

TEMA 1 – DESARROLLO CON FRAMEWORKS

MIGUEL GARRIDO LEÓN 2 – DAW

EJERCICIO 1 - A

1. Realiza los siguientes apartados de introducción a la instalación de Angular y la creación de un primer proyecto.

a. **Investigación sobre Standalone vs. NgModules:**

- Busca información sobre la definición de Standalone Components vs. NgModules.

NgModules: Es una clase que sirve como un contenedor organizativo que lo que hace es agrupar componentes y directivas y pipes que gestionan sus dependencias que serían otras funcionalidades que necesita y lo hace de forma tradicional y además todo componente debe pertenecer a uno.

Standalone Components: Es un componente autosuficiente es decir no necesita un NgModule para funcionar. El solo gestiona sus propias dependencias directamente al definirlo y así simplifica la estructura de la aplicación

Osea en resumen NgModules como una caja de herramientas organizada, pero no es la herramienta contiene componentes y directivas (Agrupar las herramientas), se tiene que declarar lo que se usa (es decir dentro de la caja de herramientas debes decir dentro de mi tengo este componente tal...), también gestiona lo que necesita, es decir: (Si tu componente tal necesita un martillo, la caja de herramientas es la que importa el martillo para que el componente lo use). Y el problema es que si quieras usar ese componente en otro lugar tienes que estar seguro de que la otra parte tenga importe también toda la caja de herramientas que tiene el componente. Ósea el componente va con la caja y en cambio Standalone Component es como una herramienta autosuficiente, es decir contiene una herramienta única y funcional, que declara lo que usa diciendo directamente a la herramienta que eres tú una herramienta funcional ya, y el gestiona lo que necesita, si necesita un martillo, la herramienta misma lo importa directamente y la ventaja de esto es que Puedes coger la herramienta y usarla en cualquier parte del proyecto sin tener que importar una caja de herramientas entera cada vez que lo vallas a usar ese componente.

- **Investiga sobre la motivación de Angular para incorporarlos.**

El principal problema que angular buscaba solucionar al incorporarlos era el excesivo código repetitivo y ceremonial y la complejidad innecesaria que estaba

generada por los NgModules y especialmente en la creación de componentes reutilizables.

Es decir el problema central era que cada componente, directiva o pipe tenía que pasar por los pasos de: Crear el componente, crear un NgModule, declarar el componente , si se iba a usar fuera exportarlo y donde se quería usar importarlo. Y entonces esto generaba una sobrecarga de Archivos, se gestionaban dependencias de forma ineficiente, y la curva de aprendizaje para aprender todo esto se hacia mucho mas elevada para alguien que comienza a querer aprender angular y los Standalone Components es la respuesta de angular para arreglar estos problemas.

- **Ventajas e inconvenientes frente a NgModules.**

Ventajas: Quita la necesidad de hacer un archivo. module.ts solo para decir que un parte existe, haciendo el arreglo de código más simple. **Mejor Rendimiento:** Ayuda a no usar partes que no son necesarias ya que el compilador sabe qué cosas se necesitan del componente, llevando a paquetes app más chicos. **Fácil Reutilización y Aprendizaje:** Los partes entran y se usan directamente, haciendo el sistema Angular más claro y bajando la difícil tarea para los nuevos creadores.

Desventajas: Llevar cosas iguales de lo simple: Las órdenes básicas de Angular como *ngIf o *ngFor (o el CommonModule) necesitan traerse por separado en cada parte que las usa, y eso puede ser molesto. **Reducción de Cierre Fuerte:** Se pierde la capa de orden centrada y oficial que daba el NgModule para juntar grandes funciones.

- **¿En qué casos sigue teniendo sentido usar NgModules ?**

En un gran proyecto (estructura) ¿por qué? Porque el módulo es el "contrato" de la empresa. Pon mucha parte útil (un grupo entero de Ventas o Clientes) y dice que solo se pueden ver algunas partes. Esto ayuda a no tener caos ni mala organización cuando también otros grupos trabajan; Los componentes que están solos dan mucha libertad, el módulo pone orden.

Agrupar Colecciones (Bibliotecas y Módulos Compartidos). ¿Por qué? Es más fácil para el creador. Si tienes diez partes que necesitan usar las mismas cinco herramientas (Pipe, Directive, Servicie), es mejor llevar un solo Share Module una vez en lugar de traer esas cinco herramientas cada vez en la sección importa de cada uno de los diez Standalone Components.

- **Incluye una tabla comparativa con:**

- Archivos característicos de cada enfoque (app.config.ts/app.routes.ts/main.ts con bootstrapApplication vs. app.module.ts/app-routing.module.ts con bootstrapModule).
- Diferencias en imports (en componentes standalone vs. en módulos).

- Impacto en el bootstrap (*arranque de la app en Angular mediante bootstrapApplication o bootstrapModule -NO confundir con el framework CSS Bootstrap-*), aprendizaje y mantenimiento.

Característica	Enfoque Tradicional (NgModules)	Enfoque Moderno (Standalone Components)
Archivos Clave (Arranque)	app.module.ts (Define la estructura) y main.ts (Usa platformBrowserDynamic().bootstrapModule(AppModule)).	main.ts (Usa bootstrapApplication(AppComponent, config)). Las configuraciones y providers iniciales van en app.config.ts y las rutas en app.routes.ts.
Unidad de Lazy Loading (Carga Perezosa)	El Módulo de Característica (FeatureModule). La aplicación se corta en bloques grandes definidos por un .module.ts.	El Componente Standalone. La aplicación se corta directamente por la página o componente asociado a la ruta.
Sintaxis de Lazy Loading	Se usa loadChildren apuntando al archivo del módulo: loadChildren: () => import('./path/to/module').then(m => m.FeatureModule)	Se usa loadComponent apuntando al archivo del componente: loadComponent: () => import('./path/to/component').then(c => c.FeatureComponent)
Diferencia en Imports de Componentes	Implícito a través del Módulo. El componente confía en que su módulo ya importó todo (por ejemplo, el CommonModule o la librería MatButtonModule).	Explícito y Directo. Cada componente SC debe listar exactamente lo que necesita en su propio array imports (ej: imports: [Nglf, MyCustomButtonComponent]).
Impacto en el Bootstrap	Pesado y Clásico. El main.ts carga el AppModule, que a su vez inicializa todo el contexto de la aplicación.	Ligero y Directo. El main.ts llama a bootstrapApplication() y arranca la aplicación usando directamente el AppComponent (que debe ser Standalone). Se elimina la capa del AppModule.
Impacto en el Aprendizaje	Curva de Aprendizaje Alta. El desarrollador debe entender por qué necesita un NgModule aparte del componente.	Curva de Aprendizaje Baja. Es más natural, te centras en el componente y sus dependencias (como en JavaScript moderno).
Impacto en el Mantenimiento	Rígido. Mover o refactorizar un componente suele obligarte a tocar múltiples archivos (.ts del componente, .module.ts, y el módulo que lo usa).	Flexible. Mover un componente es fácil; solo asegúrate de que tiene todas sus dependencias listadas en su propio imports.

EJERCICIO 1 – B

Puesta a punto del entorno:

Verifica que tienes Node y su gestor de paquetes npm.

```
C:\Users\TUF GAMING>node -v
v24.7.0
```

```
C:\Users\TUF GAMING>npm -v
11.5.1
```

Instala Angular CLI

```
C:\Users\TUF GAMING>ng v
```



```
Angular CLI: 17.3.17
Node: 24.7.0 (Unsupported)
Package Manager: npm 11.5.1
OS: win32 x64
```

```
Angular:
```

```
...
```

Package	Version
@angular-devkit/architect	0.1703.17 (cli-only)
@angular-devkit/core	17.3.17 (cli-only)
@angular-devkit/schematics	17.3.17 (cli-only)
@schematics/angular	17.3.17 (cli-only)

```
Warning: The current version of Node (24.7.0) is not supported by Angular.
```

Añade estas extensiones de Google Chrome

Angular DevTools

<https://chromewebstore.google.com/detail/angular-devtools/ienfalfidbdpebioblfackekamfmbnh>



Angular DevTools

Angular DevTools extends Chrome DevTools
adding Angular specific debugging and profiling
capabilities.

Detalles

Quitar



JSON Viewer Pro

<https://chromewebstore.google.com/detail/json-viewer-pro/eifflpmocdbdmepbjaopkkhbfmdgijcc>



JSON Viewer Pro

Detalles

Quitar



Postman Interceptor

<https://chromewebstore.google.com/detail/postman-interceptor/aicmkpgakddgnaphhhpliifpcfhicfo>



Postman Interceptor

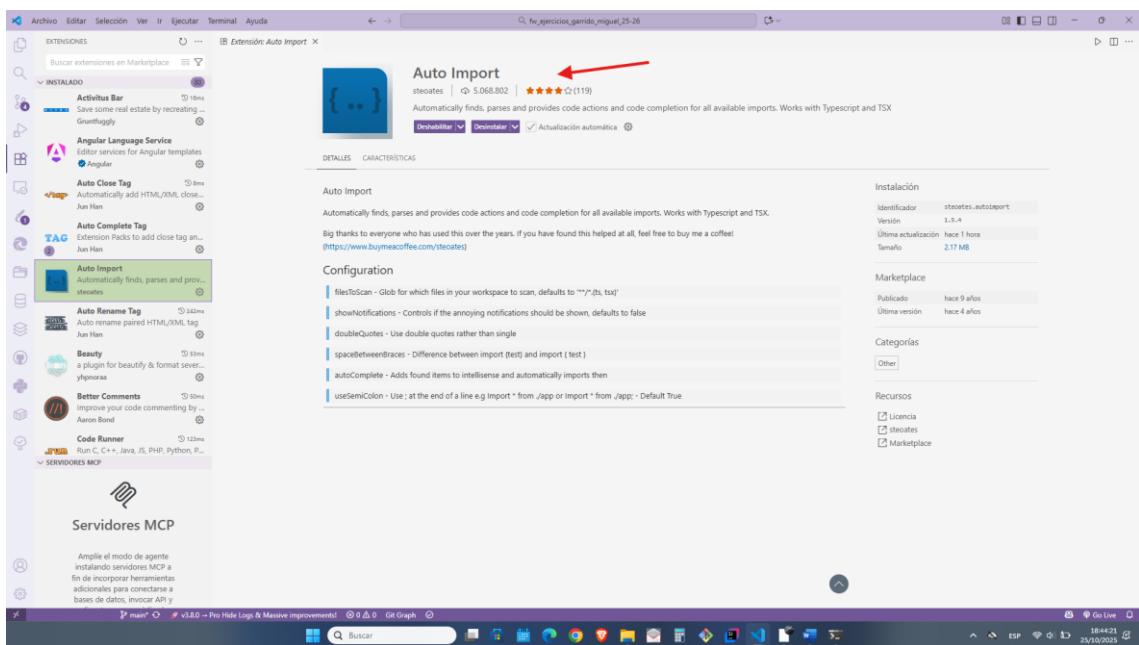
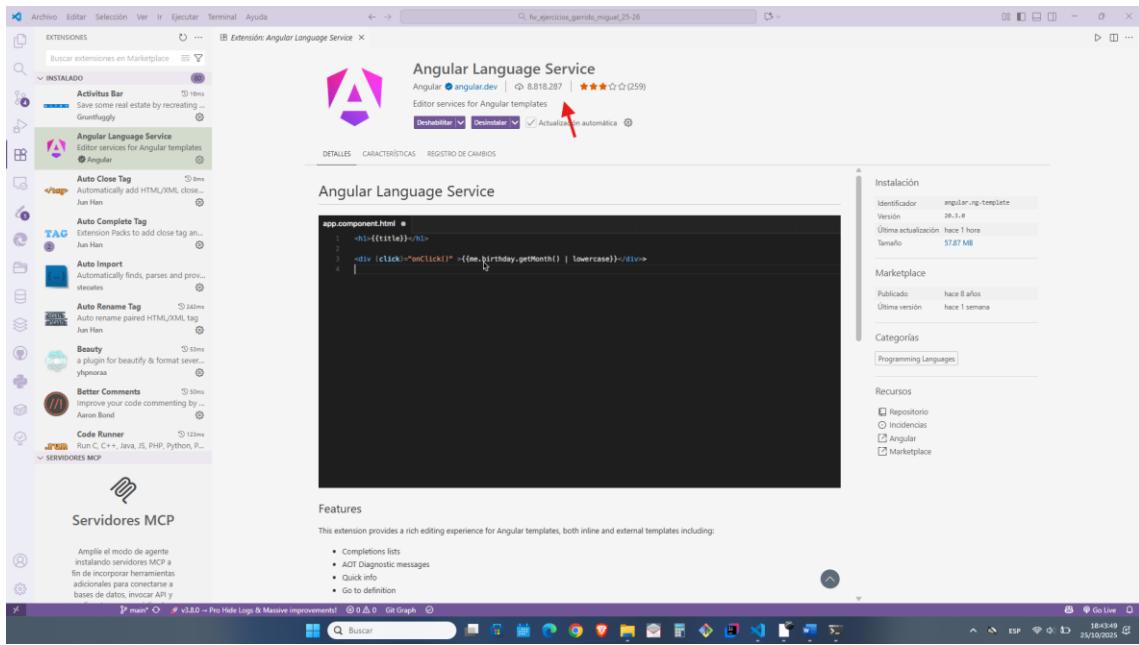
Capture requests from any website and send them to Postman Client.

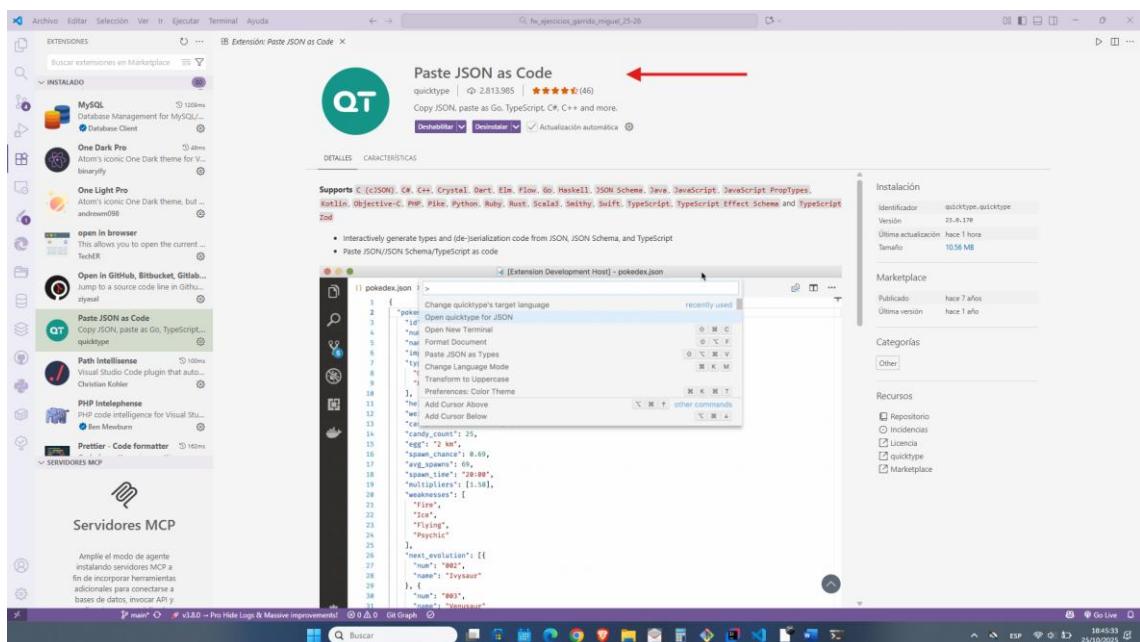
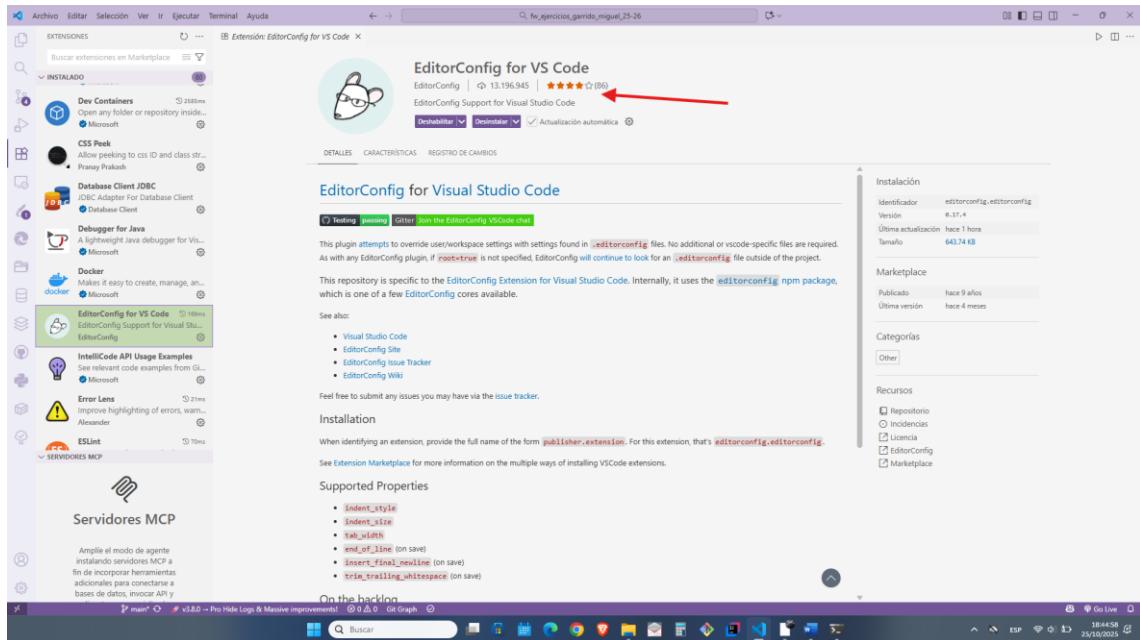
Detalles

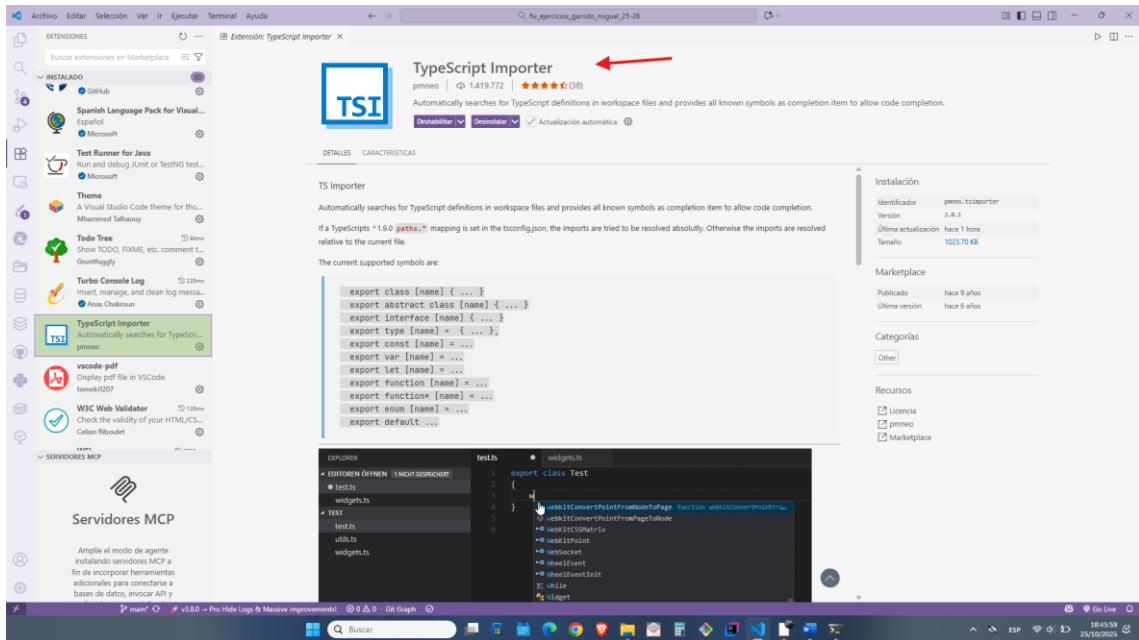
Quitar



- Agrega también estas Extensiones de VS Code



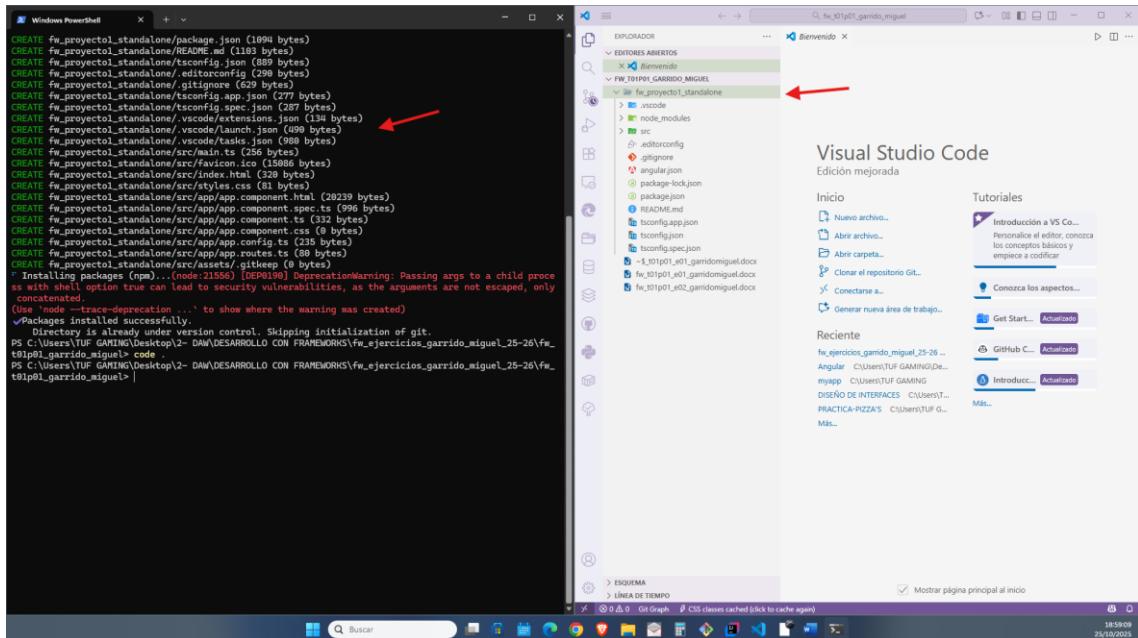




EJERCICIO 1 – C

Crea un proyecto standalone

- Crea el proyecto con nombre fw_proyecto1_standalone



- Levanta el servidor con dicho proyecto.

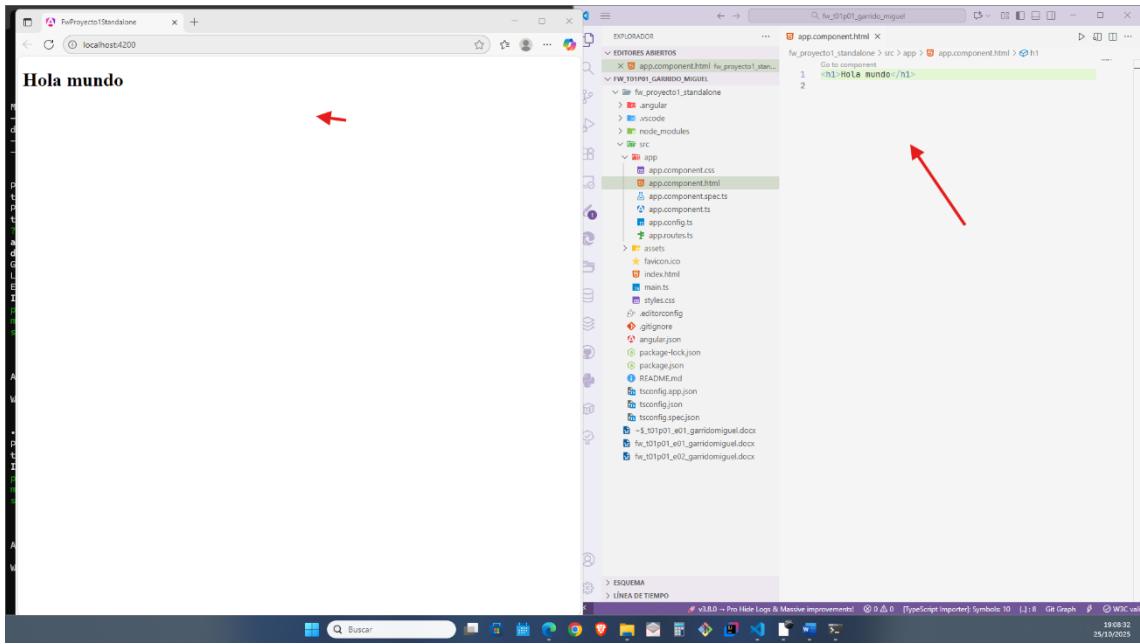
- **Analiza la estructura creada (breve descripción de para qué sirve cada fichero clave):**

• main.ts → arranque con bootstrapApplication(...).
es el punto de entrada de la aplicación es el primer archivo de TypeScript que se ejecuta. Contiene la lógica para arrancar la aplicación en el navegador y utiliza la función bootstrapApplication de Angular para iniciar el componente raíz de la aplicación.

- app.config.ts → configuración/proveedores globales.
Este archivo maneja la configuración global que solía estar en el AppModule, define los proveedores y la configuración a nivel de toda la aplicación. Esto incluye configurar el Router, el cliente HTTP, servicios de autenticación, interceptors, etc. Exporta un objeto ApplicationConfig que, a menudo, utiliza provideRouter para incluir las rutas y provideHttpClient para habilitar las peticiones HTTP.
- app.routes.ts → definición de rutas.

Este archivo define la estructura de navegación de la aplicación. Contiene un array de objetos de tipo Route donde se mapean las url de la aplicación a los componentes que deben mostrarse, es un simple array de constantes Routes.

- app.component.ts con standalone: true e imports: [...].
El componente raíz, AppComponent, es el shell de la aplicación, montado por bootstrapApplication. Su configuración clave es standalone: true (no requiere NgModule) y en su propiedad imports: [...] debe incluir las dependencias que use, siendo cruciales RouterOutlet y RouterLink para manejar la navegación y cargar las vistas de las rutas.
- Realiza una pequeña modificación en la web de inicio y verifica en el navegador que carga la app



EJERCICIO 1 – D

Crea otro proyecto con NgModules (no standalone)

- Crea el proyecto con nombre fw_proyecto2_no_standalone

```

Windows PowerShell
CREATE fw_proyecto2_no_standalone/package.json (1897 bytes)
CREATE fw_proyecto2_no_standalone/README.md (1185 bytes)
CREATE fw_proyecto2_no_standalone/tsconfig.json (298 bytes)
CREATE fw_proyecto2_no_standalone/.gitignore (298 bytes)
CREATE fw_proyecto2_no_standalone/.gitignore (629 bytes)
CREATE fw_proyecto2_no_standalone/tsconfig.app.json (277 bytes)
CREATE fw_proyecto2_no_standalone/tsconfig.spec.json (287 bytes)
CREATE fw_proyecto2_no_standalone/.vscode/tasks.json (1088 bytes)
CREATE fw_proyecto2_no_standalone/.vscode/launch.json (448 bytes)
CREATE fw_proyecto2_no_standalone/.vscode/tasks.json (988 bytes)
CREATE fw_proyecto2_no_standalone/src/main.ts (221 bytes)
CREATE fw_proyecto2_no_standalone/src/favicon.ico (15866 bytes)
CREATE fw_proyecto2_no_standalone/src/index.html (104 bytes)
CREATE fw_proyecto2_no_standalone/src/assets/icon.png (311 bytes)
CREATE fw_proyecto2_no_standalone/src/app/app-routing.module.ts (255 bytes)
CREATE fw_proyecto2_no_standalone/src/app/app.module.ts (411 bytes)
CREATE fw_proyecto2_no_standalone/src/app/app.component.html (20239 bytes)
CREATE fw_proyecto2_no_standalone/src/app/app.component.spec.ts (20239 bytes)
CREATE fw_proyecto2_no_standalone/src/app/app.component.css (237 bytes)
CREATE fw_proyecto2_no_standalone/src/app/app.component.css (0 bytes)
CREATE fw_proyecto2_no_standalone/src/assets/.gitkeep (0 bytes)
"Installing packages (npm)...(node:21836) [DEP0100] DeprecationWarning: Passing args to a child process with shell option true can lead to security vulnerabilities, as the arguments are not escaped, only
cropped[...]
[Use `node --trace-deprecation ...` to show where the warning was created]
`Packages installed successfully.
Directory is already under version control. Skipping initialization of git.
PS C:\Users\TUF GAMING\Desktop\2- DAW\DESARROLLO CON FRAMEWORKS\fw_ejercicios_garrido_miguel_25-26\fw_t0lp01_garrido_miguel> cd ..
PS C:\Users\TUF GAMING\Desktop\2- DAW\DESARROLLO CON FRAMEWORKS\fw_ejercicios_garrido_miguel_25-26\fw_t0lp01_garrido_miguel> code
PS C:\Users\TUF GAMING\Desktop\2- DAW\DESARROLLO CON FRAMEWORKS\fw_ejercicios_garrido_miguel_25-26\fw_t0lp01_garrido_miguel> ls
Directorio: C:\Users\TUF GAMING\Desktop\2- DAW\DESARROLLO CON FRAMEWORKS\fw_ejercicios_garrido_miguel_25-26\fw_t0lp01_garrido_miguel

Mode LastWriteTime Length Name
d----- 25/10/2025 19:08 fw_proyecto2_no_standalone
d----- 25/10/2025 19:12 fw_proyecto2_no_standalone
-a---- 25/10/2025 19:08 3090197 fw_t0lp01_e01_garrido_miguel.docx
-a---- 25/10/2025 18:41 91065 fw_t0lp01_e02_garrido_miguel.docx

PS C:\Users\TUF GAMING\Desktop\2- DAW\DESARROLLO CON FRAMEWORKS\fw_ejercicios_garrido_miguel_25-26\fw_t0lp01_garrido_miguel> cd .\fw_proyecto2_no_standalone
PS C:\Users\TUF GAMING\Desktop\2- DAW\DESARROLLO CON FRAMEWORKS\fw_ejercicios_garrido_miguel_25-26\fw_t0lp01_garrido_miguel> code
PS C:\Users\TUF GAMING\Desktop\2- DAW\DESARROLLO CON FRAMEWORKS\fw_ejercicios_garrido_miguel_25-26\fw_t0lp01_garrido_miguel> fw_proyecto2_no_standalone

```

- Levanta el servidor con dicho proyecto.

```

CREATE fw_proyecto2_no_standalone/src/app/app.component.css (0 bytes)
CREATE fw_proyecto2_no_standalone/src/assets/.gitkeep (0 bytes)
? Installing packages (npm)... (node:21836) [DEP0190] DeprecationWarning: Passing args to a child process with shell option true can lead to security vulnerabilities, as the arguments are not escaped, only concatenated
(Use `node --trace-deprecation ...` to show where the warning was created)
✓ Packages installed successfully.
  Directory is already under version control. Skipping initialization of git.

PS C:\Users\TUF GAMING\Desktop\2- DAW\DESARROLLO CON FRAMEWORKS\fw_ejercicios_garrido_miguel_25-26\fw_t0lp01_garrido_miguel> ls

  Directorio: C:\Users\TUF GAMING\Desktop\2- DAW\DESARROLLO CON FRAMEWORKS\fw_ejercicios_garrido_miguel_25-26\fw_t0lp01_garrido_miguel>

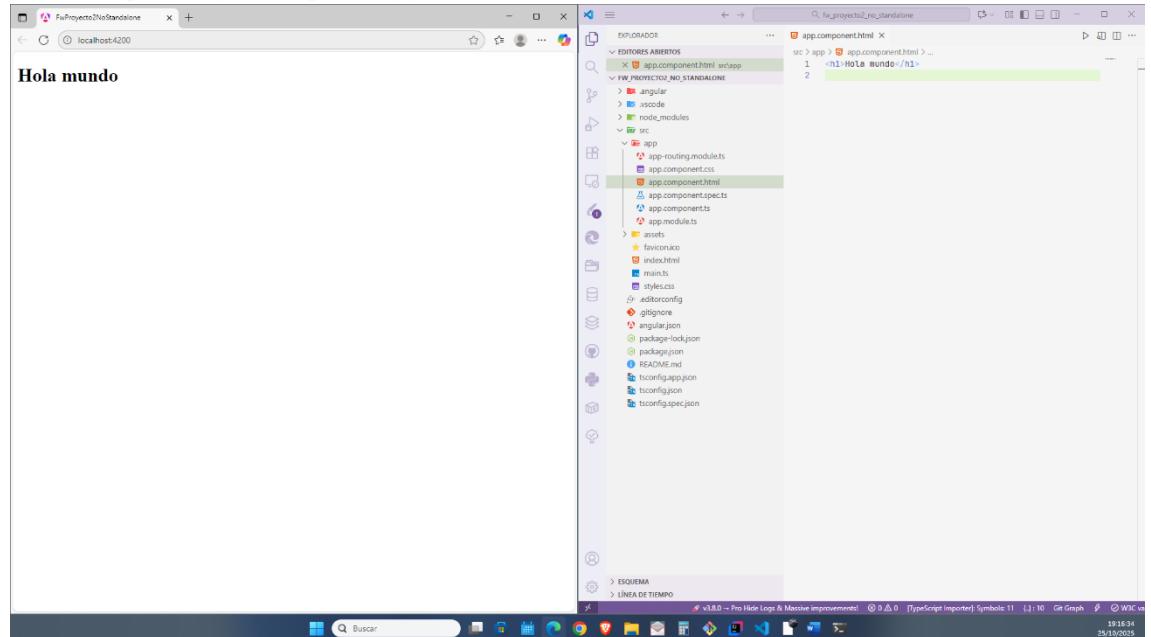
Mode LastWriteTime Length Name
d----- 25/10/2025 19:00 fw_proyecto2_no_standalone
d----- 25/10/2025 19:12 fw_proyecto2_no_standalone
-a---- 25/10/2025 19:08 3090197 fw_t0lp01_e01_garrido miguel.docx
-a---- 25/10/2025 18:41 91065 fw_t0lp01_e02_garrido miguel.docx

PS C:\Users\TUF GAMING\Desktop\2- DAW\DESARROLLO CON FRAMEWORKS\fw_ejercicios_garrido_miguel_25-26\fw_t0lp01_garrido_miguel> ng s
? 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 on how we manage this setting, see https://angular.io/analytics. No
  Global setting: enabled
  Local setting: disabled
  Effective status: disabled
Initial chunk files | Names | Raw size
polyfills.js | polyfills | 88.09 kB
main.js | main | 1.41 kB
styles.css | styles | 95 bytes
| Initial total | 111.41 kB

Application bundle generation complete. [2.921 seconds]
Match mode enabled. Watching for file changes...
→ Local: http://localhost:4200/
→ press h + enter to show help

```

- Analiza la estructura (breve):
- Presencia de app.module.ts y (si procede) app-routing.module.ts.
- main.ts con platformBrowserDynamic().bootstrapModule(AppModule).
- Realiza una pequeña modificación en la web de inicio y verifica en el navegador que carga la app.



EJERCICIO 1 – E

Comparativa técnica mediante la creación de una tabla con las diferencias clave entre:

Bootstrap: bootstrapApplication vs bootstrapModule.

Archivos generados y organización.

Dónde se declaran imports (componente vs. módulo) y dónde se declara un componente (solo en módulos).

Curva de aprendizaje y mantenimiento.

Consideraciones de migración (por qué podría interesar migrar un proyecto con módulos a standalone y en qué casos no).

Característica	Arquitectura Standalone (Moderna)	Arquitectura NgModules (Tradicional)
Arranque (Bootstrapping)	bootstrapApplication(AppComponent, appConfig) en main.ts.	platformBrowserDynamic().bootstrapModule(AppModule) en main.ts.
Punto de Entrada	El archivo main.ts carga directamente el Componente Raíz (AppComponent).	El archivo main.ts carga el Módulo Raíz (AppModule).
Configuración Global	Se define en app.config.ts usando funciones como provideRouter(), provideHttpClient(), etc.	Se define en la sección providers e imports del app.module.ts.
Archivos Clave Generados	main.ts, app.config.ts, app.routes.ts. El componente raíz tiene standalone: true.	main.ts, app.module.ts, app-routing.module.ts.
Declaración de Componente	No se requiere declaración. El componente es auto-declarable por su flag standalone: true.	Obligatorio declarar el componente en el array declarations de un NgModule.
Gestión de Dependencias (Imports)	Las dependencias se gestionan localmente en el componente, dentro del array imports: [...] del decorador @Component.	Las dependencias se importan en el array imports: [...] del @NgModule que declara el componente.
Curva de Aprendizaje	Menor y más intuitiva. Elimina la complejidad de los módulos y conceptos como declarations y exports. Se parece más a otros frameworks de frontend.	Mayor. Requiere entender la lógica de los módulos, su ámbito y la diferencia entre declarations e imports.