Manual del usuario para utilizar Angular

David Rodríguez Fernández

Jorge Manuel Ros Tobal

Jose Antonio Juan Prado

Aplicaciones Distribuídas en Internet - Grado en Ingeniería Informática - Universidad de Alicante

¿Qué es Angular?

Angular es una plataforma y un framework para crear páginas clientes utilizando **HTML** y **TypeScript**. De hecho, angular está escrito en TypeScript, aunque se puede trabajar con **javascript** sin problemas.

La arquitectura de Angular sigue ciertos conceptos. Ésta funciona con componentes que están organizados en NgModules. Los **NgModules** recolectan el código y lo convierten en sets funcionales. Por ejemplo, una app de Angular está definida por un set de NgModules. Es decir, que una app siempre tendrá al menos un módulo root que permite varias funcionalidades, como el **bootstrapping**.

Si vemos los componentes, podemos hablar de varias funcionalidades, por ejemplo, los **componentes** permiten definir vistas, que son el grupo de páginas que se muestran y se modifican en base a lo que hayamos programado. Otra funcionalidad son los servicios, los cuales proporcionan funcionalidades específicas que no tienen porqué estar relacionadas con las vistas. Estos servicios pueden ser insertados en componentes como dependencias, haciendo que el código sea reutilizable.

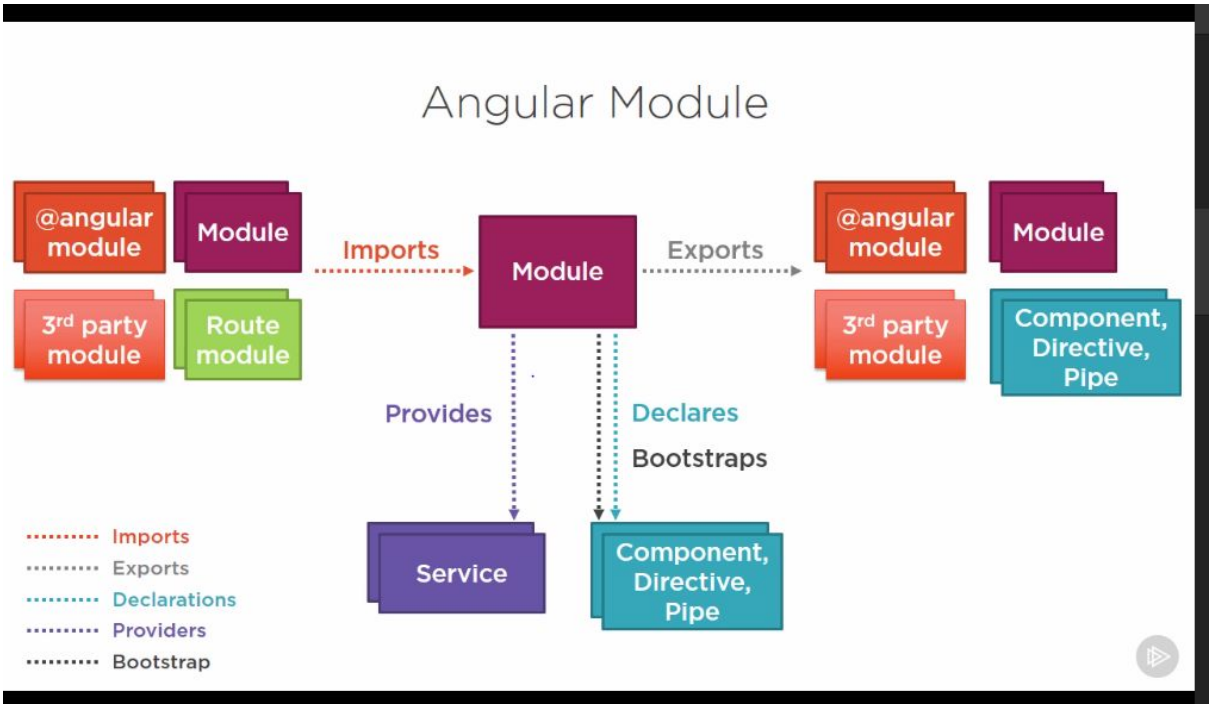
Los módulos, componentes y servicios son clases que usan **decorators**. Estos tienen su tipo y proporcionan **metadata** con la cual, Angular trabaja.
Si analizamos la metadata, tanto para componentes como para servicios, cuando trabaja con una clase componente, se asocia con una **plantilla**, la cual define una **vista**. Esta plantilla trabaja con una combinación de HTML y las directivas de Angular, que permiten modificar ese HTML. Con respecto a los servicios, la metadata proporciona la información necesaria a Angular para que los componentes puedan trabajar con la dependencia.

Módulos

Los **NgModules** de angular se diferencian de los módulos JavaScript (ES2015) y los complementan. Un módulo NgModule declara un contexto de compilación para un conjunto de componentes.

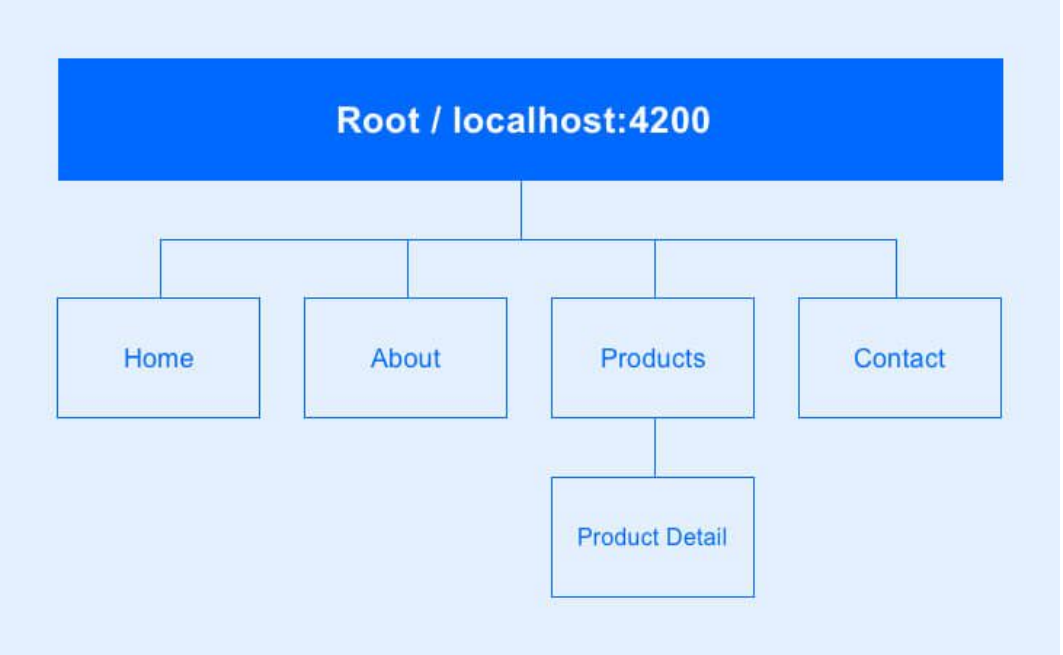
Cada app de Angular tiene un módulo root, que por lo general tiene el nombre de **AppModule,** la cual provee el mecanismo para que funcione la aplicación.

Estos módulos pueden importar funcionalidades de otros módulos, además de permitir exportar sus funcionalidades para ser usados por otros. Un ejemplo es el **router**.



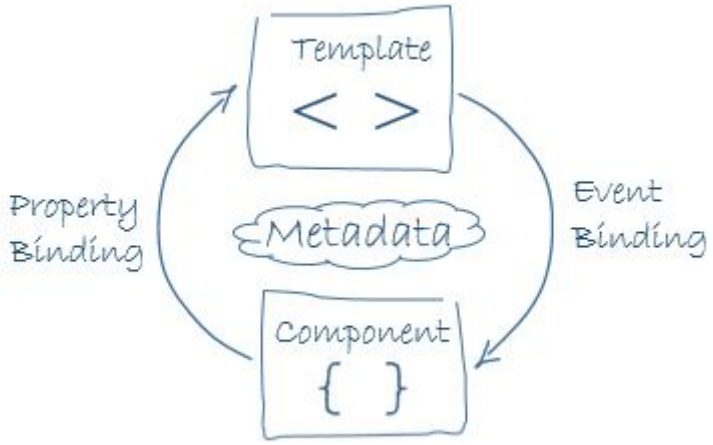
Router

Angular **router** es un NgModule que permite establecer el camino que se va a seguir por nuestra aplicación, mostrando diferentes estados y vistas. Cuando se introduce una URL, esta es procesada y se lleva a la correspondiente página. Como mencioné anteriormente, esto permite navegar de una forma predefinida por la página. Tanto por la URL como por links asociados dentro de ella. Un ejemplo de link asociado es un botón de login que te lleve a la vista de login. O que cuando logueas en la aplicación, esta te redirija a otra página automáticamente.



Componentes

Cada aplicación de Angular tiene al menos un componente, el componente root que conecta la jerarquía con las páginas. Cada componente define una clase que contiene datos y lógica de aplicación está asociada a una plantilla HTML, la cual define la vista que se mostrará al usuario final.



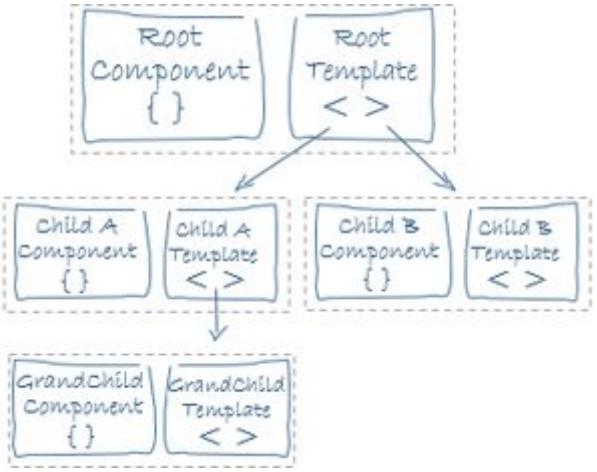
Un componente, básicamente consta de 3 archivos: un **CSS**, una plantilla **HTML** y un archivo de código **TypeScript(o javascript)**. Mediante esta composición, generamos un trozo de código completo que es encapsulado y puede ser reutilizado en distintas plantillas de la aplicación, esta es la razón de ser de los componentes. Por lo general, estos componentes se ubican como subcarpetas de la carpeta principal, la carpeta app. Los componentes se identifican mediante el decorador **"@Component**". A este decorador habrá que pasarle un objeto de JS para configurarlo y hacerlo válido. La información que se incluye se almacenará como meta data en la clase asociada al decorador. Esta configuración se compone de:

- selector: El nombre que tendrá el componente como etiqueta de HTML.
- templateURL: Referencia a la plantilla HTML de este componente.
- styleUrls: Referencia a la hoja de estilos del componente.
- providers: Un array de "providers" para servicios que el componente necesite.

Es importante reseñar que la clase que formamos como componente debemos de exportarla, dado que si no, no podríamos usarla fuera de su ámbito.

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

  msg:string = '';
```



Servicios y dependencias

Para datos o lógica que no está asociada a una vista específica y se quiere usar en distintos componentes, aquí es cuando se crea una clase servicio que será usada por uno o varios componentes. Este servicio va seguido por **@Injectable()** decorator. El decorator provee la metadata que permite a los otros componentes, se dependencias de tu clase.

El servicio puede abarcar cualquier valor, función o característica que necesite una aplicación. Normalmente, esta clase tiene un propósito y unas funcionalidades definidas para hacer hacer un trabajo específico.

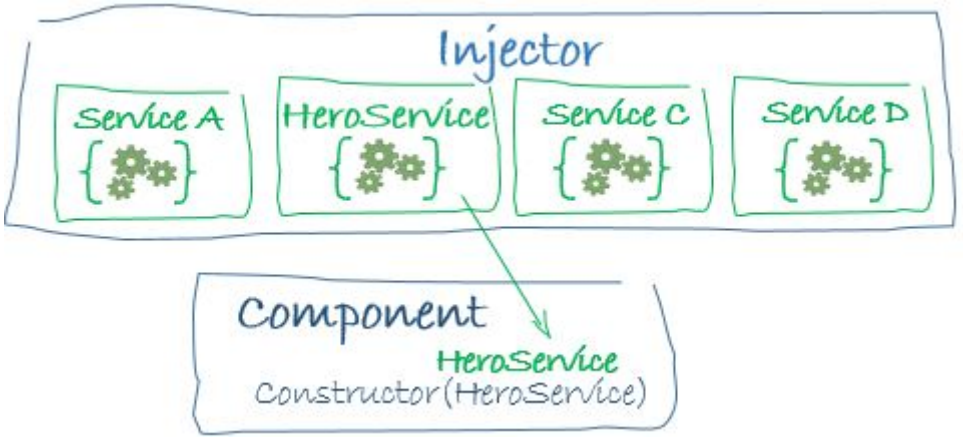
Angular distingue los componentes de los servicios para aumentar la modularidad y la reutilización, permitiendo así, que las clases de componentes sean simples.

Un posible ejemplo de servicio es el siguiente, el cual muestra datos por pantalla

```
export class Mensajes{
        saludo(msg: any)    { console.saludo(msg); }
        despedida(msg: any) { console.despedida(msg); }
        }
}
```

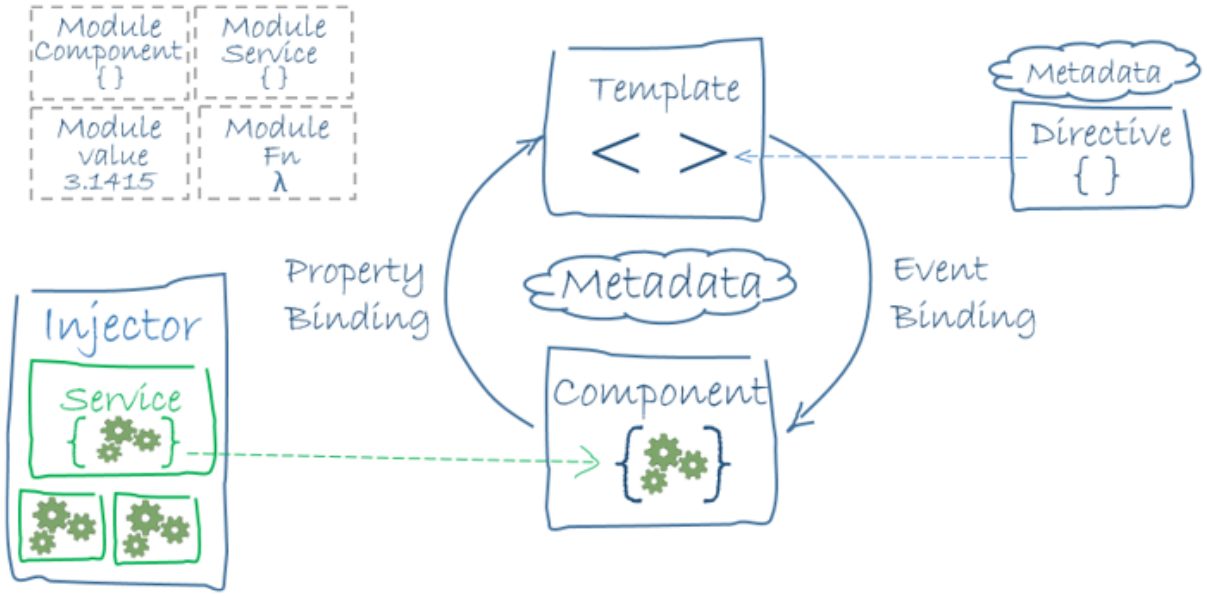
Los servicios pueden estar compuestos por otros, creando dependencia entre ellos y reutilizando sus funcionalidades.

A un componente puedes añadirle un servicio, esto es lo que se conoce como inyección de servicio. Esto hace que el componente tenga acceso a la clase que nos ofrece el servicio. Un ejemplo básico es que tenemos una clase que procesa productos, y en la vista de productos añadimos ese servicio, para que el componente procese los productos usará los servicios de la clase Producto




```
constructor(private service: HeroService) { }
```

Esta es una imagen de la página oficial de angular que muestra como se complementa todo entre sí.

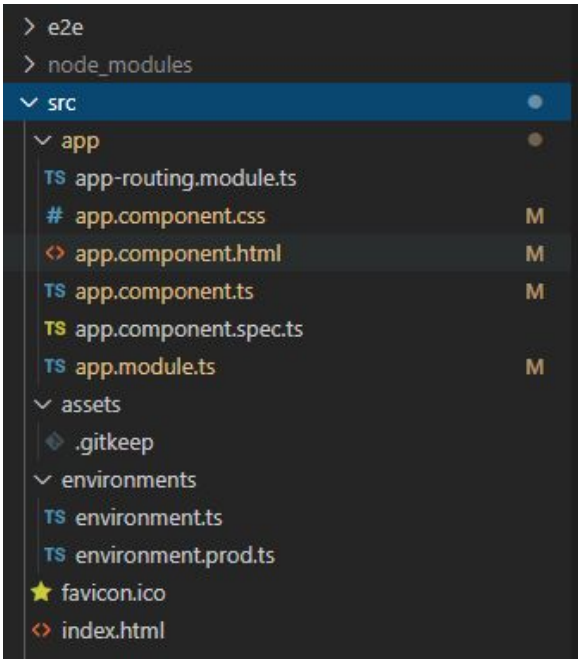


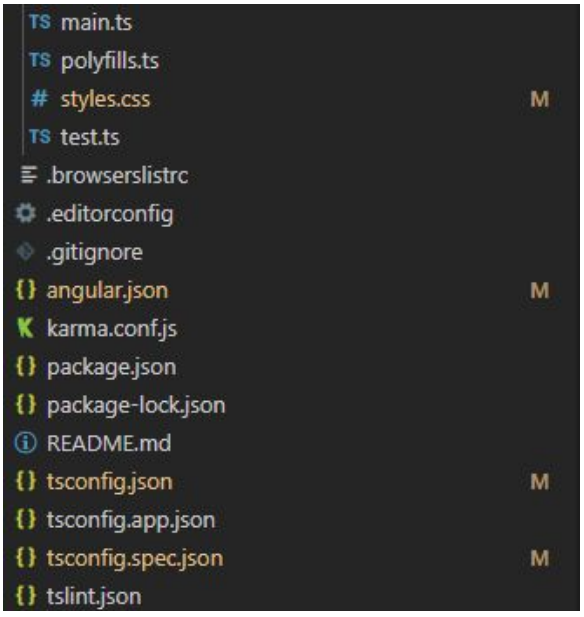
Todo junto, un **componente** y una **plantilla** definen una **vista angular**.

- Un decorator en una clase de componente agrega los metadatos, incluido un puntero a la plantilla asociada.
- Las directivas y el marcado de enlace en la plantilla de un componente modifican las vistas en función de los datos y la lógica del programa.
- El inyector de dependencia proporciona servicios a un componente, como el servicio de router, que le permite definir la navegación entre vistas.

¿Cómo funciona Angular?

Estructura de carpetas y archivos





Al realizar un nuevo proyecto, Angular nos genera un nuevo espacio de trabajo, el cual se compone de las siguientes carpetas:

- **e2e**: Su nombre proviene de las siglas “end to end”, y reúne una serie de ficheros cuya función es la realización de tests automatizados.
- **node_modules**: Esta carpeta contiene todas las dependencias de nuestro proyecto, esta carpeta puede llegar a pesar varios gigas, por lo que hay que ser cuidadoso con las dependencias que no se usen.
- **Src**: Es la carpeta en la cual trabajaremos con los módulos de angular en sí mismos, además es la más importante ya que contendrá todo el código. Éste código se estructura en dos subcarpetas:
    - **app**: Aquí se ubica toda la implementación de los componentes principales, junto a la plantilla html y los archivos de estilo css.
    - **assets**: Contendrá los archivos adicionales que hacen que el proyecto funcione.

  Además, en la raíz de la carpeta src se encuentran estos archivos:
    - **favicon.ico**: Es el icono del proyecto.
    - **index.html**: La página principal del proyecto.
    - **main.ts**: Es el archivo Type Script inicial en el cual se podrán configurar las configuraciones globales del proyecto.

- **.gitignore**: Se definen las carpetas y archivos que debe de ignorar git cuando lo añadamos a un repositorio remoto.
- **angular.json**: Contiene la configuración de angular. Cosas como rutas, versiones…
- **package.json**: Aquí es donde se encuentra la configuración de nuestra aplicación. Contiene el nombre de la app, las dependencias necesarias para que se ejecute y especifica las versiones de éstas. En definitiva, hace que sea más fácil compartir el proyecto con otra gente, ya que contiene todo lo necesario para su ejecución.
- **README.md**: Este es un fichero reservado para que el programador pueda añadir la información que considere necesaria compartir.
- **tsconfig.json**: Contiene la configuración TypeScript.

Ejemplo práctico

NOTA: Las siguientes instrucciones se realizan en el sistema operativo windows, puede que haya discrepancias en cualquier otro sistema operativo.

Antes de poder ejecutar el proyecto debemos tener instalado npm (como hemos visto a principio de curso) y angular mediante este comando:

- **npm install -g @angular/cli@latest**

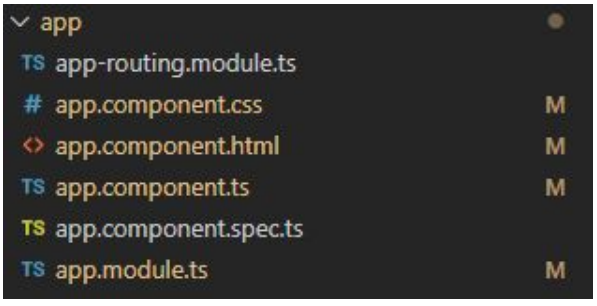
Ahora debemos crear el proyecto y descargar todas las dependencias para ello vamos a seguir los pasos mostrados a continuación.
Primero debemos abrir un terminal como administrador (Esto se debe a que durante la realización de esta práctica hemos tenido problemas). A continuación, nos debemos posicionar mediante el comando 'cd' en el directorio ejecutamos el siguiente:

- **ng new [nombre del proyecto]**

Nos preguntará si queremos realizar rutas ahí ya depende de las necesidades de cada uno, nosotros le dimos a que sí. Ahora nos preguntará qué formato de estilo queremos utilizar, nosotros seleccionamos CSS (con las flechas del teclado) y pulsamos espacio para seleccionar. Para iniciarlo debemos poner el siguiente comando:

- **ng serve**

Una vez creado el proyecto nos iremos a la estructura de carpetas y nos metemos en la carpeta 'app'. Y veremos los siguientes archivos:

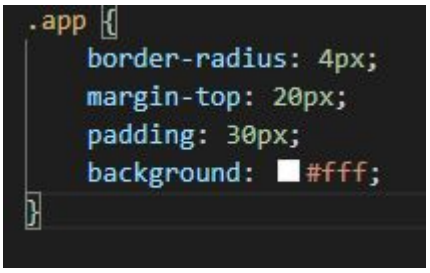


Empecemos en 'app.component.css'. Este archivo contiene toda la información de estilos de nuestro componente. Pueden crearse más para la realización de estilos de más componentes y siempre tiene que acabar con la misma extensión.

Ahora vamos a 'app.component.html' la cual contiene el código en lenguaje html para visualizar nuestras distintas tareas, datos, etc.
A continuación veremos 'app.component.ts'. Contiene toda la información de nuestro componente variables, métodos, constructores, etc. Como hemos visto en el apartado de **Componentes** debemos indicar nuestra plantilla html y nuestros estilos (en nuestro caso como hemos creado en el proyecto con CSS usamos este).
Por último vemos el apartado 'app.module.ts'. Aquí es donde importamos todos los módulos que necesitemos para la elaboración de nuestro proyecto.

A continuación mostraremos capturas de cómo hemos realizado nuestro proyecto:

- app.component.css



- app.component.html

- app.component.ts

```
cars = [
  {'name': 'A5', 'marca': 'Audi', 'color':'Gris'},
  {'name': 'RX-8', 'marca': 'Mazda', 'color':'Rojo'},
  {'name': 'Camaro', 'marca': 'Chevrolet', 'color':'A
];

model:any = {};
model2:any = {};
hideUpdate:boolean = true;

addCar():void{
  this.cars.push(this.model);
  this.msg = 'campo agregado';
}

deleteCar(i):void {
  var answer = confirm('Estas seguro querer eliminarl
  if(answer) {
    this.cars.splice(i, 1);
    this.msg = 'campo eliminado';
  }
}
```

- app.module.ts

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

import { FormsModule } from '@angular/forms';

import { AppRoutingModule } from './app-routi
import { AppComponent } from './app.component

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

Github: https://github.com/gelio98/EjemploAngularADI2020
Drive:
https://drive.google.com/file/d/1WWv8fKkQ2N7IEIy0qqfBuoIiXldcAScf/view?usp=sharing
Youtube: https://youtu.be/gW-9VOAZssw

Bibliografía

https://es.wikipedia.org/wiki/Angular_(framework)

https://angular.io/start

https://desarrolloweb.com/articulos/introduccion-angular2.html

https://www.w3schools.com/angular/

https://www.youtube.com/watch?v=kqYuyACFVkE&ab_channel=VictorRoblesWEB