

# ANGULAR JS



David Dorante Lucas  
Álvaro Sánchez Moreno  
José María Mañero Brenes  
Juan Martín Candela  
Antonio Jesús León Fernández

# ÍNDICE

<b>Módulo 1: Introducción a angular</b>	<b>3</b>
1.1. ¿Qué es angular?	3
1.1.1. Historia y evolución.	3
1.1.2. Características principales.	4
1.1.3. Diferencias entre angular y otras tecnologías:	5
1.2. Configuración del entorno de desarrollo.	7
1.2.1. Instalación de Node.js y npm.	7
1.2.2. Instalación de Angular CLI.	9
<b>Módulo 2: Conceptos básicos.</b>	<b>10</b>
2.1 Componentes en Angular.	10
2.1.1 Creación de un componente.	10
2.1.2 Propiedades y eventos.	12
2.2. Directivas en Angular.	14
2.2.1. Directivas estructurales.	14
2.2.2. Directivas de atributo	15
2.3. Servicios y dependencias.	16
2.3.1. Creación de servicios.	16
2.3.2 Inyección de dependencias	19
<b>Módulo 3: Angular CLI y Proyecto.</b>	<b>20</b>
3.1. Creación de un nuevo proyecto.	20
3.1.1. Estructura del proyecto.	23
3.1.2. Archivos principales.	24
3.2. Uso básico de Angular CLI	26
3.2.1. Comandos esenciales.	26
3.2.2. Generación de componentes, servicios, y módulos.	27

# Módulo 1: Introducción a angular

## 1.1. ¿Qué es angular?

### 1.1.1. Historia y evolución.

Angular es un marco (framework) de desarrollo de aplicaciones web desarrollado y mantenido por Google. Su historia y evolución se remontan a varios años. Aquí hay un resumen de las principales etapas:

- **AngularJS (2010):**

AngularJS fue lanzado por Google en 2010 y fue la primera versión de Angular.

Fue creado por Misko Hevery y Adam Abrons como un marco de JavaScript para construir aplicaciones de una sola página (SPA).

Introdujo la vinculación de datos bidireccional, una característica que permitía la sincronización automática entre la vista y el modelo.

- **Angular 2+ (2016 en adelante):**

Angular 2, lanzado en 2016, fue una reescritura completa de AngularJS y no fue compatible con la versión anterior.

Se reescribió en TypeScript, un superconjunto tipado de JavaScript.

Introdujo el concepto de componentes como bloques de construcción fundamentales y mejoró significativamente el rendimiento.

Adoptó un enfoque más modular y se centró en el desarrollo de aplicaciones de grandes dimensiones.

- **Angular 4 (2017):**

La versión 3 fue omitida para evitar confusiones con otra librería llamada Router 3.

Angular 4 fue una actualización menor pero trajo mejoras en el rendimiento y nuevas características.

- **Angular 5, 6, 7, 8 (2017-2019):**

Estas versiones siguieron mejorando el rendimiento, la seguridad y la facilidad de uso.

Se introdujeron características como la compilación incremental, mejoras en el servicio de HTTP y actualizaciones en Angular CLI.

- **Angular 9 (2020):**

Introdujo la compatibilidad total con TypeScript 3.7, mejoras en la compilación y la actualización de dependencias.

También presentó Ivy, una nueva arquitectura de renderizado y compilación que mejoró el rendimiento y redujo el tamaño de las aplicaciones.

- **Angular 10, 11, 12 (2020-2021):**

Estas versiones continuaron mejorando la estabilidad, el rendimiento y la facilidad de uso.

Se introdujeron características como actualizaciones en Angular CLI, mejoras en Ivy y optimizaciones de rendimiento.

- **Angular 13 (2022):**

Las versiones más recientes continuarán evolucionando y mejorando la plataforma, y es probable que introduzcan nuevas características y mejoras de rendimiento.

A lo largo de su evolución, Angular ha mantenido su posición como un marco sólido y robusto para el desarrollo de aplicaciones web, especialmente para proyectos empresariales y de gran envergadura.

La comunidad de Angular, junto con el equipo de desarrollo de Google, ha trabajado constantemente para mejorar y mantener la calidad y la eficiencia del marco.

### 1.1.2. Características principales.

Separa el frontend y el backend de una aplicación de una forma bastante clara, para evitar así que ambos se mezclen.

Permite simplificar mucho el código, ya que es un framework que nos permite realizar bastantes cosas con poco código., siguiendo el modelo vista controlador y basándose en los componentes.

Además, una de las características que tiene es que es un framework de código abierto, permitiendo así que todo el mundo pueda leerlo, usarlo y editarlo.

Angular utiliza plantillas para definir la interfaz de usuario de la aplicación, combinando estas plantillas con el código del componente para renderizar la vista final que el usuario ve.

Está construido con TypeScript, mejorando la capacidad de mantenimiento y detección de errores en el código.

Son objetos singleton, proporcionando funcionalidades compartidas en toda la aplicación.

Utiliza un sistema de inyección de dependencias para gestionar y organizar componentes, facilitando la creación, prueba y mantenimiento de aplicaciones al permitir una mejor modularidad.

### 1.1.3. Diferencias entre angular y otras tecnologías:

A continuación vamos a comparar Angular con varios frameworks que al igual que Angular, te permite construir aplicaciones de una sola página, por lo tanto podemos decir que son la competencia del mercado. Estos frameworks son Vue.js y React.

#### 1. Arquitectura y Complejidad:

- **Angular:** Utiliza una arquitectura más completa y estructurada. Proporciona una solución integral para el desarrollo de aplicaciones mediante el uso de módulos, servicios, directivas y otros conceptos.
- **React:** Es una biblioteca para construir interfaces de usuario y requiere la elección de otras bibliotecas y herramientas para tareas como enrutamiento y gestión de estado.
- **Vue.js:** Se encuentra en un punto intermedio, ofreciendo una estructura más flexible que Angular, pero más estructurada que React.

## 2. Lenguaje de Plantillas y JSX:

- **Angular:** Utiliza lenguaje de plantillas HTML con extensiones propias para la manipulación de datos.
- **React:** Emplea JSX (JavaScript XML) para definir componentes, que es una extensión de JavaScript que se parece a XML/HTML.
- **Vue.js:** Utiliza una combinación de lenguaje de plantillas y opciones de renderizado renderizadas en el lado del cliente.

## 3. Gestión de Estado:

- **Angular:** Proporciona su propio sistema de gestión de estado, principalmente a través de servicios y la arquitectura de unidireccionalidad de datos.
- **React:** Requiere soluciones externas para la gestión de estado, como Redux o el uso de context API.
- **Vue.js:** Ofrece una forma más sencilla de gestionar el estado dentro de los componentes.

## 4. Curva de Aprendizaje:

- **Angular:** Tiene una curva de aprendizaje más compleja debido a su arquitectura completa y a la variedad de conceptos que introduce.

- **React:** Es más fácil de aprender para aquellos familiarizados con JavaScript y JSX, pero puede volverse más complejo a medida que se integran más bibliotecas externas.
- **Vue.js:** Es conocido por su curva de aprendizaje suave, siendo más fácil para los principiantes.

## 5. Tamaño y Rendimiento:

- **Angular:** Puede tener un tamaño de paquete inicial más grande, pero ofrece optimizaciones para el rendimiento.
- **React:** Puede tener un tamaño más pequeño y proporcionar flexibilidad para optimizar el rendimiento según las necesidades del desarrollador.
- **Vue.js:** Tiende a tener un tamaño más pequeño que Angular y ofrece un rendimiento competitivo.

## 1.2. Configuración del entorno de desarrollo.

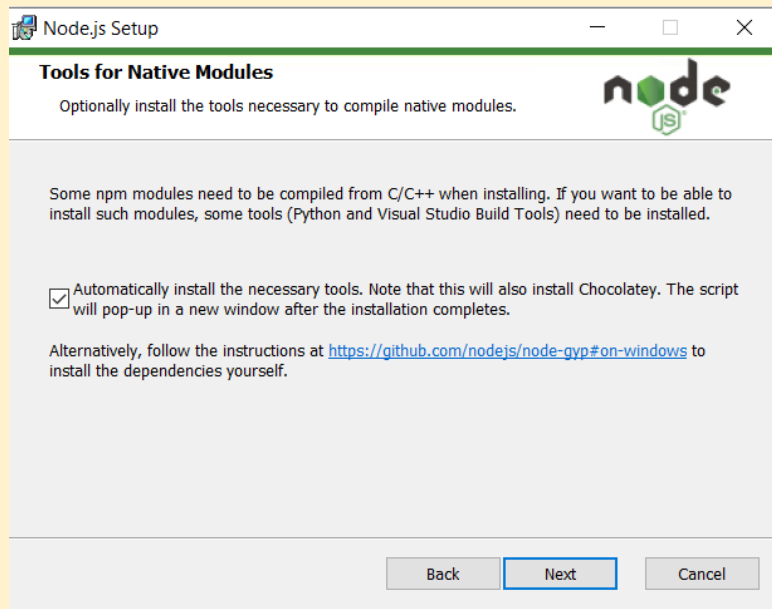
### 1.2.1. Instalación de Node.js y npm.

1.- El primer paso para llevar a cabo la instalación de node sería ir al apartado de descargas de su página principal [[Node](#)] y descargar la versión correspondiente a nuestro dispositivo.

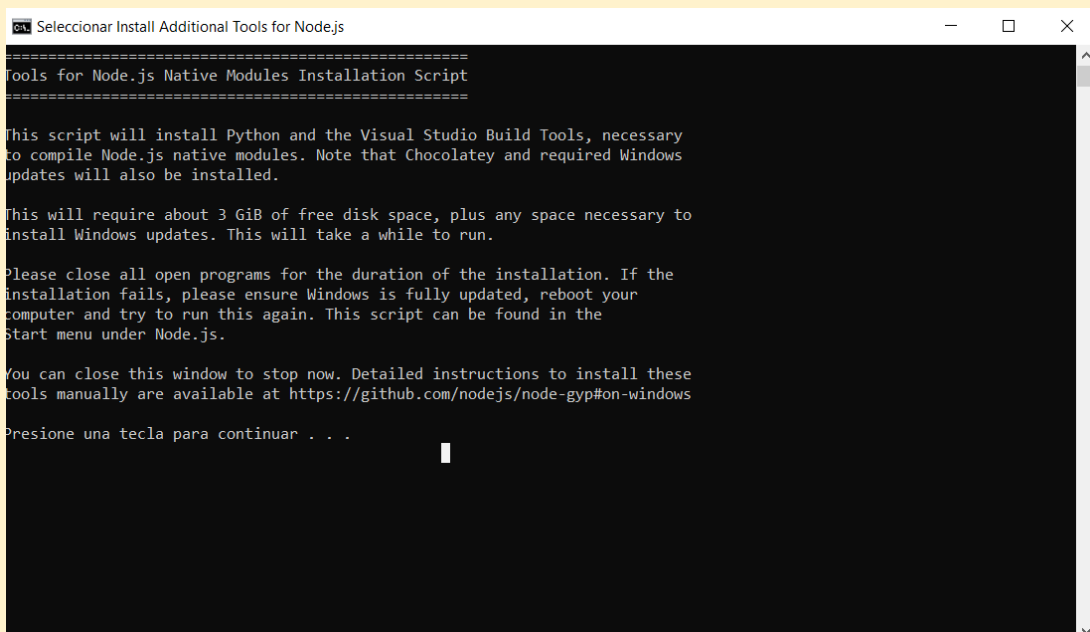
The screenshot shows the Node.js download page. It has two main tabs: 'LTS' (Recommended For Most Users) and 'Current' (Latest Features). Under 'LTS', there are three download options: 'Windows Installer' (node-v20.10.0-x64.msi), 'macOS Installer' (node-v20.10.0.pkg), and 'Source Code' (node-v20.10.0.tar.gz). Below these, there is a list of download links: 'Windows Installer (.msi)', 'Windows Binary (.zip)', 'macOS Installer (.pkg)', 'macOS Binary (.tar.gz)', 'Linux Binaries (x64)', 'Linux Binaries (ARM)', and 'Source Code'. To the right of this list is a table showing the compatibility of Node.js versions with different architectures.

32-bit	64-bit	ARM64
32-bit	64-bit	ARM64
64-bit / ARM64		
64-bit	ARM64	
64-bit		
ARMv7	ARMv8	
node-v20.10.0.tar.gz		

**2.-** Una vez descargado, procederemos a ejecutar el archivo e instalar node, presionando el botón de Next en todas las ventanas emergentes y marcando la opción que se puede observar en la anterior imagen.



**3.-** Cuando finalice la instalación de node, se abrirá una terminal con el siguiente mensaje. Simplemente se debe presionar cualquier tecla para proceder con la instalación de los componentes adicionales que son necesarios para un correcto funcionamiento de node.





**4.-** Una vez finalizada la instalación de todos los componentes necesarios, comprobaremos que la instalación se ha llevado a cabo de manera correcta escribiendo los comandos representados en la imagen dentro de la terminal. Estos comandos sirven para ver la versión de node instalada y la versión de npm instalada.

```
C:\Users\juanp>node --version
v20.10.0

C:\Users\juanp>npm --version
9.8.1
```

### 1.2.2: Instalación de Angular CLI.

**1.-** Para comenzar con la instalación de Angular CLI, ejecutaremos desde la terminal el siguiente comando “npm install -g @angular/cli”. Este comando instala de manera global el paquete de Angular CLI en nuestro equipo para que podamos utilizarlo desde cualquier ruta de nuestro equipo.

```
C:\Users\juanp>npm install -g @angular/cli
[Progress Bar] - idealTree: timing idealTree Completed in 6113ms
```

**2.-** Para comprobar que Angular CLI se ha instalado de manera correcta ejecutamos el comando: “ng version”. Al ejecutar este comando debería de salir la pantalla mostrada en la imagen anterior dónde se detalla la versión de Angular instalada y la versión de Node.

```
C:\Users\juanp>ng version
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

  ng analytics disable --global

Global setting: enabled
Local setting: No local workspace configuration file.
Effective status: enabled

Angular CLI

Angular CLI: 17.0.6
Node: 20.10.0
Package Manager: npm 9.8.1
OS: win32 x64
```

## Módulo 2: Conceptos básicos.

### 2.1 Componentes en Angular.

#### 2.1.1 Creación de un componente.

Un componente en Angular es una parte de la web que contiene vista, estilos y lógica, pudiendo crear tus propias etiquetas HTML y reutilizarlas en todas las páginas.

Los componentes de Angular pueden recibir objetos javascript complejos y arrays para su configuración, recibiendo un array de usuarios a pintar.

Dentro de los componentes también se pueden llamar o hacer uso de otros componentes, adquiriendo una estructura en forma de árbol.

Conociendo un poco qué son los componentes, vamos a pasar a la creación de estos:

#### - Angular CLI.

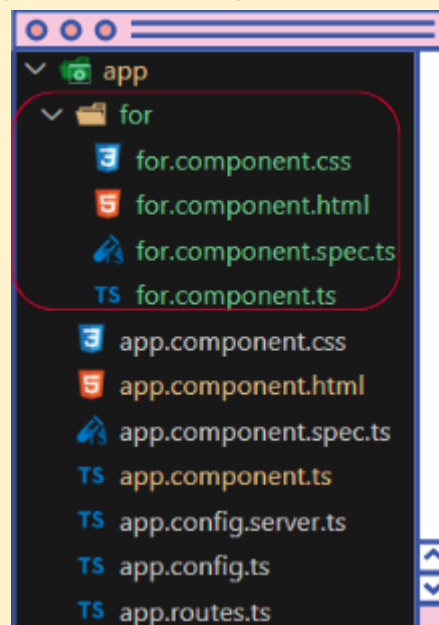
1) Abrimos el terminal y ejecutamos el siguiente comando:

```
ng generate component nombre_componente_a_crear.
```

Si queremos crear un componente con varias palabras, separamos cada palabra por guiones, en vez de por espacios.

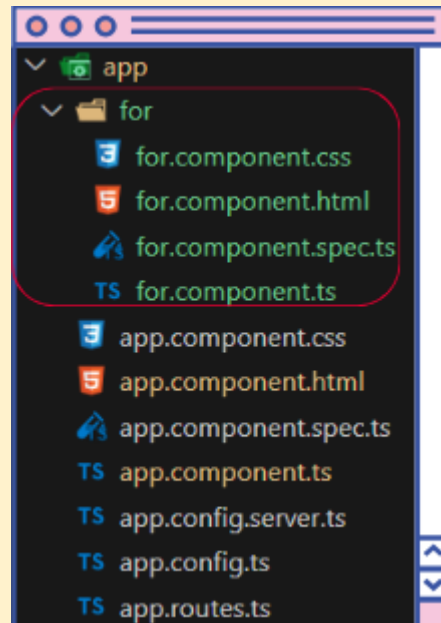
```
PS C:\Users\Davixu\2ºDAW\horaslibreconfiguracion\ProyectoAngular> cd hola_mundo
PS C:\Users\Davixu\2ºDAW\horaslibreconfiguracion\ProyectoAngular\hola_mundo> ng generate component for
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. No
Global setting: enabled
Local setting: disabled
Effective status: disabled
CREATE src/app/for/for.component.html (18 bytes)
CREATE src/app/for/for.component.spec.ts (575 bytes)
CREATE src/app/for/for.component.ts (222 bytes)
CREATE src/app/for/for.component.css (0 bytes)
```

- 2) Dentro de la carpeta app podemos comprobar como se ha creado un archivos css, un archivo html, un archivo TypeScript y un archivo TypeScript para tests.



## - Angular.

- 1) Tenemos que crear los archivos de los componentes y de las etiquetas a mano. Creamos una carpeta dentro de la carpeta app con el nombre del componente que quieras. Crearemos los 4 archivos (html, css, TypeScript):



2) Dentro de la carpeta creamos un archivo llamado “*nombre-componente.component.ts*”, por ejemplo, *form.component.ts*

3) Dentro del componente escribimos la siguiente estructura:

```
// app/for/for.component.ts
import { Component } from "@angular/core";
@Component({
  selector: "app-for",
  standalone: true,
  imports: [],
  templateUrl: "./for.component.html",
  styleUrls: ["./for.component.css"],
})
export class ForComponent {}
```

**a.- Selector:** Es el nombre que va a tener la etiqueta HTML que sirve para usar dicho componente. Para nuestro ejemplo, sería: `<app-for></app-for>`.

**b.- Standalone:** Indica que el componente es independiente.

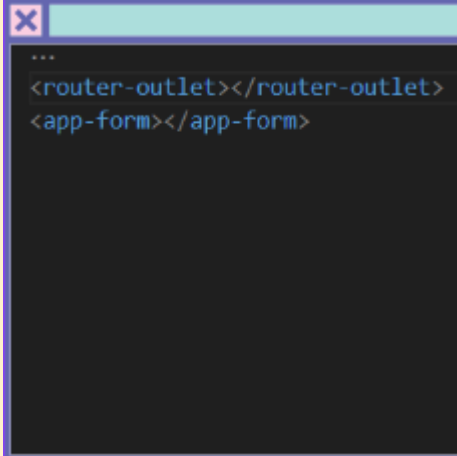
**c.- Imports:** Para importar módulos adicionales.

**d.- TemplateUrls:** Especifica la ubicación de los archivos HTML que contiene la estructura del componente.

**e.- StyleUrl:** especifica la ubicación de los archivos que aplican estilos al componente.

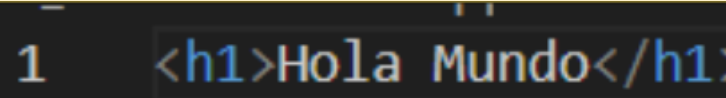
**f.- La clase ForComponent:** Declara la clase FormComponent como una clase de TypeScript y exporta el componente para que pueda ser utilizado en otros archivos si es necesario.

- 4) En el HTML del componente de la aplicación (se encuentra en la carpeta app: app.component.html) incorporamos las etiquetas que deseemos incorporar. Para incorporar un componente dentro de otro, podremos hacerlo como en el siguiente ejemplo:



```
***  
<router-outlet></router-outlet>  
<app-form></app-form>
```

- 5) En el HTML del componente creado (for.component.html) es donde podremos introducir las etiquetas y elementos de html a mostrar.



```
1 <h1>Hola Mundo</h1>
```

### 2.1.2 Propiedades y eventos.

Las propiedades de un componente en Angular se refieren a las variables o atributos definidos en la clase del componente que determinan su estado y comportamiento. Estas propiedades pueden ser utilizadas para almacenar datos, gestionar el estado interno del componente y facilitar la comunicación entre componentes.

#### - Propiedades:

- 1) **Input Properties:** son propiedades que permiten la comunicación entre componentes, decorándolos con el decorador `@Input()` en el componente hijo. Además, permiten la transferencia de datos desde el componente padre al componente hijo. Por ejemplo:

```
@Input() nombre: string;
```

- 2) **Output Properties:** se utilizan para emitir eventos desde un componente hijo al componente padre. Se suelen decorar con el decorador `@Output()` y son instancias de la clase `EventEmitter`. Por ejemplo:

```
@Output() onEvento = new EventEmitter<string>();
```

- 3) **ngModel:** se utiliza para implementar enlaces bidireccionales en los formularios, permitiendo la sincronización de datos entre el modelo y la vista. Por ejemplo:

```
<input [(ngModel)]="nombre" />
```

Los eventos son acciones que ocurren en la aplicación, como clics de ratón, cambios de entrada, cargas de datos, etc. En el contexto de Angular, los eventos generalmente se refieren a las interacciones del usuario con la interfaz de usuario o a cambios en el estado de la aplicación.

#### - Eventos:

- 1) **Event Binding:** permite la respuesta a eventos del usuario, como clics de botones, cambios en cuadros de texto, etc. Además, se

utiliza el paréntesis `()` para vincular un evento a un método en el componente. Por ejemplo:

```
<button (click)="onClick()">Haz click</button>
```

- 2) **Eventos Personalizados:** Eventos que puedes definir y emitir desde un componente hijo para ser capturados por un componente padre. Estos eventos son útiles cuando necesitas comunicación entre componentes. Por ejemplo:

```
@Output()
```

```
propagar = new EventEmitter<string>();
```

```
this.propagar.emit('Este dato viajará hacia el padre');
```

```
<p><input type="text" [(ngModel)]="mensaje">
```

```
<button (click)="onPropagar()">Propagar</button>
```

```
</p>
```

- 3) **Eventos de ciclo de vida:** Se refieren a una serie de ganchos o métodos específicos que son invocados en momentos clave durante el ciclo de vida de un componente. Estos eventos permiten a los desarrolladores ejecutar código en momentos específicos, como la inicialización, la actualización y la destrucción del componente. Ejemplo:

```
export class EjemploComponente implements OnInit {
```

```
  ngOnInit() {
```

```
    alert('Hola mundo');
```

```
  }
```

```
}
```

## 2.2. Directivas en Angular.

### 2.2.1. Directivas estructurales.

Las directivas estructurales en Angular, son un conjunto de funcionalidades que mejoran la flexibilidad y claridad del código y que permiten a los desarrolladores:

- Controlar y manipular la estructura del DOM en aplicaciones Angular.
  - Facilitar la creación, eliminación o repetición de elementos del DOM.
  - Operan según condiciones específicas o iteraciones sobre colecciones de datos.
  - Proporcionan un mecanismo declarativo para gestionar la representación visual de los componentes.
  - Capacidad para afectar directamente a la estructura del DOM.
- **Directiva @If** : (el usuarioExistente es una propiedad booleana de la clase que la ponemos por defecto en true)

```
@if (usuarioExistente) {
  <p> El usuario existe </p>
} @else {
  <p> El usuario no existe </p>
}
```

En la clase: usuarioExistente:boolean = true;



- **Directiva @for** : Teniendo en cuenta que en la clase tenemos una propiedad 'animales' que es un array de objetos (animales), con 2 atributos cada animal, el id y la especie. El track es para identificar de manera única cada elemento de la lista para realizar un seguimiento de los cambios (en versión 17+ es obligatorio ponerlo)

```
<ul>
  @for (a of animales; track a.id) {
    <li> {{a.especie}} </li>
  }
</ul>
```

En la clase: animales = [ {id=1, especie="perro"}, {etc} ]

Con este ejemplo iteramos por todos los animales (identificándose por su id) para mostrarlos en un listado.

### 2.2.2. Directivas de atributo

Las directivas de atributo en Angular, son un conjunto esencial de características que se utilizan para manipular y controlar atributos y propiedades de elementos del DOM, facilitando la personalización dinámica de atributos y estilos en la interfaz de usuario.

A diferencia de las directivas estructurales, no alteran la estructura del DOM.

Se centra en modificar atributos, estilos y comportamientos de elementos existentes.

Entre las Directivas de Atributo encontramos los diferentes tipos:

- **ngModel**: Se utiliza para implementar binding.
- **ngClass**: Se usa para añadir o eliminar varias clases CSS, para cambiar la apariencia.

- **ngStyle:** Se usa para añadir elementos que cambien la apariencia.

También podemos crear una Directiva de Atributos personalizada, según nuestras necesidades específicas.

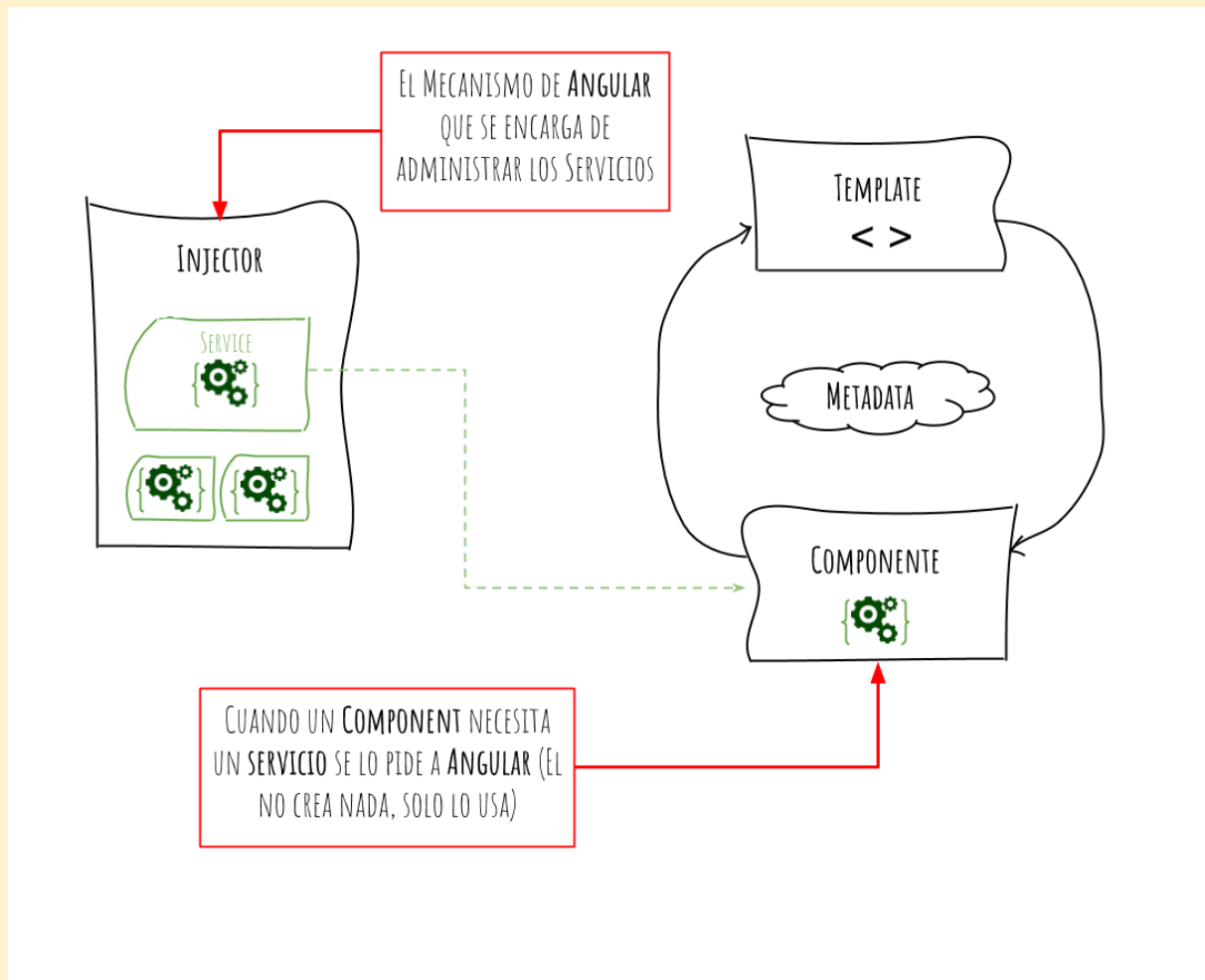
## 2.3. Servicios y dependencias.

Los servicios son clases TypeScript destinadas a encapsular lógica de negocio, manipulación de datos y funcionalidades compartidas en una aplicación Angular, mientras que las dependencias son objetos o servicios que otro objeto necesita para realizar sus servicios.

### 2.3.1. Creación de servicios.

Un servicio es una Clase con un propósito específico. Los Servicios no dependen de ningún Componente, así que varios componentes deberían poder usar el mismo servicio sin problemas. Ofrecen la posibilidad de encapsular las interacciones con el mundo exterior (por ejemplo servicios REST) o de compartir datos en toda tu aplicación. También pueden ofrecer una lógica particular que no dependa de ningún Componente.

Como los servicios están pensados para ser usados por varios componentes, Angular ofrece un mecanismo para facilitar el uso de estos Servicios. Lo que hacemos es definir los Servicios en los Módulos de Angular, éste se encargará de crearlo cuando lo considere conveniente, y luego cuando necesitemos el Servicio, se lo pedimos a Angular.



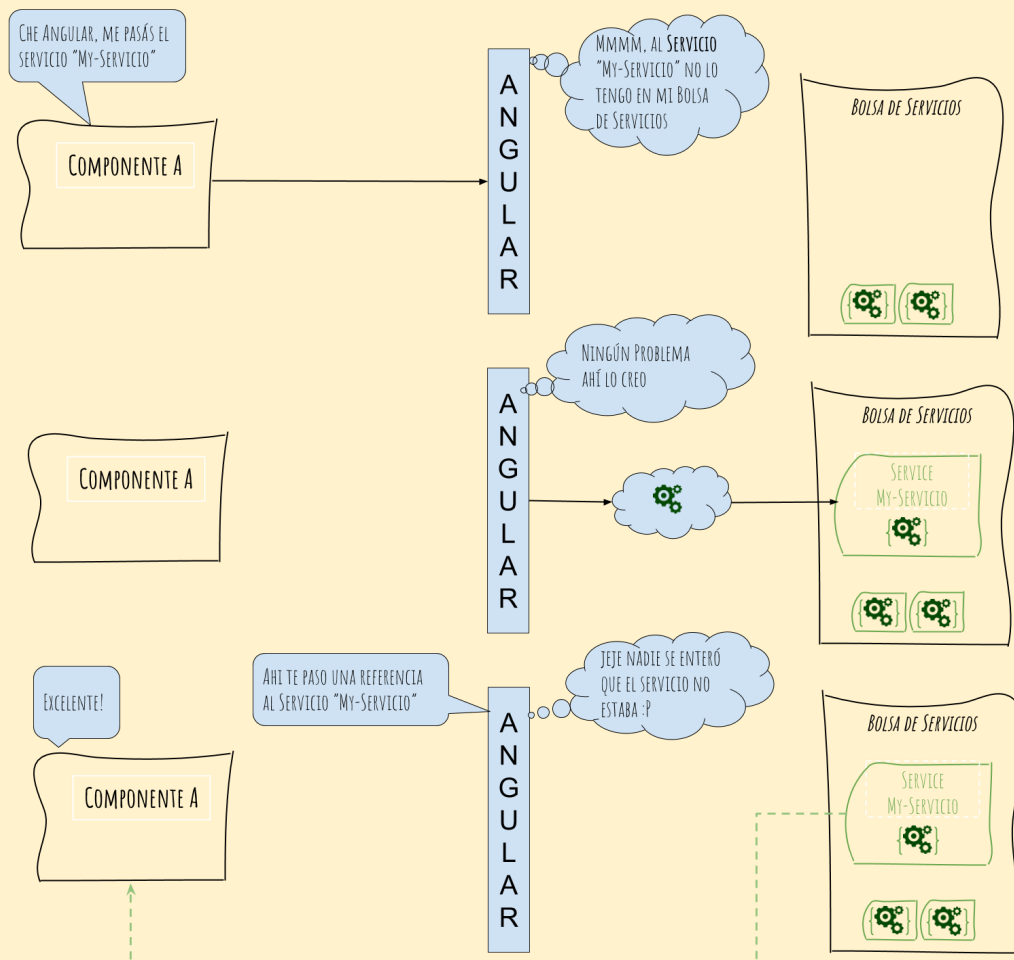
## Funcionalidad de los servicios

Se podría decir que los Servicios están para hacerle la vida más fácil a los Componentes (y sus Templates). Los Componentes solo deberían saber cómo mostrar datos en su Template, ignorando cómo conseguir esos datos a mostrar. Los Servicios sólo deberían saber cómo conseguir o procesar datos, ignorando quién los va a usar. Esto hace que los componentes sean mucho más fáciles de leer y de testear. Y todos felices y contentos.

## Explicación de cómo se crea y usa un Servicio

Los Servicios, al igual que los Component, son clases; la diferencia es que tienen la Metadata `@Injectable()`. ¿Qué significa esto de Injectable? Significa que cualquier Componente, Servicio o Directiva

puede usar este «Injectable» sin preocuparse de crearlo o destruirlo, ya que Angular se encarga de «inyectarle» el servicio para ser usado por quien lo necesite, ahorrándose muchos dolores de cabeza a la hora de crearlo o destruirlo, ya que Angular se encarga de todo eso.



Lo único que hay que hacer es indicar que un Servicio es un «servicio» con la metadata `@Injectable()` registrándolo en Angular, y entonces ya podremos usar ese servicio.

Para crear el servicio con angular cli, simplemente introduciremos en la terminal de nuestro editor (VSCode en este caso) el siguiente comando:

```
ng generate service nombre-del-servicio
```

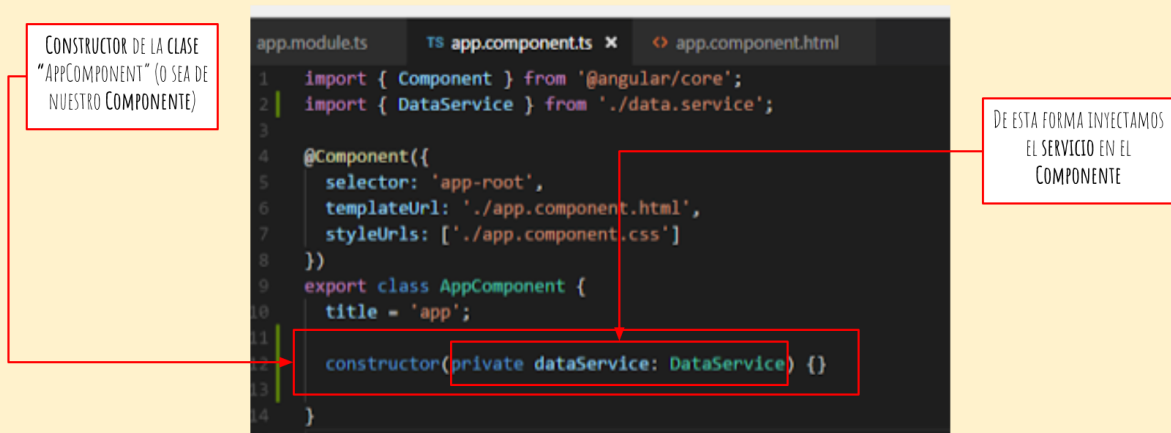
Éste se nos creará dentro de la carpeta app, igual que al crear el componente. Lo único es que no nos creará la carpeta donde contendrá

los archivos, para ello, antes del nombre del servicio a la hora de crearlo, le tendremos que poner una ruta de carpeta, por ejemplo: `ng g service carpeta-para-servicio/nombre-servicio`

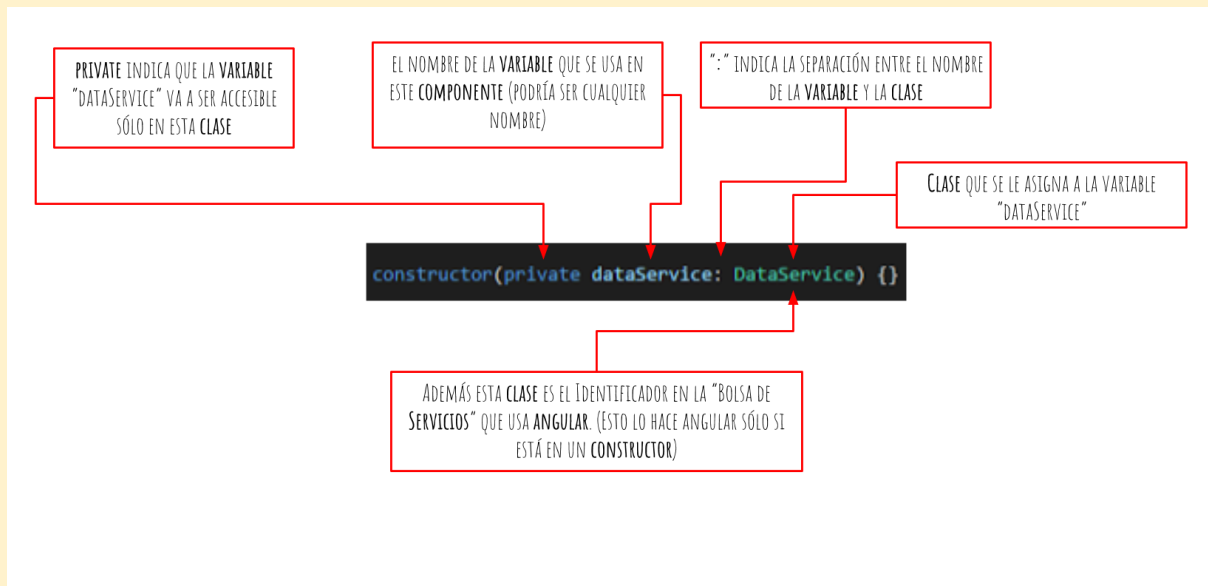
### 2.3.2 Inyección de dependencias

Se encarga de proporcionar las dependencias necesarias a un componente, servicio u otro objeto durante su creación, realizándolo de manera automática, mediante la inyección de dependencias.

- Una vez creado el servicio, pasamos con la inyección de dependencias.
- Para ello, inyectamos el servicio en el constructor del componente sobre el que estamos trabajando. Para ello, lo inyectamos de la siguiente forma. Importamos el servicio y posteriormente lo inyectamos dentro de la clase del componente.



Al haber inyectado, ya simplemente habrá que darle funcionalidad al servicio, teniendo en cuenta que habrá que utilizarlo a través de la instancia creada en la clase del componente



## Módulo 3: Angular CLI y Proyecto.

### 3.1. Creación de un nuevo proyecto.

Angular organiza el desarrollo de aplicaciones web en proyectos, proporcionando una estructura modular y escalable para gestionar el código fuente, dependencias y configuraciones. Un proyecto en Angular es la unidad principal que comprende todos los archivos y carpetas necesarias para construir y ejecutar una aplicación Angular.

1.- Lo primero que debemos hacer es introducir el siguiente comando en el terminal de VS Code. Es importante que el terminal donde introducimos este comando sea cmd y no Powershell o Git Bash.

```
C:\Users\Admin\OneDrive\Escritorio\Angular>ng new ProyectoAngular
```

2.- A continuación, nos preguntará qué hoja de estilos deseamos usar, en nuestro caso le indicamos CSS y luego nos preguntará si estamos seguros e introducimos una "y" le damos a Enter.

```
Microsoft Windows [Versión 10.0.19045.3803]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Admin\OneDrive\Escritorio\Angular>ng new ProyectoAngular
? Which stylesheet format would you like to use? CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? (y/N) █
```

**3.-** Por último, nos pregunta si queremos habilitar el Server Side Rendering (la generación de la página ocurre en el servidor cada vez que un usuario solicita una página), y el Static Site Generation (la generación de la página ocurre durante la fase de compilación, antes de que la aplicación se despliegue en producción) cuya función es mejorar el rendimiento y la experiencia del usuario. Para habilitarlo escribimos “y” y pulsamos en Enter.

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  GITLENS

Microsoft Windows [Versión 10.0.19045.3803]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Admin\OneDrive\Escritorio\Angular>ng new ProyectoAngular
? Which stylesheet format would you like to use? CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? (y/N) y█
```

Cabe destacar que si introducimos una “n” y pulsamos Enter, nos creará un proyecto más simple, pero que no es recomendable debido a que no va a tener las características que nombramos anteriormente ya que no habilitamos el Server Side Rendering y el Static Site Generation, por lo tanto empeorará el rendimiento y la experiencia del usuario. Por lo tanto, esto hará que no aparezca el archivo server.ts.

**4.-** En teoría ya está creado nuestro proyecto, ahora ejecutamos el siguiente comando para asegurarnos que tenemos instaladas todas las dependencias en nuestro proyecto. Cabe destacar que este comando es opcional aunque recomendable, ya que asegura que la instalación ha sido correcta y tienes todas las dependencias instaladas como podemos ver en la primera línea que nos muestra el “up to date”.

```

C:\Users\Admin\OneDrive\Escritorio\Angular\ProyectoAngular>npm install

up to date, audited 987 packages in 5s

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

4 moderate severity vulnerabilities

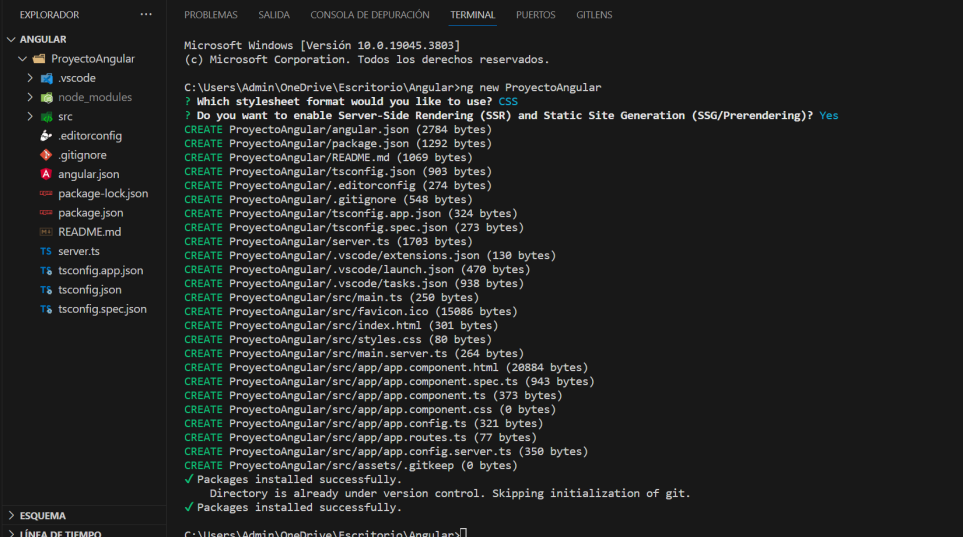
To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

```

También es importante tener en cuenta que debes estar situado en tu carpeta por lo tanto tendrás que introducir el comando `cd ruta-del-proyecto` o te aparecerá un error.

**5.-** Una vez que ya hemos seguido todos los pasos nos debe aparecer lo siguiente en nuestro editor de código VS Code.



```

EXPLORADOR  ...  PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  GITLENS
└─ ANGULAR
  └─ ProyectoAngular
    ├── .vscode
    ├── node_modules
    ├── src
    ├── .editorconfig
    ├── .gitignore
    ├── angular.json
    ├── package-lock.json
    ├── package.json
    ├── README.md
    ├── server.ts
    ├── tsconfig.app.json
    ├── tsconfig.json
    └── tsconfig.spec.json

Microsoft Windows [Versión 10.0.19045.3803]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\Admin\OneDrive\Escritorio\Angular>ng new ProyectoAngular
? Which stylesheet format would you like to use? CSS
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? Yes
CREATE ProyectoAngular/angular.json (2784 bytes)
CREATE ProyectoAngular/package.json (1292 bytes)
CREATE ProyectoAngular/README.md (1069 bytes)
CREATE ProyectoAngular/tsconfig.json (903 bytes)
CREATE ProyectoAngular/.editorconfig (274 bytes)
CREATE ProyectoAngular/.gitignore (548 bytes)
CREATE ProyectoAngular/tsconfig.app.json (324 bytes)
CREATE ProyectoAngular/tsconfig.spec.json (273 bytes)
CREATE ProyectoAngular/server.ts (1703 bytes)
CREATE ProyectoAngular/.vscode/extensions.json (130 bytes)
CREATE ProyectoAngular/.vscode/launch.json (470 bytes)
CREATE ProyectoAngular/.vscode/tasks.json (938 bytes)
CREATE ProyectoAngular/src/main.ts (250 bytes)
CREATE ProyectoAngular/src/favicon.ico (15086 bytes)
CREATE ProyectoAngular/src/index.html (301 bytes)
CREATE ProyectoAngular/src/styles.css (60 bytes)
CREATE ProyectoAngular/src/main.server.ts (264 bytes)
CREATE ProyectoAngular/src/app/app.component.html (20884 bytes)
CREATE ProyectoAngular/src/app/app.component.spec.ts (943 bytes)
CREATE ProyectoAngular/src/app/app.component.ts (373 bytes)
CREATE ProyectoAngular/src/app/app.component.css (0 bytes)
CREATE ProyectoAngular/src/app/app.config.ts (321 bytes)
CREATE ProyectoAngular/src/app/app.routes.ts (77 bytes)
CREATE ProyectoAngular/src/app/app.config.server.ts (350 bytes)
CREATE ProyectoAngular/src/assets/.gitkeep (0 bytes)
✓ Packages installed successfully.
  Directory is already under version control. Skipping initialization of git.
✓ Packages installed successfully.

C:\Users\Admin\OneDrive\Escritorio\Angular>

```

Finalmente, introducimos el siguiente comando para iniciar el servidor de desarrollo de Angular, introducimos una “y” y pulsamos en Enter. Como podemos ver nos aparece entre otras cosas la url localhost que debemos introducir en nuestro navegador.



```

C:\Users\Admin\OneDrive\Escritorio\Angular\ProyectoAngular>ng serve
? Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.io/analytics. Yes

Thank you for sharing pseudonymous usage data. Should you change your mind, the following
command will disable this feature entirely:

  ng analytics disable

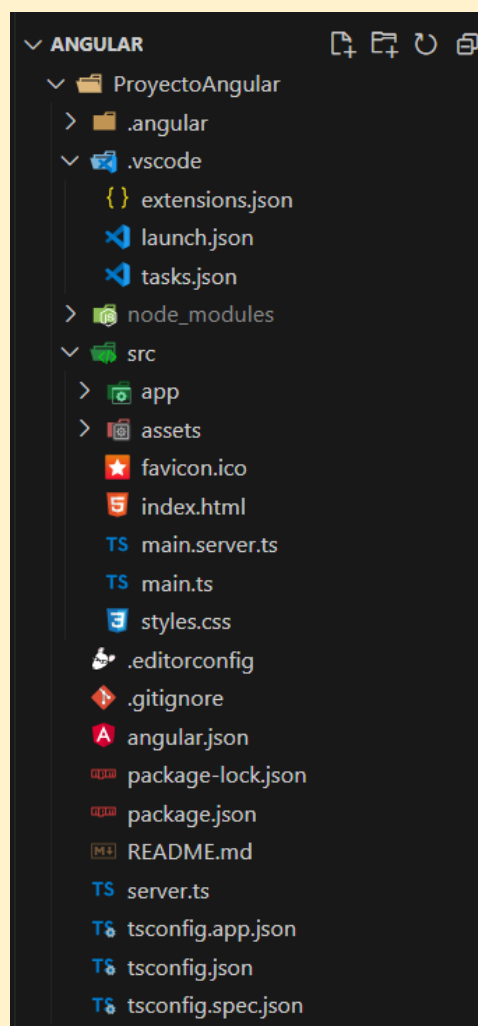
Global setting: enabled
Local setting: enabled
Effective status: enabled

Initial Chunk Files | Names          | Raw Size
polyfills.js        | polyfills      | 82.71 kB |
main.js             | main           | 23.32 kB |
styles.css          | styles        | 95 bytes |
                    | Initial Total  | 106.13 kB

Application bundle generation complete. [17.687 seconds]
Watch mode enabled. Watching for file changes...
→ Local:  http://localhost:4200/

```

### 3.1.1. Estructura del proyecto.



Como podemos observar en la imagen anterior se encuentra nuestra estructura del proyecto de Angular con los diferentes archivos y directorios que aparecen en la imagen.

### 3.1.2. Archivos principales.

Los archivos principales de nuestro proyecto son los siguientes:

- Directorio **.vscode** : contiene configuraciones específicas para Visual Studio Code
  - **extensions.json**: Archivo autogenerado por Angular CLI dónde se indican recomendaciones de extensiones descargables en VSCODE que aportarán ayuda a la hora de utilizar Angular.
- Directorio **node\_modules** : En este directorio se encuentran las dependencias de Node.js y Angular que son necesarias para ejecutar y desarrollar la aplicación.
- Directorio **src** :
  - **assets** : Directorio que contendrá los recursos del proyecto como pueden ser las fuentes, las imágenes, iconos, etc.
  - **styles.css** : Archivo dónde aparecen los estilos globales de nuestra aplicación.
  - **main.ts** : Es el punto de entrada de nuestra aplicación, es decir es el primer archivo que se carga, que importa los ficheros que contiene, etc. En resumen, inicializa la aplicación, viene por defecto y no es necesario modificar nada.

- **index.html** : contiene la etiqueta `<app-root></app-root>` que sería la raíz de nuestra aplicación donde vamos a poner todos nuestros componentes. También aparece el head donde podemos añadir links, algún favicon, estilos, cambiar lenguaje, etc.
  - \* **main.server.ts** : Facilita la inicialización de la aplicación Angular en el lado del servidor mediante el uso de la función ``bootstrapApplication``. (Este archivo solo aparece cuando habilitamos el Server Side Rendering y el Static Site Generation).
  - **favicon.ico** : Es un icono.
- Archivo **.editorconfig** : Contiene configuración sobre el editor la cual viene por defecto y no es necesario modificar nada.
- Archivo **.gitignore** : Aquí aparecen los archivos y directorios los cuales excluirá git del control de versiones.
- Archivo **angular.json** : Aquí podemos ver una configuración de angular donde aparece el tipo de proyecto, la arquitectura donde está toda la configuración, etc.
- Archivo **package.json** : En primer lugar nos aparecen los scripts que utilizaremos, por lo tanto no es necesario utilizar los comandos anteriores.
- Por otro lado contiene las dependencias de angular también vienen por defecto y tiene su parte positiva y negativa. La parte positiva es que viene en el proyecto, por lo tanto no es necesario perder tiempo instalando dependencias por nuestra cuenta (animaciones, formulario, enrutado, observables, testing, etc.). La parte mala es que muchas de las dependencias ya instaladas puede ser que no sean utilizadas en el proyecto, por lo que estarían ocupando hueco de más.
- Archivo **package.lock.json** : Contiene más dependencias de angular y vienen por defecto.

- Archivo **README.md** : Contiene información sobre cómo levantar el entorno de desarrollo, ejecutar tests unitarios, generar un nuevo componente, etc. Podemos decir que contiene una guía con información y sus respectivos comandos sobre cómo realizar algunas acciones sobre el proyecto.
- Archivo \* **server.ts** : Es un archivo TypeScript que contiene la implementación de un servicio. (Este archivo solo aparece cuando habilitamos el Server Side Rendering y el Static Site Generation).
- Archivo **tsconfig.app.json** : Contiene configuración de typescript para la app al igual que para los tests, esta viene ya por defecto por lo que no hará falta modificar nada.
- Archivo **tsconfig.json** de nuevo más configuración de typescript más general y también sin necesidad de modificar nada.
- Archivo **tsconfig.spec.json** configuración de typescript que viene instalado y su configuración viene ya por defecto por lo que no tendremos que cambiar nada aquí. Este archivo en específico sirve para la configuración de los tests.

## 3.2. Uso básico de Angular CLI

Angular CLI (Command Line Interface) es una interfaz de línea de comandos que proporciona utilidades y comandos para desarrollar, construir, probar y desplegar aplicaciones Angular de manera más eficiente. Fue creado por el equipo de Angular para facilitar y agilizar el desarrollo de aplicaciones Angular.

### 3.2.1. Comandos esenciales.

Los comandos más importantes para el uso básico de Angular CLI son:

- **ng new** : Crea un nuevo proyecto Angular.

**ng new nombre-del-proyecto**

- **ng serve** : Inicia el servidor de desarrollo y abre la aplicación en un navegador.

**ng serve**

- **ng build** : Compila la aplicación para producción.

**ng build**

- **ng test** : Ejecuta pruebas unitarias utilizando Karma.

**ng test**

- **ng e2e** : Ejecuta pruebas E2E (end to end) utilizando Protractor

**ng e2e**

- **ng generate o ng g** : Genera componentes, servicios, módulos y otros artefactos.

**ng generate component nombre-del-componente**

**ng g service nombre-del-servicio**

**ng g module nombre-del-módulo**

- **ng lint** : Ejecuta TSLint para analizar el código en busca de posibles problemas.

**ng lint**

### 3.2.2. Generación de componentes, servicios, y módulos.

A continuación vamos a ver los diferentes comandos para la generación de componentes, servicios y módulos:

- Generar un componente :
  - **ng generate component nombre-del-componente**
  - **ng g component nombre-del-componente**
- Generar un servicio :

- `ng generate service nombre-del-servicio`
- `ng g service nombre-del-servicio`
- Generar un módulo :
  - `ng generate module nombre-del-módulo`
  - `ng g module nombre-del-módulo`

Estos comandos sirven para crear de forma rápida y estructurada los diferentes componentes, servicios y módulos, aunque también puedes personalizar aún más estos comandos según tus necesidades específicas.