

Verifica el entorno de programación (Solo si estás trabajando con las PC del centro de cómputo)

Abre una terminal, consola o línea de comandos.

Ejecutar en la terminal

```
### Verifica la versión de NodeJS ###
node -v

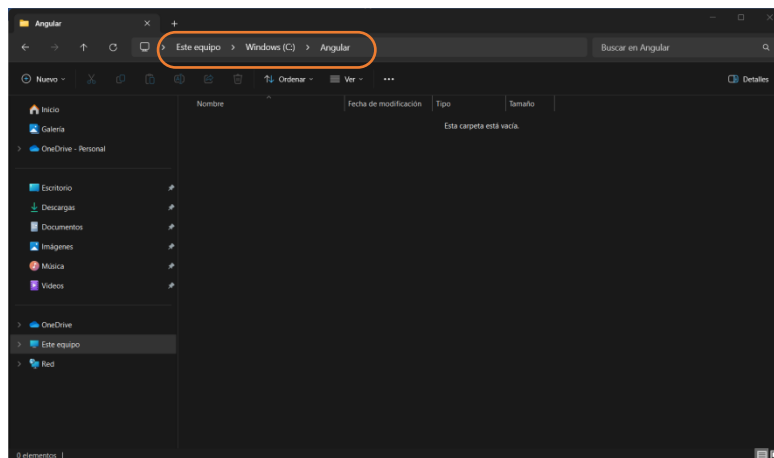
### Verifica la versión de NPM ###
npm -v

### Verifica la versión de Angular ###
ng version
```

Si todos los comandos te retornan las versiones puedes continuar con el desarrollo de la guía, de lo contrario deberás instalar el entorno nuevamente.

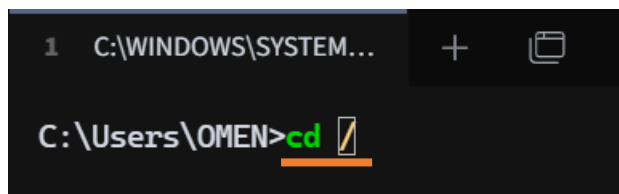
Creando nuestro repositorio de proyectos (Solo si estás trabajando con las PC del centro de cómputo)

Antes de crear el proyecto es recomendable tener una carpeta donde alojaremos nuestros proyectos de Angular, por lo cual es necesario que abras un explorador de archivos y agregues una carpeta llamada **Angular**, preferiblemente en la raíz **C:/** de tu unidad de almacenamiento.



Creando nuestra aplicación eCommerce

En la ventana de la terminal digita el comando **cd /** y presiona enter:



En la terminal, muévase a la carpeta recién creada usando el comando **cd Angular** y luego presiona enter:

```
1 C:\WINDOWS\system3... + 
C:\>cd Angular █
```

Cuando el prompt cambie estamos listos para la creación del proyecto, para ello digita en la terminal **ng new ecommerce-TUCARNET** (Por favor **no** ser literales, cambien la palabra TUCARNET por el número de carnet de la UFG):

Ejecutar en la terminal

```
ng new ecommerce-TUCARNET
### Ej. ng new ecommerce-JD100100 ###
```

Selecciona CSS como formato de hojas de estilo que se utilizaran en el proyecto

```
~/Documentos/angular
ng new ecommerce-JD100100
? 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 ]
```

No utilizaremos SSR (Server Side Rendering), así que cuando nos pregunte colocamos N y simplemente presionamos Enter.

```
~/Documentos/angular
ng new ecommerce-JD100100
✓ 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) N
```

Nos notificará cuando el proyecto este finalizado.

```
✓ Packages installed successfully.
```

Luego, abre los archivos en tu editor VSCode.

Cámbiate de directorio con el comando **cd** y el nombre de tu proyecto:

Ejecutar en la terminal

```
cd ecommerce-TUCARNET
### Ej. cd ecommerce-JD100100 ###
```

Una vez dentro del proyecto ejecuta el comando **code .** y presiona **enter**:

Ejecutar en la terminal

```
C:/Angular/ecommerce-TUCARNET> code .
```

* **nota que hay un espacio y luego un punto (.) en la instrucción.**

Instalación de TailwindCSS

Para crear una interfaz agradable para nuestra aplicación, usaremos la Tailwindcss.


<https://tailwindcss.com/docs/installation/framework-guides/angular>

Para instalar Tailwind CSS, en la terminal (con el proyecto detenido) ingresa el siguiente comando:

```
npm install tailwindcss @tailwindcss/postcss postcss --force
```

Una vez finalice el proceso te mostrará algo similar a esto:

```
~/Documentos/angular/ecommerce-JD100100 git:(master) (5.888s)
npm install tailwindcss @tailwindcss/postcss postcss --force
npm warn using --force Recommended protections disabled.
added 13 packages, removed 2 packages, changed 1 package, and audited 947 packages in 6s
161 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

Luego tenemos que crear un archivo de configuración postcss. En la raíz de tu proyecto y utilizando la opción “Nuevo archivo...” , agrega uno llamado **.postcssrc.json** (lleva un punto al inicio)

Abre el archivo recién creado y agrega el código siguiente, guarda los cambios y cierra el archivo:

```
1 {
2   "plugins": {
3     "@tailwindcss/postcss": {}
4   }
5 }
```

Abre tu archivo **src/app/styles.css**, borra el comentario y agrega lo siguiente:

```
1 @import "tailwindcss";
```

Guarda los cambios y cierra el archivo.

Ahora ejecutaremos la aplicación desde tu terminal:

```
ng serve -o
```

La aplicación correrá en <http://localhost:4200/> y se mostrará en tu navegador predeterminado.

Posteriormente debes detener la ejecución presionando **CTRL+C** en la terminal.

Creando nuestros componentes

Primero crearemos dos recursos de tipo **componente**, para ello:

- Componente header:

Ejecutar en la terminal

```
ng generate component components/header --skip-tests  
## lo pueden simplificar: ng g c components/header --skip-tests
```

- Componente footer

Ejecutar en la terminal

```
ng generate component components/footer --skip-tests  
## lo pueden simplificar: ng g c components/footer --skip-tests
```

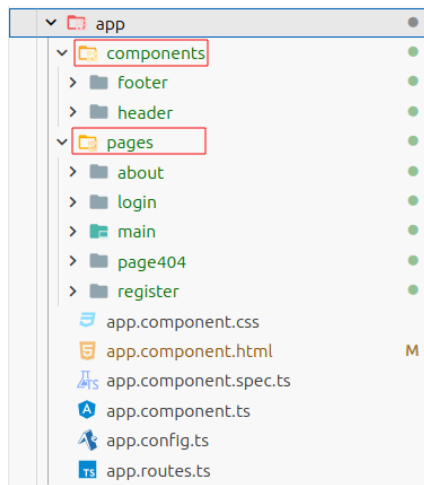
Luego crearemos recursos de tipo **página**, para ello:

Ejecutar en la terminal

```
ng generate component pages/main --skip-tests  
## lo pueden simplificar: ng g c pages/main --skip-tests
```

Repite el mismo paso y genera las páginas **about**, **login**, **register** y **page404**, siempre dentro de la carpeta pages.

De momento nuestro proyecto tendrá una estructura similar a la siguiente:



Creando el enrutamiento de componentes

Antes de modificar el archivo de rutas, vamos a preparar nuestros componentes (páginas) recién creados para exportarlos por defecto.

Abre tu archivo `src/app/pages/about.component.ts` y agrégle la palabra “default” a la exportación de la clase:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-about',
5   imports: [],
6   templateUrl: './about.component.html',
7   styleUrls: ['./about.component.css']
8 })
9 export default class AboutComponent {
10
11 }
```

Repite el mismo proceso para los archivos `src/app/pages/main.component.ts`, `src/app/pages/login.component.ts`, `src/app/pages/register.component.ts` y `src/app/pages/page404.component.ts`

Ahora abre tu archivo `src/app/app.routes.ts` y configura tus rutas como el siguiente código:

Si has seguido la guía al pie de la letra y has respetado los nombres de los componentes puedes copiar esta configuración del siguiente Gist:

<https://gist.github.com/rortiz-ufg/4a9599b6d7fb0544e0c8fae52b8fab67>

En caso contrario deberás digitarla con los nombres que le asignaste a tus componentes.

```
1 import { Routes } from '@angular/router';
2 import Page404Component from '../pages/page404/page404.component';
3
4 export const routes: Routes = [
5   {
6     path: '',
7     redirectTo: '/main',
8     pathMatch: 'full'
9   },
10  {
11    path: 'about',
12    loadChildren: () => import('../pages/about/about.component'),
13  },
14  {
15    path: 'login',
16    loadChildren: () => import('../pages/login/login.component'),
17  },
18  {
19    path: 'main',
20    loadChildren: () => import('../pages/main/main.component'),
21  },
22  {
23    path: 'register',
24    loadChildren: () => import('../pages/register/register.component'),
25  },
26  {
27    path: 'not-found',
28    component: Page404Component,
29  },
30  { // Esta debe ser la última ruta
31    path: '**',
32    redirectTo: '/not-found',
33  },
34 ];
```

Modificando el componente principal

Vamos a modificar el HTML del archivo `src/app/app.component.html`, borra **todo** el contenido de ese archivo **exceptuando** la directiva `Routeroutlet`, el código debe quedar similar al siguiente:

```
1 <!-- * * * * * HEADER COMPONENT GOES HERE * * * * * -->
2
3 <router-outlet />
4
5 <!-- * * * * * FOOTER COMPONENT GOES HERE * * * * * -->
```

Guarda los cambios, por el momento solo necesitamos dejar definido donde se cargarán nuestros componentes basados en las rutas.

Editando nuestro componente Header

Abre tu archivo `src/app/components/header.component.html`

Ve al sitio de [tailwindcomponents](https://www.tailwindcomponents.com/component/navbar-famms) y busca un navbar de tu elección, que sea parecido al siguiente para mantener algo estandarizado:



Home Pages Products Contact  

Cuando encuentres el navbar adecuado copia el código con el botón de la parte superior derecha



y pégalo en tu archivo sustituyendo cualquier código que esté generado por defecto.

Si quieres utilizar el mismo de la imagen por referencia utiliza el siguiente enlace:

<https://www.creative-tim.com/twcomponents/component/navbar-famms>

Agregando RouterLink al componente principal

Agregaremos el `Routerlink` para poder navegar entre nuestros componentes por medio de sus rutas.

Abre el archivo `src/app/header.component.ts` y agrega el `import` de `RouterLink`

```
1 import { Component } from '@angular/core';
2 import { RouterLink } from '@angular/router';
3
4 @Component({
5   selector: 'app-header',
6   imports: [RouterLink],
7   templateUrl: './header.component.html',
8   styleUrls: ['./header.component.css']
9 })
10 export class HeaderComponent {
11
12 }
```

Ahora ya podemos agregar la navegación en la plantilla, comencemos modificando header.component.html.

Busca (dependiendo de la plantilla que utilizaste) si hay elementos anchor tag `<a>` o elementos tipo list-item `` y modifícalos.

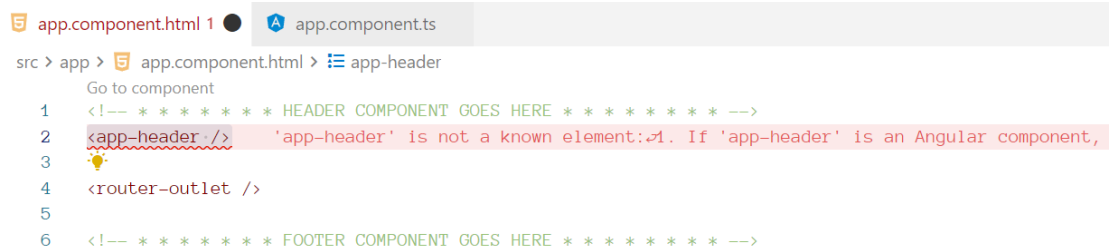
```
1 <ul class="lg:flex hidden items-center space-x-10">
2 <li class="text-lg font-semibold hover:text-red-500 transition duration-200 cursor-pointer">Home</li> <!-- Cambiar por Inicio -->
3 <li class="text-lg font-semibold hover:text-red-500 transition duration-200 cursor-pointer">Pages</li> <!-- Cambiar por Conócenos -->
```

Dentro de las etiquetas agrega la propiedad `routerLink` y asígnale la ruta a la que debe navegar:

```
1 <li class="text-lg font-semibold hover:text-red-500 transition duration-200 cursor-pointer" routerLink="/main">Inicio</li>
2 <li class="text-lg font-semibold hover:text-red-500 transition duration-200 cursor-pointer" routerLink="/about">Conócenos</li>
3 <li class="text-lg font-semibold hover:text-red-500 transition duration-200 cursor-pointer" routerLink="/login">Ingresar</li>
4 <li class="text-lg font-semibold hover:text-red-500 transition duration-200 cursor-pointer" routerLink="/register">Registrarse</li>
```

Importando componente header

Abre tu archivo `src/app/app.component.html` y modifícalo de la siguiente forma, agregando la etiqueta de referencia a tu componente header:



```
src > app > app.component.html > app-header
Go to component
1 <!-- * * * * * HEADER COMPONENT GOES HERE * * * * * -->
2 <app-header /> 'app-header' is not a known element:1. If 'app-header' is an Angular component,
3 it must have a corresponding NgModule declaration in the module it's being used in.
4 <router-outlet />
5
6 <!-- * * * * * FOOTER COMPONENT GOES HERE * * * * * -->
```

Si tienes las extensiones de VSCode instaladas es posible que te lo marque como error, eso es normal porque hace falta agregar la referencia o import.

Abre tu archivo `src/app/app.component.ts` y en el arreglo de imports agrega el HeaderComponent. Asegúrate de importarlo en la parte superior.

```
1 import { Component } from '@angular/core';
2 import { RouterOutlet } from '@angular/router';
3 import { HeaderComponent } from '../components/header/header.component';
4
5 @Component({
6   selector: 'app-root',
7   imports: [RouterOutlet, HeaderComponent],
8   templateUrl: './app.component.html',
9   styleUrls: ['./app.component.css']
10 })
11 export class AppComponent {
12   title = 'ecommerce-JD100100';
13 }
```

Guarda los cambios y el error de tu archivo `app.component.html` debe haber desaparecido.

Editando nuestro componente Footer

Abre tu archivo `src/app/components/footer.component.html` puedes copiar y pegar el código del siguiente Gist:

<https://gist.github.com/rortiz-ufg/72ba7c00d95cc4f5152588b39949092e>

Guarda los cambios y agrega el componente footer a tu archivo `app.component` (repite el procedimiento que hiciste con el header)

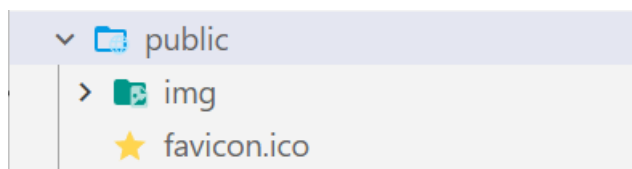
En tu archivo `src/app/app.component.html`, modifica la etiqueta `<routeroutlet>` envolviéndola en un `<div>` con las siguientes reglas de tailwind:

```
1  <!-- * * * * * HEADER COMPONENT GOES HERE * * * * * -->
2  <app-header />
3
4  <div class="min-h-svh bg-gray-100 flex items-center justify-center space-x-3">
5    <router-outlet></router-outlet>
6  </div>
7
8  <!-- * * * * * FOOTER COMPONENT GOES HERE * * * * * -->
9  <app-footer />
```

Editando nuestro componente Main

Descarga el archivo comprimido que contiene las imágenes que usaremos desde la plataforma virtual.

Descomprime la carpeta **img**, córtala y pégala en la carpeta **public** de tu proyecto



Ahora abre tu archivo `src/app/pages/main.component.ts`, vamos a crear algunas propiedades para hacer el enlace (interpolación) con el template (html)

```

1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-main',
5   imports: [],
6   templateUrl: './main.component.html',
7   styleUrls: ['./main.component.css']
8 })
9 export default class MainComponent {
10   title = 'Productos';
11
12   categories = [
13     {id: 1, name: "2x2", img: "img/2x2.avif"},
14     {id: 2, name: "3x3", img: "img/3x3.avif"},
15     {id: 3, name: "Pyraminx", img: "img/pyraminx.avif"},
16     {id: 4, name: "Megaminx", img: "img/megaminx.avif"},
17   ]
18
19   products = [
20     {id: 1, name: "Qiyi 2x2", type: "stickered", price: 4.50, img: "img/Qiyi2x2Stickered.webp"},
21     {id: 2, name: "Qiyi 3x3", type: "stickered", price: 3.50, img: "img/Qiyi3x3Stickered.webp"},
22     {id: 3, name: "Qiyi Pyraminx", type: "stickerless", price: 7.50, img: "img/QiyiPyraminxStickerless.webp"},
23     {id: 4, name: "Qiheng Megaminx", type: "stickerless", price: 11.0, img: "img/QihengMegaminxStickerless.webp"},
24     {id: 5, name: "Moyu 2x2", type: "stickerless", price: 4.65, img: "img/Moyu2x2Stickerless.webp"},
25     {id: 6, name: "Moyu 3x3", type: "stickerless", price: 4.25, img: "img/Moyu3x3Stickerless.webp"},
26     {id: 7, name: "Moyu Pyraminx", type: "stickerless", price: 7.75, img: "img/MoyuPyraminxStickerless.webp"},
27     {id: 8, name: "YJMG Megaminx", type: "stickerless", price: 11.0, img: "img/YJMGCMegaminxStickerless.webp"},
28   ]
29 }

```

Ya puedes modificar el archivo `src/app/pages/main.component.html`, comienza borrando su contenido.

Copia y pega el código del siguiente Gist, con esto lograrás tener el esqueleto de tu página main.

- Recuerda no solo copies y pegues, la idea es que intentes entender que es lo que realiza cada parte.
- Los comentarios son importantes incluso en el HTML, como puedes observar hay comentarios que limitan las secciones de la página.
- No te preocupes mucho de momento por las clases de Tailwind, como podrás notar Tailwind se ha preocupado para que con la práctica te sean bien intuitivas de utilizar. Nota que los nombres de dichas clases hacen referencia a la acción que ejecutaran al momento de pintarse en el navegador.

<https://gist.github.com/rortiz-ufg/5064a101198300375ecd881ac56c7ae4>

Verifica el funcionamiento del proyecto en el navegador.

Editando otros componentes

Ahora busca bloques de código (templates o snippets) en el sitio de [tailwindcomponents](https://tailwindcomponents.com) para las páginas **about**, **login**, **register**, **page404** y agrégalos a los archivos html correspondientes a cada página.

Verifica el funcionamiento en el navegador y utiliza los enlaces del el header para poder navegar entre los diferentes componentes.

Para la entrega:

Comprime todos los archivos de tu proyecto, pero evita agregar la carpeta **node_modules** y la carpeta **.angular** ya que estas pesan demasiado y no será posible alojarlo en la plataforma virtual.