

<b>Nombre del instructor</b>	Adolfo Centeno Tellez		
<b>Área temática</b>	Ciencia de datos		
<b>Rol</b>	Data Scientist		
<b>Competencia</b>	Full Stack		
<b>Subcompetencia</b>	Frontend		
<b>Objetivo</b> <b>Módulo</b> <b>Conocimiento</b>			
<b>Módulo Conocimiento (5h en total )</b>			
	Duración	Objetivos	Subtemas
Tema 2 - Angular basico y el enfoque basado en componentes	1 hora	Crear un proyecto básico usando angular CLI, analizar la arquitectura del proyecto y entender el enfoque de componentes que usa angular para crear aplicaciones, finalmente desarrollar un proyecto usando componentes.	1.- Crear un proyecto en Angular, configuración y ejecución 2.- Definición de componente en Angular y sus características 3.- Creación de una aplicación simple basada en componentes

### Introducción al tema

En el tema anterior se creó el ambiente de desarrollo para Angular, formado por git, nodejs 20.0 y Angular 18.0. Con este stack tecnológico estamos habilitados para crear proyectos nuevos usando el Angular CLI.

Al finalizar este módulo Identificarás serás capaz de crear un proyecto básico usando angular CLI, entender cada una de las opciones que muestra el asistente, entender el proceso de creación y configuración del proyecto generado. Además entenderás la arquitectura del proyecto, el enfoque basado en componentes que usa angular para crear aplicaciones, finalmente desarrollarás un proyecto que está dividido en componentes y aprenderás cómo integrar la aplicación completa desde el componente principal.

### SUBTEMA 1: Crear un proyecto en Angular, configuración y ejecución

Angular es un framework de software abierto basado en componentes y con fuerte enfoque en el Desarrollo Dirigido por pruebas. Dentro de sus principales características están el que está basado en componentes y cuenta con una CLI altamente especializado. Angular CLI está representado por la herramienta o comando ng.

Existen diferentes comandos para realizar diferentes tareas ejemplo de alguna de ellas:

- ng help
- ng version
- ng new
- ng test
- ng build
- ng generate component
- ng generate service

En esta sección crearemos nuestro primer proyecto en Angular al cual denominaremos **mycv**.

#### Windows

Abrir git bash

#### Linux

Abrir una consola de su shell predeterminado

**Mac**

Abrir la terminal

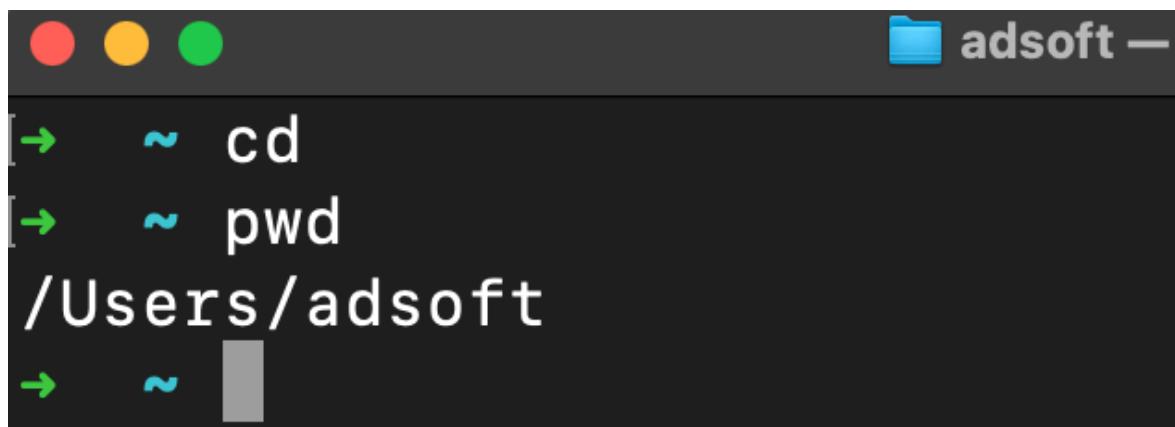
Para crear un proyecto ejecutaremos las siguientes tareas:

Para asegurarnos estar en nuestro carpeta personal de trabajo, tecleamos el comando **cd** sin parámetros:

```
$ cd
```

Posteriormente visualizamos la ruta de trabajo actual con **pwd**

```
$ pwd
```



```
[→ ~ cd  
[→ ~ pwd  
/Users/adsoft  
→ ~
```

Antes de crear nuestro proyecto, asegurarnos de que tenemos activo nuestro nodejs en la versión 20.x.x

```
$ node --version
```



```
[→ ~ node --version  
v20.11.1  
→ ~
```

En caso esté activa otra versión, usar el comando **nvm use v20.x.x**

```
→ ~ nvm use v20.11.1
Now using node v20.11.1 (npm v10.2.4)
→ ~
```

Una vez en la versión de nodejs correcta, creamos nuestro proyecto con el CLI de angular.

```
$ ng new mycv --no-standalone
```

```
→ ~ ng new mycv --no-standalone

? Which stylesheet format would you like to use? (Use arrow keys)
> CSS [ https://developer.mozilla.org/docs/Web/CSS ]
Sass (SCSS) [ https://sass-lang.com/documentation/syntax#scss ]
Sass (Indented) [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
Less [ http://lesscss.org ]
```

Seleccionamos **CSS** como el tipo de hojas de estilo para nuestro proyecto, tecleando <ENTER> sobre la opción resaltada.

```
→ ~ ng new mycv --no-standalone

? Which stylesheet format would you like to use? CSS [ https://developer.mozilla.org/docs/Web/CSS ]
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? (y/N)
```

Presionar **N** en la opción de habilitar **SSR** y presionar la tecla <ENTER>

```
[→ ~ ng new mycv --no-standalone

? Which stylesheet format would you like to use? CSS [https://developer.mozilla.org/docs/Web/CSS]
? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No
CREATE mycv/README.md (1065 bytes)
CREATE mycv/.editorconfig (274 bytes)
CREATE mycv/.gitignore (587 bytes)
CREATE mycv/angular.json (2835 bytes)
CREATE mycv/package.json (1035 bytes)
CREATE mycv/tsconfig.json (1021 bytes)
CREATE mycv/tsconfig.app.json (424 bytes)
CREATE mycv/tsconfig.spec.json (434 bytes)
CREATE mycv/.vscode/extensions.json (130 bytes)
CREATE mycv/.vscode/launch.json (470 bytes)
CREATE mycv/.vscode/tasks.json (938 bytes)
CREATE mycv/src/main.ts (248 bytes)
CREATE mycv/src/index.html (290 bytes)
CREATE mycv/src/styles.css (80 bytes)
CREATE mycv/src/app/app-routing.module.ts (245 bytes)
CREATE mycv/src/app/app.module.ts (393 bytes)
CREATE mycv/src/app/app.component.css (0 bytes)
CREATE mycv/src/app/app.component.html (19903 bytes)
CREATE mycv/src/app/app.component.spec.ts (1037 bytes)
CREATE mycv/src/app/app.component.ts (205 bytes)
CREATE mycv/public/favicon.ico (15086 bytes)
" Installing packages (npm)...
```

Posteriormente CLI de angular creará una carpeta llamada **mycv**, dentro de mycv Angular CLI creará un template de nuestro proyecto y descarga los paquetes de nodejs necesarios para que funcione la plantilla inicial.

```
[CREATE mycv/public/favicon.ico (15086 bytes)
✓ Packages installed successfully.
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
      Successfully initialized git.
→ ~ ]
```

Una vez terminado el proceso de creación del proyecto, se configura esta carpeta mycv, como un repositorio de git.

Ahora nos movemos a la carpeta de nuestro proyecto con el comando:

\$ cd **mycv**

```
[→ ~ cd mycv
→ mycv git:(master) ]
```

Podemos visualizar la estructura de nuestro proyecto con el comando: ls -la

\$ **ls -la**

```
[→ mycv git:(master) ls -la
total 1016
drwxr-xr-x  16 adsoft  staff      512 Nov 18 16:26 .
drwxr-x---+ 76 adsoft  staff     2432 Nov 18 16:32 ..
-rw-r--r--   1 adsoft  staff      274 Nov 18 16:24 .editorconfig
drwxr-xr-x  12 adsoft  staff      384 Nov 18 16:26 .git
-rw-r--r--   1 adsoft  staff      587 Nov 18 16:24 .gitignore
drwxr-xr-x   5 adsoft  staff      160 Nov 18 16:24 .vscode
-rw-r--r--   1 adsoft  staff     1065 Nov 18 16:24 README.md
-rw-r--r--   1 adsoft  staff     2582 Nov 18 16:24 angular.json
drwxr-xr-x  567 adsoft  staff    18144 Nov 18 16:26 node_modules
-rw-r--r--   1 adsoft  staff    484245 Nov 18 16:26 package-lock.json
-rw-r--r--   1 adsoft  staff     1035 Nov 18 16:24 package.json
drwxr-xr-x   3 adsoft  staff      96 Nov 18 16:24 public
drwxr-xr-x   6 adsoft  staff     192 Nov 18 16:24 src
-rw-r--r--   1 adsoft  staff     424 Nov 18 16:24 tsconfig.app.json
-rw-r--r--   1 adsoft  staff    1021 Nov 18 16:24 tsconfig.json
-rw-r--r--   1 adsoft  staff     434 Nov 18 16:24 tsconfig.spec.json
→ mycv git:(master) ]
```

Podemos visualizar algunos de los siguientes archivos y carpetas:

- node\_modules
- package.json
- .git
- src
- public

Podemos probar nuestro proyecto inicial con el comando: ng serve

\$ **ng serve**

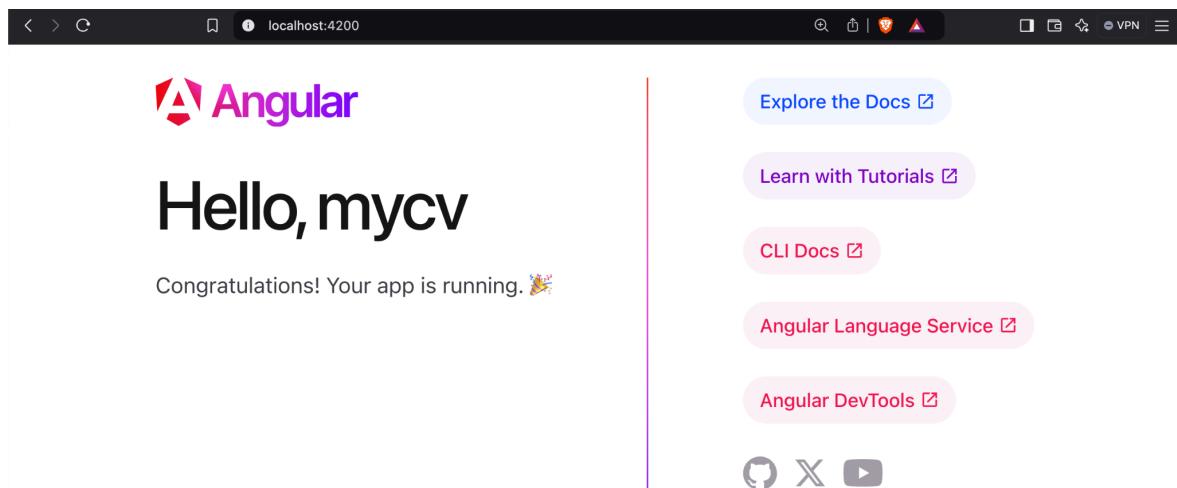
```
mycv — ng serve — ng — esbuild -v _CFBundleIdentifier=com.apple.Terminal TMPDIR=/var/folders/36/rk953hg90lb4n8wvdj3w0jf
→ mycv git:(master) ng serve
Initial chunk files | Names | Raw size
polyfills.js | polyfills | 90.20 kB
main.js | main | 22.80 kB
styles.css | styles | 95 bytes

| Initial total | 113.10 kB

Application bundle generation complete. [4.334 seconds]

Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
→ Local: http://localhost:4200/
→ press h + enter to show help
```

Podemos visualizar el template abriendo nuestro navegador y capturar la dirección: <http://localhost:4200>



Ya tienes algunas herramientas que te permitirán construir una aplicación Web simple usando Angular CLI, y arrancarla en modo frontend usando el comando `ng serve`, así como visualizar la aplicación en la dirección: <http://localhost:4200>.

### Sala de inspiración

El conocimiento que has adquirido hasta ahora, te permite construir y ejecutar una aplicación simple de Angular.

- ¿Harás una ejercicio mental, analizando qué aplicaciones web te gustaría realizar en tu organización usando Angular?

- ¿A qué tipo de sectores económicos o actividades podría beneficiar la creación de aplicaciones web usando este modelo de trabajo ? Lista al menos 3.
- Podrías comprobar si puedes visualizar una aplicación de forma local ?, verifica que puerto usa angular por default
- Si alguna vez has hecho una página Web en otra tecnología, ¿Como te imaginas podrías ponerla en un servidor real ?

Las empresas en la actualidad usan Angular porque permite crear plataformas web de manera muy práctica y sencilla.

### **Sala de pruebas**

Lee cada una de las preguntas y selecciona la respuesta que consideres correcta de acuerdo a tu experiencia hasta ahora.

**1. ¿Cómo crearías un proyecto en Angular llamado *mysite* ?**

<b>Opción a:</b>  ng mysite	Incorrecta:  Esta sintaxis es incorrecta.
<b>Opción b:</b>  ng serve mysite	Incorrecta:  Esta sintaxis es incorrecta.
<b>Opción c:</b>  ng mysite new	Incorrecta:  Esta sintaxis es incorrecta.
<b>Opción d:</b>  <b>ng new mysite</b>	<b>CORRECTA:</b>  <i>Sintaxis correcta, esta operación creará un template de proyecto angular llamado mysite en una carpeta del mismo nombre.</i>

**2. Cómo ejecutas localmente un proyecto de Angular**

<b>Opción a:</b>  ng test	Incorrecta  ng test permite ejecutar solo las pruebas automatizadas del proyecto.
<b>Opción b:</b>  ng build	Incorrecta  ng build, permite crear una versión distribuible del proyecto en términos de HTML, CSS y Javascript
<b>Opción c:</b>  ng serve	<b>CORRECTA</b>  ng serve, permite ejecutar un proyecto angular de forma local y lo visualiza en la dirección: <a href="http://localhost:4200">http://localhost:4200</a>
<b>Opción d:</b>  ng new	Incorrecta:  ng new permite crear un proyecto nuevo de angular.

### 3.- En qué puerto se ejecuta una aplicación Angular

<b>Opción a:</b>  80	Incorrecta:  El puerto por default donde se ejecuta una aplicación en angular es 4200.
<b>Opción b:</b>  8080	Incorrecta:  El puerto por default donde se ejecuta una aplicación en angular es 4200.

Opción c:  4200	<b>CORRECTA</b>  <i>4200 es el puerto donde angular visualiza los proyectos de forma local.</i>
Opción d:  8081	Incorrecta:  El puerto por default donde se ejecuta una aplicación en angular es 4200.

### SUBTEMA 2: Definición de componente en Angular y sus características

Una de las principales características de Angular es que tiene un enfoque basado en componentes, por tanto en angular las aplicaciones están formadas por siempre por componentes como base fundamental.

*En Angular, un componente es una unidad básica de construcción de una aplicación que encapsula la lógica, la plantilla (HTML) y los estilos (CSS) de una parte específica de la interfaz de usuario. Un componente es un bloque de construcción reutilizable y autónomo que representa una parte de la interfaz de usuario y puede tener su propio comportamiento y datos.*

Un componente está formado por 4 elementos:

1. Template (HTML): El archivo de plantilla HTML que define la estructura visual del componente.
2. Logic (TypeScript): El archivo de lógica TypeScript que contiene la funcionalidad del componente, incluyendo métodos y variables.

3. Styles (CSS): El archivo de estilos CSS que define la apariencia visual del componente.
4. Pruebas(.spec.ts) El archivo de pruebas es usado para codificar pruebas automatizadas (unitarias y de integración).

Los componentes en Angular se utilizan para crear elementos visuales interactivos y reutilizables en una aplicación. Pueden ser utilizados para representar elementos como:

- Una barra de navegación
- Un formulario de registro
- Una lista de elementos
- Un detalle de un elemento

Los componentes en Angular tienen varias características importantes:

- Reutilizable: Los componentes pueden ser reutilizados en diferentes partes de la aplicación.
- Autónomo: Cada componente tiene su propio espacio de nombres y no afecta a otros componentes.
- Encapsulación: Los componentes encapsulan su lógica y estilos, lo que facilita la maintainability y la escalabilidad de la aplicación.
- Inyección de dependencias: Los componentes pueden recibir dependencias injectadas, como servicios o otros componentes, para interactuar con ellos.

En resumen, los componentes en Angular son la base fundamental de las aplicaciones en este framework y permiten crear interfaces de usuario flexibles, escalables y mantenibles.

Veamos con detenimiento la siguiente pantalla de <https://youtube.com>

Analiza cómo dividirías en componentes esta aplicación?

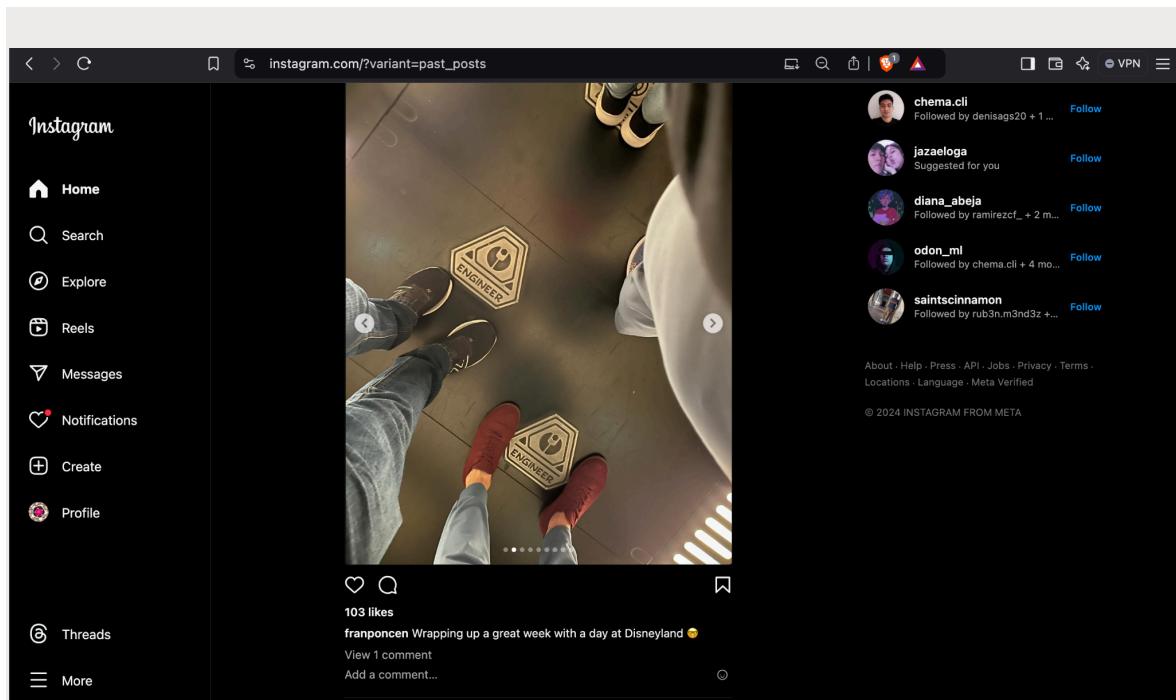
## Plantilla Nivel de Conocimiento



Posibles componentes: **menú lateral, búsqueda, perfil de usuario, recomendaciones, visor de video, comentarios, canales**, entre otros.

Ahora veamos esta plataforma con detenimiento,

Analiza cómo dividirías en componentes esta aplicación? cual es tu respuesta .



## Estructura de una aplicación en Angular

En esta sección analizaremos la estructura de una aplicación en Angular, para ver la lista de archivos y carpetas creadas en el template usaremos:

```
$ ls -la
```

```
.angular
.editorconfig
.git
.gitignore
.vscode
README.md
angular.json
node_modules
package-lock.json
package.json
public
src
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
```

Podemos destacar los siguientes archivos y carpetas más importantes:

1. **.git** - Carpeta oculta que indica que nuestro proyecto Angular es un repositorio de git, almacena todos los archivos de versiones de código.
2. **.gitignore** - Archivo de configuración de git que almacena los archivos, o carpetas que deben ignorarse al subir el repositorio a github.com
3. **node\_modules** - Carpeta donde se guardan todos los archivos que forman los paquetes de javascript instalados en el proyecto, regularmente ocupa mucho espacio y está en el archivo .gitignore para no ser subido a la nube de github.com
4. **package.json** - Archivo de configuración en formato .json que guarda todas las dependencias que forman parte del proyecto.
5. **src** - Carpeta donde almacena todos los archivos que forman parte del código fuente del proyecto.

Ahora analizaremos a detalle el código de la aplicación que se encuentra en la carpeta src. Dentro de nuestro terminal podemos visualizar la carpeta src.

```
$ ls -la src
```

```
[→ mycv git:(master) ls -la src
total 24
drwxr-xr-x  6 adsoft  staff  192 Nov 18 16:24 .
drwxr-xr-x 17 adsoft  staff  544 Nov 18 18:03 ..
drwxr-xr-x  8 adsoft  staff  256 Nov 18 16:24 app
-rw-r--r--  1 adsoft  staff  290 Nov 18 16:24 index.html
-rw-r--r--  1 adsoft  staff  250 Nov 18 16:24 main.ts
-rw-r--r--  1 adsoft  staff   80 Nov 18 16:24 styles.css]
```

Podemos ver el index.html que es el archivo de arranque de nuestra aplicación, con su respectiva hoja de estilos global que aplica a todo el proyecto: styles.css.

Podemos visualizar el archivo index.html

```
→ mycv git:(master) cat src/index.html
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Mycv</title>
    <base href="/">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
    <app-root></app-root>
</body>
</html>
```

Podemos analizar que dentro de la etiqueta <body>, el archivo index.html llama al componente principal <app-root></app-root> el cual está ubicado en la carpeta src/app.

Podemos ver el contenido de de la carpeta app con:

```
$ ls -la src/app
```

```
→ mycv git:(master) ls -la src/app
total 72
drwxr-xr-x  8 adsoft  staff   256 Nov 18 16:24 .
drwxr-xr-x  6 adsoft  staff   192 Nov 18 16:24 ..
-rw-r--r--  1 adsoft  staff     0 Nov 18 16:24 app.component.css
-rw-r--r--  1 adsoft  staff  19903 Nov 18 16:24 app.component.html
-rw-r--r--  1 adsoft  staff   910 Nov 18 16:24 app.component.spec.ts
-rw-r--r--  1 adsoft  staff   300 Nov 18 16:24 app.component.ts
-rw-r--r--  1 adsoft  staff   310 Nov 18 16:24 app.config.ts
-rw-r--r--  1 adsoft  staff    77 Nov 18 16:24 app.routes.ts
```

En este punto podemos ver los 4 elementos del componente principal cuyo nombre siempre empieza con: **app.component**

1. app.component.ts

Es el elemento principal del componente, contiene el Código TypeScript que define el nombre del componente: **AppComponent**, el nombre de su selector : **app-root** que es el nombre con el que puede ser insertado en cualquier parte del proyecto. Análogamente define como se llama el archivo HTML y CSS del componente.

```
[→ mycv git:(master) cat src/app/app.component.ts
import { Component } from '@angular/core';

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

## 2. app.component.css

Almacena las hojas de estilo aplicables sólo al componente app.component, puedes visualizar el contenido actual con:

```
→ mycv git:(master) cat src/app/app.component.css  
→ mycv git:(master)
```

### 3. app.component.html

Almacena los elementos visuales HTML del template creado por el CLI, podemos visualizar con:

```
|> mycv git:(master) cat src/app/app.component.html
<!-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * -->
<!-- * * * * * * * * * * * * * The content below * * * * * * * * * * * * * -->
<!-- * * * * * * * * * * * is only a placeholder * * * * * * * * * * * * * -->
<!-- * * * * * * * * * * * and can be replaced. * * * * * * * * * * * * * -->
<!-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * -->
<!-- * * * * * * * * Delete the template below * * * * * * * * * * * * * -->
<!-- * * * * * * * * to get started with your project! * * * * * * * -->
<!-- * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * -->

<style>
:host {
  --bright-blue: oklch(51.01% 0.274 263.83);
  --electric-violet: oklch(53.18% 0.28 296.97);
  --french-violet: oklch(47.66% 0.246 305.88);
  --vivid-pink: oklch(69.02% 0.277 332.77);
  --hot-red: oklch(61.42% 0.238 15.34);
  --orange-red: oklch(63.32% 0.24 31.68);
```

#### 4. app.component.spec.ts

Contiene pruebas automatizadas para probar el código TypeScript del componente, utiliza una nomenclatura especial que se analizará en el tema 4. Podemos ver su contenido con el siguiente comando:

```
[→ mycv git:(master) cat src/app/app.component.spec.ts
import { TestBed } from '@angular/core/testing';
import { RouterModule } from '@angular/router';
import { AppComponent } from './app.component';

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

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

  it(`should have as title 'mycv'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('mycv');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
    expect(compiled.querySelector('h1')?.textContent).toContain('Hello, mycv');
  });
});
```

Los archivos app.config.ts y app.routes serán analizados en las siguientes secciones del curso.

Como práctica vamos a modificar el archivo app.component.html, utilizando el editor de texto nano, abrimos el archivo:

```
[→ mycv git:(master) nano src/app/app.component.html
```

```
UW PICO 5.09                               File: src/app/app.component.html

!-- * * * * * * * * * * * * * * * * * * * * * * * * * * -->
<!-- * * * * * * * * * * * * * * * * * * * * * * * * * -->
<!-- * * * * * * * * * is only a placeholder * * * * * * * * -->
<!-- * * * * * * * * * and can be replaced. * * * * * * * * -->
<!-- * * * * * * * * * * * * * * * * * * * * * * * * * -->
<!-- * * * * * * * * * Delete the template below * * * * * * * * -->
<!-- * * * * * * * * to get started with your project! * * * * * * * -->
<!-- * * * * * * * * * * * * * * * * * * * * * * * * * * -->

<style>
:host {
    --bright-blue: oklch(51.01% 0.274 263.83);
    --electric-violet: oklch(53.18% 0.28 296.97);
    --french-violet: oklch(47.66% 0.246 305.88);
    --vivid-pink: oklch(69.02% 0.277 332.77);
    --hot-red: oklch(61.42% 0.238 15.34);
    --orange-red: oklch(63.32% 0.24 31.68);

    --gray-900: oklch(19.37% 0.006 300.98);
    --gray-700: oklch(36.98% 0.014 302.71);
    --gray-400: oklch(70.9% 0.015 304.04);

    --red-to-pink-to-purple-vertical-gradient: linear-gradient(
        180deg,
        var(--orange-red) 0%,
        var(--vivid-pink) 50%,
        var(--electric-violet) 100%
    )
}

^G Get Help      ^O WriteOut     ^R Read File     ^Y Prev Pg     ^K Cut Text     ^C Cur Pos
^X Exit         ^J Justify      ^W Where is      ^V Next Pg      ^U UnCut Text   ^T To Spell
```

Eliminamos todas las líneas y solo dejamos nuestro nombre con la etiqueta:

<h3> Adolfo Centeno Tellez</h3>

```
UW PICO 5.09                               File: src/app/app.component.html                               Modified

<h3>Adolfo Centeno Tellez</h3>

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ? [Y] Yes [N] No
```

Salimos y guardamos con la tecla Ctrl-X, presionamos Y y <enter>

Repetimos el comando: \$ ng serve

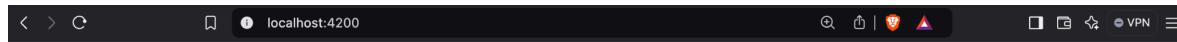
```
[→ mycv git:(master) ✘ ng serve
Initial chunk files | Names          | Raw size
polyfills.js         | polyfills      | 90.20 kB |
main.js              | main           | 1.51 kB |
styles.css           | styles          | 95 bytes |

| Initial total | 91.81 kB

Application bundle generation complete. [2.337 seconds]

Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
  → Local: http://localhost:4200/
  → press h + enter to show help
```

podemos ver los cambios en el navegador:



**Adolfo Centeno Tellez**

Hasta ahora ya hemos analizado la estructura de un proyecto en Angular, analizando sus archivos y carpetas principales. Además hemos revisado el componente principal denominado AppComponent y sus 4 archivos: typescript (.ts), html, css y pruebas (.spec.ts).

### Sala de inspiración

Todas las plataformas web están basadas en componentes, se construyen de forma modular y se integran en un componente principal. Esto permite trabajar en equipos de desarrollo y distribuir las tareas basadas en componentes, trabajarlas de manera paralela e integrarlas continuamente.

### Sala de pruebas

Lee cada una de las preguntas y selecciona la respuesta que consideres correcta

1.- *Archivo del componente principal de Angular que contiene el código Typescript*

<b>Opción a:</b>  app.component.html	Incorrecta:  app.component.html guarda las etiquetas que forman la interfaz visual.
<b>Opción b:</b>  app.component.css	Incorrecta:  app.component.css almacena hojas de estilo para mejorar la apariencia del HTML
<b>Opción c:</b>  <b>app.component.ts</b>	<b>CORRECTA:</b>  app.component.ts contiene el código Typescript con el nombre de la clase del componente, su selector y nombre del html y css.
<b>Opción d:</b>  app.component.spec.ts	Incorrecta:  app.component.spec.ts guarda las pruebas automatizadas para probar el código Typescript.

2.- *Archivo del componente principal de Angular que contiene el código de las pruebas automatizadas:*

<b>Opción a:</b>  app.component.html	Incorrecta:  app.component.html guarda las etiquetas que forman la interfaz visual.
<b>Opción b:</b>  app.component.css	Incorrecta:  app.component.css almacena hojas de estilo para mejorar la apariencia del HTML
<b>Opción c:</b>  app.component.ts	Incorrecta:  app.component.ts contiene el código Typescript con el nombre

	de la clase del componente, su selector y nombre del html y css.
<b>Opción d:</b>  app.component.spec.ts	<b>CORRECTA:</b>  app.component.spec.ts guarda las pruebas automatizadas para probar el código Typescript

3.- Archivo del componente principal de Angular que contiene las etiquetas para construir la interfaz visual del componente

<b>Opción a:</b>  app.component.html	<b>CORRECTA</b>  app.component.html guarda las etiquetas que forman la interfaz visual.
<b>Opción b:</b>  app.component.css	Incorrecta:  app.component.css almacena hojas de estilo para mejorar la apariencia del HTML
<b>Opción c:</b>  app.component.ts	Incorrecta:  app.component.ts contiene el código Typescript con el nombre de la clase del componente, su selector y nombre del html y css.
<b>Opción d:</b>  app.component.spec.ts	Incorrecta:  app.component.spec.ts guarda las pruebas automatizadas para probar el código Typescript



## SUBTEMA 3: Creación de una aplicación simple basada en componentes.

En esta sección crearemos una aplicación basada en componentes, usaremos el proyecto mycv del tema anterior, para crear nuestro propio CV estilo Elon Musk en una aplicación web.

La idea es tener nuestro CV en una sola página como se muestra en la siguiente ilustración:

Elon Musk



elonmusk@teslamotors.com

650-681-5000

Los Angeles, USA

elon.musk

---

**Work Experience**
**Skills & Competences**

<div style="border-bottom: 1px solid black; padding-bottom: 10px;"> <b>06/2006 - Present</b>  <b>Chairman</b>  <b>SolarCity</b> </div> <div style="margin-top: 5px;"> <b>Accomplishments</b>                      Created a collaboration between SolarCity and Tesla to use electric vehicle batteries to smooth the impact of rooftop solar on the power grid.                      Provided the initial concept and financial capital.                 </div> <div style="border-bottom: 1px solid black; margin-top: 10px;"> <b>02/2004 - Present</b>  <b>CEO and Product Architect</b>  <b>Tesla Motors</b> </div> <div style="margin-top: 5px;"> <b>Accomplishments</b>                      Currently oversee the company's product strategy – including the design, engineering and manufacturing of more and more affordable electric vehicles for mainstream consumers.                      Insisted on using carbon fiber composite materials in the hull to minimize weight, developed the battery module and even some elements of design, like the headlights.                      Received Global Green 2006 product design award for Tesla Roadster design.                 </div> <div style="border-bottom: 1px solid black; margin-top: 10px;"> <b>06/2002 - Present</b>  <b>CEO and CTO</b>  <b>SpaceX</b> </div> <div style="margin-top: 5px;"> <b>Accomplishments</b>                      Plans to reduce space transportation costs to enable people to colonize Mars.                      Oversees the development of rockets and spacecraft for missions to Earth orbit and ultimately to other planets.                      Developed the Falcon 9 spacecraft which replaced the space shuttle when it retired in 2011.                 </div> <div style="border-bottom: 1px solid black; margin-top: 10px;"> <b>03/1999 - 10/2002</b>  <b>CEO</b>  <b>X.com and PayPal</b> </div> <div style="margin-top: 5px;"> <b>Accomplishments</b>                      Involved in the development of new business models, conducted a successful viral marketing campaign, which led to a rapid increase in the number of customers.                      Created a method of securely transferring money using a recipient's e-mail address.                 </div> <div style="border-bottom: 1px solid black; margin-top: 10px;"> <b>01/1995 - 02/1995</b>  <b>Co-founder</b>  <b>Zip2</b> </div> <div style="margin-top: 5px;"> <b>Accomplishments</b>                      Created a platform where newspapers – including credible ones as New York Times – could offer their customers some additional commercial services.                 </div> <div style="border-bottom: 1px solid black; margin-top: 10px;"> <b>Education</b> </div> <div style="border-bottom: 1px solid black; margin-top: 10px;"> <b>09/1992 - 06/1995</b>  <b>Bachelor of Science in Economics</b>                      Wharton School of the University of Pennsylvania                 </div> <div style="border-bottom: 1px solid black; margin-top: 10px;"> <b>09/1992 - 06/1995</b>  <b>Bachelor of Science in Physics</b>                      Penn's College of Arts and Sciences                 </div>	<div style="border-bottom: 1px solid black; padding-bottom: 10px;"> <b>Thinking through first principles</b> <div style="display: flex; justify-content: space-around; width: 100%;"> <span></span> <span></span> <span></span> <span></span> </div> </div> <div style="border-bottom: 1px solid black; margin-top: 5px;"> <b>Micromanaging</b> <div style="display: flex; justify-content: space-around; width: 100%;"> <span></span> <span></span> </div> </div> <div style="border-bottom: 1px solid black; margin-top: 5px;"> <b>Goal oriented</b> <div style="display: flex; justify-content: space-around; width: 100%;"> <span></span> <span></span> </div> </div> <div style="border-bottom: 1px solid black; margin-top: 5px;"> <b>Future focused</b> <div style="display: flex; justify-content: space-around; width: 100%;"> <span></span> <span></span> </div> </div> <div style="border-bottom: 1px solid black; margin-top: 5px;"> <b>Critical thinking</b> <div style="display: flex; justify-content: space-around; width: 100%;"> <span></span> <span></span> </div> </div> <div style="border-bottom: 1px solid black; margin-top: 5px;"> <b>Resiliency</b> <div style="display: flex; justify-content: space-around; width: 100%;"> <span></span> <span></span> </div> </div> <div style="border-bottom: 1px solid black; margin-top: 5px;"> <b>Verbal and written communication</b> <div style="display: flex; justify-content: space-around; width: 100%;"> <span></span> <span></span> </div> </div> <div style="border-bottom: 1px solid black; margin-top: 5px;"> <b>Leadership</b> <div style="display: flex; justify-content: space-around; width: 100%;"> <span></span> <span></span> </div> </div> <div style="border-bottom: 1px solid black; margin-top: 5px;"> <b>Creativity</b> <div style="display: flex; justify-content: space-around; width: 100%;"> <span></span> <span></span> </div> </div> <div style="border-bottom: 1px solid black; margin-top: 5px;"> <b>Time Management</b> <div style="display: flex; justify-content: space-around; width: 100%;"> <span></span> <span></span> </div> </div> <div style="border-bottom: 1px solid black; margin-top: 10px;"> <b>Achievements &amp; Certificates</b> </div> <div style="margin-top: 5px;"> <b>IEEE Honorary Membership (2015)</b>  <small>Given to people who have rendered meritorious service to humanity in the IEEE's designated fields of interest.</small> </div> <div style="margin-top: 5px;"> <b>Businessperson of the Year by Fortune Magazine (2013)</b>  <small>Prize received for the following companies: 'SpaceX', 'Tesla Motors' and 'SolarCity'</small> </div> <div style="margin-top: 5px;"> <b>FIA Gold Space Medal (2010)</b>  <small>One of the highest honors in the aerospace industry, shared with prominent personalities like Neil Armstrong and John Glenn.</small> </div> <div style="margin-top: 5px;"> <b>Honorary doctorate in Design from the Art Center College of Design</b> </div> <div style="margin-top: 5px;"> <b>Honorary doctorate (DUniv) in Aerospace Engineering from the University of Surrey</b> </div> <div style="margin-top: 5px;"> <b>Honorary doctorate of Engineering and Technology from Yale University</b> </div> <div style="border-bottom: 1px solid black; margin-top: 10px;"> <b>Languages</b> </div> <div style="display: flex; justify-content: space-between; margin-top: 5px;"> <span>English</span> <span>Afrikaans</span> </div> <div style="border-bottom: 1px solid black; margin-top: 10px;"> <b>Interests</b> </div> <div style="margin-top: 5px;">         Physics   Sustainability   Philanthropy   Extraterrestrial life   Alternative energy sources   Space engineering   Reading   Video games     </div>
--	---



24

The Learning Gate | Tecnológico de Monterrey

Como primer paso identificamos cuantos y cuales componentes formarán nuestro proyecto:

- 1.- Header
- 2.- Work Experience
- 3.- Education
- 4.- Skills
- 5.- Certificates
- 6.- Languages
- 7.- Interests

*NOTA: Estos 7 componentes serán insertados en el componente principal AppComponent mediante el ID de sus selectores.*

Ahora crearemos cada uno de nuestros componentes usando el CLI con el comando: **ng generate component <component-name>**

**Para crear los componentes nos aseguramos de que estemos en el root del proyecto:**

```
[→ mycv git:(master) ✘ ls -la
total 1016
drwxr-xr-x  17 adsoft  staff   544 Nov 18 18:03 .
drwxr-x---+ 76 adsoft  staff  2432 Nov 18 23:20 ..
drwxr-xr-x  3 adsoft  staff   96 Nov 18 18:03 .angular
-rw-r--r--  1 adsoft  staff  274 Nov 18 16:24 .editorconfig
drwxr-xr-x 12 adsoft  staff  384 Nov 18 16:26 .git
-rw-r--r--  1 adsoft  staff  587 Nov 18 16:24 .gitignore
drwxr-xr-x  5 adsoft  staff  160 Nov 18 16:24 .vscode
-rw-r--r--  1 adsoft  staff  1065 Nov 18 16:24 README.md
-rw-r--r--  1 adsoft  staff  2582 Nov 18 16:24 angular.json
drwxr-xr-x 567 adsoft  staff 18144 Nov 18 16:26 node_modules
-rw-r--r--  1 adsoft  staff 484245 Nov 18 16:26 package-lock.json
-rw-r--r--  1 adsoft  staff  1035 Nov 18 16:24 package.json
drwxr-xr-x  3 adsoft  staff   96 Nov 18 16:24 public
drwxr-xr-x  6 adsoft  staff  192 Nov 18 16:24 src
-rw-r--r--  1 adsoft  staff  424 Nov 18 16:24 tsconfig.app.json
-rw-r--r--  1 adsoft  staff 1021 Nov 18 16:24 tsconfig.json
-rw-r--r--  1 adsoft  staff  434 Nov 18 16:24 tsconfig.spec.json
→ mycv git:(master) ✘ ]
```

*Como tip, debemos tener a la vista el archivo package.json o la carpeta node\_modules.*

Crear el componente header usando:

\$ ng generate component header

```
→ mycv git:(master) ✘ ng generate component header

CREATE src/app/header/header.component.css (0 bytes)
CREATE src/app/header/header.component.html (21 bytes)
CREATE src/app/header/header.component.spec.ts (597 bytes)
CREATE src/app/header/header.component.ts (199 bytes)
UPDATE src/app/app.module.ts (475 bytes)
→ mycv git:(master) ✘
```

Cada vez que generamos un componente se crea una carpeta con su nombre. Podemos ver el componente header dentro de la carpeta src/app, usamos el comando:

\$ ls -la src/app/header

```
[→ mycv git:(master) ✘ ls -la src/app/header
total 24
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:22 .
drwxr-xr-x  9 adsoft  staff  288 Nov 18 23:22 ..
-rw-r--r--  1 adsoft  staff     0 Nov 18 23:22 header.component.css
-rw-r--r--  1 adsoft  staff    21 Nov 18 23:22 header.component.html
-rw-r--r--  1 adsoft  staff   592 Nov 18 23:22 header.component.spec.ts
-rw-r--r--  1 adsoft  staff   234 Nov 18 23:22 header.component.ts
→ mycv git:(master) ✘
```

Ahora creamos el componente work-experience con el comando:

\$ ng generate component work-experience

```
→ mycv git:(master) ✘ ng generate component work-experience

CREATE src/app/work-experience/work-experience.component.css (0 bytes)
CREATE src/app/work-experience/work-experience.component.html (30 bytes)
CREATE src/app/work-experience/work-experience.component.spec.ts (654 bytes)
CREATE src/app/work-experience/work-experience.component.ts (234 bytes)
UPDATE src/app/app.module.ts (591 bytes)
→ mycv git:(master) ✘
```

visualizamos el componente work-experience:

```
→ mycv git:(master) ✘ ls -la src/app/work-experience
total 24
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:31 .
drwxr-xr-x 10 adsoft  staff  320 Nov 18 23:31 ..
-rw-r--r--  1 adsoft  staff   0 Nov 18 23:31 work-experience.component.css
-rw-r--r--  1 adsoft  staff   30 Nov 18 23:31 work-experience.component.html
-rw-r--r--  1 adsoft  staff  649 Nov 18 23:31 work-experience.component.spec.ts
-rw-r--r--  1 adsoft  staff  269 Nov 18 23:31 work-experience.component.ts
→ mycv git:(master) ✘
```

Creamos el componente education con el comando abreviado:

\$ ng g c education

```
→ mycv git:(master) ✘ ng g c education

CREATE src/app/education/education.component.css (0 bytes)
CREATE src/app/education/education.component.html (24 bytes)
CREATE src/app/education/education.component.spec.ts (618 bytes)
CREATE src/app/education/education.component.ts (211 bytes)
UPDATE src/app/app.module.ts (685 bytes)
→ mycv git:(master) ✘
```

visualizamos los archivos del componente education

```
[→ mycv git:(master) ✘ ls -la src/app/education
total 24
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:34 .
drwxr-xr-x 11 adsoft  staff  352 Nov 18 23:34 ..
-rw-r--r--  1 adsoft  staff   0 Nov 18 23:34 education.component.css
-rw-r--r--  1 adsoft  staff   24 Nov 18 23:34 education.component.html
-rw-r--r--  1 adsoft  staff  613 Nov 18 23:34 education.component.spec.ts
-rw-r--r--  1 adsoft  staff  246 Nov 18 23:34 education.component.ts
→ mycv git:(master) ✘
```

Es turno del componente skills:

\$ ng g c skills

```
→ mycv git:(master) ✘ ng g c skills

CREATE src/app/skills/skills.component.css (0 bytes)
CREATE src/app/skills/skills.component.html (21 bytes)
CREATE src/app/skills/skills.component.spec.ts (597 bytes)
CREATE src/app/skills/skills.component.ts (199 bytes)
UPDATE src/app/app.module.ts (767 bytes)
→ mycv git:(master) ✘
```

### visualizamos el componente skills

```
[→ mycv git:(master) × ls -la src/app/skills
total 24
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:36 .
drwxr-xr-x 12 adsoft  staff  384 Nov 18 23:36 ..
-rw-r--r--  1 adsoft  staff   0 Nov 18 23:36 skills.component.css
-rw-r--r--  1 adsoft  staff   21 Nov 18 23:36 skills.component.html
-rw-r--r--  1 adsoft  staff  592 Nov 18 23:36 skills.component.spec.ts
-rw-r--r--  1 adsoft  staff  234 Nov 18 23:36 skills.component.ts
→ mycv git:(master) × ]
```

### Creamos ahora el componente certificates y los visualizamos con los comandos:

- ng g c certificates
- ls -la src/app/certificates

```
[→ mycv git:(master) × ng g c certificates
CREATE src/app/certificates/certificates.component.css (0 bytes)
CREATE src/app/certificates/certificates.component.html (27 bytes)
CREATE src/app/certificates/certificates.component.spec.ts (639 bytes)
CREATE src/app/certificates/certificates.component.ts (223 bytes)
UPDATE src/app/app.module.ts (873 bytes)
→ mycv git:(master) × ]
```

### Creamos el componente languages y los visualizamos con los comandos:

- ng g c languages
- ls -la src/app/languages

```
[→ mycv git:(master) × ng g c languages
CREATE src/app/languages/languages.component.css (0 bytes)
CREATE src/app/languages/languages.component.html (24 bytes)
CREATE src/app/languages/languages.component.spec.ts (618 bytes)
CREATE src/app/languages/languages.component.ts (211 bytes)
UPDATE src/app/app.module.ts (967 bytes)
[→ mycv git:(master) × ls -la src/app/languages
total 24
drwxr-xr-x  6 adsoft  staff  192 Dec 26 01:47 .
drwxr-xr-x 14 adsoft  staff  448 Dec 26 01:47 ..
-rw-r--r--  1 adsoft  staff   0 Dec 26 01:47 languages.component.css
-rw-r--r--  1 adsoft  staff   24 Dec 26 01:47 languages.component.html
-rw-r--r--  1 adsoft  staff  618 Dec 26 01:47 languages.component.spec.ts
-rw-r--r--  1 adsoft  staff  211 Dec 26 01:47 languages.component.ts
→ mycv git:(master) × ]
```

### Finalmente creamos el componente interests y los visualizamos con los comandos:

- **ng g c interests**
- **ls -la src/app/interests**

```
→ mycv git:(master) ✘ ng g c interests

CREATE src/app/interests/interests.component.css (0 bytes)
CREATE src/app/interests/interests.component.html (24 bytes)
CREATE src/app/interests/interests.component.spec.ts (618 bytes)
CREATE src/app/interests/interests.component.ts (211 bytes)
UPDATE src/app/app.module.ts (1061 bytes)
[→ mycv git:(master) ✘ ls -la src/app/interests
total 24
drwxr-xr-x  6 adsoft  staff  192 Dec 26 01:49 .
drwxr-xr-x 15 adsoft  staff  480 Dec 26 01:49 ..
-rw-r--r--  1 adsoft  staff     0 Dec 26 01:49 interests.component.css
-rw-r--r--  1 adsoft  staff    24 Dec 26 01:49 interests.component.html
-rw-r--r--  1 adsoft  staff   618 Dec 26 01:49 interests.component.spec.ts
-rw-r--r--  1 adsoft  staff   211 Dec 26 01:49 interests.component.ts
→ mycv git:(master) ✘
```

Podemos visualizar todos los componentes app dentro de la carpeta de código src.

\$ **ls -la src/app**

```
[→ mycv git:(master) ✘ ls -la src/app
total 40
drwxr-xr-x 15 adsoft  staff  480 Nov 19 16:54 .
drwxr-xr-x  6 adsoft  staff  192 Nov 18 16:24 ..
-rw-r--r--  1 adsoft  staff     0 Nov 18 16:24 app.component.css
-rw-r--r--  1 adsoft  staff   420 Nov 19 16:54 app.component.html
-rw-r--r--  1 adsoft  staff   910 Nov 18 16:24 app.component.spec.ts
-rw-r--r--  1 adsoft  staff   957 Nov 19 14:08 app.component.ts
-rw-r--r--  1 adsoft  staff   310 Nov 18 16:24 app.config.ts
-rw-r--r--  1 adsoft  staff     77 Nov 18 16:24 app.routes.ts
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:39 certificates
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:34 education
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:22 header
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:42 interests
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:41 languages
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:36 skills
drwxr-xr-x  6 adsoft  staff  192 Nov 18 23:31 work-experience
→ mycv git:(master) ✘
```

Una vez creado todos los componentes, probaremos con insertar el componente header dentro del componente principal app.component.

**NOTA:** cada componente nuevo, tiene por default un template html como el siguiente:

```
|→ mycv git:(master) ✘ cat src/app/header/header.component.html
<p>header works!</p>
```

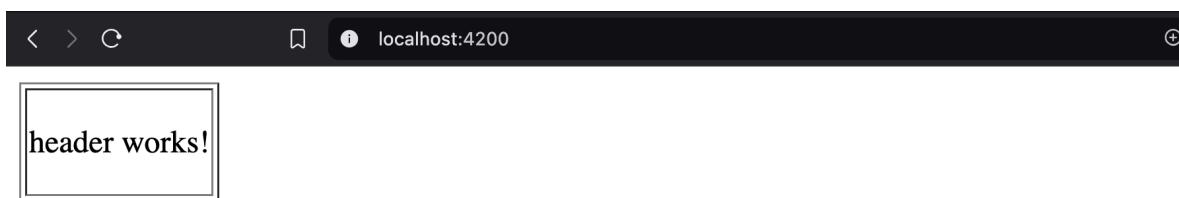
Ahora modificaremos app.component.html para insertar el componente header en un template básico html.

```
$ nano src/app/app.component.html
```

```
1 <table border="1">
2   <tr>
3     <td colspan="2">
4       <app-header></app-header>
5     </td>
6   </tr>
7 </table>
```

Ahora podemos probar como va el proyecto con header insertado en el template:

```
$ ng serve
```



Podemos comprobar que el componente header completo ahora forma parte de nuestra aplicación global, al haber sido insertado en una celda de la tabla html.

Ahora modificaremos el archivo app.component.html para incluir el resto de los componentes en el template.

\$ nano src/app/app.component.html

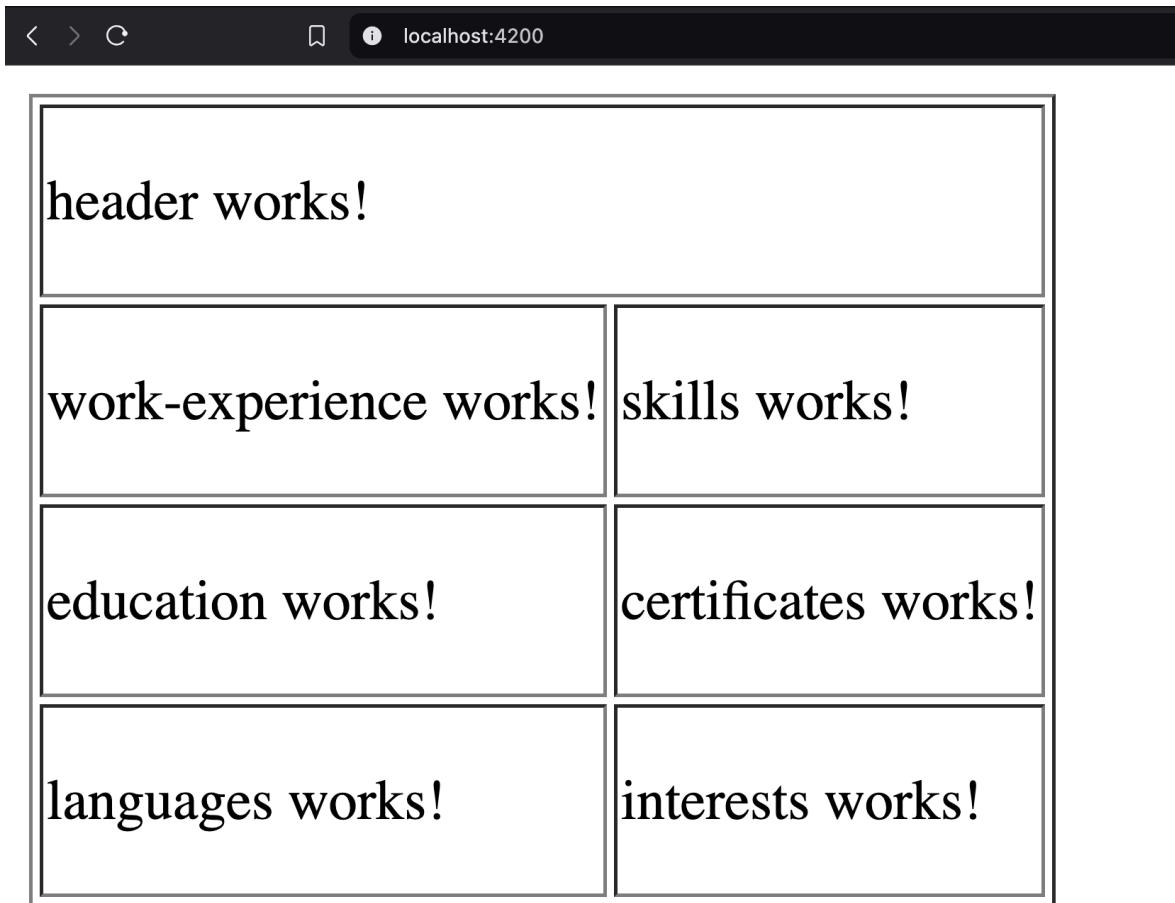
```

1 <table border="1">
2
3 <tr>
4     <td colspan="2"> <app-header></app-header> </td>
5 </tr>
6 <tr>
7     <td> <app-work-experience></app-work-experience> </td>
8     <td> <app-skills></app-skills> </td>
9 </tr>
10
11 <tr>
12     <td> <app-education></app-education> </td>
13     <td> <app-certificates></app-certificates> </td>
14 </tr>
15
16 <tr>
17     <td> <app-languages></app-languages> </td>
18     <td> <app-interests></app-interests> </td>
19 </tr>
20
21 </table>
```

Hasta ahora hemos creado un contenedor de componentes en app.component.ts, y podemos visualizarlo con

\$ ng serve

Abrimos el navegador en <http://localhost:4200>



A screenshot of a web browser window titled "localhost:4200". The page displays a grid of six rectangular boxes arranged in two columns and three rows. The top row contains one box with the text "header works!". The second row contains two boxes: the left one with "work-experience works!" and the right one with "skills works!". The third row contains two boxes: the left one with "education works!" and the right one with "certificates works!". The bottom row contains two boxes: the left one with "languages works!" and the right one with "interests works!". The browser interface includes standard navigation buttons (back, forward, search) and a status bar at the bottom.

header works!	
work-experience works!	skills works!
education works!	certificates works!
languages works!	interests works!

Hasta ahora hemos creado la estructura básica de nuestra aplicación, creando múltiples componentes que representan funcionalidad específica del proyecto, y los hemos insertado en el componente principal de la aplicación.

### Sala de inspiración

Piensa en diferentes plataformas de software a nivel global con que trabajes en el día a día, usando el enfoque basado en componentes como podrías diseñar alguna de esas plataformas? ¿Estás de acuerdo en que no es posible hacer un sitio web global en una sola página ?

Ten en cuenta que una plataforma web de gran escala, no es construida por una sola persona, sino por un grupo de desarrolladores grande, entonces las aplicaciones se dividen en módulos o componentes, y cada desarrollador construye 1 o más componentes de forma paralela que se integran cada determinado tiempo.

### Sala de pruebas

**Pregunta 1:** Cual es nombre del selector del componente principal de la aplicación app.component.ts ?

Opciones de respuesta	Retroalimentación
<b>Opción a:</b> app-component	Incorrecta:  El nombre correcto del selector del componente principal es: app-root
<b>Opción b:</b> <b>app-root</b>	<b>Correcta:</b>  <b>Correcto, el selector del componente AppComment es: app-root</b>
<b>Opción c:</b> app-main	Incorrecta:  El nombre correcto del selector del componente principal es: app-root
<b>Opción d:</b> app-component.ts	Incorrecta:  El nombre correcto del selector del componente principal es: app-root

**Pregunta 2:** Cuál es el comando correcto para crear un componente llamado test:

Opciones de respuesta	Retroalimentación
<b>Opción a:</b> ng generate test	Incorrecta:

## Plantilla Nivel de Conocimiento

	<p>La sintaxis correcta para crear el componente test es: ng generate component test</p>
<b>Opción b:</b> ng g s test	<p>Incorrecta:</p> <p>La sintaxis correcta para crear el componente test es: ng generate component test</p>
<b>Opción c:</b> ng component test	<p>Incorrecta:</p> <p>La sintaxis correcta para crear el componente test es: ng generate component test</p>
<b>Opción d:</b> ng generate component test	<p>Correcta:</p> <p>La sintaxis correcta para crear el componente test es: ng generate component test</p>

**Pregunta 3:** Con qué etiqueta insertas el componente **test** en el HTML de otro componente

Opciones de respuesta	Retroalimentación
<b>Opción a:</b> <test></test>	<p>Incorrecta:</p> <p>Si el componente se llama test, por default su selector es app-test, por tanto las etiquetas para insertar el componente son: &lt;app-test&gt;&lt;/app-test&gt;</p>

Opción b:  <app-test></app-test>	Correcta:  Las etiquetas correctas para insertar el componente test en otro son: <app-test></app-test>
Opción c:  <test><test/>	Incorrecta:  Si el componente se llama test, por default su selector es app-test, por tanto las etiquetas para insertar el componente son: <app-test></app-test>
Opción d:  <app-test><app-test/>	Incorrecta:  Si el componente se llama test, por default su selector es app-test, por tanto las etiquetas para insertar el componente son: <app-test></app-test>

### Contenido (NUTRE /SIGNIFICA)-

Hasta esta sección ya hemos estudiado de forma general cómo crear una ambiente de desarrollo para Angular con las tecnologías git, nvm y angular. Además hemos creado un proyecto angular para crear nuestro propio CV y creado los componentes para desarrollar cada sección. También ya los hemos insertado en el componente principal y tenemos un prototipo inicial de la aplicación funcionando.

### Pruébate ( de los 3 temas )

Este es el momento ideal para realizar algunos ejercicios adicionales con angular y su enfoque basado en componentes..

Ingrasa a: <https://www.amazon.com/> analiza la pagina principal detenidamente

Analiza cuáles componentes serían necesarios para crear una aplicación similar usando Angular 18.

Crea un proyecto nuevo llamado **mystore**, crea cada componente que hayas analizado, modifica el HTML del componente principal e inserta cada componente generado.

Repite los paso anteriores para: <https://open.spotify.com/>

### Ideas para llevar

Actualmente muchas áreas de la ciencia requieren aplicaciones web para visualizar datos; las aplicaciones web son altamente escalables y al ser aplicaciones grandes deben dividirse en componentes para ser administrados y distribuidos en equipos de desarrolladores.

- A través de los ejercicios prácticos ya tienes las habilidades para construir proyectos en angular basados en componentes e integrarlos en el componente principal.
- Los proyectos creados te dan cierta experiencia para mejorar tus habilidades para participar en proyectos web basados en componentes.
- ¿Cómo integrar esta experiencia en los proyectos de tu organización?

El siguiente paso es replicar estos análisis en la información que manipulas como parte de tus responsabilidades laborales y/o de investigación.

### Referencias

Twersky, Emma (2022-06-03). ["Angular v14 is now available!"](#). Angular Blog. Retrieved 2024-12-18.

Gechev, Minko (2024-05-23). "[Meet Angular v19](#)". Medium. Retrieved 2024-12-02.

Fluin, Stephen (6 February 2020). "[Version 9 of Angular Now Available — Project Ivy has arrived!](#)". blog.angular.io. Retrieved 2024-12-15

Gechev, Minko (2024-05-23). "[Angular v18 is now available!](#)". Medium. Retrieved 2024-12-10.

Gechev, Minko (8 November 2023). "[Introducing Angular v17](#)". Medium. Angular Blog. Retrieved 2024-12-16.

### Material de consulta

<https://docs.angular.lat/guide/architecture-components>

<https://ngchallenges.gitbook.io/project/componentes>

<https://codingpotions.com/angular-componentes/>

<https://medium.com/notasdeangular/componentes-en-angular-f25138b00c83>

<https://www.comsoft-mexico.com/blog/que-son-los-componentes-en-angular-y-como-crearlos/>

<https://desarolloweb.com/articulos/practica-angular-modulos-componentes-servicios.html>