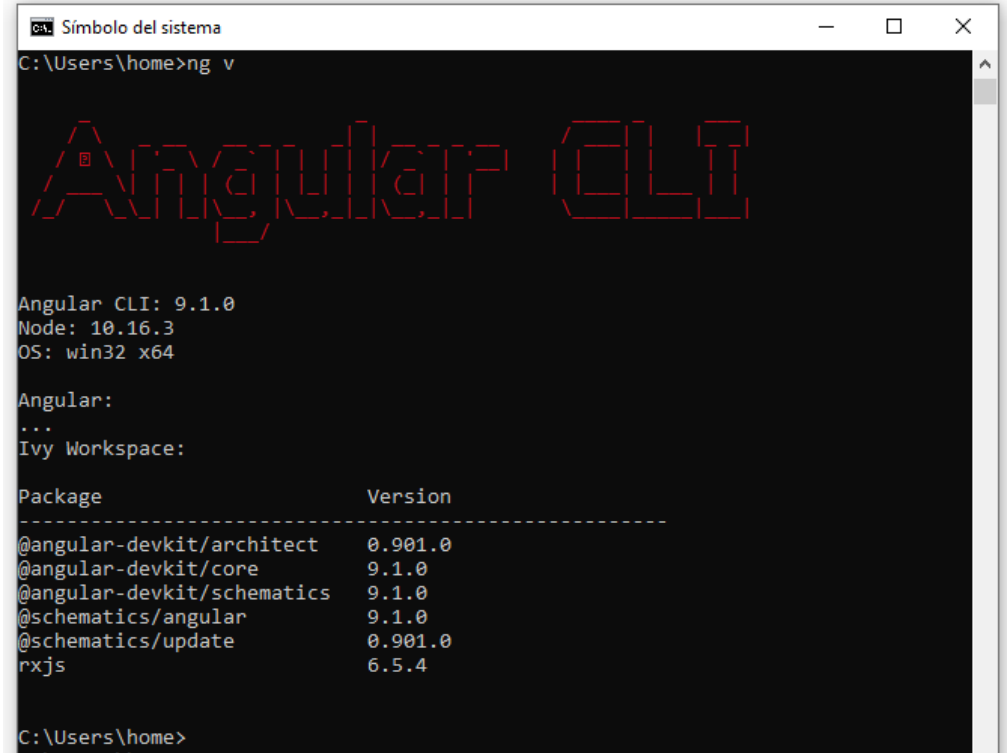
> @angular/cli@9.1.0 postinstall C:\Users\home\AppData\Roaming\npm\node_modules\@angular\cli
> node ./bin/postinstall/script.js

+ @angular/cli@9.1.0
added 49 packages from 37 contributors, removed 40 packages and updated 84 packages in 29.435s

C:\Users\home>
```

e) Verificar que esté instalado *Angular*

ng v



```
Símbolo del sistema
C:\Users\home>ng v

Angular CLI
Angular CLI: 9.1.0
Node: 10.16.3
OS: win32 x64

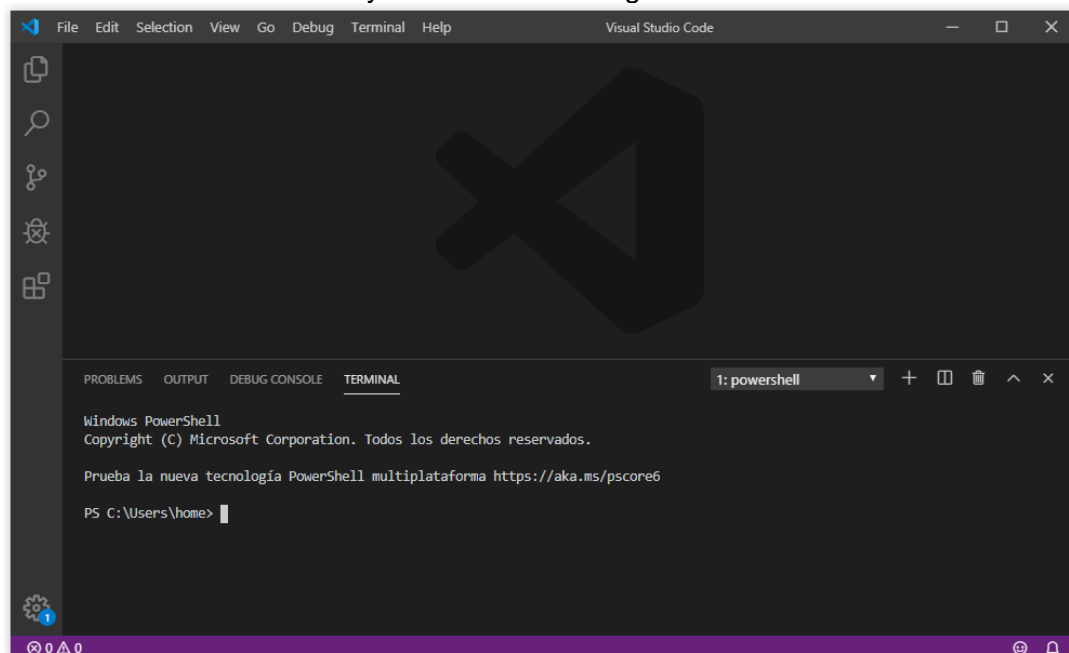
Angular:
...
Ivy Workspace:

Package                       Version
-----
@angular-devkit/architect     0.901.0
@angular-devkit/core          9.1.0
@angular-devkit/schematics     9.1.0
@schematics/angular           9.1.0
@schematics/update             0.901.0
rxjs                           6.5.4

C:\Users\home>
```

2. Creación de un proyecto en *Angular*.

a) Iniciar **Visual Studio Code** y con CTRL + Ñ cargar la consola de comandos



b) Crear el proyecto una vez ubicado en la carpeta de destino

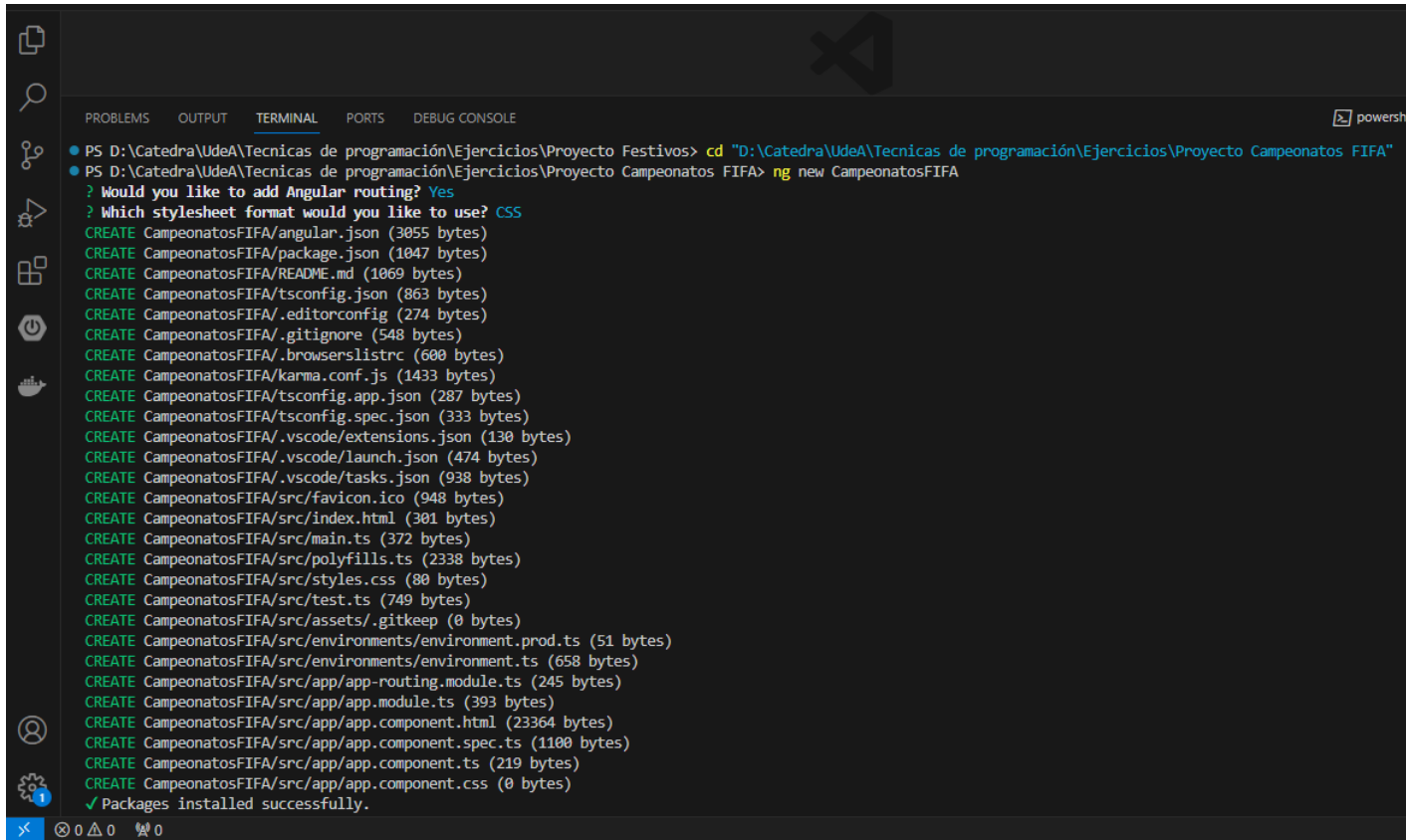
```
ng new nombre_proyecto
```

En la ejecución de esta instrucción, es *Angular CLI* quien crea el siguiente Scaffolding (andamiaje) de la aplicación (para lograr una estructura apropiada para el proyecto y facilitar el trabajo en equipo, ya que siempre se mantiene la misma estructura):

```
.
├── e2e/
├── node_modules/
├── src/
├── .editorconfig
├── .gitignore
├── angular-cli.json
├── package.json
├── tslint.json
├── karma.conf.js
├── protractor.conf.js
└── README.md
```

Elemento	Descripción
<b>editorconfig</b>	Este archivo ayuda a los desarrolladores a definir y mantener estilos de codificación uniformes entre los diferentes editores y entornos de desarrollo. En caso, <i>Angular CLI</i> proporciona unos estilos de codificación por defecto para todos los archivos([*]) y para los archivos en Markdown([*.md])

<b>node_modules y package.json</b>	Esta carpeta contiene todos los paquetes descargados mediante el gestor de paquetes de <i>JavaScript</i> , <b>npm</b> . Los paquetes incluidos en esta carpeta se encuentran definidos en el archivo <i>package.json</i> , dentro de <i>dependencies</i> y <i>devDependencies</i> , acompañados de la versión del paquete instalada. En este archivo también tenemos información general del proyecto, como por ejemplo, los script que podemos ejecutar, el nombre del proyecto, licencia, versión, etc
<b>gitignore</b>	Especifica que carpetas o archivos tiene que ignorar el sistema de control de versiones <b>Git</b>
<b>README.md</b>	Contiene información básica sobre los comandos que podemos ejecutar
<b>karma.conf.js y protractor.conf.js</b>	Dan información a <b>Karma</b> y <b>Protractor</b> para poder ejecutar pruebas del proyecto. La herramienta de testeo en Angular por defecto es <b>Jasmine</b> . Estas herramientas permiten hacer pruebas unitarias que son rápidas y permiten probar partes aisladas del código, mientras que las de extremo a extremo son lentas y comprueban el sistema entero.
<b>tslint.json</b>	Angular CLI proporciona un <b>linter</b> (utilidad para reparar automáticamente cientos de problemas), que se puede ejecutar con el comando <code>ng lint</code> y que tiene su configuración en este archivo. Este contiene especificadas una serie de reglas para que el código <i>Typescript</i> mantenga una estructura y pueda ser más entendible y fácil de mantener
<b>angular-cli.json</b>	Aquí se encuentra la información que <i>Angular CLI</i> usa para saber por qué archivo empezar la ejecución de la aplicación, pero también incluye información del proyecto, de las apps que lo componen y de lo que usan esas apps para funcionar



```

PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Festivos> cd "D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA"
PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA> ng new CampeonatosFIFA
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE CampeonatosFIFA/angular.json (3055 bytes)
CREATE CampeonatosFIFA/package.json (1047 bytes)
CREATE CampeonatosFIFA/README.md (1069 bytes)
CREATE CampeonatosFIFA/tsconfig.json (863 bytes)
CREATE CampeonatosFIFA/.editorconfig (274 bytes)
CREATE CampeonatosFIFA/.gitignore (548 bytes)
CREATE CampeonatosFIFA/.browserslistrc (600 bytes)
CREATE CampeonatosFIFA/karma.conf.js (1433 bytes)
CREATE CampeonatosFIFA/tsconfig.app.json (287 bytes)
CREATE CampeonatosFIFA/tsconfig.spec.json (333 bytes)
CREATE CampeonatosFIFA/.vscode/extensions.json (130 bytes)
CREATE CampeonatosFIFA/.vscode/launch.json (474 bytes)
CREATE CampeonatosFIFA/.vscode/tasks.json (938 bytes)
CREATE CampeonatosFIFA/src/favicon.ico (948 bytes)
CREATE CampeonatosFIFA/src/index.html (301 bytes)
CREATE CampeonatosFIFA/src/main.ts (372 bytes)
CREATE CampeonatosFIFA/src/polyfills.ts (2338 bytes)
CREATE CampeonatosFIFA/src/styles.css (80 bytes)
CREATE CampeonatosFIFA/src/test.ts (749 bytes)
CREATE CampeonatosFIFA/src/assets/.gitkeep (0 bytes)
CREATE CampeonatosFIFA/src/environments/environment.prod.ts (51 bytes)
CREATE CampeonatosFIFA/src/environments/environment.ts (658 bytes)
CREATE CampeonatosFIFA/src/app/app-routing.module.ts (245 bytes)
CREATE CampeonatosFIFA/src/app/app.module.ts (393 bytes)
CREATE CampeonatosFIFA/src/app/app.component.html (23364 bytes)
CREATE CampeonatosFIFA/src/app/app.component.spec.ts (1100 bytes)
CREATE CampeonatosFIFA/src/app/app.component.ts (219 bytes)
CREATE CampeonatosFIFA/src/app/app.component.css (0 bytes)
✓ Packages installed successfully.
  
```

Dentro de la carpeta `e2e` se incluye lo siguiente:

	Elemento	Descripción
<pre> . ├── e2e/ │   ├── app.po.ts │   ├── app.e2e-spec.ts │   └── tsconfig.json </pre>	<b><code>app.po.ts</code></b> y <b><code>app.e2e-spec.ts</code></b>	Se tiene un test básico de <i>Angular-CLI</i> que comprueba que se muestra correctamente la frase <i>"app works!"</i>
<b><code>tsconfig.json</code></b>	Indica que estamos en un proyecto <i>Typescript</i> y especifica las opciones del compilador y los archivos que necesita para poder compilar el proyecto.	



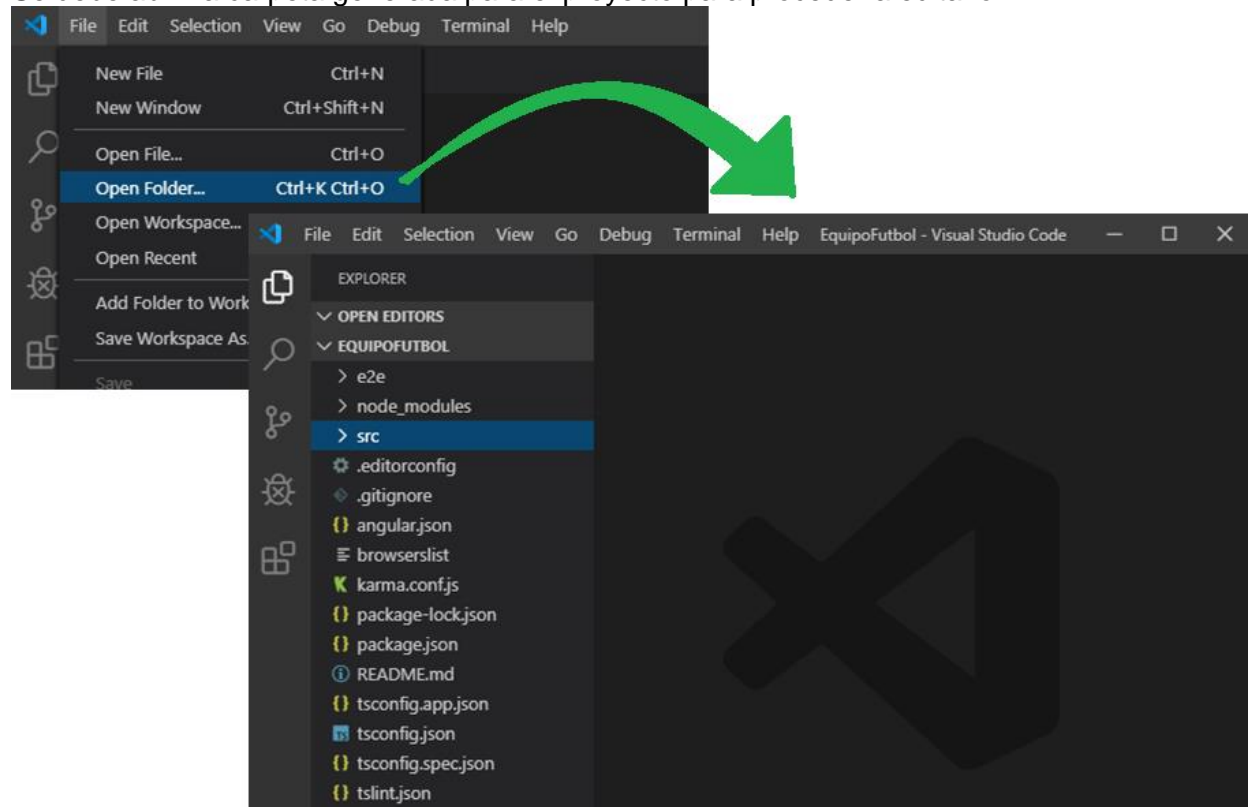
Y finalmente, dentro de la carpeta *src*, la cual contiene la aplicación, se tiene:

```

.
├── src/
│   ├── app/
│   │   ├── app.component.html
│   │   ├── app.component.css
│   │   ├── app.component.spec.ts
│   │   ├── app.component.ts
│   │   ├── app.module.ts
│   │   └── index.ts
│   ├── assets/
│   ├── .gitkeep
│   ├── enviroments/
│   │   ├── environments.ts
│   │   └── environments.prod.ts
│   ├── favicon.ico
│   ├── index.html
│   ├── main.ts
│   ├── polyfills.ts
│   ├── styles.css
│   ├── test.ts
│   ├── tsconfig.json
│   └── typings.d.ts
  
```

Elemento	Descripción
<b><i>index.html</i></b>	Es la página <i>HTML</i> principal de la aplicación
<b><i>style.css</i></b>	Establece los estilos generales para la aplicación
<b><i>app</i></b>	Carpeta que contiene el código de todos los componentes. En esta carpeta es donde <i>Angular CLI</i> facilita el scaffolding al aportar una serie de comandos ( <i>ng generate</i> ) para la creación automática de los componentes, filtros, servicios, etc. que componen la aplicación y manteniendo siempre la misma estructura.

c) Se debe abrir la carpeta generada para el proyecto para proceder a editarlo

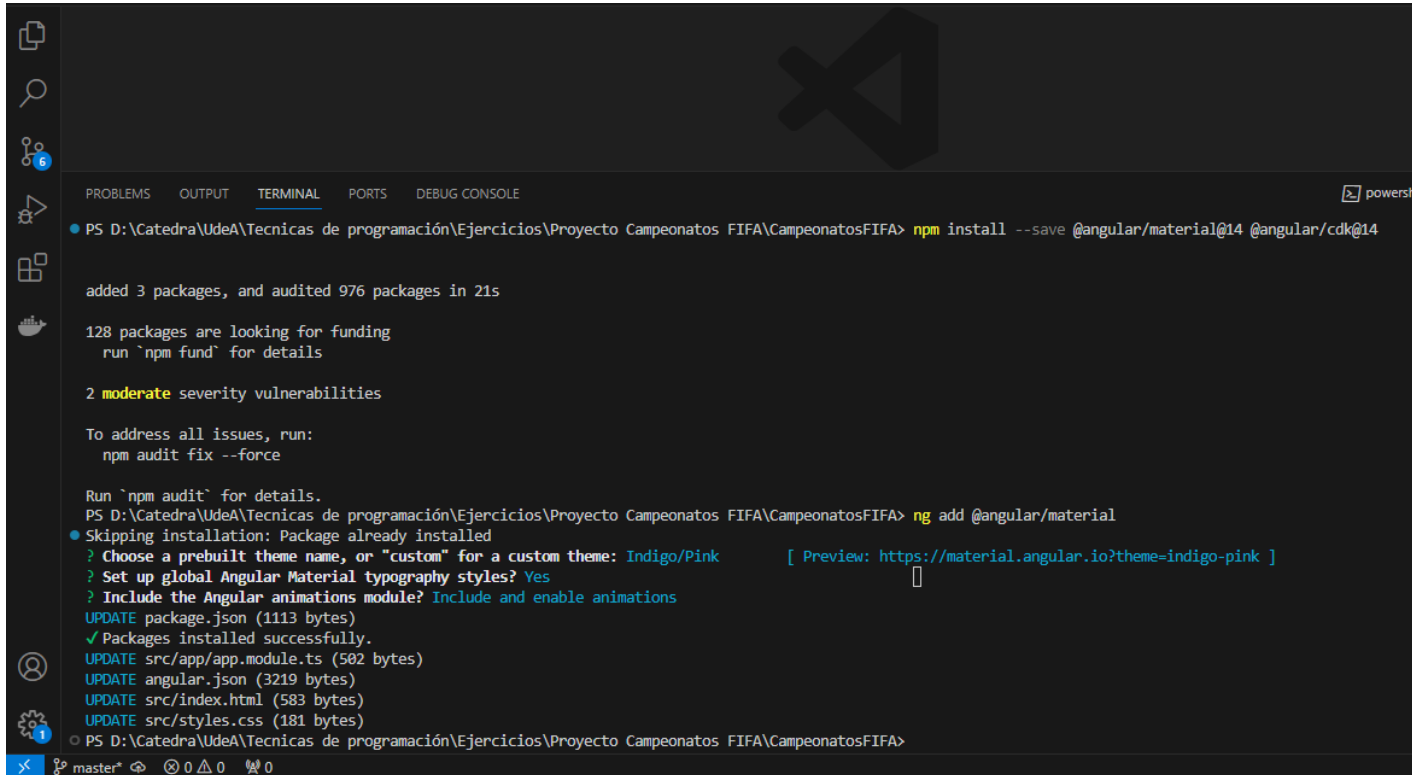


d) Agregar al proyecto los módulos **NodeJS**

Por ejemplo, para instalar la librería *Angular Material*, se tendrían los siguientes comandos:

```
npm install --save @angular/material @angular/cdk
ng add @angular/material
```

El primero instala el paquete y el segundo lo agrega al proyecto



```

PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA> npm install --save @angular/material@14 @angular/cdk@14

added 3 packages, and audited 976 packages in 21s

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

2 moderate severity vulnerabilities

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA> ng add @angular/material
• Skipping installation: Package already installed
? Choose a prebuilt theme name, or "custom" for a custom theme: Indigo/Pink    [ Preview: https://material.angular.io?theme=indigo-pink ]
? Set up global Angular Material typography styles? Yes
? Include the Angular animations module? Include and enable animations
UPDATE package.json (1113 bytes)
✓ Packages installed successfully.
UPDATE src/app/app.module.ts (502 bytes)
UPDATE angular.json (3219 bytes)
UPDATE src/index.html (583 bytes)
UPDATE src/styles.css (181 bytes)
PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA>
  
```

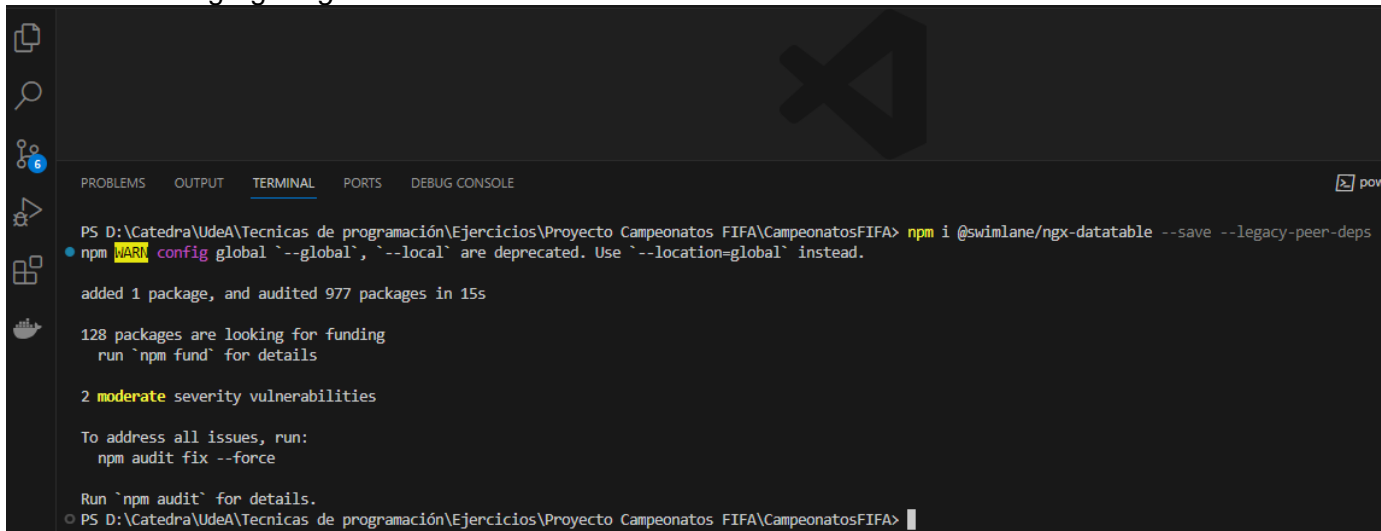
**Angular Material** es una biblioteca de diseño y componentes de interfaz de usuario desarrollada por Google para la creación de aplicaciones web y móviles con un diseño moderno y coherente. Esta biblioteca proporciona una amplia gama de elementos visuales y funcionalidades predefinidas, como botones, cuadros de diálogo, barras de navegación y más, que siguen las pautas de diseño de Material Design. Angular Material facilita la creación de aplicaciones atractivas y funcionales al ofrecer componentes reutilizables y una apariencia consistente, lo que agiliza el proceso de desarrollo y mejora la experiencia del usuario.

Otros módulos muy comúnmente utilizados son:

Módulo	Descripción	Instrucción
<b>Bootstrap</b>	Es un framework CSS y Javascript diseñado para la creación de interfaces limpias y con un diseño responsive	<code>npm install bootstrap</code>
<b>Awesome</b>	<b>FontAwesome</b> es una biblioteca CSS donde se tiene una gran cantidad de	<code>npm install font-awesome</code>

	iconos vectoriales para utilizar en las aplicaciones	
<b>ngx-datatable</b>	Es una biblioteca que ofrece una manera eficaz de presentar y administrar datos tabulares en aplicaciones web, con una amplia gama de características personalizables, como filtrado, clasificación, paginación y más. Además, es altamente responsiva	<pre>npm i @swimlane/ngx-datatable --save --legacy-peer-deps</pre>

Para agregar *ngx-datatable*:



```

PS D:\Catedra\UdeA\Técnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA> npm i @swimlane/ngx-datatable --save --legacy-peer-deps
npm WARN config global '--global', '--local' are deprecated. Use '--location=global' instead.

added 1 package, and audited 977 packages in 15s

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

2 moderate severity vulnerabilities

To address all issues, run:
  npm audit fix --force

Run `npm audit` for details.
PS D:\Catedra\UdeA\Técnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA>
  
```

Ahora, para poder editar el aplicativo, es importante conocer un poco sobre la arquitectura de Angular y abordar introductoriamente que son los Módulos, Componentes y Servicios

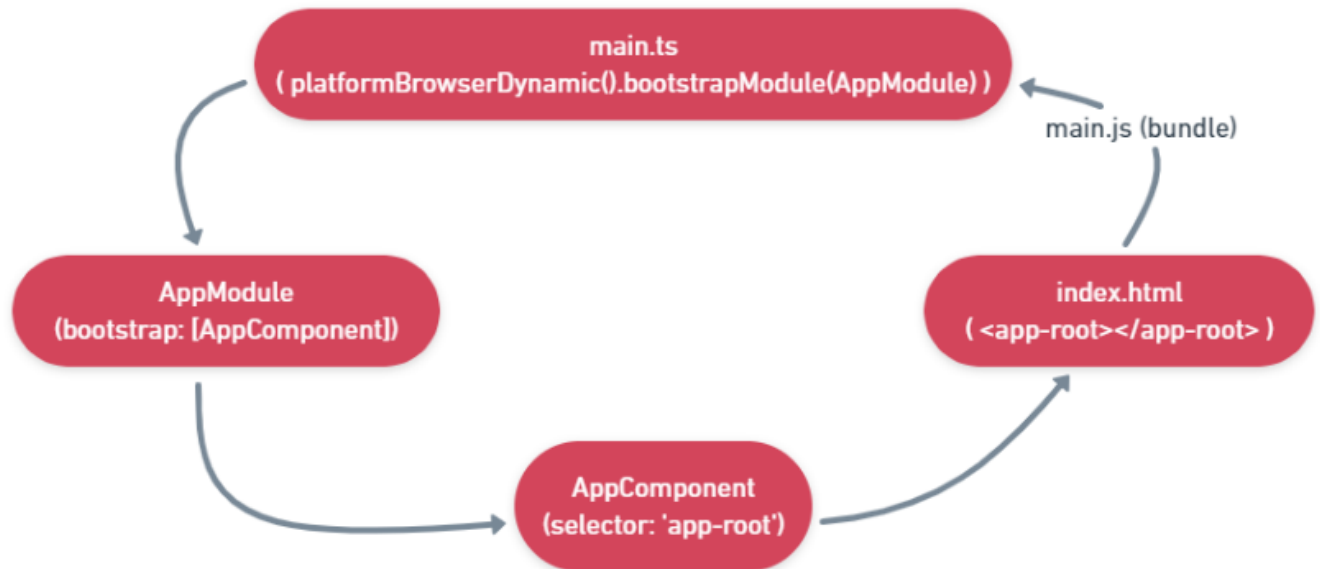


**Angular** es una plataforma y framework utilizado para escribir aplicaciones web en HTML y **Typescript**. Esta cuenta con diferentes librerías; muchas son parte del Core y son necesarias para el funcionamiento correcto de nuestras aplicaciones, y otras son opcionales.

La creación de aplicaciones con *Angular* consiste básicamente en generar **Plantillas (Templates)** usando *HTML*, los cuales se controlan con la lógica creada en los **Componentes**, que serán exportados como clases. Así mismo, se agrega lógica a unos **Servicios** para manejar la data que la aplicación tendrá y finalmente se encapsulan los Componentes y Servicios en **Módulos (NgModules)**.

La aplicación inicia su ejecución mediante el **Proceso de Carga (Bootstrapping)** del **Módulo Raíz** (generalmente llamado "**AppModule**" el cual es responsable de cargar otros módulos, componentes y servicios.). *Angular* toma el control y muestra contenido en el navegador,

reaccionando a la interacción de los usuarios que utilicen la aplicación de acuerdo a las instrucciones que se dieron en la lógica (código).



### Qué son los Módulos

Un **NgModule** declara un contexto de compilación para un conjunto de componentes. Un NgModule puede asociar sus componentes con código relacionado, como servicios, para formar unidades funcionales.

Cada aplicación generada con Angular cuenta con un módulo de raíz llamado convencionalmente **AppModule**, el cual provee el mecanismo de arranque que inicia la aplicación. Una aplicación generalmente contiene varios módulos funcionales.

Como en Javascript (y en muchos lenguajes con programación funcional), un módulo puede importar funcionalidades de otros módulos, y exportar sus propias funcionalidades.

Una buena práctica es crear diferentes módulos para la aplicación y organizar el código de manera funcional, para poder crear aplicaciones bien estructuradas cuando son complejas y escalables. Además, de esta forma se puede sacar provecho de la **Carga Diferida (Lazy Loading)**, el cual es un patrón de diseño que consiste en retrasar la carga o inicialización de un objeto hasta el mismo momento de su utilización) y así mejorar el rendimiento de la aplicación.

### Qué son los Componentes

Cada aplicación de Angular tiene al menos un componente. Al igual que el módulo de raíz, existe el **Componente Raíz (Root Component)**, que conecta una jerarquía de componentes con el DOM (**Document Object Model**, el cual es una interfaz que proporciona un conjunto estándar de objetos para representar documentos HTML, XHTML y XML). Cada componente define una clase que contiene data y lógica, y está vinculada a la plantilla HTML.

### Que son los Templates

Una **Plantilla (Template)** es una mezcla de HTML con Angular markup (tags personalizados de Angular). Las directivas de un Template proveen lógica de programación, y hacen **data binding** (enlace a los data de la aplicación) con las vistas.

Hay dos tipos de data binding:

- **Event binding** o enlace de eventos, que responden a la interacción del usuario al modificar algún input en la aplicación, actualizando la data.
- **Property binding** o enlace de propiedades, que permite agregar valores modificados desde la data al HTML

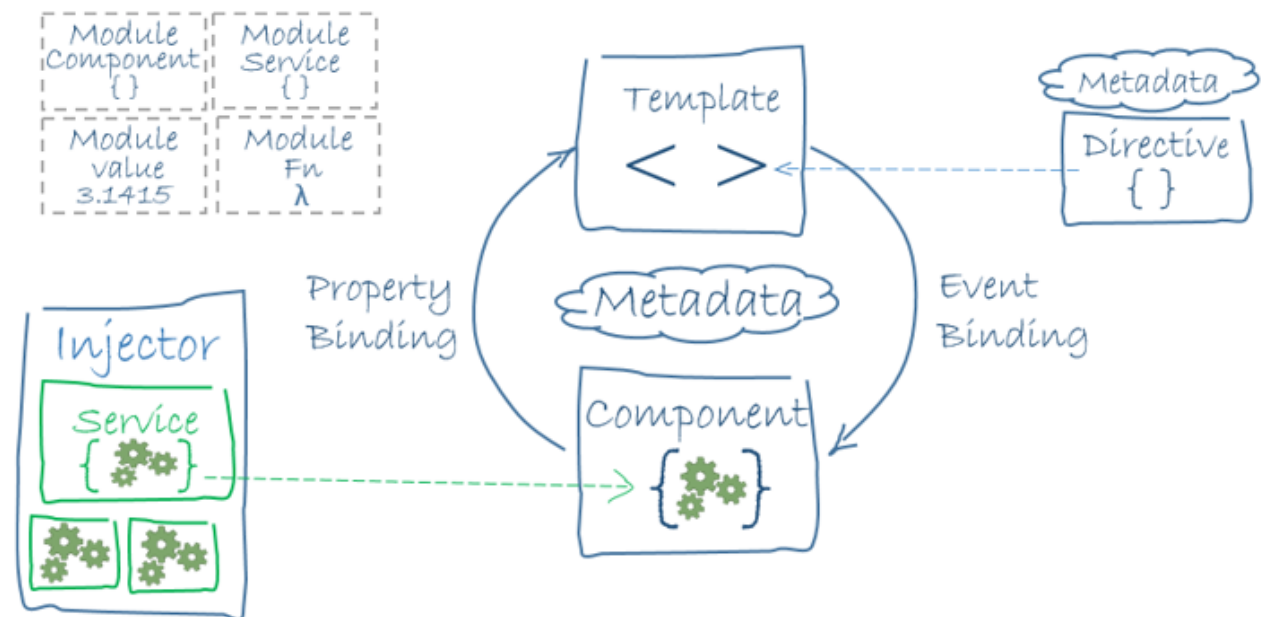
### Qué son Servicios e Inyección de Dependencias



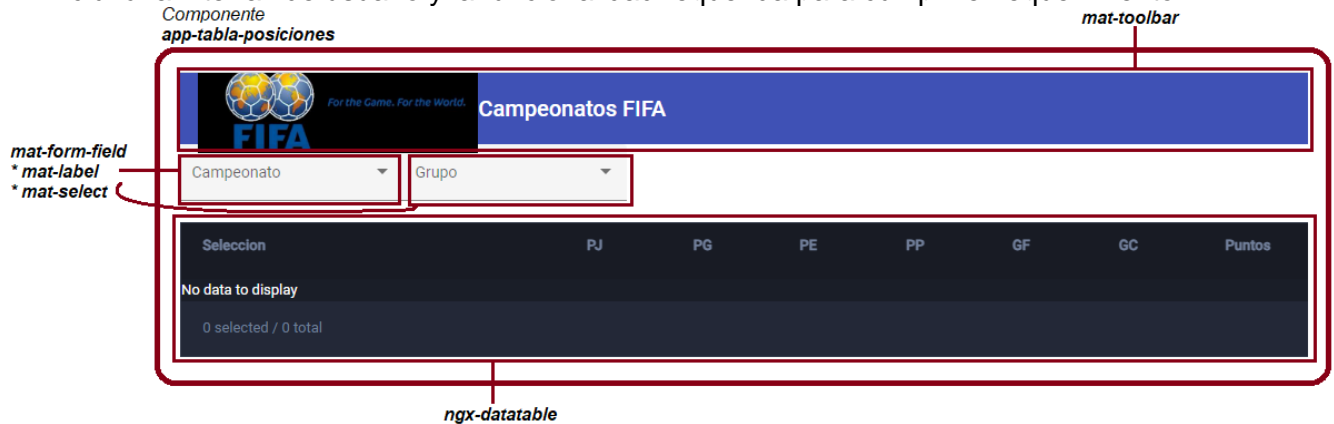
Toda la data o lógica que no está asociada directamente a una vista y que se quiere utilizar en diferentes partes de la aplicación y entre diferentes componentes, puede ser escrita en un **Servicio**. Tal como un componente, los servicios son exportados como clases. Los servicios cuentan con el decorator **@Injectable()** que provee metadata que permite que los servicios sean inyectados en componentes como dependencias.

Las Inyección de Dependencias (**Dependency injection**) permite manejar las clases de los componentes de forma ligera y eficiente. No obtienen datos del servidor, validan los user input o logs directamente en la consola; delegan la obtención de datos a los servicios.

Estos fueron algunos conceptos básicos sobre los principales bloques de arquitectura de una aplicación en Angular. El siguiente diagrama muestra cómo se relacionan estas piezas básicas:



- Con lo anterior en mente, se comenzará el desarrollo agregando un componente que incluirá la interfaz de usuario y la funcionalidad requerida para cumplir el requerimiento:



Para ello se agregará una carpeta al proyecto (como subcarpeta de la carpeta *app*) denominada “*componentes*” y luego se ejecutará el comando de creación de componentes (*ng generate component*)

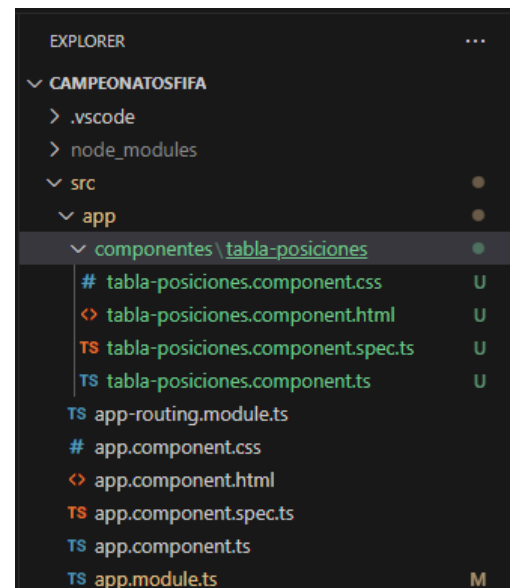
```
ng g c TablaPosiciones
```

```

PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE
PS D:\Catedra\UdeA\Técnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA> cd "D:\Catedra\UdeA\Técnicas de programación\Ejercicios\Proyecto CampeonatosFIFA\src\app\componentes"
PS D:\Catedra\UdeA\Técnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\componentes> ng g c TablaPosiciones
CREATE src/app/componentes/tabla-posiciones/tabla-posiciones.component.html (31 bytes)
CREATE src/app/componentes/tabla-posiciones/tabla-posiciones.component.spec.ts (663 bytes)
CREATE src/app/componentes/tabla-posiciones/tabla-posiciones.component.ts (314 bytes)
CREATE src/app/componentes/tabla-posiciones/tabla-posiciones.component.css (0 bytes)
UPDATE src/app/app.module.ts (634 bytes)
PS D:\Catedra\UdeA\Técnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\componentes>
  
```

El resultado es que se agrega al proyecto los archivos:

- tabla-posiciones.component.html* que contendrá el *HTML* para la interfaz de usuario
- tabla-posiciones.component.ts* que contendrá el código *TypeScript* que desarrollará la funcionalidad del componente



El código *TypeScript* de este componente sería:

```

import { Component, OnInit } from '@angular/core';
import { Campeonato } from 'src/app/entidades/campeonato';
  
```

```
import { ColumnMode, SelectionType } from '@swimlane/ngx-datatable';
import { TablaPosicion } from 'src/app/entidades/tabla-posicion';
import { CampeonatoService } from 'src/app/servicios/campeonato.service';
import { Grupo } from 'src/app/entidades/grupo';
import { GrupoService } from 'src/app/servicios/grupo.service';

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

  public idCampeonato: number = 0;
  public campeonatos: Campeonato[] = [];
  public idGrupo: number = 0;
  public grupos: Grupo[] = [];

  public columnas = [
    { name: 'Seleccion', prop: 'pais', width: 300 },
    { name: 'PJ', prop: 'pJ', width: 50 },
    { name: 'PG', prop: 'pG', width: 50 },
    { name: 'PE', prop: 'pE', width: 50 },
    { name: 'PP', prop: 'pP', width: 50 },
    { name: 'GF', prop: 'gF', width: 50 },
    { name: 'GC', prop: 'gC', width: 50 },
    { name: 'Puntos', prop: 'puntos', width: 50 },
  ];
  public posiciones: TablaPosicion[] = [];
  public modoColumna = ColumnMode;
  public tipoSeleccion = SelectionType;
  public posicionSeleccionada: TablaPosicion | undefined;

  constructor(private campeonatoService: CampeonatoService,
    private grupoService: GrupoService,
  ) {
  }

  ngOnInit(): void {
    this.listarCampeonatos();
  }

  public listarCampeonatos() {
    this.campeonatoService.listar()
      .subscribe(data => {
        this.campeonatos = data;
      },
      err => {
        window.alert(err.message)
      });
  }

  public seleccionarCampeonato() {
    this.campeonatoService.listarGrupos(this.idCampeonato)
      .subscribe(data => {
        this.grupos = data;
      });
  }
}
```



```
    },  
    err => {  
        window.alert(err.message)  
    });  
}  
  
public seleccionarGrupo() {  
    this.grupoService.obtenerTablaPosiciones(this.idGrupo)  
        .subscribe(data => {  
            this.posiciones = data;  
        },  
        err => {  
            window.alert(err.message)  
        });  
}  
  
public onActivate(event: any) {  
    if (event.type == 'click') {  
        this.posicionSeleccionada = event.row;  
    }  
}  
}
```

En este código:

- Se declaran las variables *idCampeonato* e *idGrupo* mediante las cuales se seleccionan el campeonato y el grupo para la consulta respectiva
- Se declara un arreglo de objetos *Campeonato* (denominado *campeonatos*) que se llenará con la consulta a la API mediante el servicio respectivo (objeto de la clase *CampeonatoService*) y que servirá de fuente de datos para la lista desplegable respectiva
- Se declara un arreglo de objetos *Grupo* que se llenará con la consulta a la API mediante el servicio respectivo (objeto de la clase *GrupoService*) y que servirá de fuente de datos para la lista desplegable respectiva
- Se declara un arreglo de objetos que definen los atributos de las columnas de la rejilla de datos donde se desplegará la tabla de posiciones consultada:
  - **name**: atributo con el título del encabezado
  - **prop**: atributo con el nombre del campo que se desplegará
  - **width**: atributo con el ancho de la columna
- Se declara un arreglo de objetos *TablaPosicion* que se llenará con la consulta a la API mediante el servicio *GrupoService* y que servirá de fuente para la rejilla de datos respectiva
- Se declara un objeto de la clase *TablaPosicion* denominado *posicionSeleccionada* el cual corresponde a la fila que se seleccione en la rejilla de datos
- Se declara un objeto de la clase *ColumnMode* que se utiliza para especificar el modo de las columnas en una tabla de datos. Estos modos pueden ser
  - **Standard**: Este es el modo predeterminado. Las columnas se ajustarán automáticamente según el contenido, pero si el contenido es demasiado ancho para la columna, se cortará y no se mostrará completo.
  - **Flex**: En este modo, las columnas se expandirán y contraerán automáticamente para ajustarse al espacio disponible. Esto permite que las columnas se ajusten dinámicamente al tamaño del contenedor de la tabla.
  - **Force**: En este modo, las columnas se ajustarán automáticamente según el contenido, pero si el contenido es demasiado ancho, las columnas se



reducirán para adaptarse al espacio disponible. Esto puede resultar en que el contenido se reduzca o abrevie.

- **ForceFit:** Similar al modo *Force*, pero las columnas se ajustarán para asegurarse de que todas las columnas se ajusten en el espacio disponible, sin reducir el contenido.
- **Fixed:** En este modo, las columnas tendrán un ancho fijo que se establece manualmente y no cambiará según el contenido. Esto significa que no se ajustarán automáticamente para adaptarse al contenido
- Se declara un objeto de la clase *SelectionType* que se utiliza para definir el tipo de selección que se aplicará a las filas de una tabla de datos. Los tipos de selección disponibles son:
  - **Single:** En este modo, solo se permite seleccionar una sola fila a la vez. Si se selecciona una fila diferente, la selección previa se deselecta automáticamente. Este modo es útil cuando se necesita seleccionar una fila específica de la tabla.
  - **Multi:** En este modo, se permite la selección de múltiples filas de la tabla de datos al mismo tiempo. Los usuarios pueden seleccionar varias filas sin deselectar las filas previamente seleccionadas. Esto es útil cuando se necesita realizar acciones en varias filas a la vez, como eliminar o editar múltiples elementos.
  - **MultiClick:** Similar al modo *Multi*, pero las filas se seleccionan o deselectan haciendo clic en ellas en lugar de usar casillas de verificación de selección. Esto permite una selección más intuitiva haciendo clic directamente en las filas.
  - **Checkbox:** En este modo, las filas se seleccionan utilizando casillas de verificación en lugar de seleccionar toda la fila. Esto es útil cuando se desea una interfaz de usuario que permita a los usuarios seleccionar filas mediante casillas de verificación Standard
- Para los objetos que representan los servicios (*campeonatoService* y *grupoService*) se utiliza la inyección de dependencias. *Angular* utiliza un sistema de inyección de dependencias para proporcionar instancias de objetos o servicios a las clases que los necesitan. En lugar de que las clases creen sus propias instancias de objetos, *Angular* se encarga de proporcionar esas instancias de manera eficiente y coherente. Se utiliza principalmente en componentes y servicios. Generalmente se inyectan dependencias en el constructor de un componente o servicio
- La clase correspondiente a este componente implementa la interfaz *OnInit*. Esta es parte del ciclo de vida de un componente en *Angular* y se utiliza para definir un método que se ejecutará cuando se inicialice el componente. En este caso se llamará al método *listarCampeonatos()*
- El método *listarCampeonatos()* se suscribe al método *listar()* del servicio de *Campeonato*. Suscribirse a un método generalmente significa observar un flujo de datos emitido por una función o método asíncronico. Esto se hace comúnmente utilizando observables. Y es que, en este caso, el método *listar()* del servicio devuelve un *Observable* al consumir asíncronicamente un método de la *API*. Al suscribirse a un observable, generalmente se proporcionan dos funciones de devolución de llamada: una para manejar los datos emitidos por el observable y otra para manejar errores en caso de que ocurran.  
En la primera función, se asigna la información de la respuesta de la *API* al arreglo de objetos de la clase *Campeonato* y en la segunda se muestra el mensaje de error recibido

- Adscrito al evento *click* de las opciones de la lista desplegable que muestra los campeonatos se tiene el método *seleccionarCampeonato()* el cual también se suscribe a un método que devuelve un *Observable*. En este caso al método *listarGrupos()* del servicio de *Campeonato*, y en el caso de la primera función, se asigna la respuesta al arreglo de objetos de la clase *Grupo*.  
Se debe observar que este método requiere el parámetro *idCampeonato* el cual se obtiene de la lista desplegable
- Adscrito al evento *click* de las opciones de la lista desplegable que muestra los grupos de un campeonato, se tiene el método *seleccionarGrupo()* el cual también se suscribe a un método que devuelve un *Observable*. En este caso al método *obtenerTablaPosiciones()* del servicio de *Grupo*, y en el caso de la primera función, se asigna la respuesta al arreglo de objetos de la clase *TablaPosiciones*.
- Por último, se tiene un método que responde a un evento de la rejilla de datos. El evento *activate* en el componente *ngx-datatable* se utiliza para capturar la activación de una fila en la rejilla de datos. Esto puede ocurrir cuando un usuario hace clic en una fila específica. El evento *activate* proporciona información sobre la fila activada, como los datos de la fila y la fila misma.  
En este caso, se valida que el tipo de evento sea un *click*, en cuyo caso se asigna la fila seleccionada un objeto de la clase *TablaPosicion*

Y el código HTML de este componente sería:

```
<mat-toolbar class="mat-primary mat-toolbar-single-row mat-elevation-z8
color-barra" color="background">
  
  <div class="center">
    Campeonatos FIFA
  </div>
</mat-toolbar>

<mat-form-field appearance="fill">
  <mat-label>Campeonato</mat-label>
  <mat-select [(value)]="idCampeonato">
    <mat-option *ngFor="let c of campeonatos" [(value)]="c.id"
(click)="seleccionarCampeonato()" ">
      {{c.nombre}}
    </mat-option>
  </mat-select>
</mat-form-field>
<mat-form-field appearance="fill">
  <mat-label>Grupo</mat-label>
  <mat-select [(value)]="idGrupo">
    <mat-option *ngFor="let g of grupos" [(value)]="g.id"
(click)="seleccionarGrupo()" ">
      {{g.nombre}}
    </mat-option>
  </mat-select>
</mat-form-field>

<ngx-datatable class="dark" [headerHeight]="50" [footerHeight]="50"
[rowHeight]="20" [rows]="posiciones"
[columns]="columnas" [columnMode]="modoColumna.force" [limit]="10"
[selectionType]="tipoSeleccion.single"
(activate)="onActivate($event)" [rowHeight]="'auto'">

</ngx-datatable>
```

En este código:

- Se tiene inicialmente un componente **mat-toolbar** el cual es un componente de la librería *Angular Material* que se utiliza para crear barras de herramientas (*toolbars*) en aplicaciones web desarrolladas con Angular. Las barras de herramientas son áreas de la interfaz de usuario que suelen contener botones, enlaces, títulos y otros elementos de navegación o de acción.

En este caso, se utiliza para mostrar el logo de la FIFA y un título para la página

- El componente **mat-form-field** es otro componente de la librería *Angular Material* que se utiliza para crear formularios en aplicaciones web desarrolladas con Angular. Proporciona un contenedor flexible para los elementos de entrada de un formulario, como campos de texto, selectores, casillas de verificación, etc.

Una característica común de *mat-form-field* es la capacidad de mostrar etiquetas flotantes. Cuando un usuario coloca el cursor en un campo de entrada, la etiqueta se eleva para proporcionar información sobre el tipo de entrada esperado.

En este caso para el primer *mat-form-field* se coloca dentro una etiqueta (componente **mat-label**) que indica que se está leyendo el campeonato, y un componente *mat-select*



- El componente **mat-select** es parte de la librería *Angular Material* y se utiliza para crear listas desplegables (selects) en aplicaciones web desarrolladas con Angular. Una lista desplegable es un elemento de formulario que permite a los usuarios seleccionar una opción de una lista predefinida. Entre las características de este componente, permite al usuario seleccionar una opción de una lista de elementos que se puede cargar dinámicamente desde una fuente de datos.

En este ejercicio, se utiliza *mat-select* para crear una lista desplegable de campeonatos. Las opciones se generan dinámicamente utilizando **\*ngFor** para iterar sobre el arreglo *campeonatos*. La propiedad **[(value)]** se utiliza para vincular la selección del usuario al modelo de datos en el componente. En este caso a la variable *idCampeonato*.

Cada opción es definida por un componente **mat-option** y está asociada a un valor específico utilizando la propiedad **[value]**. Esto permite vincular la selección del usuario con un valor específico en el componente de Angular.

- De manera similar, se tiene otra lista desplegable para listar los grupos del campeonato seleccionado y esta se llena iterando el arreglo *grupos*
- Finalmente se tiene un componente **ngx-datatable** correspondiente a la rejilla de datos los cuales tienen los siguientes atributos:
  - **[rows]**: Esta propiedad se utiliza para especificar los datos que se mostrarán en la tabla. Debe asignarse a un arreglo de objetos que representan las filas de datos, en este caso el arreglo *posiciones*
  - **[columns]**: Se utiliza para definir las columnas de la tabla. Debes asignar un arreglo de objetos de columna que describen las características de cada



columna, como el título y el nombre de la propiedad de datos que se mostrará. En este caso el arreglo *columns*

- **[columnMode]:** Permite especificar el modo de las columnas. Los valores comunes incluyen 'standard', 'flex', 'force', 'force-fill' y 'fixed'. Controla cómo se ajustan las columnas en la tabla.
- **[headerHeight]:** Define la altura de la fila de encabezado.
- **[footerHeight]:** Define la altura de la fila de pie de página.
- **[rowHeight]:** Establece la altura de las filas de datos.
- **[scrollbarV]:** Indica si se debe mostrar una barra de desplazamiento vertical.
- **[scrollbarH]:** Indica si se debe mostrar una barra de desplazamiento horizontal.
- **[virtualization]:** Habilita o deshabilita la virtualización de filas para un rendimiento mejorado en tablas largas.
- **[selectionType]:** Define el tipo de selección que se utilizará en la tabla. Puede ser 'single', 'multi', 'multiClick', o 'checkbox'.
- **[selected]:** Un arreglo de objetos que representa las filas seleccionadas en la tabla.
- **(activate):** Se utiliza para capturar el evento de activación de una fila en la tabla.
- **(select):** Permite manejar eventos cuando se selecciona una fila.
- **(deselect):** Permite manejar eventos cuando se deselecciona una fila.

Selección	PJ	PG	PE	PP	GF	GC	Puntos
Colombia	3	2	0	1	5	2	6
Japón	3	1	1	1	4	4	4
Senegal	3	1	1	1	4	4	4
Polonia	3	1	0	2	2	5	3
0 selected / 4 total							

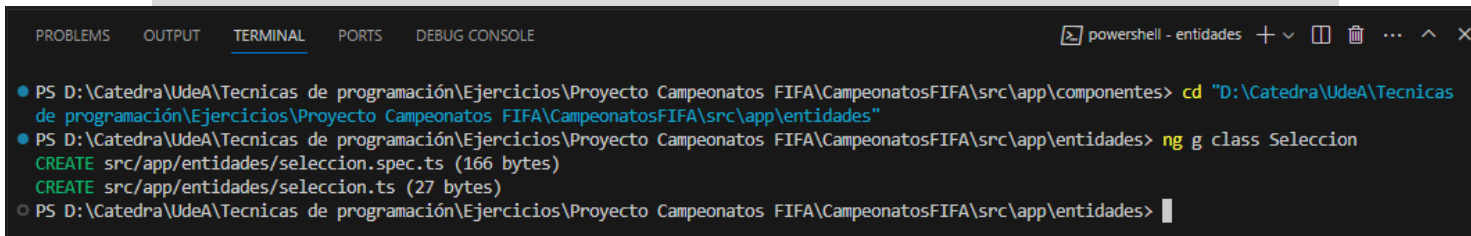
Ahora bien, para poder garantizar la funcionalidad del anterior componente, se requieren agregar los siguientes elementos:

Elemento	Archivos	Descripción
<b>Clase Campeonato</b>	• <i>campeonato.ts</i>	Define la entidad respectiva. Las entidades representan la información a procesar. En este caso, la procedente de la API a través de los servicios
<b>Clase Selección</b>	• <i>seleccion.ts</i>	Define la entidad respectiva
<b>Clase Grupo</b>	• <i>grupo.ts</i>	Define la entidad respectiva
<b>Clase GrupoSelección</b>	• <i>grupo-selección.ts</i>	Define la entidad respectiva
<b>Clase TablaPosiciones</b>	• <i>tabla-posicion.ts</i>	Define la entidad respectiva
<b>Servicio Campeonato</b>	• <i>campeonato.sevice.ts</i>	Define la funcionalidad que accede a la API a través del controlador de <i>Campeonatos</i>
<b>Servicio Grupo</b>	• <i>grupo.sevice.ts</i>	Define la funcionalidad que accede a la API a través del controlador de <i>Grupos</i>

<b>Modulo</b> <b>ReferenciasMaterial</b>	<ul style="list-style-type: none"> <li>referencias-material.module.ts</li> </ul>	Organiza funcionalidades relacionadas con la librería <i>Material</i>
---	--	---

- Para ello se agregará otra carpeta al proyecto (también como subcarpeta de la carpeta *app*) denominada “*entidades*” y luego se ejecutará el comando de creación de clases (*ng generate class*)

```
ng g class Seleccion
```

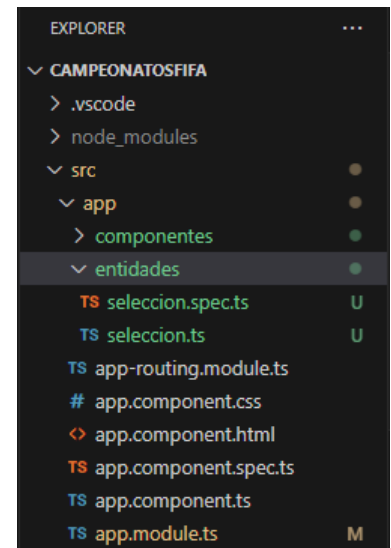


```

PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\componentes> cd "D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\entidades"
PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\entidades> ng g class Seleccion
CREATE src/app/entidades/seleccion.spec.ts (166 bytes)
CREATE src/app/entidades/seleccion.ts (27 bytes)
PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\entidades>
  
```

El resultado es que se agrega al proyecto el archivo:

- seleccion.ts* que contendrá el código *TypeScript* en el que se definirá la estructura de la clase



El código *TypeScript* de esta clase sería:

```

export class Seleccion {
  constructor(
    public id: number,
    public nombre: string,
    public entidad: string,
  ) {
  }
}
  
```

Donde se definen los atributos de la clase *Selección* en el método constructor.

- En la misma carpeta se agregarán las demás clases:

```

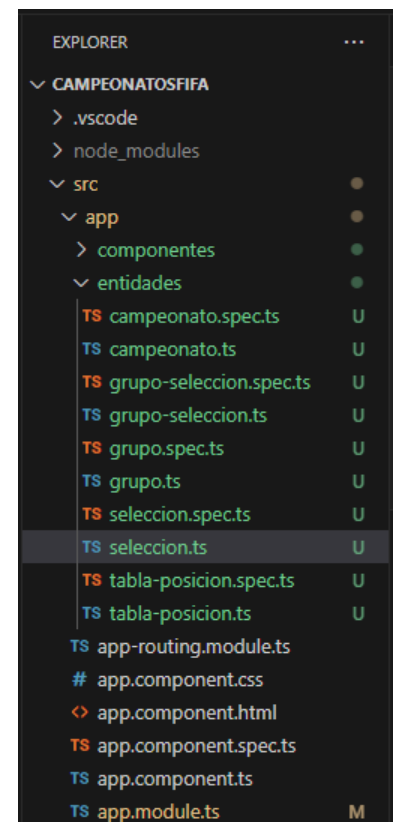
ng g class Campeonato
ng g class Grupo
ng g class GrupoSeleccion
ng g class TablaPoscion
  
```

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE powershell - entidades + v [ ] [ ] ...

de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\entidades"
● PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\entidades> ng g class Campeonato
CREATE src/app/entidades/campeonato.spec.ts (170 bytes)
CREATE src/app/entidades/campeonato.ts (28 bytes)
● PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\entidades> ng g class Grupo
CREATE src/app/entidades/grupo.spec.ts (150 bytes)
CREATE src/app/entidades/grupo.ts (23 bytes)
● PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\entidades> ng g class GrupoSeleccion
CREATE src/app/entidades/grupo-seleccion.spec.ts (187 bytes)
CREATE src/app/entidades/grupo-seleccion.ts (32 bytes)
● PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\entidades> ng g class TablaPosicion
CREATE src/app/entidades/tabla-posicion.spec.ts (183 bytes)
CREATE src/app/entidades/tabla-posicion.ts (31 bytes)
● PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\entidades> |
```

El resultado es que se agregan al proyecto los archivos:

- *campeonato.ts* que contendrá el código *TypeScript* en el que se definirá la estructura de la clase *Campeonato*
- *grupo.ts* que contendrá el código *TypeScript* en el que se definirá la estructura de la clase *Grupo*
- *grupo-seleccion.ts* que contendrá el código *TypeScript* en el que se definirá la estructura de la clase *GrupoSeleccion*
- *tabla-posicion.ts* que contendrá el código *TypeScript* en el que se definirá la estructura de la clase *TablaPosicion*



El código *TypeScript* del archivo *campeonato.ts* sería:

```
import { Seleccion } from "../seleccion";

export class Campeonato {
  constructor(public id: number,
    public nombre: string,
    public pais: Seleccion,
    public año: number
  ) {

  }
}
```

Donde se definen los atributos de la clase *Campeonato* en el método constructor. Obsérvese la referencia a la clase *Seleccion*



El código *TypeScript* del archivo *grupo.ts* sería:

```
export class Grupo {
  constructor(
    public id: number,
    public nombre: string,
  ) {
  }
}
```

Donde se definen los atributos de la clase *Grupo* en el método constructor.

El código *TypeScript* del archivo *grupo-seleccion.ts* sería:

```
import { Grupo } from "../grupo";
import { Seleccion } from "../seleccion";

export class GrupoSeleccion {
  constructor(public grupo: Grupo,
    public pais: Seleccion,
  ) {
  }
}
```

Donde se definen los atributos de la clase *GrupoSeleccion* en el método constructor. Obsérvese las referencias a las clases *Selección* y *Grupo*.

Por último, el código *TypeScript* del archivo *tabla-posicion.ts* sería:

```
export class TablaPosicion {
  constructor(
    public id: number,
    public pais: string,
    public pJ: number,
    public pG: number,
    public pE: number,
    public pP: number,
    public gF: number,
    public gC: number,
    public puntos: number,
  ) {
  }
}
```

Donde se definen los atributos de la clase *TablaPosicion* en el método constructor.

- Ahora se agregará otra carpeta al proyecto (también como subcarpeta de la carpeta *app*) denominada “*servicios*” y luego se ejecutará el comando de creación de servicios (*ng generate service*)

```
ng g s Campeonato
```

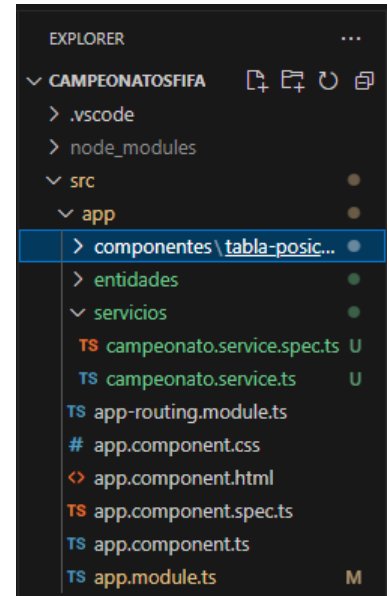
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

powershell - servicios + v [ ] [ ]

- PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\entidades> cd "D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\servicios"
- PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\servicios> ng g s Campeonato  
CREATE src/app/servicios/campeonato.service.spec.ts (377 bytes)  
CREATE src/app/servicios/campeonato.service.ts (139 bytes)
- PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\servicios> |

El resultado es que se agrega al proyecto el archivo:

- *campeonato.services.ts* que contendrá el código *TypeScript* en el que se implementarán las solicitudes al controlador *Campeonato* de la *API*



El código *TypeScript* de esta clase sería:

```
import { HttpClient } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { Observable } from "rxjs";
import { environment } from "src/environments/environment";
import { Campeonato } from "../entidades/campeonato";
import { Grupo } from "../entidades/grupo";

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

  url: string;

  constructor(
    private http: HttpClient
  ) {
    this.url = `${environment.urlAPI}campeonatos`;
  }

  public listar(): Observable<Campeonato[]> {
    let urlT = `${this.url}/listar`;
    return this.http.get<Campeonato[]>(urlT);
  }

  public listarGrupos(id: number): Observable<Grupo[]> {
    let urlT = `${this.url}/grupos/${id}`;
    return this.http.get<Grupo[]>(urlT);
  }
}
```

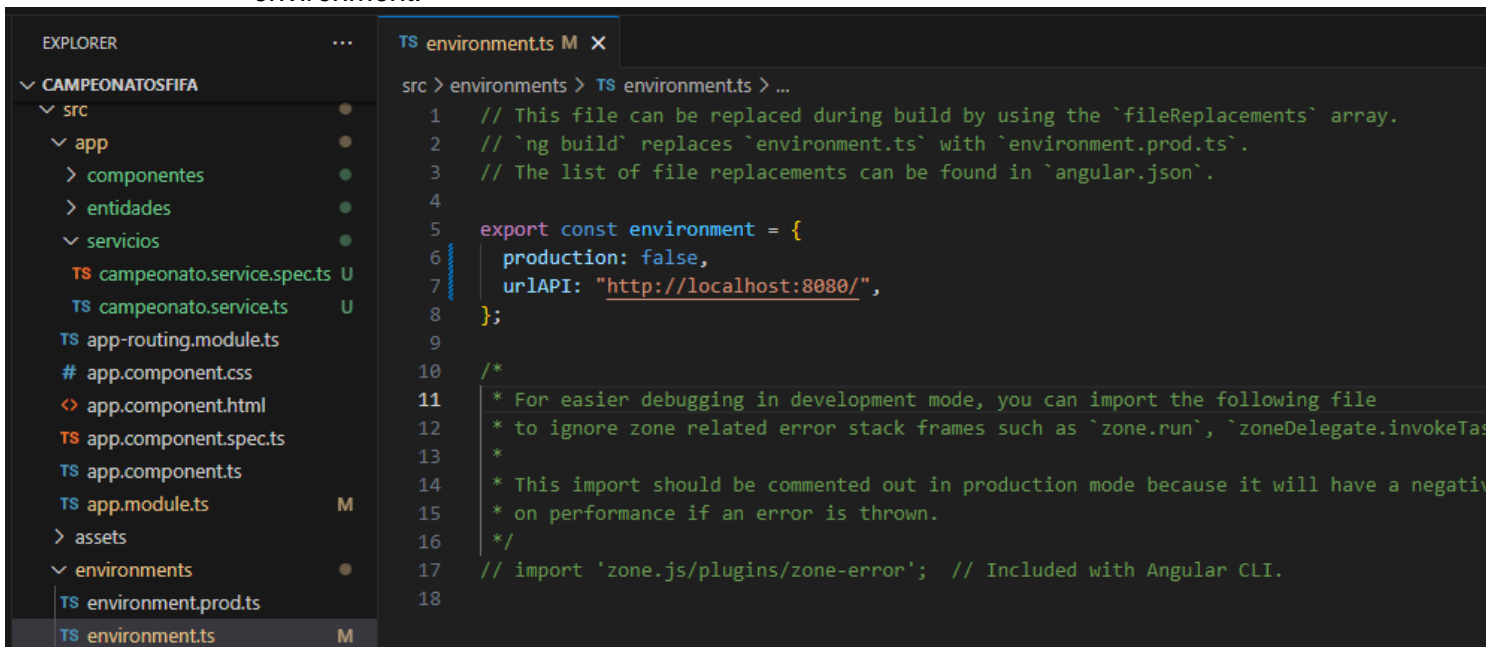
En este código:



- Se declara una variable de clase que almacena la URL del controlador. Esta se inicializa en el método constructor, concatenando la variable de entorno *urlAPI* (proveniente del archivo *environment.ts*) con la palabra “campeonatos”.

El archivo *environment.ts* es un archivo de configuración utilizado para definir variables de entorno y configuraciones específicas de un entorno de desarrollo. Este archivo se encuentra en la carpeta *src/environments* de todo proyecto *Angular* y se utiliza para proporcionar valores que pueden variar entre entornos, como el entorno de desarrollo y el entorno de producción. Esto es particularmente útil para separar configuraciones que difieren entre entornos, como las URL de las *API*, las claves de acceso a servicios externos u otras variables de configuración.

En un proyecto *Angular*, predeterminadamente, se proporcionan dos archivos *environment*.



The screenshot shows the VS Code interface. On the left, the Explorer panel displays the project structure: **CAMPEONATOSFIFA** > **src** > **app** > **servicios** > **TS campeonato.service.spec.ts** (U), **TS campeonato.service.ts** (U), **TS app-routing.module.ts**, **# app.component.css**, **<> app.component.html**, **TS app.component.spec.ts**, **TS app.component.ts**, **TS app.module.ts** (M), **> assets**, **> environments** > **TS environment.prod.ts**, **TS environment.ts** (M). The main editor shows the content of **TS environment.ts**:

```
1 // This file can be replaced during build by using the `fileReplacements` array.
2 // `ng build` replaces `environment.ts` with `environment.prod.ts`.
3 // The list of file replacements can be found in `angular.json`.
4
5 export const environment = {
6   production: false,
7   urlAPI: "http://localhost:8080/",
8 };
9
10 /*
11  * For easier debugging in development mode, you can import the following file
12  * to ignore zone related error stack frames such as `zone.run`, `zoneDelegate.invokeTask`
13  *
14  * This import should be commented out in production mode because it will have a negative
15  * on performance if an error is thrown.
16  */
17 // import 'zone.js/plugins/zone-error'; // Included with Angular CLI.
18
```

- ***environment.ts***: El cual se utiliza para configuraciones específicas del entorno de desarrollo como por ejemplo variables correspondientes a las URL de *APIs* de prueba
- ***environment.prod.ts***: Este archivo se utiliza para configuraciones específicas del entorno de producción. Por ejemplo, la URL de producción de una *API*.

En este ejercicio, se tendrá el siguiente código para el archivo *environment.ts*:

```
export const environment = {
  production: false,
  urlAPI: "http://localhost:8080/",
};
```

En el que se puede observar la declaración la variable de entorno *urlAPI* almacenando la URL de la *API* que va a ser utilizada.

Para acceder a las variables de entorno en una aplicación *Angular*, se debe importar el objeto *environment* y usar sus propiedades en el código, como en este caso para completar la URL de la *API*

```
import { environment } from "src/environments/environment";
...
```

```
this.url = `${environment.urlAPI}campeonatos`  
...
```

- El método *listar()* es un método con características muy especiales. En primer lugar, el tipo de dato de salida tiene la notación *Observable<>*.

La notación ***Observable<>*** en *Angular* se refiere a la clase *Observable* proporcionada por la librería ***RxJS*** (sigla de *Reactive Extensions for JavaScript* en inglés). *RxJS* es ampliamente utilizado para manejar flujos de datos asíncronicos, como respuestas de *API*, eventos del usuario y otros tipos de datos que llegan con el tiempo. El *Observable* es una parte fundamental de la programación reactiva en *Angular*.

Un ***Observable*** es una secuencia de eventos que pueden incluir valores, errores o notificaciones de completitud. Algunas de las operaciones y características comunes incluyen:

- ***Suscripción***: se suscribe a un *Observable* para escuchar y reaccionar a los eventos emitidos por él
- ***Operadores***: sirven para transformar, filtrar, mapear o combinar los datos en el flujo
- ***Manejo de Errores***: se pueden manejar errores que se produzcan en el flujo de datos utilizando operadores como *catchError* o *retry*
- ***Composición***: Se Pueden combinar múltiples *Observables* en uno solo utilizando operadores como *merge*, *concat*, *forkJoin*, etc
- ***Unsubscribing***: Es importante desuscribirse de los *Observable* cuando ya no se necesite escuchar los datos para evitar pérdidas de memoria. Se Pueden hacer llamando al método ***unsubscribe()*** en el objeto de suscripción

Dentro del cuerpo del método *listar()*, la primera instrucción es una concatenación que completa el endpoint del método de la api a consumirse. En este caso la URL completa sería:

```
http://localhost:8080/campeonatos/listar
```

Resultado de concatenar la URL del controlador (que a su vez resulta de la concatenación del servidor de *API* con el nombre del controlador) con el nombre del método.

Seguido, Se hace uso del método genérico de la clase ***HttpClient, get<T>*** el cual representa una solicitud *HTTP GET* a un servidor con el tipo genérico ***<T>***. Por ejemplo, la instrucción en cuestión:

```
return this.http.get<Campeonato[]>(urlT);
```

Significa que se está haciendo un llamado al endpoint cuyo método esté direccionado mediante la URL almacenada en la variable *urlT* en una solicitud de tipo *GET* que debe devolver un arreglo de objetos *Campeonato*

La clase *HttpClient* en *Angular* es un módulo que proporciona una forma simplificada y basada en observables para realizar peticiones *HTTP* a servidores remotos. Permite a los desarrolladores realizar solicitudes *HTTP*, como *GET*, *POST*, *PUT*, *DELETE*, etc., gestionar las respuestas, y trabajar con datos *JSON* u

otros formatos de datos de manera eficiente y asincrónica, lo que facilita la comunicación con servicios web y la obtención de datos en aplicaciones Angular.

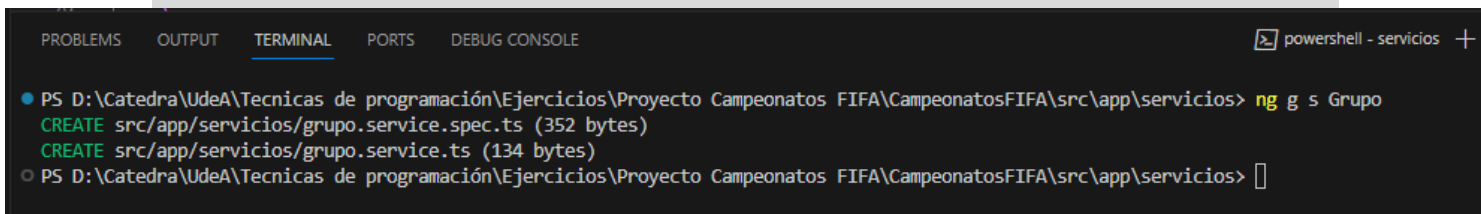
- El método *listarGrupos()*, de manera similar al anterior método, realiza una solicitud *GET* en este caso al endpoint:

```
http://localhost:8080/campeonatos/grupos/{id}
```

el cual pasa en la URL un parámetro de entrada que corresponde a la clave primaria del Campeonato del que se desea consultar la lista de grupos registrados y los cuales son retornados también a través de un *Observable*

- Ahora se ejecutará de nuevo el comando de creación de servicios para la información asociada a los Grupos

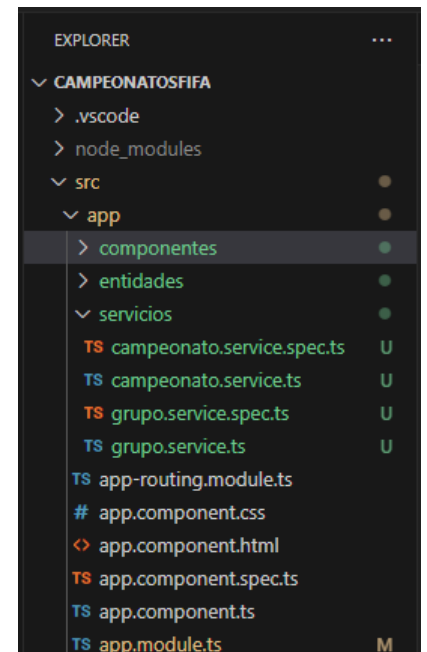
```
ng g s Grupo
```



```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE powershell - servicios +
● PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\servicios> ng g s Grupo
CREATE src/app/servicios/grupo.service.spec.ts (352 bytes)
CREATE src/app/servicios/grupo.service.ts (134 bytes)
○ PS D:\Catedra\UdeA\Tecnicas de programación\Ejercicios\Proyecto Campeonatos FIFA\CampeonatosFIFA\src\app\servicios> []
```

El resultado es que se agrega al proyecto el archivo:

- *grupo.services.ts* que contendrá el código *TypeScript* en el que se implementarán las solicitudes al controlador *Grupo* de la API



El código *TypeScript* de esta clase sería:

```
import { HttpClient } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { Observable } from "rxjs";
import { environment } from "src/environments/environment";
```

```
import { TablaPosicion } from "../entidades/tabla-posicion";

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

  url: string;

  constructor(
    private http: HttpClient
  ) {
    this.url = `${environment.urlAPI}grupos`;
  }

  public obtenerTablaPosiciones(id: number):
  Observable<TablaPosicion[]> {
    let urlT = `${this.url}/posiciones/${id}`;
    return this.http.get<TablaPosicion[]>(urlT);
  }
}
```

En este código:

- De manera similar al anterior servicio, se tiene una variable que almacena la URL del controlador:

<http://localhost:8080/grupos>

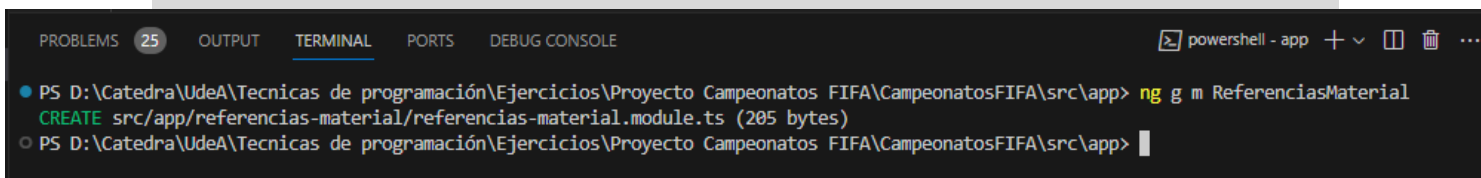
- El método *obtenerTablaPosiciones()*, de manera similar a los del anterior servicio, realiza una solicitud *GET* al endpoint:

<http://localhost:8080/grupos/posiciones/{id}>

el cual pasa en la URL un parámetro de entrada que corresponde a la clave primaria del Grupo del que se desea consultar la tabla de posiciones la cual es retornada a través de un *Observable*

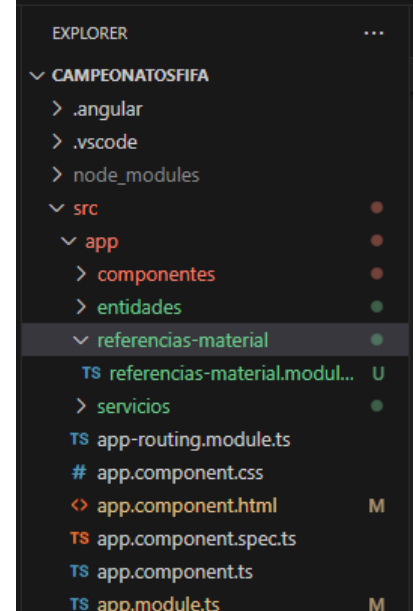
- Para poder acceder a todas las funcionalidades que ofrece la librería *Material*, o más específicamente a las requeridas por el proyecto (como *label*, *input*, *select*, entre otros), se va a agregar un módulo que las deje disponibles. Para ello se ejecutará el comando de creación de módulos (*ng generate module*)

```
ng g m ReferenciasMaterial
```



El resultado es que se agrega al proyecto el archivo:

- *referencias-material.module.ts* que contendrá el código *TypeScript* en el que se exportarán todas las referencias a los elementos más utilizados de la librería *Material*



El código *TypeScript* de esta clase sería:

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

import { MatToolbarModule } from '@angular/material/toolbar';
import { MatCardModule } from '@angular/material/card';
import { MatButtonModule } from '@angular/material/button';
import { MatInputModule } from '@angular/material/input';
import { MatFormFieldModule } from '@angular/material/form-field';
import { MatSidenavModule } from '@angular/material/sidenav';
import { MatIconModule } from '@angular/material/icon';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatListModule } from '@angular/material/list';
import { MatExpansionModule } from '@angular/material/expansion';
import { MatRadioModule } from '@angular/material/radio';
import { MatTabsModule } from '@angular/material/tabs';
import { MatDialogModule } from '@angular/material/dialog';
import { MatDividerModule } from '@angular/material/divider';
import { MatSelectModule } from '@angular/material/select';

import { MatCheckboxModule } from '@angular/material/checkbox';

import { MatDatepickerModule } from '@angular/material/datepicker';
import { MatNativeDateModule } from '@angular/material/core';

import { MatAutocompleteModule } from '@angular/material/autocomplete';

import { MatMenuModule } from '@angular/material/menu';

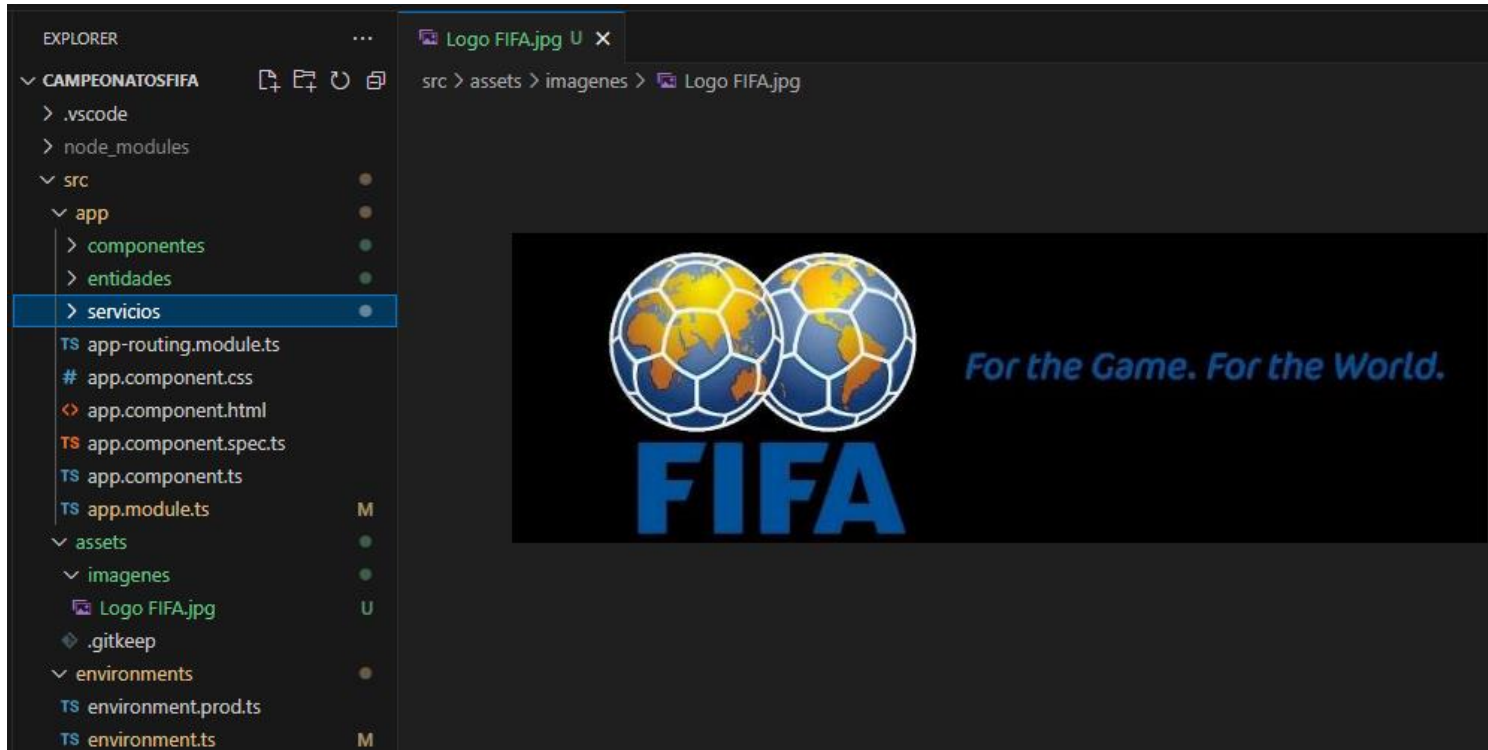
import { MatProgressSpinnerModule } from '@angular/material/progress-spinner';
import { CommonModule } from '@angular/common';

@NgModule({
  exports: [
    MatToolbarModule,
    MatCardModule,
```

```
MatFormFieldModule,  
MatInputModule,  
MatButtonModule,  
MatSidenavModule,  
MatIconModule,  
BrowserAnimationsModule,  
MatListModule,  
MatExpansionModule,  
MatRadioModule,  
MatTabsModule,  
MatDialogModule,  
MatDividerModule,  
MatMenuModule,  
MatSelectModule,  
MatCheckboxModule,  
  
MatNativeDateModule,  
MatDatepickerModule,  
MatAutocompleteModule,  
  
MatProgressSpinnerModule  
]  
})  
  
export class ReferenciasMaterialModule { }
```

Un módulo en Angular se define mediante la decoración **@NgModule**. En este caso se utiliza la instrucción **exports** para especificar qué componentes quedarán disponibles para otros módulos o componentes fuera del módulo actual. Esto significa que los elementos marcados como "exportados" en este módulo pueden ser utilizados por otros módulos de la aplicación.

- Ahora se agregará una carpeta denominada *"imágenes"* dentro de la carpeta *"assets"*



La carpeta "assets" es el lugar donde se almacenan archivos estáticos que se deben incluir en la aplicación, como imágenes, fuentes, archivos de configuración, documentos y otros recursos no relacionados con el código fuente de la aplicación. Estos archivos son copiados tal cual, a la carpeta de salida de la aplicación durante el proceso de construcción, lo que los hace accesibles a través de la URL de la aplicación.

La estructura de archivos dentro de la carpeta "assets" se mantendrá en la carpeta de salida, lo que significa que, si se tienen archivos en subdirectorios dentro de ella, podrán ser accedidos utilizando las rutas correspondientes en la aplicación Angular.

Dentro de la carpeta "imágenes" se copiará el archivo con el logo de la *FIFA* para ambientar la aplicación, el cual es accedido desde el componente *TablaPosiciones*

- Para culminar la edición del aplicativo, se editarán los archivos *app.component.html*, *app.module.ts* y *styles.css*.

El archivo *app.component.html* es el punto de entrada visual de la aplicación y en este caso contendrá una referencia al componente *TablaPosiciones*. El tag a utilizar es el que se definió en el atributo **selector** de la decoración *@Component* de este componente:

```
@Component({
  selector: 'app-tabla-posiciones',
  templateUrl: './tabla-posiciones.component.html',
  styleUrls: ['./tabla-posiciones.component.css']
})
```

El código HTML de este componente sería:

```
<app-tabla-posiciones></app-tabla-posiciones>
```



En Angular, el archivo *app.module.ts* es una parte clave que define y configura el módulo principal de la aplicación. Este módulo principal suele llamarse **AppModule** y desempeña varios roles importantes:

- Contiene la decoración **@NgModule** que especifica los metadatos esenciales, como los componentes, directivas, pipes y servicios que pertenecen a este módulo. Además, incluye configuraciones importantes, como las rutas y otros módulos que el módulo principal importa
- También especifica el componente principal que se utilizará como punto de entrada para la aplicación mediante la propiedad **Bootstrap**
- A través de la propiedad **imports**, importa otros módulos necesarios para la aplicación, como **BrowserModule**, el cual proporciona funcionalidades esenciales para aplicaciones web
- En la propiedad **declarations**, se incluyen los componentes, directivas y pipes que pertenecen a este módulo. Esto permite a Angular reconocer y utilizar estos elementos en la aplicación

El código *TypeScript* de esta clase sería:

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { TablaPosicionesComponent } from './componentes/tabla-posiciones/tabla-posiciones.component';
import { ReferenciasMaterialModule } from './referencias-material/referencias-material.module';
import { NgxDatatableModule } from '@swimlane/ngx-datatable';

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

El archivo *styles.css* es un archivo de estilo global en una aplicación Angular. Este archivo es utilizado para definir estilos que se aplicarán a toda la aplicación y afectarán a todos los componentes y elementos en ella. Es el lugar común para definir reglas de estilo que se aplican de manera global en toda la aplicación Angular.

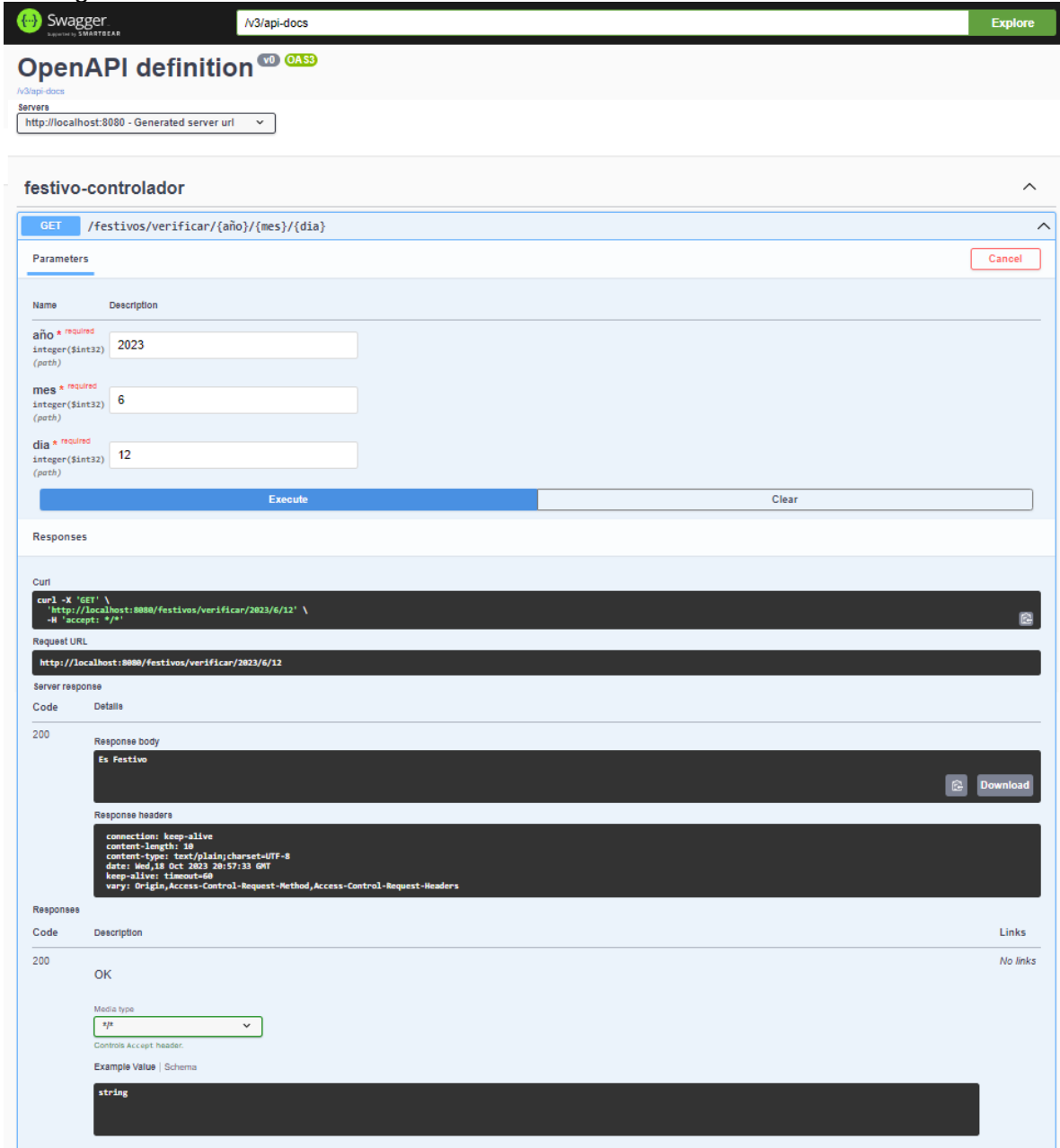
En este proyecto, se importarán estilos provistos por las librerías *Material* y *ngx-datatable* para disponer temáticas en los elementos visuales a utilizar:



Con todo lo anterior implementado, se puede proceder a la ejecución del aplicativo mediante el comando:

Esto activará la compilación del proyecto completo y la respectiva generación del ejecutable que luego será desplegado en el navegador de internet

2. Elaborar un aplicativo cliente Web en *Angular CLI* para una *API RESTfull* en *Spring Boot* que cumple con la siguiente documentación:



The image shows the Swagger UI for an API endpoint. The top bar includes the Swagger logo, the text "Swagger", and a search bar containing "/v3/api-docs". A green "Explore" button is on the right. Below the top bar, the text "OpenAPI definition" is displayed, followed by "v0" and "OAS3". A "Servers" section shows a dropdown menu with "http://localhost:8080 - Generated server url". The main section is titled "festivo-controlador" and shows a GET endpoint: "/festivos/verificar/{año}/{mes}/{dia}". The "Parameters" section lists three required path parameters: "año" (integer, 2023), "mes" (integer, 6), and "dia" (integer, 12). Below the parameters are "Execute" and "Clear" buttons. The "Responses" section shows a 200 status code with a response body "Es Festivo" and response headers including "connection: keep-alive", "content-length: 10", "content-type: text/plain; charset=UTF-8", "date: Wed, 18 Oct 2023 20:57:33 GMT", "keep-alive: timeout=60", and "vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers". A "Curl" section shows the command: "curl -X 'GET' \ 'https://localhost:8080/festivos/verificar/2023/6/12' \ -H 'accept: \*/\*'". A "Request URL" section shows the URL: "http://localhost:8080/festivos/verificar/2023/6/12". A "Server response" section shows the response code "200" and the response body "Es Festivo". A "Response" section shows the response code "200" and the response description "OK". A "Media type" dropdown menu is set to "\*/\*", and an "Example Value" field shows "string".

Un primer método que verifica si una fecha corresponde a un festivo

Swagger  [Explore](#)

OpenAPI definition <sup>v0</sup> **OAS3**

[/v3/api-docs](#)

**Servers**  
[http://localhost:8080 - Generated server url](#)

### festivo-controlador

**GET** /festivos/verificar/{año}/{mes}/{dia}

**GET** /festivos/obtener/{año}

**Parameters** [Cancel](#)

Name	Description
año <sup>*</sup> required	
integer(int32)	2023
(path)	

[Execute](#) [Clear](#)

**Responses**

**Curl**

```
curl -X 'GET' \
'http://localhost:8080/festivos/obtener/2023' \
-H 'accept: */*'
```

**Request URL**  
[http://localhost:8080/festivos/obtener/2023](#)

**Server response**

Code	Details
200	<p><b>Response body</b></p> <pre>{   "festivo": "Año nuevo",   "fecha": "2023-01-01T05:00:00.000+00:00" }, {   "festivo": "Santos Reyes",   "fecha": "2023-01-09T05:00:00.000+00:00" }, {   "festivo": "San José",   "fecha": "2023-03-20T05:00:00.000+00:00" }, {   "festivo": "Jueves Santo",   "fecha": "2023-04-06T05:00:00.000+00:00" }, {   "festivo": "Viernes Santo",   "fecha": "2023-04-07T05:00:00.000+00:00" }, {   "festivo": "Domingo de Pascua",   "fecha": "2023-04-09T05:00:00.000+00:00" }, {   "festivo": "Día del Trabajo",   "fecha": "2023-05-01T05:00:00.000+00:00" } }</pre> <p><a href="#">Download</a></p> <p><b>Response headers</b></p> <pre>connection: keep-alive content-type: application/json date: Wed, 18 Oct 2023 21:18:22 GMT keep-alive: timeout=60 transfer-encoding: chunked vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers</pre>

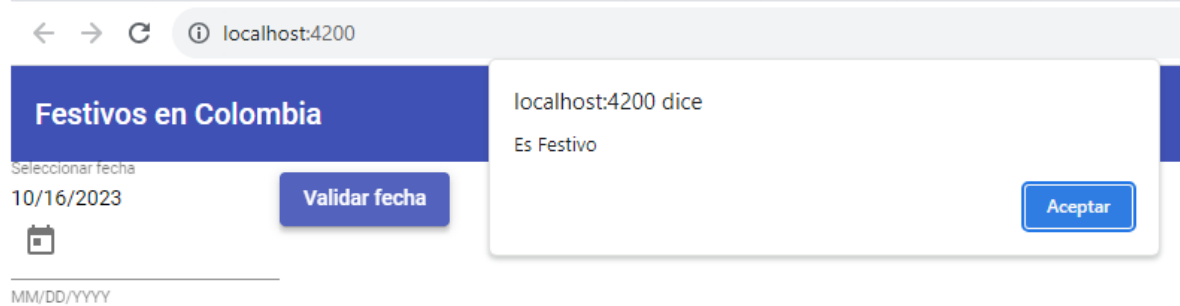
**Responses**

Code	Description	Links
200	<p><b>OK</b></p> <p>Media type</p> <p><a href="#">*/*</a></p> <p>Controls Accept header</p> <p><a href="#">Example Value</a>   <a href="#">Schema</a></p> <pre>{   "festivo": "string",   "fecha": "2023-10-18T21:18:22.121Z" }</pre>	No links

Y un segundo método que devuelve la lista de festivos de un año.

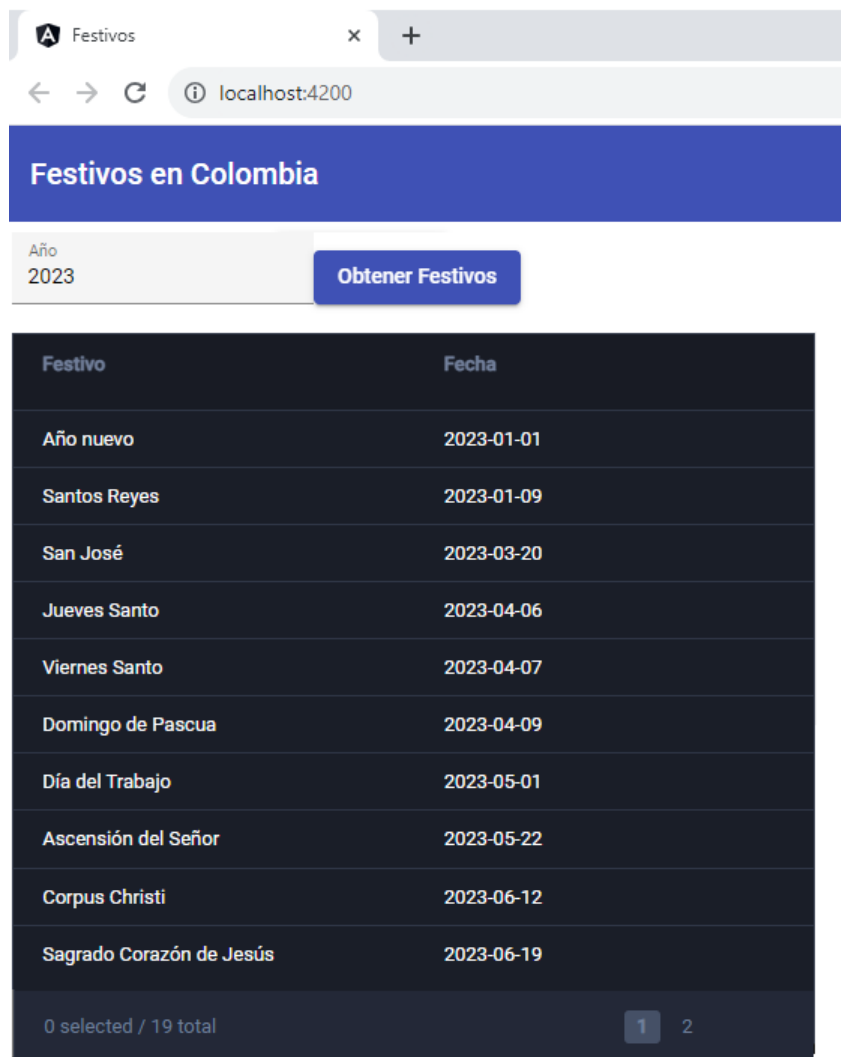
El aplicativo web debe permitir:

- La lectura de una fecha y mostrar si es festivo o no con base en la respuesta de la API



localhost:4200 dice  
Es Festivo

- La lectura de un año y listar los festivos encontrados, también con base en la respuesta de la API



Festivo	Fecha
Año nuevo	2023-01-01
Santos Reyes	2023-01-09
San José	2023-03-20
Jueves Santo	2023-04-06
Viernes Santo	2023-04-07
Domingo de Pascua	2023-04-09
Día del Trabajo	2023-05-01
Ascensión del Señor	2023-05-22
Corpus Christi	2023-06-12
Sagrado Corazón de Jesús	2023-06-19

0 selected / 19 total

El aplicativo debe incluir:

- El servicio que se conecte a la API
- Las entidades para recibir los datos
- Uso de librerías de diseño como Material y ngx-datatable