



Clase 02. ANGULAR

# ***COMPONENTES Y ELEMENTOS DE UN PROYECTO ANGULAR***

***RECUERDA PONER A GRABAR LA  
CLASE***





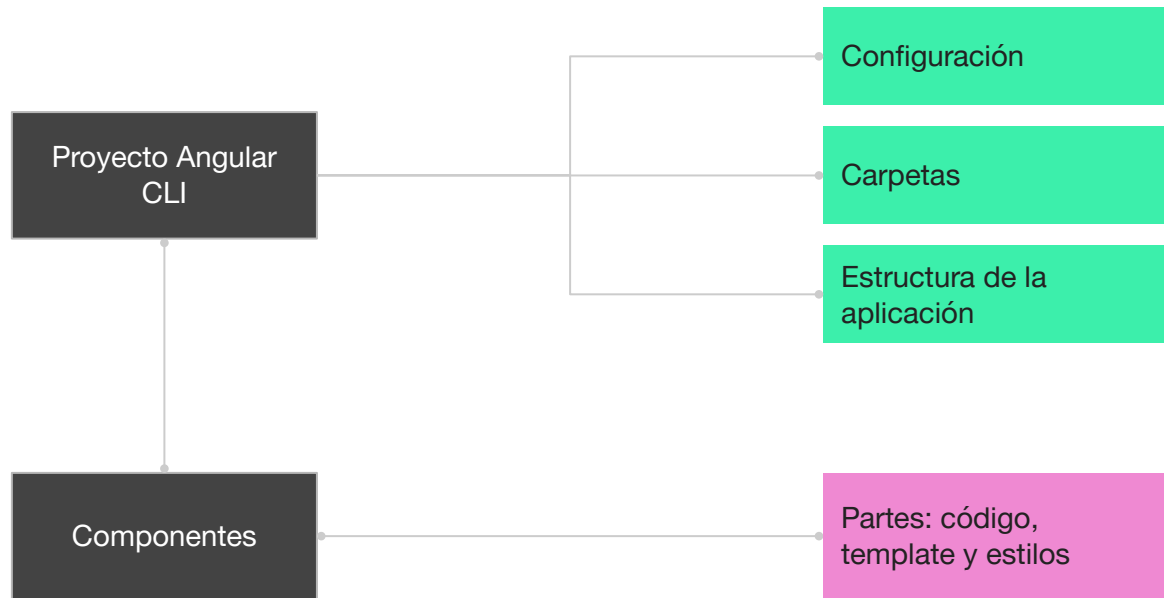
## ***OBJETIVOS DE LA CLASE***

- Analizar la estructura de un proyecto basado en Angular CLI.
- Identificar los componentes y técnicas de componentización.

# ***MAPA DE CONCEPTOS***

# MAPA DE CONCEPTOS CLASE

¡Para  
recordar!



# ***CRONOGRAMA DEL CURSO***

## Clase 1



### **Introducción al curso y a Angular**



Crear el primer proyecto en Angular



Subir el proyecto a GitHub



Instalación de herramientas en vivo

## Clase 2



### **Componentes y Elementos de un proyecto Angular**



Flujo de desarrollo Angular



Componentes



Ejemplo en vivo



Layouts

## Clase 3



### **Typescript**



Interfaces



Funciones y tipado genérico



Ejemplo en vivo

# ***GLOSARIO:***

## ***Clase 1***

**Angular:** Es un framework Front End de código abierto desarrollado y mantenido por Google. Se utiliza para crear páginas web de tipo SPA.

**SPA:** Se le llama Single Page Application (SPA) a las páginas que cargan sólo al inicio y, luego, las sucesivas actualizaciones se producen sin necesidad de refrescarla en forma completa.

**Angular.CLI:** Es una herramienta que nos va a permitir la interacción con el framework.

***¡EMPEZAMOS!***



***CODER HOUSE***



# ***ANÁLISIS DE ESTRUCTURA***

# ***ANÁLISIS DE ESTRUCTURA***

Si bien un proyecto Angular puede ser creado partiendo de un simple archivo **index.html** o una estructura básica de página web, hoy nadie empezaría un proyecto en Angular sin partir de una **estructura de archivos, carpetas y herramientas** configuradas y optimizadas para la tarea 📁.



# ***ANÁLISIS DE ESTRUCTURA***

La tarea del armado de una estructura y configuración de herramientas se lleva a cabo mediante el uso de la herramienta Angular CLI propuesta por el mismo Framework 😊.



# ***RECORDEMOS***

***¿CÓMO CREAR UN PROYECTO?***



***CODER HOUSE***

# ***CREANDO TU PROYECTO***

## ***BREVE REPASO***

1

Verifica tu  
instalación



`ng version`

2

Crea tu primer  
proyecto



`ng new my-project`

3

Accede a la carpeta  
del proyecto



`cd my-project`

# CREANDO TU PROYECTO

## BREVE REPASO

4

Ejecución del  
servidor



```
ng serve
```

Ejecuta nuestro entorno de desarrollo dejando habilitado el puerto donde corre nuestra aplicación por defecto.



```
ng serve -o
```

Hace lo mismo que el comando **ng serve** pero corre un navegador en el puerto por defecto - **Recomendado** 🚀.



```
ng serve -o  
--port=3500
```

Con este comando podemos elegir el puerto en el que se va a ejecutar nuestro proyecto.















# ***ANGULAR BOILERPLATE***

Con la ejecución de creación de un proyecto Angular mediante el comando **ng new** de ANGULAR-CLI, se genera una estructura plantilla del tipo **boilerplate**. Este tipo de estructura no es más que un proyecto inicial con las configuraciones y herramientas necesarias de desarrollo 🙌.

Entre estas configuraciones se incluye una plantilla básica de archivos y carpetas que estaremos analizando a continuación.



✓ MY-PROJECT

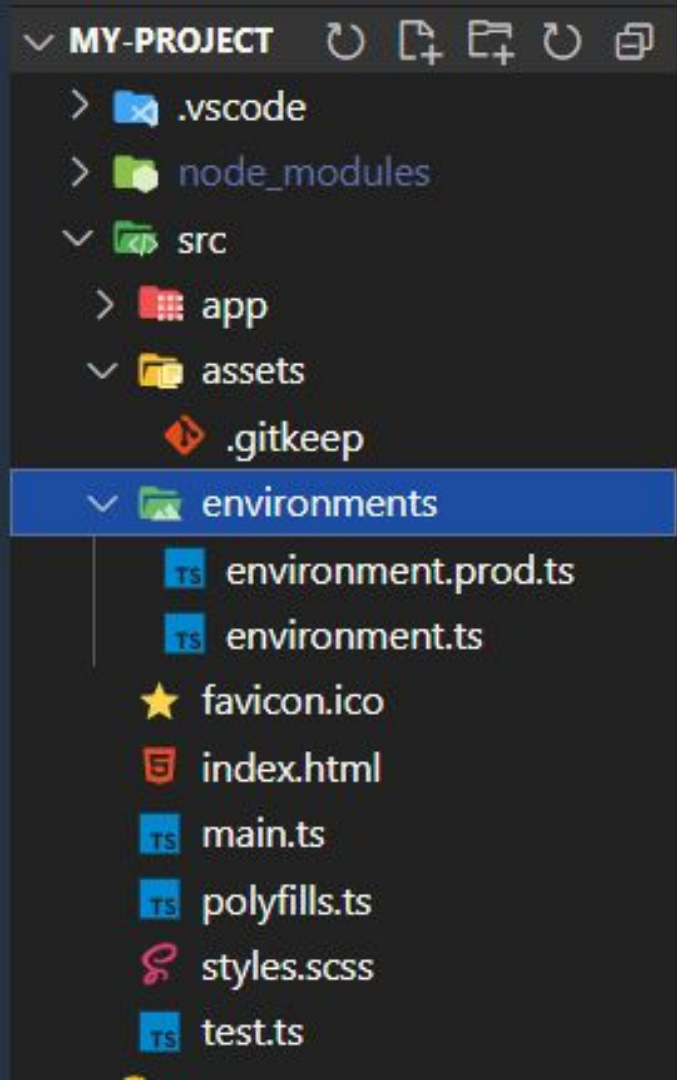
- >  .vscode
- >  node\_modules
- >  src
-  .browserslistrc
-  .editorconfig
-  .gitignore
-  angular.json
-  karma.conf.js
-  package-lock.json
-  package.json
-  README.md
-  tsconfig.app.json
-  tsconfig.json
-  tsconfig.spec.json

# ***ESTRUCTURA***

## ***ARCHIVOS DE CONFIGURACIÓN***

- gitignore
- .browserslistrc
- .editorconfig
- angular.json
- karma.config
- package-lock.json
- package.json
- README.md
- tsconfig.app.json
- tsconfig.json
- tsconfig.spec.json

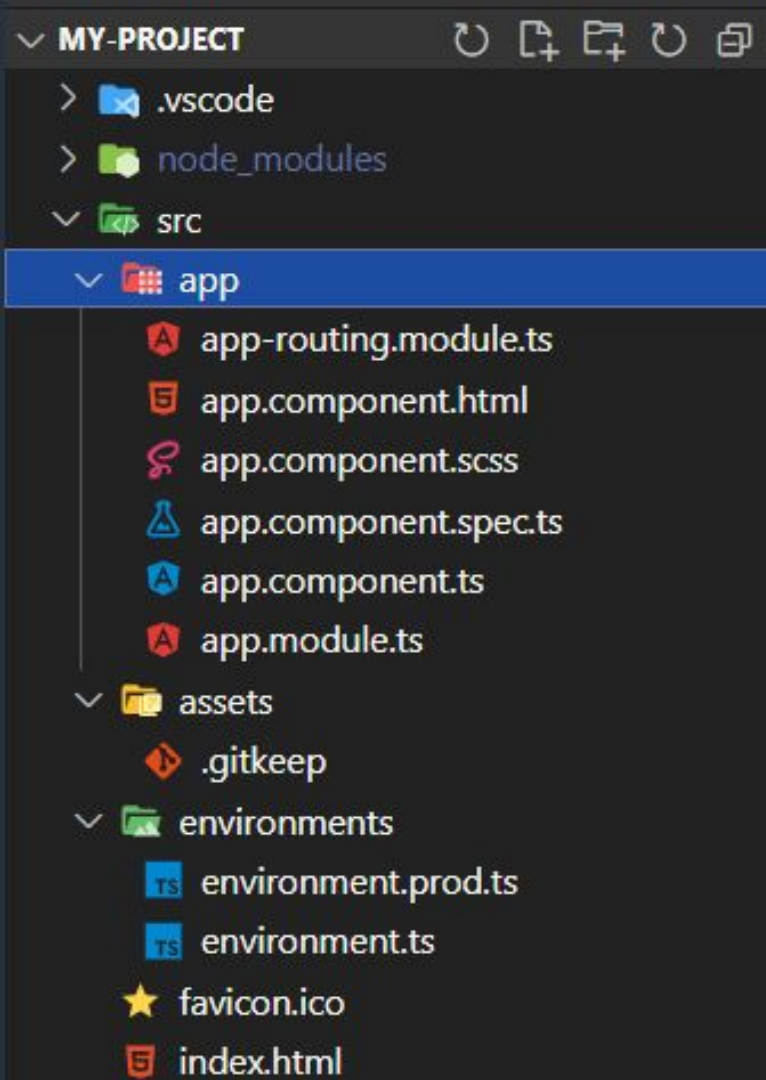




# ***ESTRUCTURA***

## ***CARPETA SRC***

- Carpeta assets
- Carpeta environments
- favicon.ico
- index.html
- main.ts
- polyfill.ts
- styles.scss
- test.ts



# ***ESTRUCTURA***

## ***CARPETA SRC/APP***

- app-routing.module.ts
- app.component.html
- app.component.scss
- app.component.spec.ts
- app.component.ts
- App.module.ts

Esta es la estructura inicial. En una app real encontraremos muchos más elementos.

# ***CICLO DE EDICIÓN, EJECUCIÓN Y DEPURACIÓN***

# CICLO

Ejecución del  
servidor “Angular”

1

```
Command Prompt
C:\curso\v2>cd my-project
C:\curso\v2\my-project>code .
C:\curso\v2\my-project>ng serve
  Browser application bundle generation complete.

Initial Chunk Files | Names          | Raw Size
vendor.js           | vendor         | 2.48 MB
polyfills.js        | polyfills      | 339.24 KB
styles.css, styles.js | styles        | 212.57 KB
main.js             | main           | 53.58 KB
runtime.js           | runtime        | 6.48 KB
                    | Initial Total  | 2.68 MB

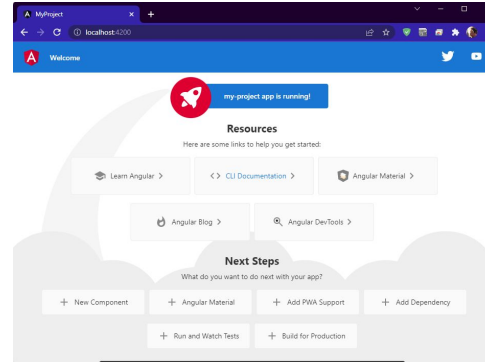
Build at: 2022-02-24T08:06:46.138Z - Hash: d580bcc7dde3d561 - Time: 9706ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
✓ Browser application bundle generation complete.
✓ unchanged chunks
Build at: 2022-02-24T08:06:46.789Z - Hash: d580bcc7dde3d561 - Time: 207ms
✓ Compiled successfully.
```

Navegar a  
la app:

**localhost: 4200**

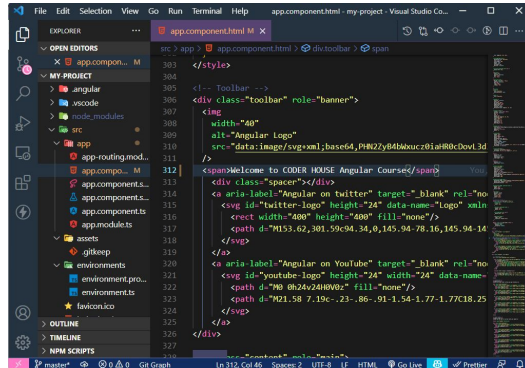
2



# CICLO

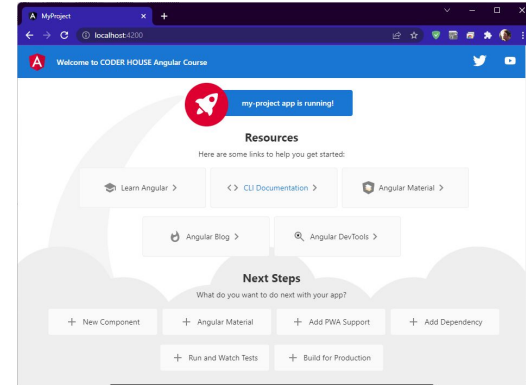
Editar y salvar el  
código

3



Refresh automático  
en el navegador

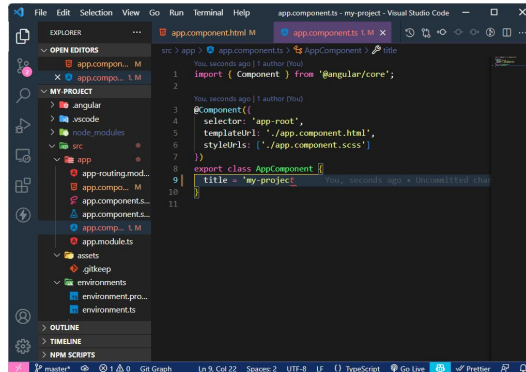
4



# CICLO

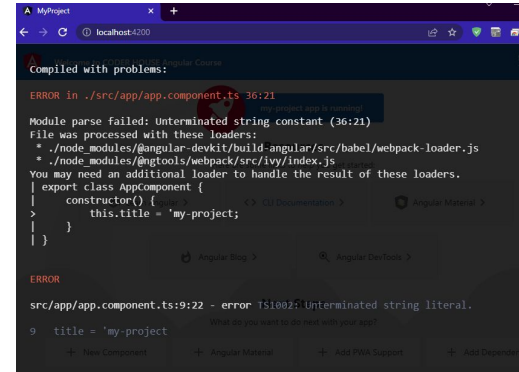
Generamos un error  
y salvamos

5



Error en el navegador

6





# ***FLUJO DE DESARROLLO ANGULAR***

Ejecutar, editar, visualizar cambios automáticos y ver errores 🤖.

**Tiempo estimado:** 10 minutos.

***CODER HOUSE***

# ***FLUJO DE DESARROLLO ANGULAR***

Desafío  
generico



- Ejecuta el proyecto creado anteriormente.
- Personaliza la página de muestra.
- Revisa los cambios.
- Genera errores y observa qué sucede.

**Tiempo estimado:** 10 minutos.

***CODER HOUSE***





***BREAK***


**¡5/10 MINUTOS Y VOLVEMOS!**

# ***COMPONENTES***



# ¿QUÉ ES UN COMPONENTE?

Un componente representa una porción (o toda) de la aplicación y está contenido dentro de un módulo. Cada componente **define una clase** que contiene datos y lógica y está asociado con una plantilla HTML y CSS.

Se puede decir que **un componente está orientado a la experiencia del usuario** .

# COMPONENTES

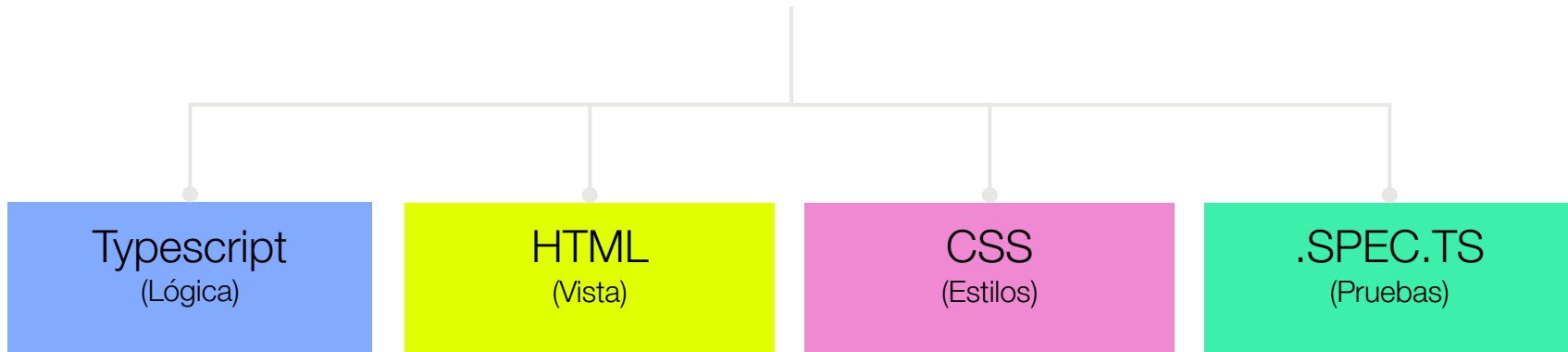
Como podemos ver aquí, nuestra aplicación es un **componente** que vive en la única página HTML de nuestra aplicación 🤖:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MyProject</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-s
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Nuestra aplicación

# COMPONENTE BÁSICO

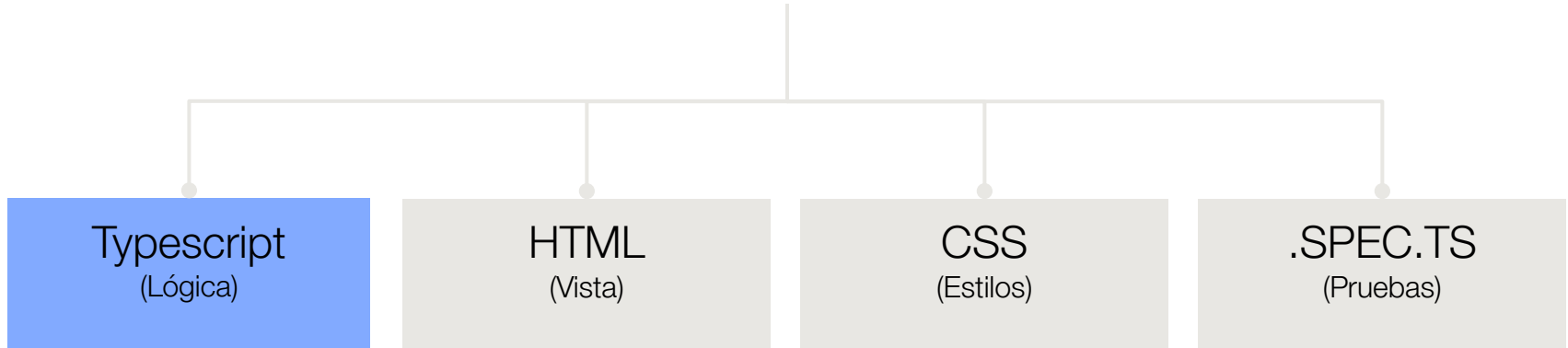
## PARTES/ARCHIVOS



**! Nota importante:** Un componente puede tener uno o más archivos, este es un factor que dependerá de la necesidad de tu proyecto.

# ***COMPONENTE BÁSICO***

## ***PARTES/ARCHIVOS***



# COMPONENTE BÁSICO

## APP COMPONENT TS



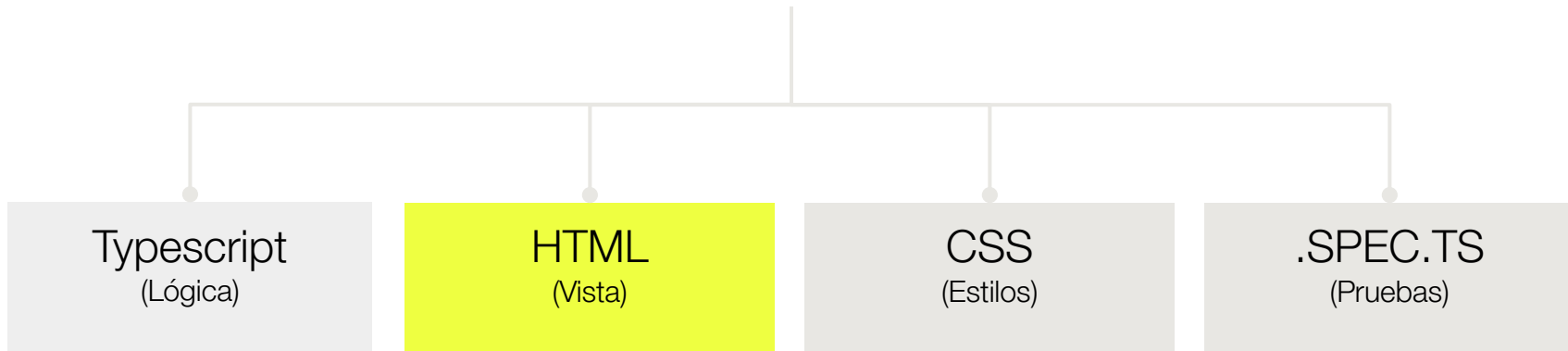
Esta parte del componente encapsula la **lógica** de la aplicación y referencia a los archivos de vista y estilo de este mismo 🙌

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

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

# ***COMPONENTE BÁSICO***

## ***PARTES/ARCHIVOS***





# COMPONENTE BÁSICO

## APP COMPONENT HTML

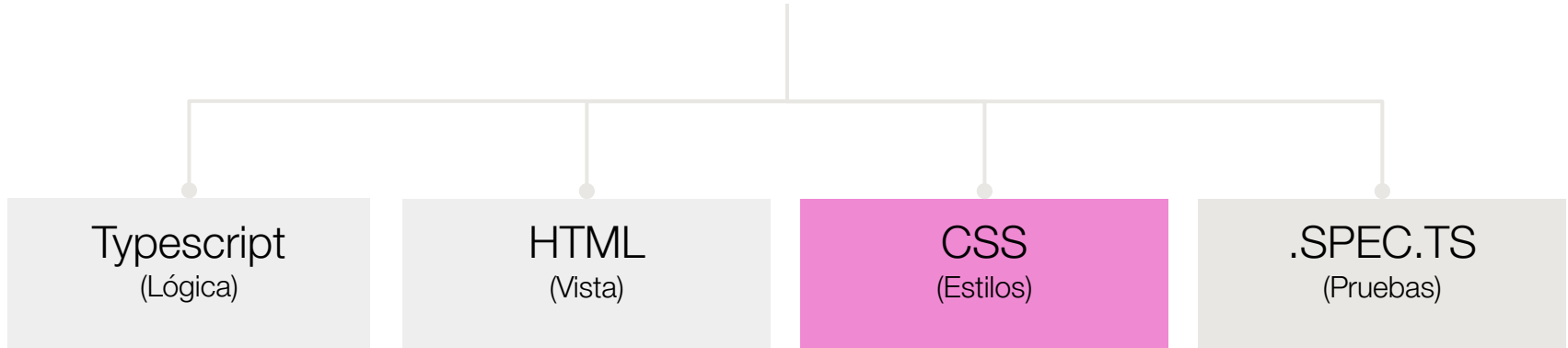


Esta parte del componente referencia estructura de la **vista**, básicamente cómo van a estar organizados los datos que queremos mostrar 🖥️. Se vincula a los estilos por las definiciones establecidas en el archivo de lógica:

```
<h1 class="titulo">Bienvenidos</h1>  
<h2 class="subtitulo">{{ title }}</h2>
```

# ***COMPONENTE BÁSICO***

## ***PARTES/ARCHIVOS***



# COMPONENTE BÁSICO

## APP COMPONENT CSS

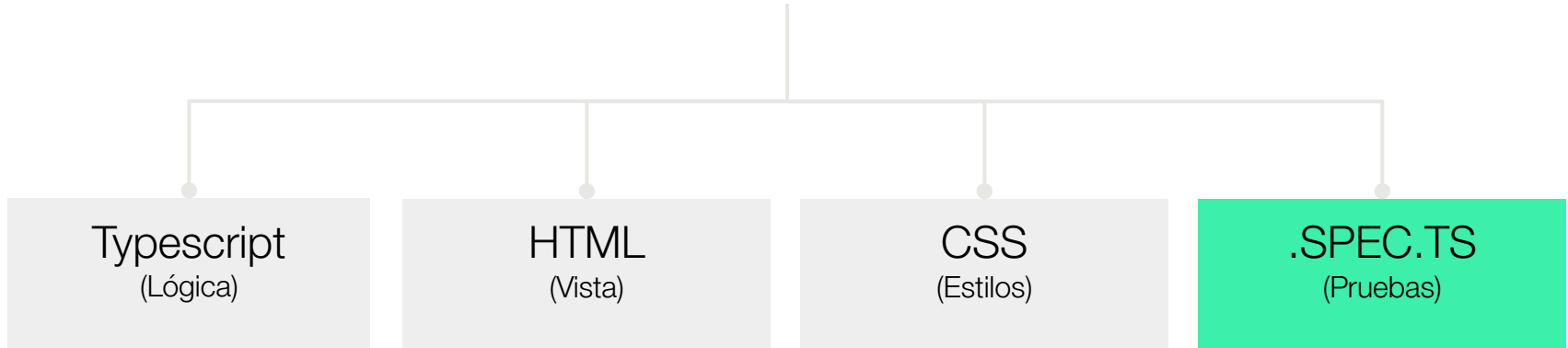


Este archivo se encarga de darle el **formato visual** al componente. Se encuentra diferenciado en el archivo de lógica 📁

```
.titulo {  
  color: red;  
}  
  
.subtitulo {  
  color: yellow;  
}
```

# ***COMPONENTE BÁSICO***

## ***PARTES/ARCHIVOS***



# COMPONENTE BÁSICO

## APP COMPONENT SPEC.TS



Este archivo se encarga de las **pruebas** del componente 🙌. Es muy común a la hora de construir un componente. Sin embargo, aún no lo vamos a utilizar. Por ello, no lo crearemos en esta instancia.

```
import { TestBed } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from '../app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [ RouterTestingModule ],
      declarations: [ AppComponent ],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });
});
```

# COMPONENTES

**AHORA BIEN, ¿CÓMO LOS CREAMOS?** 🤔

Ejemplo  
en vivo



Desde Angular.CLI

Generación Manual

# COMPONENTES

AHORA BIEN, ¿CÓMO LOS CREAMOS? 🤔

Ejemplo  
en vivo



Desde Angular.CLI

Generación Manual



```
ng generate component <name>  
[options]
```



```
ng g c <name> [options]
```

# COMPONENTES

AHORA BIEN, ¿CÓMO LOS CREAMOS? 🤔

Ejemplo  
en vivo



Desde Angular.CLI

Generación Manual

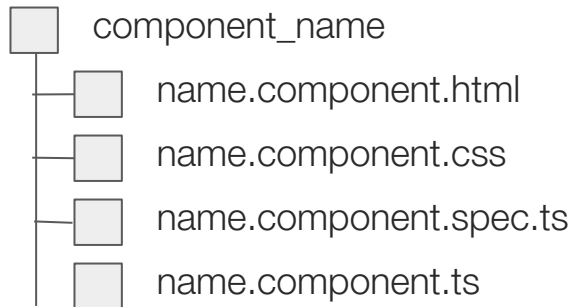


`ng generate component <name>  
[options]`



`ng g c <name> [options]`

Para la generación manual se crea una carpeta con los siguientes archivos:





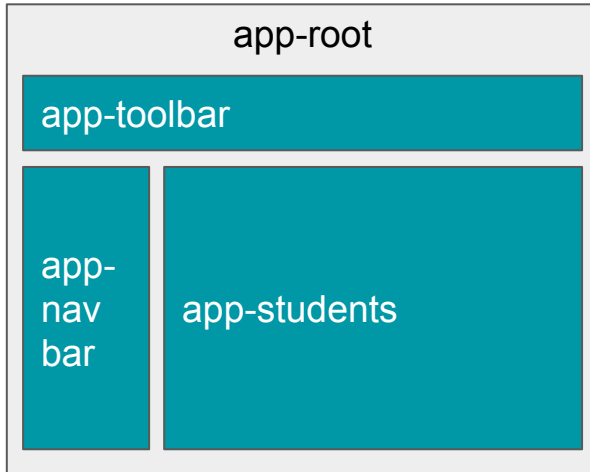
# ***COMPONENTIZACIÓN DE UNA APLICACIÓN***

# ***COMPONENTIZACIÓN***

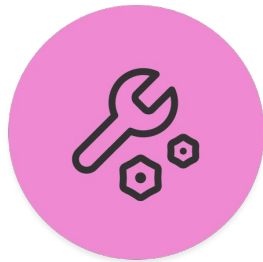
Como vimos, nuestra aplicación es un componente: **app-root**. También es conocido como componente principal o componente raíz 🌳. Dentro de él vamos a ir agregando los demás componentes.

De esta manera, se va generando una **jerarquía de componentes** padres-hijo dentro de nuestra aplicación.

# COMPONENTIZACIÓN



```
app.component.html M X
src > app > app.component.html > ...
Go to component | You, seconds ago | 1 author (You)
1 <div class="main-grid-layout">
2   <app-toolbar class="toolbar"></app-toolbar>
3   <app-navbar class="navbar"></app-navbar>
4   <app-students class="students"></app-students>
5 </div>
6
```



# ***COMPONENTIZACIÓN***

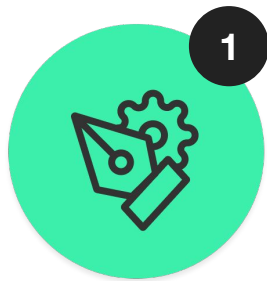
En esta instancia, podrás sumar tres componentes nuevos dentro del **app-root** y utilizar el **layout grid de css** 🙌.

**Tiempo estimado:** 15 minutos.



- Agregar los componentes student, toolbar y navbar con angular-cli.
- Definir el css necesario para darle un layout utilizando css grid,
- Darles contenido y estilos a cada componente.

**Tiempo estimado:** 15 minutos.



# ***INTEGRANDO CONOCIMIENTOS***

En esta instancia crearás un layout, un toolbar y un área de contenido 🚀.

# INTEGRANDO CONOCIMIENTOS

**Formato:** Se debe entregar un proyecto con el formato ANGULAR-CLI. Lo vamos a nombrar como "Componentes+Apellido".

**Sugerencia:** Utiliza ANGULAR-CLI para generar un proyecto con componentes personalizados.

Algunas ideas que pueden desarrollar:

Página personal con enlaces a sus redes sociales.

Página que muestre destinos turísticos en forma de tarjeta.

Página de descripción de un producto.

Desafío  
entregable



**>> Consigna:** Crear un layout general con menú a la izquierda, un toolbar y un área de contenido. Cada uno de los apartados del layout será un componente independiente y tendrá su propia hoja de estilos y contenido html. Todos ellos integrados en el app.component.html

**>>Aspectos a incluir en el entregable:**

Se espera la entrega de un proyecto configurado funcional utilizando creación de componentes. Se valorará la utilización de la librería bootstrap para la colocación de los componentes.

***¿PREGUNTAS?***

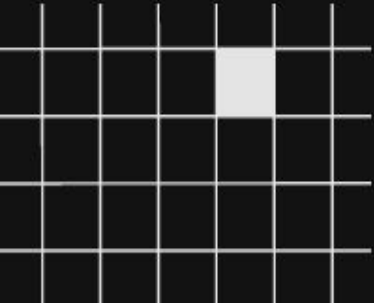






# ***¡MUCHAS GRACIAS!***

## **Resumen de lo visto en clase hoy:**

- ANGULAR CLI. Estructura y componentes.
  - Ciclo de ejecución y edición.
- 



***OPINA Y VALORA ESTA CLASE***

***#DEMOCRATIZANDO LA EDUCACIÓN***

***CODER HOUSE***