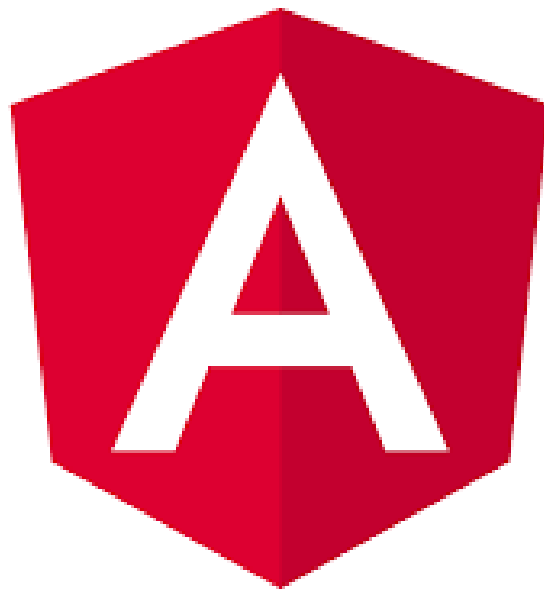


ANGULAR



TEMA 01 – ANGULAR – PARTE 01
MIKEL ARROYO GOMEZ

MIKELARROYOGOMEZ@GMAIL.COM | 2ºDAW

Contenido

Introducción a la instalación de Angular y la creación de un primer proyecto. ...	2
a. Investigación sobre Standalone vs. NgModules	2
Definición de Standalone Components vs. NgModules.	2
Investiga sobre la motivación de Angular para incorporarlos.	2
Ventajas e inconvenientes frente a NgModules.	3
¿En qué casos sigue teniendo sentido usar NgModules ?	3
Tabla comparativa:	3
b. Puesta a punto del entorno	4
Verifica que tienes Node y su gestor de paquetes npm	4
Instala Angular CLI	4
Extensiones de Google Chrome	5
Extensiones de VS Code	6
c. Crea un proyecto standalone	7
Crea el proyecto con nombre fw_proyecto1_standalone	7
Levanta el servidor con dicho proyecto.	8
Analiza la estructura creada (breve descripción de para qué sirve cada fichero clave):	8
<i>main.ts</i> →	8
Realiza una pequeña modificación en la web de inicio y verifica en el navegador que carga la app.	9
d. Crea otro proyecto con NgModules (no standalone)	9
Crea el proyecto con nombre fw_proyecto2_no_standalone	9
Levanta el servidor con dicho proyecto.	10
Analiza la estructura (breve):	11
Realiza una pequeña modificación en la web de inicio y verifica en el navegador que carga la app.	11
e. Comparativa técnica:	12

Introducción a la instalación de Angular y la creación de un primer proyecto.

a. Investigación sobre Standalone vs. NgModules

Definición de Standalone Components vs. NgModules.

Angular antes usaba los NgModules, que eran como cajas donde se metían todos los componentes de una aplicación. Cada vez que creabas uno nuevo, tenías que registrarlo dentro de un módulo.

Ahora, con las versiones nuevas (como la v17), Angular permite usar los Standalone Components, que son componentes que funcionan por sí solos sin necesidad de estar dentro de un módulo. Esto hace que todo sea más fácil y rápido de crear, sobre todo para proyectos pequeños o para empezar a aprender Angular.

En resumen:

- NgModules: sistema antiguo, más organizado pero más complicado.
- Standalone Components: sistema nuevo, más simple y recomendado por Angular hoy en día.

Investiga sobre la motivación de Angular para incorporarlos.

Cuando Angular presentó los *Standalone Components*, lo hizo porque quería que crear una aplicación fuera más fácil, rápido y moderno.

Antes, todo componente tenía que estar dentro de un módulo (un archivo llamado NgModule) y eso añadía muchos pasos extra.

Con los Standalone Components:

- Ya no necesitas ese “archivo módulo” para casi todo.
- Puedes arrancar la aplicación de forma más directa.
- Es más fácil para quien empieza, porque menos archivos y menos abstracción.
- También mejora el rendimiento en muchos casos, porque al reducir lo que carga la app, se puede optimizar mejor.

En pocas palabras: Angular decidió hacerlo para que sea menos “pesado” aprenderlo, desarrollarlo y mantenerlo, sin perder potencia.

Ventajas e inconvenientes frente a NgModules.

La principal ventaja de los *Standalone Components* es que simplifican mucho el trabajo. Ya no hace falta crear ni usar módulos para cada cosa, por lo que el proyecto tiene menos archivos y menos pasos. También son más fáciles de entender, sobre todo para quien empieza, y permiten que la aplicación arranque más rápido.

Por otro lado, el inconveniente es que algunas librerías o proyectos antiguos todavía usan módulos, así que a veces hay que mezclar los dos sistemas. En proyectos grandes, los NgModules también siguen siendo útiles porque ayudan a mantener el código más ordenado.

¿En qué casos sigue teniendo sentido usar NgModules ?

Aunque los *Standalone Components* son el sistema recomendado en Angular, los NgModules todavía tienen su utilidad.

Se siguen usando en proyectos grandes o antiguos, donde ya hay muchos módulos creados y cambiarlo todo sería complicado.

También tienen sentido cuando se usan librerías que aún necesitan módulos o cuando un equipo quiere organizar el código por secciones (por ejemplo, un módulo para usuarios, otro para productos, etc.).

En resumen, los NgModules siguen siendo prácticos cuando se busca orden y separación por partes en proyectos grandes o ya existentes.

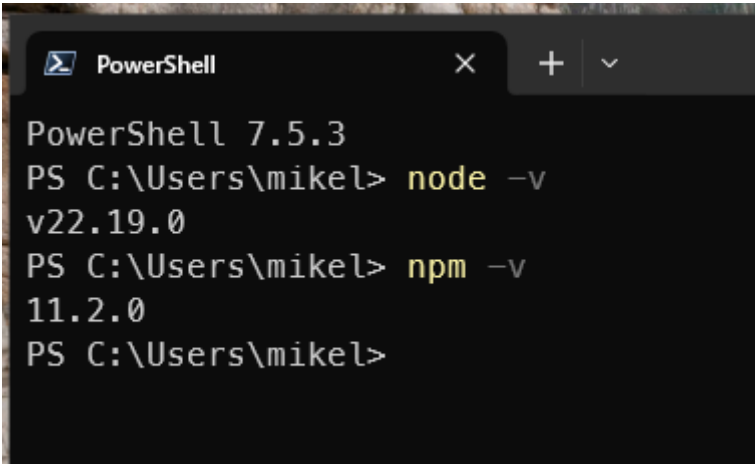
Tabla comparativa:

Aspecto	Standalone Components	NgModules
Archivos característicos	main.ts, app.config.ts, app.routes.ts (y AppComponent con standalone: true)	main.ts, app.module.ts, app-routing.module.ts (y AppComponent declarado en el módulo)
Diferencias en imports	Los imports (componentes, directivas, módulos de soporte) se hacen directamente en el componente con imports: [...]	Los imports se hacen en el módulo, en el @NgModule({ imports: [...] }). Los componentes se declaran en declarations: [...].

	o en la configuración de arranque (bootstrapApplication).	
Impacto en el bootstrap / arranque, aprendizaje y mantenimiento	Arranque más sencillo: se usa bootstrapApplication(AppComponent, { providers: [...] }). Menos archivos, curva más suave para principiantes, mantenimiento más ligero para proyectos nuevos.	Arranque tradicional: platformBrowserDynamic().bootstrapModule(AppModule). Más estructurado, puede tener más “boilerplate”, mejor para proyectos grandes o históricos.

b. Puesta a punto del entorno

Verifica que tienes Node y su gestor de paquetes npm



```

PowerShell 7.5.3
PS C:\Users\mikel> node -v
v22.19.0
PS C:\Users\mikel> npm -v
11.2.0
PS C:\Users\mikel>

```

El resultado mostró que tengo:

Node.js v22.19.0

npm v11.2.0

Esto confirma que ambos están correctamente instalados y listos para usar con Angular.

Instala Angular CLI

Utilizamos el comando: `npm install -g @angular/cli`

El `-g` significa que se instalará de forma global, para que puedas usarla desde cualquier carpeta del ordenador.

```
PowerShell
added 346 packages in 30s

65 packages are looking for funding
  run `npm fund` for details
PS C:\Users\mikel> ng version

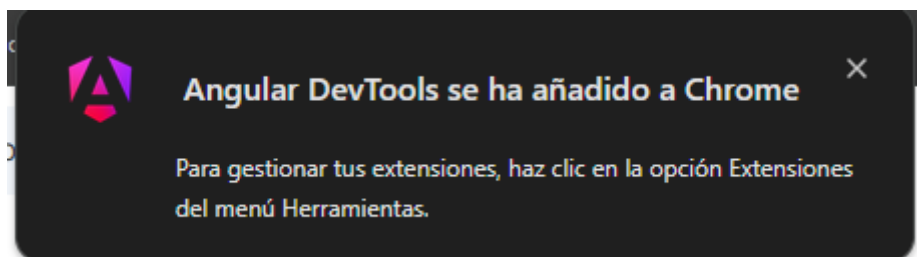
Angular CLI: 20.3.6
Node: 22.19.0
Package Manager: npm 11.2.0
OS: win32 x64

Angular: <error>

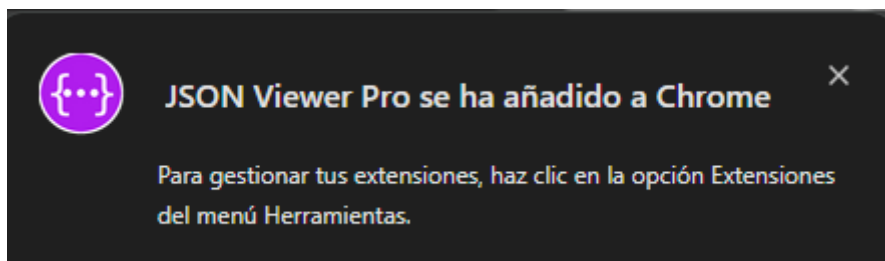
Package                                Version
```

Extensiones de Google Chrome

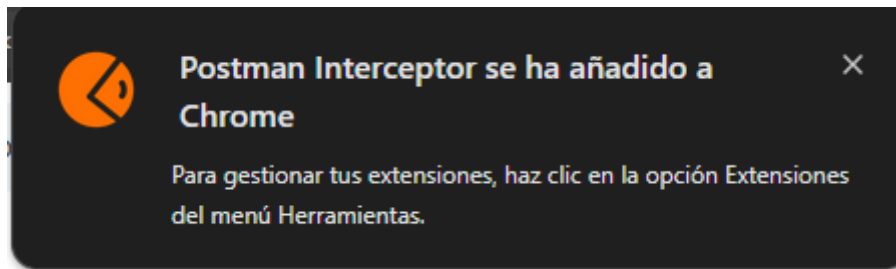
Angular DevTools



JSON Viewer Pro

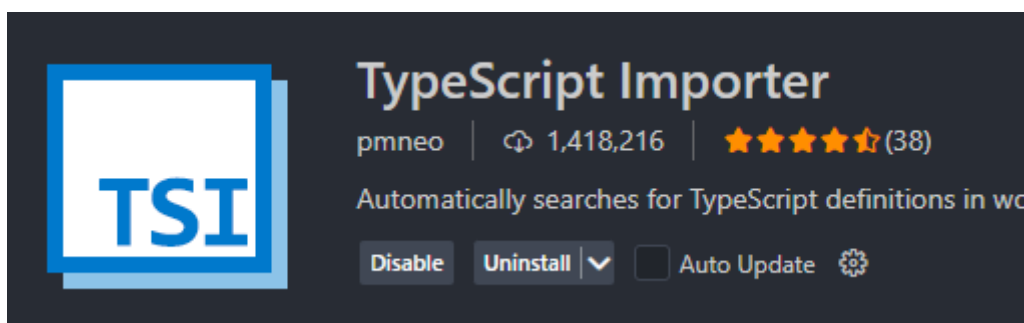
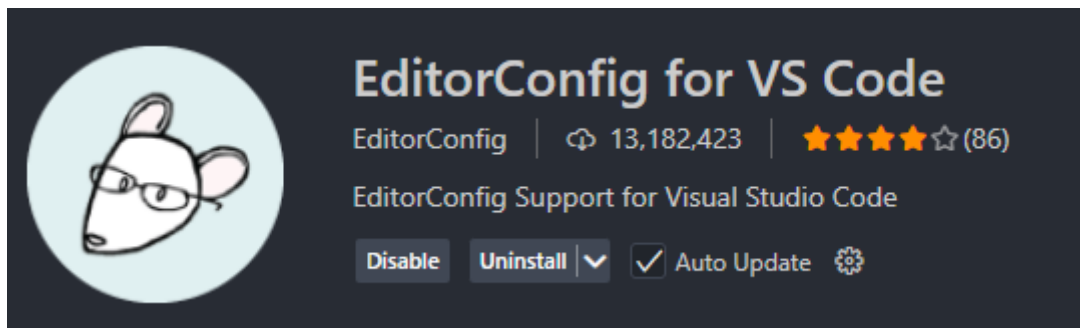
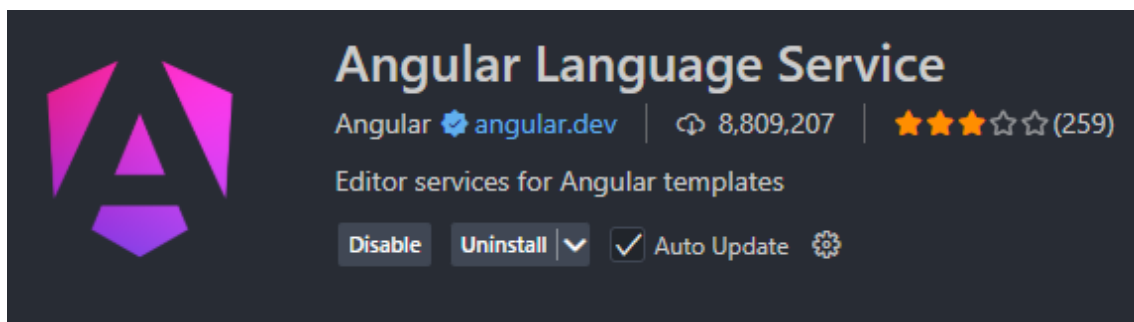


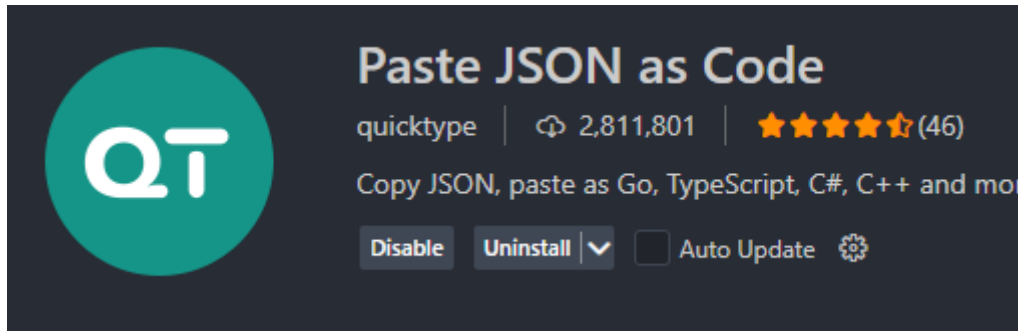
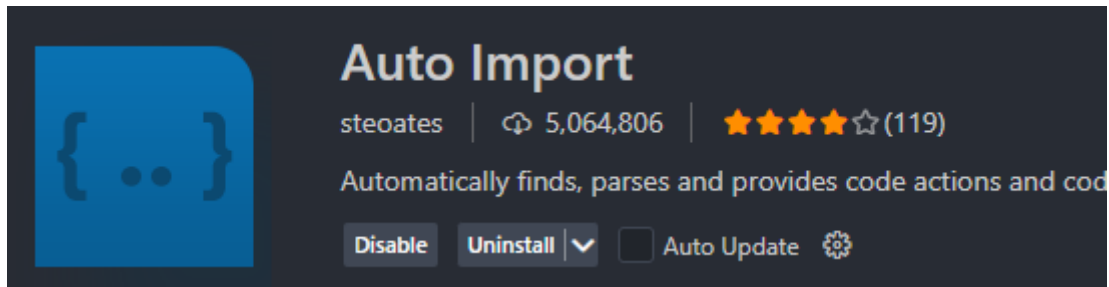
Postman Interceptor



Extensiones de VS Code

Screenshots sacadas desde mi Visual:



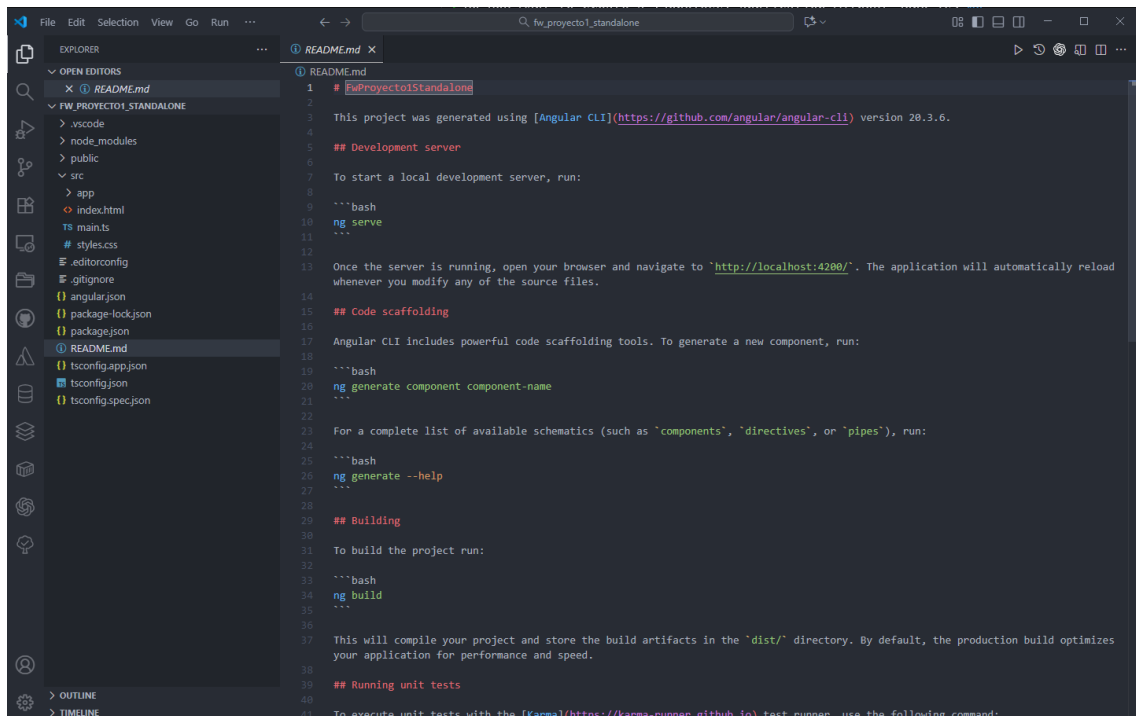


c. Crea un proyecto standalone

Crea el proyecto con nombre fw_proyecto1_standalone

Utilizamos comando ng new fw_proyecto1_standalone para iniciar la creación de proyecto y respondemos una serie de preguntas:

```
✓ 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
✓ Do you want to create a 'zoneless' application without zone.js? No
✓ Which AI tools do you want to configure with Angular best practices? https://angular.dev/ai/develop-with-ai None
CREATE fw_proyecto1_standalone/angular.json (2530 bytes)
```



Levanta el servidor con dicho proyecto.

```
PS C:\Users\mikel\fw_proyecto1_standalone> ng serve
Initial chunk files | Names          | Raw size
main.js            | main           | 48.45 kB |
polyfills.js       | polyfills      | 95 bytes |
styles.css         | styles         | 95 bytes |

| Initial total | 48.64 kB

Application bundle generation complete. [4.784 seconds] - 2025-10-22T19:27:50.376Z

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
```



Hello, fw_proyecto1_standalone

Congratulations! Your app is running. 🎉

[Explore the Docs](#)

[Learn with Tutorials](#)

[Prompt and best practices for AI](#)

[CLI Docs](#)

[Angular Language Service](#)

[Angular DevTools](#)



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

main.ts → *arranque con bootstrapApplication(...)*.

Es el archivo que pone en marcha la aplicación. Usa la función `bootstrapApplication(AppComponent)` para arrancar el componente principal (como si fuera el “botón de encendido” de la app).

app.config.ts → *configuración/proveedores globales*.

Guarda la configuración general del proyecto. Aquí se añaden los proveedores globales, como el enrutador o el manejo de peticiones HTTP.

app.routes.ts → *definición de rutas*.

Define las rutas o caminos de la web. Indica qué componente se muestra cuando el usuario entra a una dirección concreta.

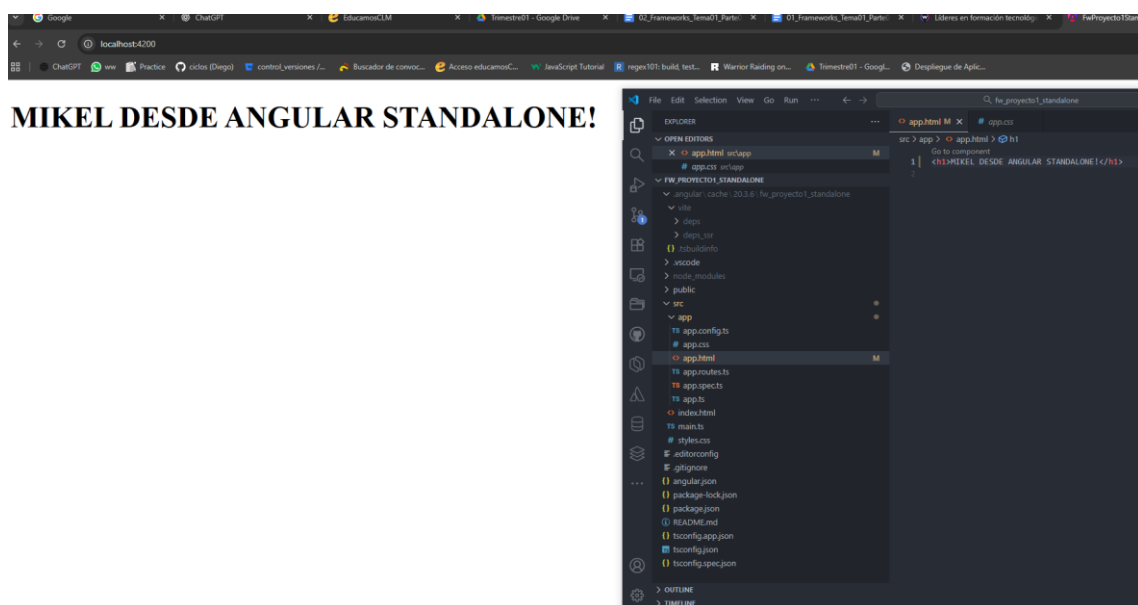
app.component.ts con standalone: true e imports: [...].

Es el componente principal de la aplicación.

Está marcado con standalone: true, lo que significa que no necesita un módulo para funcionar. Dentro de él también se indican los imports (otros módulos o componentes que usa) y se enlaza con su HTML y CSS.

Realiza una pequeña modificación en la web de inicio y verifica en el navegador que carga la app.

Modificamos el archivo app.html dentro de la carpeta app de src



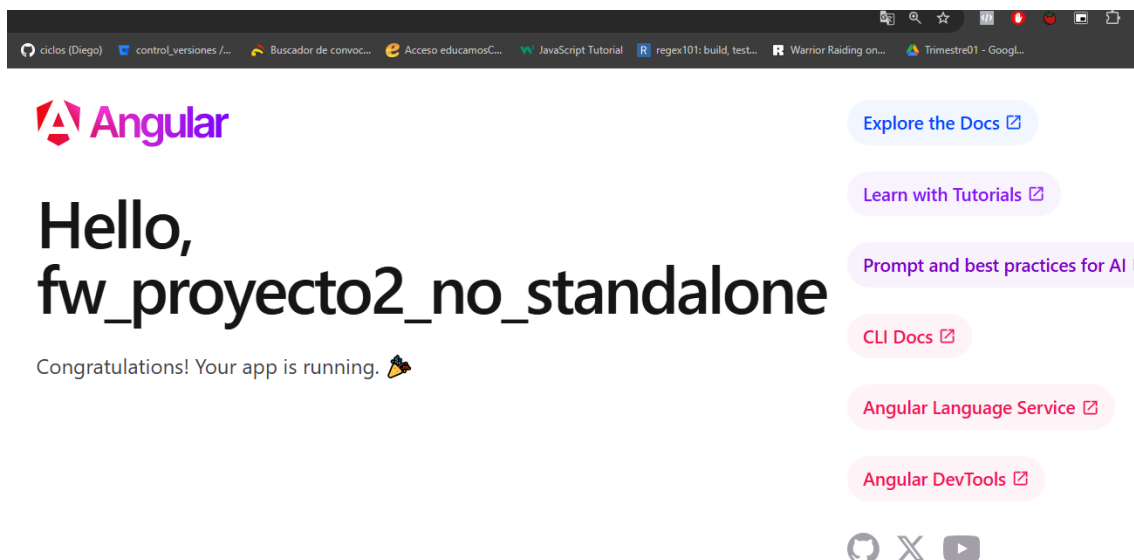
d.Crea otro proyecto con NgModules (no standalone)

Crea el proyecto con nombre fw_proyecto2_no_standalone

Para crear un proyecto no standalone utilizamos el comando “ng new fw_proyecto2_no_standalone --no-standalone” ya que desde la versión 17 por defecto el proyecto se crea en standalone y no te hace pregunta si lo quieres con con Ngmodules.

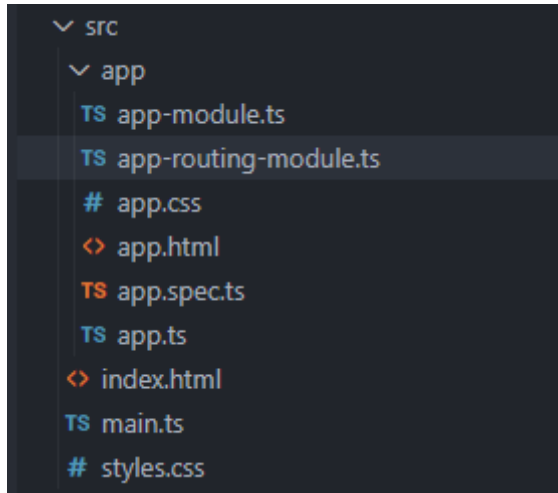
```
1 # FwProyecto2NoStandalone
2
3 This project was generated using [Angular CLI](https://
4 github.com/angular/angular-cli) version 20.3.6.
5
6 ## Development server
7
8 To start a local development server, run:
9
10 ```bash
11 ng serve
12 ```
13
14 Once the server is running, open your browser and navigate
15 to `http://localhost:4200/`. The application will
16 automatically reload whenever you modify any of the source
17 files.
18
19 ## Code scaffolding
20
21 Angular CLI includes powerful code scaffolding tools. To
22 generate a new component, run:
23
24 ```bash
25 ng generate component component-name
26 ```
27
28 For a complete list of available schematics (such as
29 `components`, `directives`, or `pipes`), run:
30
31 ```bash
32 ng generate --help
33 ```
34
35 ## Building
36
37 To build the project run:
```

Levanta el servidor con dicho proyecto.



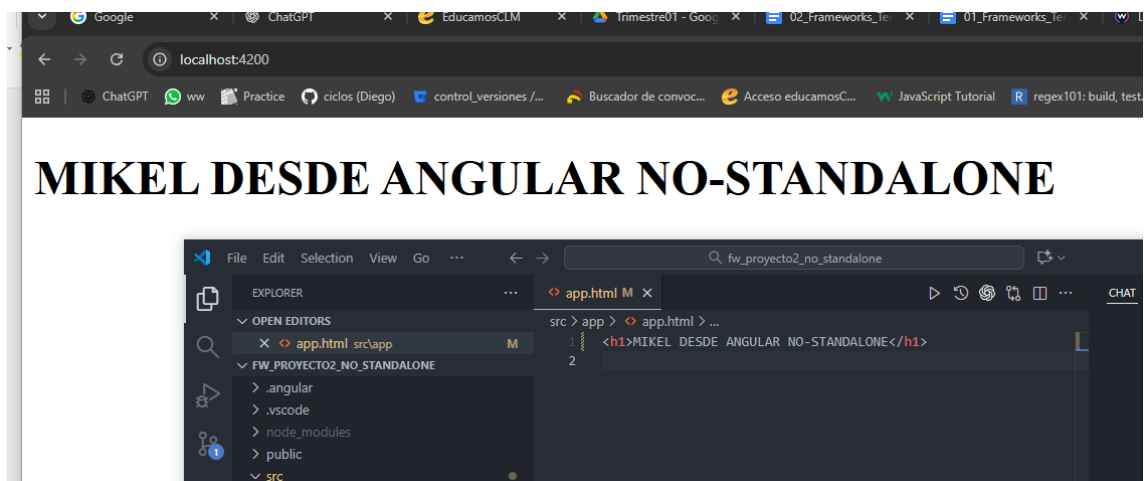
Analiza la estructura (breve):

En este proyecto aparecen los archivos `app.module.ts` y `app-routing.module.ts`, que se encargan de organizar los componentes y las rutas. El archivo `main.ts` inicia la aplicación con `platformBrowserDynamic().bootstrapModule(AppModule)`, lo que indica que el proyecto usa el sistema clásico de NgModules.



Realiza una pequeña modificación en la web de inicio y verifica en el navegador que carga la app.

Modificamos el archivo de `app.html` dentro de `app` en `src` igual que en el proyecto anterior.



e.Comparativa técnica:

Aspecto	Standalone Components	NgModules (no standalone)
Bootstrap (arranque)	Usa <code>bootstrapApplication(AppComponent)</code> directamente en <code>main.ts</code> .	Usa <code>platformBrowserDynamic().bootstrapModule(AppModule)</code> a través del módulo principal.
Archivos generados y organización	Se crean archivos como <code>app.config.ts</code> y <code>app.routes.ts</code> . No existe <code>app.module.ts</code> . La estructura es más simple.	Se generan <code>app.module.ts</code> y <code>app-routing.module.ts</code> . La estructura es más dividida y tradicional.
Dónde se declaran imports	En el propio componente (<code>imports: [...]</code>) o en la configuración del bootstrap.	En el módulo (<code>imports: [...]</code>) dentro del decorador <code>@NgModule</code> .
Dónde se declara un componente	No se declara en ningún módulo, solo se marca como <code>standalone: true</code> .	Se declara dentro del <code>@NgModule</code> en <code>declarations: [...]</code> .
Curva de aprendizaje y mantenimiento	Más fácil para empezar, menos archivos y más directo de mantener.	Requiere entender módulos, imports y declaraciones; más trabajo de organización.
Consideraciones de migración	Migrar a standalone puede simplificar proyectos y reducir código.	No siempre conviene migrar si el proyecto es grande o usa librerías que dependen de módulos.