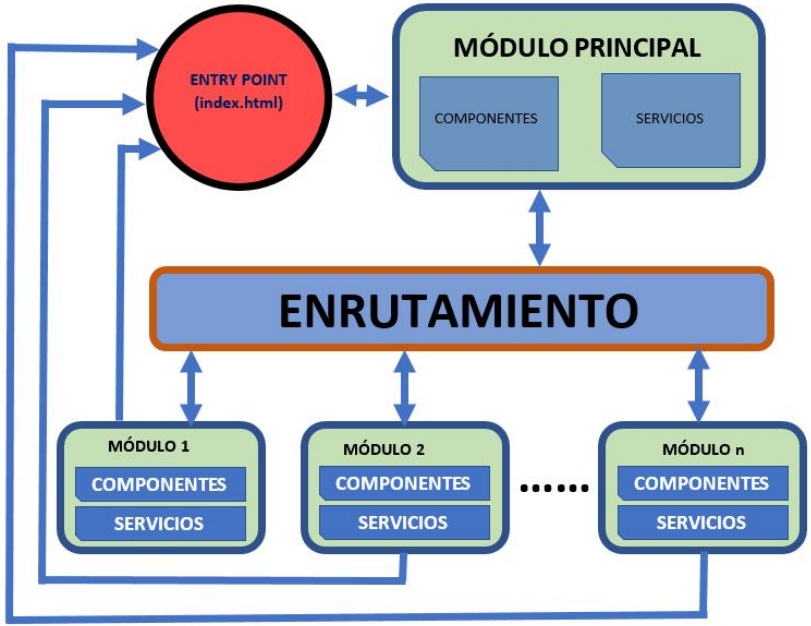
Módulo 1: Introducción a Angular

Módulo 1: Introducción a Angular

- Conceptos básicos sobre Angular
- Instalación del software necesario
- Entorno de desarrollo
- Breve introducción a Typescript
- Breve introducción a Angular CLI

Angular

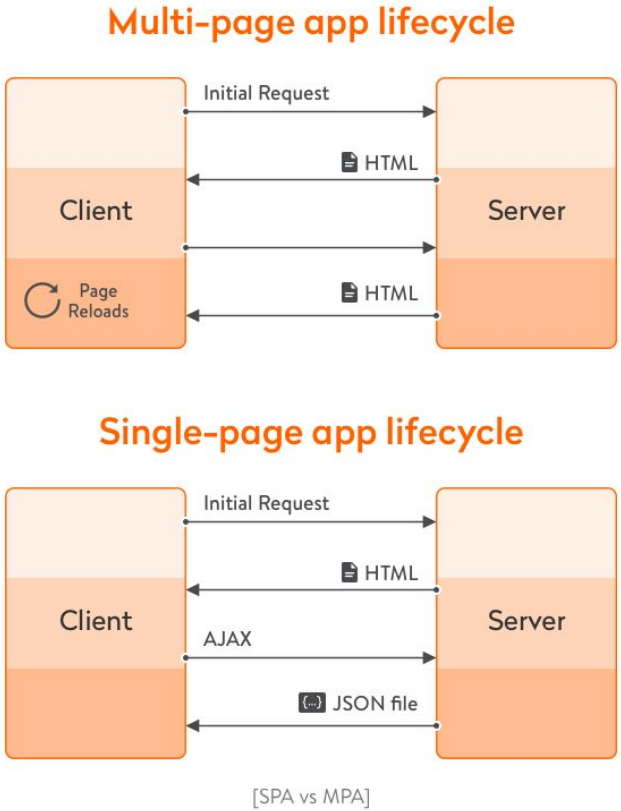
- Es un **framework** para desarrollo de aplicaciones SPA utilizando HTML y **Typescript**
- Escrito y basado en **Typescript**
- Basado en **módulos**, **componentes** y **servicios** (*clases Typescript con decoradores*)
- Extiende el código HTML con **etiquetas propias**



https://httpmasters.es/2018/05/02/estructura-de-una-aplicacion-angular-5/

Single Page Application (SPA)

- SPA vs. MPA (Multiple Page Application)
- Aplicaciones cliente completas programadas con HTML, CSS y **Javascript**
- **Vistas** vs. Páginas
- Comunicación con el servidor a través de **API's**
- Frameworks para crear SPA's: **Angular**, React, Vue, EmberJS, Polymer, Svelte, etc.



[SPA vs MPA]

https://yalantis.com/blog/single-page-apps-vs-multiple-page-apps/

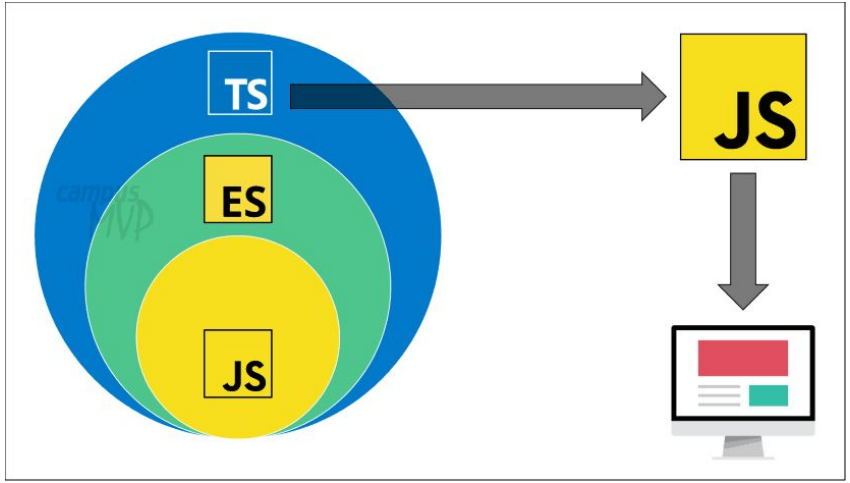
Javascript

- Creado en 1995
- En 1997 se crea un comité de la ECMA para su estandarización. Se diseña el DOM (Document Model Object) para evitar incompatibilidades de navegadores
- Las nuevas versiones de Javascript se regirán por ECMAScript
- En 2011 se aprueba ES5
- En 2014 se aprueba ES6 (funciones de flecha, let y const, clases, template strings, promesas, etc.)
- Versiones de Javascript: https://www.w3schools.com/js/js_versions.asp
- Compatibilidad con navegadores según versión de ECMASscript: https://kangax.github.io/compat-table/es6/

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide

Typescript

- Es un lenguaje de programación Open Source creado por Microsoft en 2012
- **Superset** de Javascript
- **Transpilación** vs compilación
- Sitios oficiales:
  - https://www.typescriptlang.org
  - https://github.com/Microsoft/TypeScript
  - https://www.typescriptlang.org/play



https://www.campusmvp.es/recursos/post/typescript-contra-javascript-cual-deberias-utilizar.aspx

Typescript

- Características principales:
  - Tipado estático (autocompletado de código, recomendación de argumentos de función, documentación, etc.)
  - Orientado a objetos con clases
  - Interfaces
  - Tipos genéricos
  - Casting de datos
  - Argumentos con tipo
  - Tipo de retorno de las funciones
  - Tooling: soporte para herramientas en tiempo de desarrollo. Su servicio *tsserver* expone al compilador y otros servicios del lenguaje y los expone como un servicio para los editores de código. Esto permite disponer de herramientas fundamentales como el *Intellisense*.

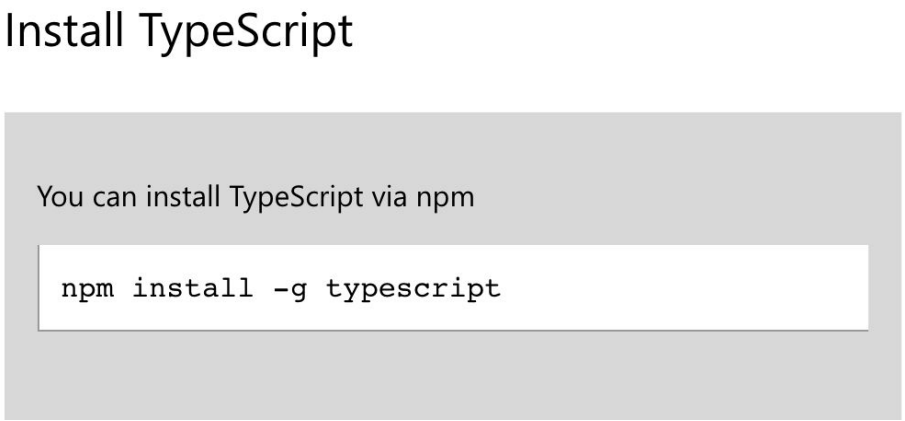
Typescript - Ventajas frente a Javascript

- Tipado estático (y opcional) permite verificar corrección del código y produce menos errores
- Permite que los editores de código ofrezcan autocompletar, refactorizar, *ir a*, etc.
- Sintaxis similar a Javascript y a Java o C#
- El compilador siempre generará código Javascript compatible con los navegadores
- *Pone orden* en Javascript. Código mucho más entendible

Typescript - Instalación

Typescript - Instalación

- Para instalar Typescript se necesita **npm**, el gestor de paquetes de **NodeJS**



https://www.typescriptlang.org/#installation

Typescript - Declaración de variables

- **let**
  - `let a: number = 5;`
- **const**
  - `const str = 'this is a string';`

https://medium.com/@tatymolys/var-let-y-const-donde-cuando-y-por-qu%C3%A9-d4a0ee66883b
https://www.tektutorialshub.com/typescript/typescript-let-var-const/

Typescript - Tipos de datos

- **number**
  - `let a: number = 5;`

```ts
/**
 * number
 */
let a: number;
a = 10;

const b = 10;
const c = 5.123;

console.log('a: ', a, '- b:', b, '- c: ', c);
console.log('¿El tipo de a es number?: ', typeof a === 'number');
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Tipos de datos

- **string**
  - `let str = 'this is a string';`

```
/**
 * string
 */
let fullName: string = "Bob Bobbington";
let sentence: string = `Hello, my name is ${fullName}`;
let color: string = 'green';

console.log(fullName);
console.log(sentence);
console.log(color);
console.log('¿El tipo de color es string?: ', typeof color === 'string');
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Tipos de datos

- **boolean**
  - `let isFinished = false;`

```ts
/**
 * boolean
 */
let isFinished: boolean = true;
let isLoaded = false;

console.log(isFinished);
console.log('¿El tipo de isLoaded es boolean?: ', typeof isLoaded === 'boolean');
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Tipos de datos

- **any**
  - `let myVble: any;`

```
/**
 * any
 * Puede tener cualquier valor y, además, puede variar en tiempo de ejecución
 */
let myVariable: any;
myVariable = 5;
console.log('myVariable: ', myVariable, ' --- tipo: ', typeof (myVariable));
myVariable = 'this is a string';
console.log('myVariable: ', myVariable, ' --- tipo: ', typeof (myVariable));
myVariable = true;
console.log('myVariable: ', myVariable, ' --- tipo: ', typeof (myVariable));
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Tipos de datos

- **array**
  - `let myColors= [‘red’, ‘green’, ‘blue’];`

```ts
/**
 * array
 */
let myArray: number[] = [1, 2, 3, 5];
let myArray2 = [1, 'My String', false, [7, 8, 9], { name: 'Juan', surname: 'Pérez' }];
myArray.push(6);
myArray2.push(6);
console.log(myArray);
console.log(myArray2);
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Tipos de datos

- **undefined**
  - `let myVbleWithouValue;`

```
/**
 * undefined
 */
let vbleWithoutValue;

console.log(vbleWithoutValue);
console.log('¿El tipo de vbleWithoutValue es undefined?: ', typeof vbleWithoutValue === 'undefined');

vbleWithoutValue = 45;
console.log(vbleWithoutValue);
console.log('¿El tipo de vbleWithoutValue es undefined?: ', typeof vbleWithoutValue === 'undefined');
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Tipos de datos

- **null**
  - `let myNullVble: null = null;`

```ts
/**
 * null
 */
let vbleWithNullValue: null = null;

console.log(vbleWithNullValue);
// console.log('¿El tipo de vbleWithNullValue es null?: ', typeof vbleWithNullValue == 'null');
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Tipos de datos

- **enum**
  - `enum Color {Red, Green, Blue};`
  - `let c: Color = Color.Green;`

```
/**
 * enum
 */
enum Color {
  Red,
  Green,
  Blue
}
let col: Color = Color.Green;
console.log('El color es: ', col);
console.log('Cuál es el tipo de col?:  ', typeof (col));
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Tipos de datos

- **Object**
  - *Object is a type that represent the non-primitive type*
  - `let myObj = {name: 'Juan', age: 45};`

```
/**
 * Object represented in JSON
 */
let myObj = {
  name: 'Juan',
  age: '45'
}

console.log('El nombre es: ', myObj.name);
console.log('La edad es:  ', myObj.age);
console.log('myObj es un objeto?: ', myObj instanceof Object);
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Clases

```ts
/**
 * Basic Class
 */

export class Album {

  title = '';
  pubDate = new Date();
  numberOfSongs = 5;

  constructor(title: string, pubDate: Date, numberOfSongs: number) {
    this.title = title;
    this.pubDate = pubDate;
    this.numberOfSongs = numberOfSongs;
  }

  playSong(songNumber: number) {
    console.log('Playing song number ', songNumber, '...');
  }
}
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Clases con *Class Expression*

```typescript
/**
 * Class Expresion
 */
export class Artist {
  constructor(public name: string, public bornDate: Date) { }

  public calcAge() {
    return new Date().getFullYear() - this.bornDate.getFullYear();
  }
}
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Clases

```
import { Album, Artist } from "./03.Classes";

let myAlbum = new Album('myTitle', new Date('2020-01-20'), 10);
let myArtits = new Artist('Album name', new Date('1950-01-31'));

console.log(myAlbum.playSong(5));
console.log(myArtits.calcAge());
console.log(myArtits instanceof Album);
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Typescript - Interfaces

```typescript
interface IPerson {
  id: number;
  name: string;
  fiscalId: string;
  isMarried: boolean;
}


let obj1: IPerson;

obj1 = {
  id: 1,
  name: 'Juan',
  fiscalId: '44.444.555F',
  isMarried: false
}
```

https://www.typescriptlang.org/docs/handbook/basic-types.html

Angular - Instalación

Angular - Extensiones para VS Code

Extensiones recomendadas:

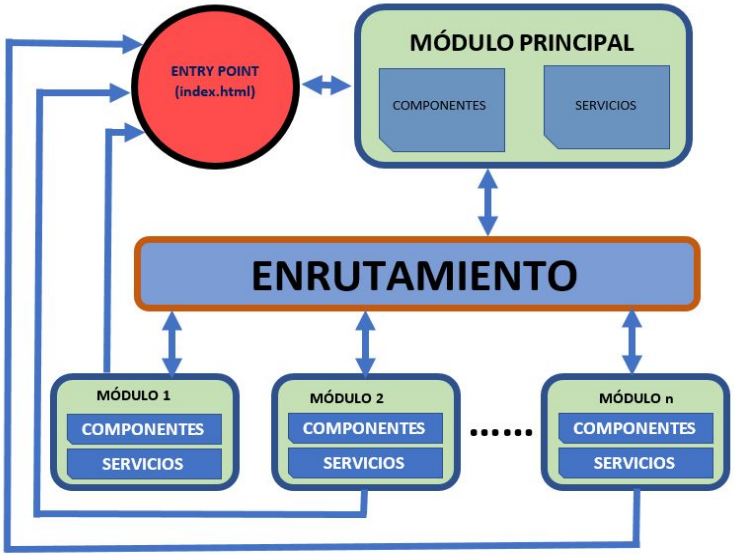
- Básicas
  - Angular Essentials
  - Typescript importer
  - JSON to TS
- Recomendadas
  - Angular2-switcher
  - Bracket Pair Colorizer 2

Angular

- Es un **framework** para desarrollo de aplicaciones SPA utilizando HTML y **Typescript**
- Escrito y basado en Typescript
- Basado en **módulos**, **componentes** y **servicios** (clases Typescript con decoradores)
- Extiende el código HTML con **etiquetas propias**

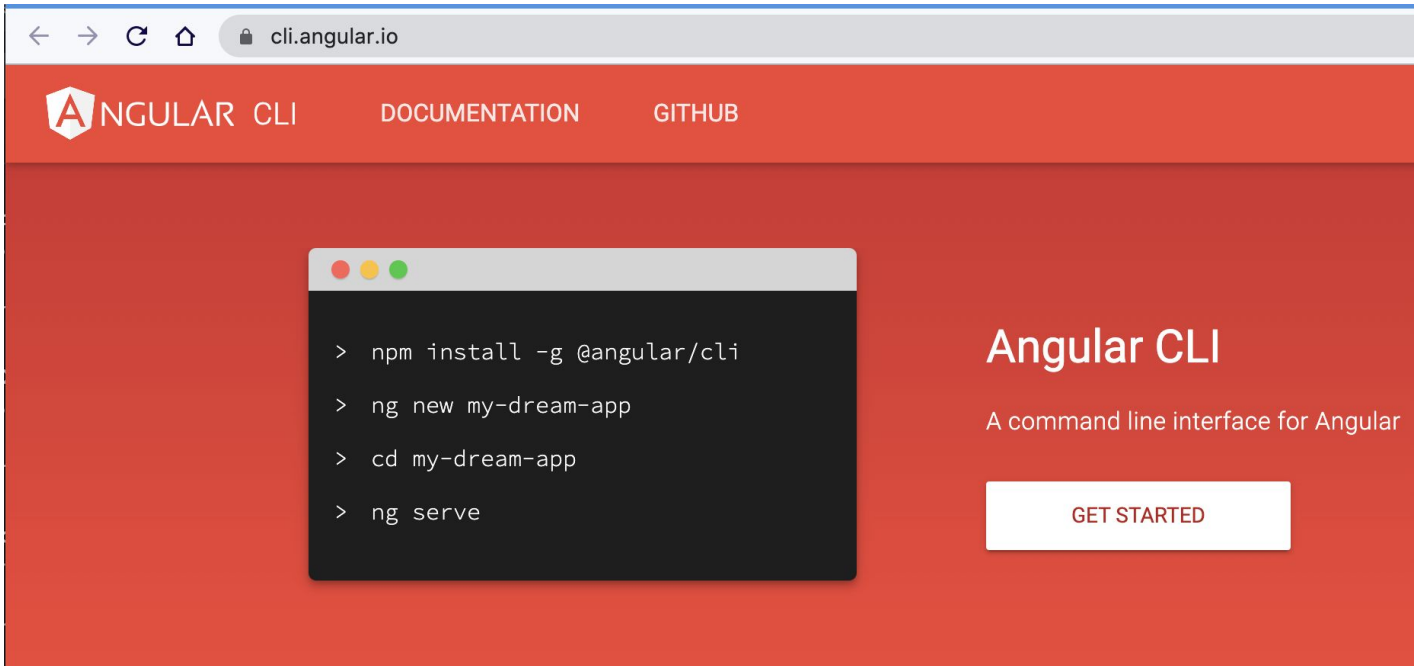
VERSION	STATUS	RELEASED	ACTIVE ENDS	LTS ENDS
^10.0.0	Active	Jun 24, 2020	Dec 24, 2020	Dec 24, 2021
^9.0.0	Active	Feb 06, 2020	Aug 06, 2020	Aug 06, 2021
^8.0.0	LTS	May 28, 2019	Nov 28, 2019	Nov 28, 2020

https://github.com/angular/angular/blob/master/CHANGELOG.md

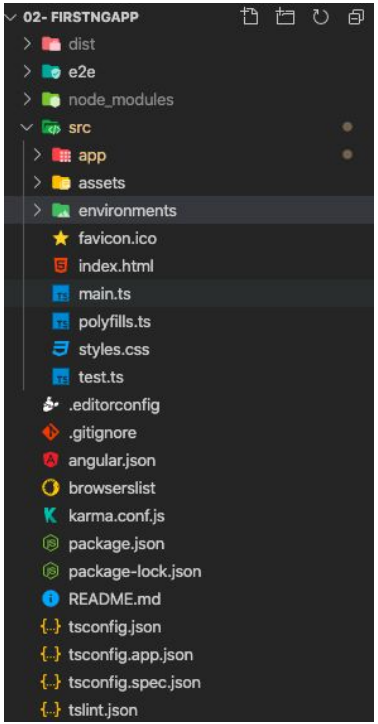


https://httpmasters.es/2018/05/02/estructura-de-una-aplicacion-angular-5/

Angular - Instalación

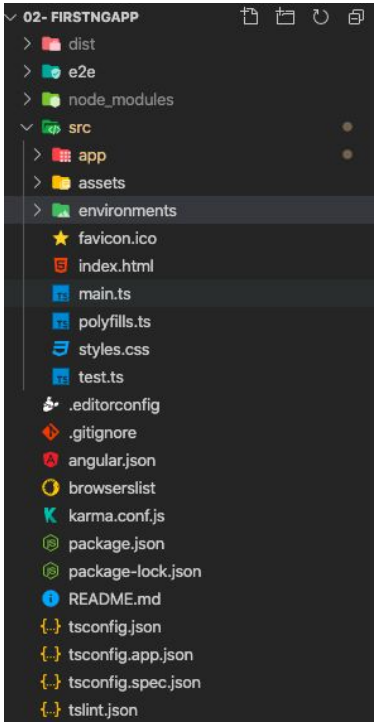


Angular - Estructura de una aplicación



- **dist**: La aplicación para publicar en el servidor web de producción. Se genera sólo cuando se compila para producción (`ng build --prod`)
- **e2e**: Ficheros para realizar pruebas end-to-end automáticas
- **node_modules**: Carpeta con las dependencias del proyecto, es decir, librerías y herramientas necesarias en el proyecto
- **src**: Fuentes de la aplicación
  - **app**: Ficheros fuente principales de la aplicación (módulos, componentes, etc.)
  - **assets**: Recursos estáticos que necesita la aplicación (imágenes, css, etc.)
  - **environments**: Configuraciones y variables de entorno que se utilizarán tanto en desarrollo como en producción
  - **favicon.ico**: Archivo icono del proyecto
  - **index.html**: Página principal (y única) de la aplicación
  - **main.ts**: Archivo Typescript de inicio de la aplicación
  - **polyfills.ts**: Archivo que contiene *polyfills* que tienen como objetivo que navegadores antiguos se comporten correctamente con los estilos y scripts generados por Angular

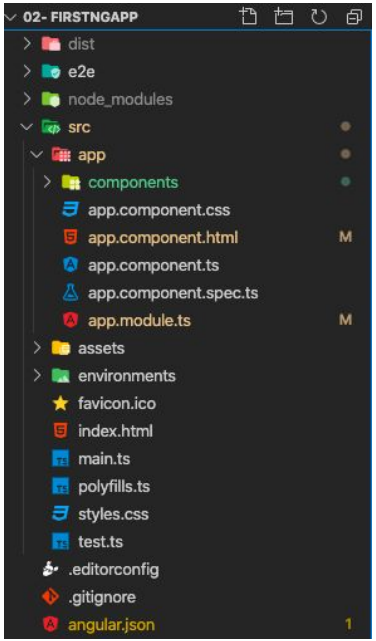
Angular - Estructura de una aplicación



- **.editorconfig**: Configuración del editor VSCode
- **.gitignore**: Carpetas/ficheros que git debe ignorar
- **angular.json** (1): contiene la configuración del propio CLI
- **package.json**: Configuración de la aplicación y registra las dependencias de librerías y scripts necesarios para su despliegue y ejecución.
- **README.md**: Información/documentación sobre la aplicación.
- **tsconfig.json**: Contiene la configuración de TypeScript para transpilar a Javascript
- **tslint.json**: Reglas del linter de Typescript

(1)https://medium.com/angular-chile/angular-cli-workspace-bajo-la-lupa-417b9e7eb836#:~:text=JSON%20Schema%20es%20un%20vocabulario,%2Flib%2Fconfig%2Fschema

Angular - Estructura de una aplicación



- **app.component.css**: Estilos CSS para el componente
- **app.component.hml**: Vista del componente
- **app.component.ts**: Clase Typescript del componente
- **app.component.spec.ts**: Test creado para probar el componente
- **app.module.ts**: Módulo principal de la aplicación