

## Lab – directivas de administración Angular

El presente proyecto tiene como objetivo crear una aplicación completa que tenga procesos diversos y use operaciones como:

- Routing
- Lectura de datos json
- Manejo de imágenes
- Acceso a servicios API
- Uso de base de datos MySQL o PostgreSQL
- Uso de multiples modulos
- Uso de Pipe y operaciones de transformación de datos

Los cuales se estarán realizando progresivamente en este Lab y los que vendrán

Se crea el proyecto con el nombre app-servicios-generales



Una vez creado, se debe ingresar al directorio del proyecto.

A continuación, se implementa los elementos generales del proyecto.

### Elementos Básicos del proyecto

#### a. styles.css

```
body {  
  padding-top: 3.5rem;
```

```
font-family: "Roboto", sans-serif;
}
```

## **b. index.html**

En el archivo.index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Aplicacion de Uso Angular</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">

  <!-- Global site tag (gtag.js) - Google Analytics -->
  <script                                     async=""
src="https://www.googletagmanager.com/gtag/js?id=YOUR-
ID"></script>
  <script>
    window.dataLayer = window.dataLayer || [];
    function gtag() { dataLayer.push(arguments); }
    gtag('js', new Date());

    gtag('config', 'YOUR-ID');
  </script>
  <link rel="manifest" href="manifest.webmanifest">
  <meta name="theme-color" content="#1976d2">
  <link rel="stylesheet"
```

```
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
```

```
    integrity="sha384-  
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm  
"
```

```
    crossorigin="anonymous">
```

```
</head>
```

```
<body>
```

```
    <app-root></app-root>
```

```
    <noscript>Por favor, active JavaScript para seguir utilizando  
esta aplicación.</noscript>
```

```
</body>
```

```
</html>
```

### c. environment.ts

La carpeta `src/environments/` de un proyecto contiene el archivo de configuración base, `environment.ts`, que proporciona un entorno por defecto. Puedes añadir valores predeterminados por defecto para entornos adicionales, como producción y puesta en marcha, en archivos de configuración específicos del objetivo.

Un entorno de aplicación de Angular es información de configuración JSON que indica al sistema de compilación qué archivos debe cambiar cuando utilizas `ng build` y `ng serve`. Digamos que tienes una API REST de back-end desplegada en un servidor que proporciona servicios a tu aplicación Angular.

### Añadiendo múltiples entornos:

Abra su proyecto Angular en un IDE preferido. Navega a `src>environments`, allí puedes ver dos archivos `environment.ts` y `environment.prod.ts`. Sí, este es el lugar, donde se puede añadir más entornos personalizados. Pero debes seguir la siguiente convención de nombres para los archivos que se crean.

### Utilizando environment:

```
import { environment } from "../../environments/environment";
```

Ahora tienes múltiples entornos, pero no necesitas importar (o) usar todos los archivos dentro de tu aplicación.

Como se muestra en la captura de pantalla anterior, sólo importa el archivo de entorno donde lo necesites.

### Servir localmente:

Para servir la aplicación localmente con un entorno diferente, siga el siguiente comando

```
ng serve --env=<envName> (eg: ng serve --env=prod)
```

### Construir una aplicación:

Para construir la aplicación con un entorno diferente, siga el siguiente comando

```
ng build --prod --env=<envName> (eg: ng build --prod --env=dev)
```

también puedes usar `ng build --prod --env=staging` y `qa`, `dev`, `prod`, etc. Para llevar a cabo la construcción (o) desplegar la configuración de la aplicación docker y etc...

La mayoría de los proyectos de software se realizan de forma ágil, por lo que a menudo utilizamos la entrega continua. La idea es entregar versiones en ciclos cortos mediante un proceso automatizado de lanzamiento de software. Este proceso se lleva a cabo mediante la construcción de una tubería correspondiente que normalmente comprueba el código, instala las dependencias, ejecuta las pruebas y construye un paquete de producción.

A continuación, este artefacto de compilación pasa por varias etapas en las que se puede probar. A continuación, se muestra un ejemplo de configuración de etapas:

**DEV:** entorno de desarrollo que es utilizado principalmente por los desarrolladores. Un nuevo despliegue se activa automáticamente al enviar un commit a la rama de desarrollo.

**TEST:** entorno de pruebas que se utiliza principalmente para las pruebas automatizadas y las pruebas de usuario. Un nuevo despliegue se activa automáticamente al enviar un commit a la rama master.

**STAGING:** este entorno debe ser lo más parecido posible al entorno PROD. Se utiliza para las pruebas finales de aceptación antes de que se lance manualmente un despliegue PROD del artefacto de construcción.

**PROD:** el entorno "final" que es utilizado por los clientes, el despliegue se activa manualmente

Debe notarse que en la carpeta `src/environment` hay un archivo de entorno para desarrollo y otro para producción. Se va a utilizar esta característica para permitir utilizar diferentes URLs de host de la API dependiendo de si estamos en modo desarrollo o producción:

**environment.ts:**

```
export const environment = {  
  producción: false,  
  apiHost: https://api.local.com  
}
```

**environment.prod.ts:**

```
export const environment = {  
  production: true,  
  apiHost: https://api.production-url.com  
};
```

En el caso de nuestro proyecto, el código en **environment.ts** es:

```
// This file can be replaced during build by using the  
`fileReplacements` array.
```

```
// `ng build` replaces `environment.ts` with  
`environment.prod.ts`.
```

```
// The list of file replacements can be found in `angular.json`.
```

```
export const environment = {  
  production: false,  
  application:  
  {  
    name: 'angular-starter',  
    angular: 'Angular 13.1.1',  
    bootstrap: 'Bootstrap 5.1.3',  
    fontawesome: 'Font Awesome 5.15.4',  
  },  
  urlNews: './assets/params/json/mock/trailers.json',  
  urlMovies: './assets/params/json/mock/movies.json',  
  /* urlNews: 'http://localhost:5004/trailers', */  
  // url: 'https://api.ganatan.com/tutorials',
```

```

config: {
  /* SELECT ONE OF THOSE CONFIGURATIONS */
  /* LOCAL JSON (NO CRUD) */
  api: false,
  url: './assets/params/json/crud/',
  /* LOCAL REST API CRUD WITH POSTGRESQL */
  /* api: true,
  url: 'http://localhost:5004/', */
},
};

```

### **main.ts**

El trabajo de main.ts es arrancar la aplicación. Carga todo y controla el inicio de la aplicación. main.ts no es un módulo sino un simple archivo de script, ejecutado de arriba a abajo y puede tener cualquier otro nombre de archivo. Lo que realiza es importar todos los módulos necesarios para iniciar la aplicación, verificar la configuración del entorno, configurar el entorno BUILD para la producción si es necesario (el modo de desarrollo es el predeterminado) y ejecutar la función bootstrapModule que realmente configura e inicia el código de la aplicación.

La única otra cosa que lo afecta como un archivo .ts es tsconfig.json, que maneja la transpilación a javascript. pero lo hace mediante el patrón de nombre de archivo \*.ts, no haciendo referencia a archivos individualmente.

En el caso de este proyecto, el código de este archivo es el siguiente:

```

import { enableProdMode } from '@angular/core';

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

//import { AppBrowserModule } from './app/app.browser.module';
import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

```

```
platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));

function bootstrap() {
  platformBrowserDynamic().bootstrapModule(AppBrowserModule)
    .catch(err => console.error(err));
};

if (document.readyState === 'complete') {
  bootstrap();
} else {
  document.addEventListener('DOMContentLoaded', bootstrap);
}
```

Donde se aprecian las siguientes librerías y metodos:

**a. método `platformBrowserDynamic().bootstrapModule(AppModule)`**

La primera parte de la declaración `platformBrowserDynamic()` crea una plataforma. Esta se describe como el punto de entrada para Angular en una página web. Cada página tiene exactamente una plataforma, y los servicios (como la reflexión) que son comunes a todas las aplicaciones de Angular que se ejecutan en la página están vinculados a su alcance. Cada aplicación se crea a partir del módulo utilizando el método `bootstrapModule`. Este es exactamente el método que se usa en `main.ts`. Entonces, la declaración que se muestra en los documentos primero crea una plataforma y luego la instancia de la aplicación.

Cuando se crea la aplicación, Angular verifica la propiedad de arranque del módulo utilizado para arrancar la aplicación (`AppModule`).

**b. `document.readyState()`**

La propiedad `Document.readyState` de un `document` describe el estado de carga del documento.

**Valores**

El `readyState` de un documento puede tener uno de los siguientes valores:

- **loading**
  - El document todavía esta cargando.
- **interactive**

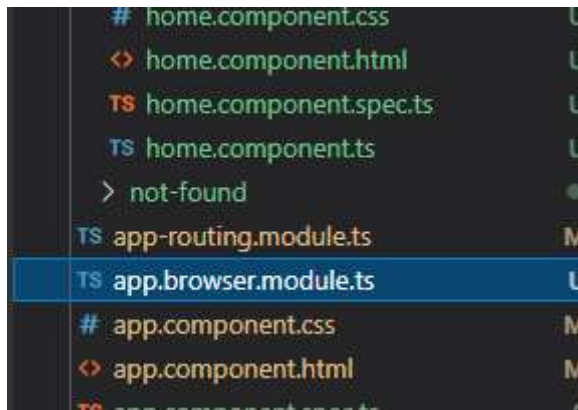
- El documento ha terminado de cargar y ha sido analizado pero los sub-recursos como imágenes, estilos y frames aún siguen cargando. El estado indica que el evento `DOMContentLoaded` (en-US) ha sido disparado.
- **complete**
  - El documento y todos los sub-recursos han cargado completamente. El estado indica que el evento `load` ha sido disparado.
  - Cuando el valor de esta propiedad cambia, un evento `readystatechange` (en-US) se dispara en el objeto `document`.

#### c. Método `document.addEventListener('DOMContentLoaded', funcion)`

El evento `DOMContentLoaded` esperará a que las etiquetas `<script>` terminen a menos que sean `<script async>` o se añadan dinámicamente, lo cual puede dispararse en cualquier momento independientemente de si el DOM está cargado o no. Este evento se dispara cuando el documento HTML inicial ha sido completamente cargado y analizado, sin esperar a que las hojas de estilo, las imágenes y los subcuadros terminen de cargarse.

#### d. `platformBrowserDynamic`

La importación de `platformBrowserDynamic` permite tener acceso al método que permite ejecutar el módulo en un navegador. La librería a la que pertenece soporta la compilación JIT y la ejecución de aplicaciones Angular en diferentes navegadores soportados



Como se observa, se invoca a un archivo denominado `app.browser.module` cuyo código es el siguiente:

#### **app.browser.module**

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { BrowserModule, BrowserTransferStateModule } from '@angular/platform-browser';
import { ServiceWorkerModule } from '@angular/service-worker';
```



```

import { AppModule } from './app.module';
import { AppComponent } from './app.component';
import { environment } from '../environments/environment';

@NgModule({
  declarations: [],
  imports: [
    CommonModule,
    BrowserModule.withServerTransition({ appId: 'angular-
starter' }),
    BrowserTransferStateModule,
    ServiceWorkerModule.register('ngsw-worker.js', {
      enabled: environment.production,
      // Registre el ServiceWorker tan pronto como la aplicación
sea estable
      // o después de 30 segundos (lo que ocurra primero).
      registrationStrategy: 'registerWhenStable:30000'
    })
  ],
  bootstrap: [AppComponent],
})
export class AppBrowserModule { }

```

### Uso de service-worker

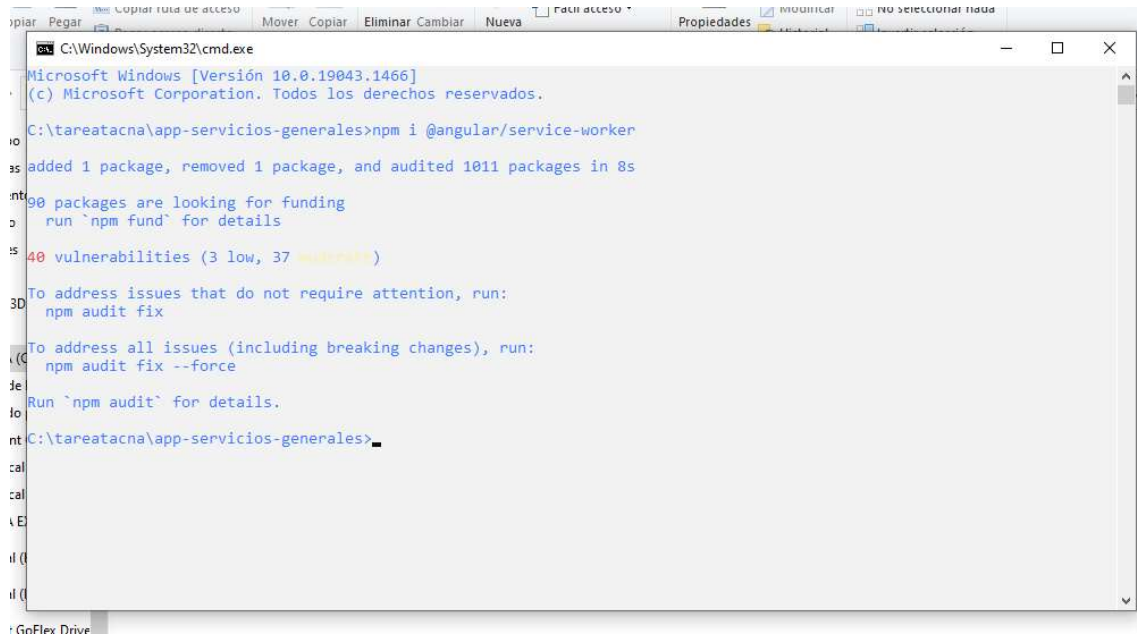
Implementa un service worker para aplicaciones Angular. Agrega un service worker a una aplicación Angular es uno de los pasos para convertirla en una aplicación web progresiva (también conocida como PWA).

Los service workers aumentan el modelo de despliegue web tradicional y permiten a las aplicaciones ofrecer una experiencia de usuario con la fiabilidad y el rendimiento a la par que el código que se escribe para ejecutarse en su sistema operativo y hardware. Añadir un service worker a una aplicación Angular es uno de los pasos para convertir una aplicación en una Progressive Web App (también conocida como PWA).

En su forma más simple, un service worker es un script que se ejecuta en el navegador web y gestiona el almacenamiento en caché de una aplicación

Los service worker funcionan como un proxy de red. Interceptan todas las solicitudes HTTP salientes realizadas por la aplicación y pueden elegir cómo responder a ellas.

`npm i @angular/service-worker`



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Versión 10.0.19043.1466]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\tareatacna\app-servicios-generales>npm i @angular/service-worker

added 1 package, removed 1 package, and audited 1011 packages in 8s

90 packages are looking for funding
  run `npm fund` for details

40 vulnerabilities (3 low, 37 moderate)

To address issues that do not require attention, run:
  npm audit fix

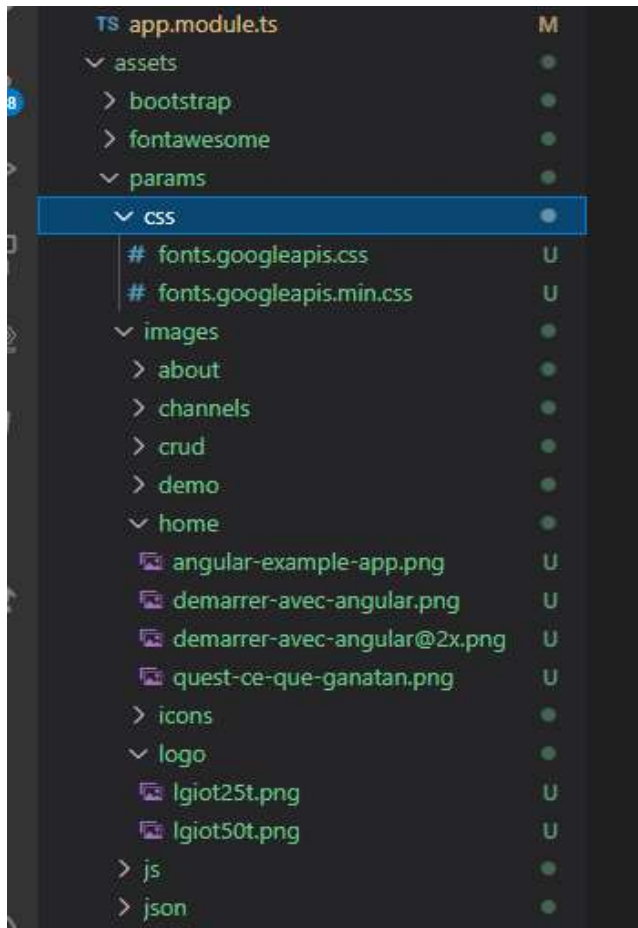
To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.

C:\tareatacna\app-servicios-generales>
```

## Carpeta assets

Para el desarrollo de este proyecto, se colocará en la carpeta assets los archivos siguientes:



Se observa la ruta y carpetas donde están los archivos css y png.

## Implementación del componente APP

A continuación, se implementa el código para el componente app de este proyecto.

### app.component.html

```
<app-header></app-header>

<main>

  <router-outlet></router-outlet>

</main>

<app-footer></app-footer>
```

### app.component.ts

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

@Component({
  selector: 'app-root',
```

```
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css']
  })
```

```
export class AppComponent {
  title = 'Aplicacion Web 2022';
  version = 'Angular version 13.1.1';
}
```

#### **app.component.css**

```
.navbar.navbar-dark .navbar-nav .nav-item .nav-link {
  color: white;
  font-weight: 500;
  border-top: 1px solid #09238d;
  border-bottom: 1px solid #09238d;
}

.navbar.navbar-dark .navbar-nav .nav-item .nav-link:hover {
  color: yellow;
  border-top: 1px solid yellow;
  border-bottom: 1px solid yellow;
}

.nga-navbar {
  -webkit-box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.16), 0 11px 10px 0
  rgba(0, 0, 0, 0.12);
  box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.16), 0 11px 10px 0
  rgba(0, 0, 0, 0.12);
  background-color: #09238d;
}

.nga-navbar-logo {
  font-weight: 700;
}

.nga-navbar-logo:hover {
  color: rgba(255, 255, 255, 0.75);
}
```

```
}  
.nga-btn-navbar {  
  color: #fff;  
  background-color: #1976d2;  
  border-color: #0d6efd;  
}  
.nga-btn-navbar:hover {  
  color: white;  
  background-color: #0b5ed7;  
  border-color: #0a58ca;  
}  
.nga-logo {  
  font-weight: 700;  
}  
.nga-logo:hover {  
  color: rgba(255, 255, 255, 0.75);  
}  
  
.nga-footer {  
  background-color: #212121;  
  color: white;  
}  
.nga-footer a {  
  color: white;  
  text-decoration: none  
}  
.nga-footer a:hover,  
.nga-footer a:focus {  
  color: yellow;  
  text-decoration: underline;
```

```

}

.nga-footer .hint {
  background-color: #1976d2;
}

.nga-footer .hint:hover {
  opacity: 0.8;
}

```

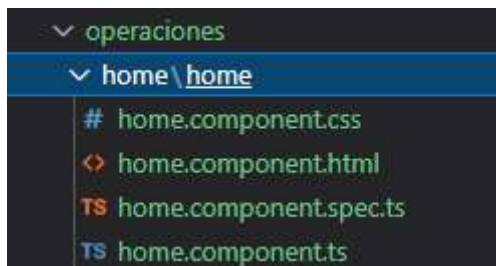
De esta forma se establecen los elementos básicos del proyecto

## Implementación de los componentes Iniciales

Se implementan los elementos iniciales de la vista del proyecto.

### a. HomeComponent

Se creara una carpeta llamada operaciones, donde se alojara este componente



#### home.component.ts

En este archivo se utilizarán los siguientes librerías y funciones pertenecientes a @angular:

### a. isPlatformBrowser

Funcion que devuelve si un platformId representa una plataforma de navegador.

*isPlatformBrowser(platformId: Object): booleano*

Parámetros : platformId Objeto

Devuelve: devuelve un valor booleano que indica si un id de plataforma representa una plataforma de navegador.

### b. PLATFORM\_ID

Un token que indica una ID de plataforma opaca.

```
const PLATFORM_ID: InjectionToken<Object>;
```

Al llamar a `platformBrowserDynamic()` (o `platformDynamicServer()` en el caso de una aplicación `@Angular/universal`) en `main.ts`, se crea un inyector que `PlatformModule` configura. `PLATFORM_ID` se proporciona aquí.

El valor es una cadena: `'navegador'` (o `'servidor'` para una aplicación universal).

### c. Meta y Title

Angular proporciona los servicios `Meta` y `Title` que le permiten obtener o establecer las etiquetas meta y el título HTML.

**Meta:** Un servicio para gestionar las etiquetas HTML `<meta>`.

```
class Meta {  
  
  • addTag(tag: MetaDefinition, forceCreation: boolean = false): HTMLMetaElement  
    | null  
  • addTags(tags: MetaDefinition[], forceCreation: boolean = false):  
    HTMLMetaElement[]  
  • getTag(attrSelector: string): HTMLMetaElement | null  
  • getTags(attrSelector: string): HTMLMetaElement[]  
  • updateTag(tag: MetaDefinition, selector?: string): HTMLMetaElement | null  
  • removeTag(attrSelector: string): void  
  • removeTagElement(meta: HTMLMetaElement): void  
  
}
```

**Title:** Un servicio que se puede utilizar para obtener y establecer el título de un documento HTML actual.

```
class Title {  
  
  getTitle(): string  
  
  setTitle(newTitle: string)  
  
}
```

A continuación, el código de ***home.component.ts***:

```
import { Component, OnInit } from '@angular/core';  
import { isPlatformBrowser } from '@angular/common';  
import { Meta, Title } from '@angular/platform-browser';  
import { environment } from 'src/environments/environment';  
import { Inject, PLATFORM_ID } from '@angular/core';  
  
@Component({
```

```

    selector: 'app-home',
    templateUrl: './home.component.html',
    styleUrls: ['./home.component.css']
  })
export class HomeComponent implements OnInit {
  name = environment.application.name;
  angular = environment.application.angular;
  bootstrap = environment.application.bootstrap;
  fontawesome = environment.application.fontawesome;
  features: any;

  constructor(
    @Inject(PLATFORM_ID) private platformId: object,
    private meta: Meta,
    private titleService: Title
  ) { }

  ngOnInit() {
    if (isPlatformBrowser(this.platformId)) {
      let navMain = document.getElementById('navbarCollapse');
      if (navMain) {
        navMain.onclick = function () {
          if (navMain) {
            navMain.classList.remove("show");
          }
        }
      }
    }
  }

  this.titleService.setTitle('Aplicacion Web en Angular 12');

```



```

    this.meta.addTag({
      name: 'author',
      content: 'jorge guerra'
    });
    this.meta.updateTag(
      {
        name: 'description',
        content: 'Esta aplicacion fue desarrollada con Angular
version 13.1.1 y bootstrap 5.1.3' +
          ' Se aplica Routing, Lazy loading, Server side rendering
y Progressive Web App (PWA)'
      });
  }

```

```

loadScript(name: string): void {
  if (isPlatformBrowser(this.platformId)) {
    const s = document.createElement('script');
    s.type = 'text/javascript';
    s.src = name;
    s.async = false;
    document.getElementsByTagName('head')[0].appendChild(s);
  }
}

```

#### **home.component.html**

```

<section class="nga-gradient text-white">
  <div class="container col-xxl-8 px-4 py-5">
    <div class="row align-items-center">
      <div class="col-lg-7 p-3 p-lg-5 pt-lg-3">
        <div class="text-center">
          <h1 class="mt-2 display-4 fw-bold">Angular</h1>

```

```

<h2 class="h1">Aplicacion Web - General</h2>

<hr>

<h2 class="h5 mt-4">{{ angular }}</h2>

<h2 class="h5">{{ bootstrap }}</h2>

<h2 class="h5">{{ fontawesome }}</h2>

<div class="col-lg-12 mx-auto mt-4">

  <div class="d-grid gap-2 d-sm-flex justify-content-sm-center">

    <a class="nga-btn-home" routerLink="/angular">Comenzando</a>

  </div>

</div>

</div>

</div>

<div class="col-lg-4 offset-lg-1 p-0">

  <div class="position-lg-absolute top-0 left-0 overflow-hidden">

  </div>

</div>

</div>

</div>

</div>

</section>

<div class="container px-4 py-5">

  <h2 class="display-6 fw-bold pb-2 text-center">Características</h2>

  <hr>

  <div class="row g-4 py-2 row-cols-1 row-cols-md-2 row-cols-lg-4">

    <div class="col">

      <div class="nga-card-step p-4">

```

```

<a routerLink="/bootstrap">
  <i class="text-primary fab fa-bootstrap fa-2x mb-2"></i>
  <h3 class="h4 text-primary fw-bold">Bootstrap</h3>
  <p>Como usar Botones, Alertas, Paginacion, Tablas, Collapses</p>
</a>
</div>
</div>
<div class="col">
  <div class="nga-card-step p-4">
    <a routerLink="/services">
      <i class="text-primary fas fa-handshake fa-2x mb-2"></i>
      <h3 class="h4 text-primary fw-bold">Servicios</h3>
      <p>Usar servicios para ver una lista de reproducción y un reproductor de
youtube</p>
    </a>
  </div>
</div>
<div class="col">
  <div class="nga-card-step p-4">
    <a routerLink="/components">
      <i class="text-primary far fa-clone fa-2x mb-2"></i>
      <h3 class="h4 text-primary fw-bold">Componentes</h3>
      <p>Componente de canal con entrada, salida y emisor de eventos</p>
    </a>
  </div>
</div>
<div class="col">
  <div class="nga-card-step p-4">
    <a routerLink="/httpClient">
      <i class="text-primary fas fa-network-wired fa-2x mb-2"></i>

```

```
<h3 class="h4 text-primary fw-bold">HttpClient</h3>
<p>Usar una API externa con Modulo HttpClient</p>
</a>
</div>
</div>
<div class="col">
<div class="nga-card-step p-4">
<a routerLink="/forms">
<i class="text-primary far fa-file-alt fa-2x mb-2"></i>
<h3 class="h4 text-primary fw-bold">Forms</h3>
<p>Seleccione Formularios Dirigidos por Plantilla o Formularios Reactivos</p>
</a>
</div>
</div>
<div class="col">
<div class="nga-card-step p-4">
<a routerLink="/modal">
<i class="text-primary fab fa-angular fa-2x mb-2"></i>
<h3 class="h4 text-primary fw-bold">Modal</h3>
<p>Crear Modal con servicio y sin jquery</p>
</a>
</div>
</div>
<div class="col">
<div class="nga-card-step p-4">
<a routerLink="/crud">
<i class="text-primary fab fa-node-js fa-2x mb-2"></i>
<h3 class="h4 text-primary fw-bold">CRUD</h3>
<p>Crear y Usar API Rest</p>
```

```
    </a>
  </div>
</div>
</div>
</div>
```

### **home.component.css**

```
.nga-gradient {
  padding: 3rem 0;
  background: linear-gradient(225deg, #0d47a1, #42a5f5);
}
```

```
.nga-btn-home {
  font-family: inherit;
  color: inherit;
  display: inline-block;
  line-height: 32px;
  line-height: 3.2rem;
  padding: 0 16px;
  font-size: 14px;
  font-size: 1.4rem;
  font-weight: 400;
  border-radius: 3px;
  text-decoration: none;
  overflow: hidden;
  border: none;
  color: white;
}
```

```
.nga-btn-home {  
  background-color: black;  
  padding: 2px 34px 0;  
  font-size: 16px;  
  font-size: 1.6rem;  
  font-weight: 600;  
  line-height: 30px;  
  line-height: 3rem;  
  border-radius: 48px;  
  box-shadow: 0 2px 5px 0 rgba(0, 0, 0, .26);  
  box-sizing: border-box;  
  cursor: pointer  
}
```

```
.nga-btn-home:hover {  
  background-color: #09238d;  
  color: yellow;  
  text-decoration: underline;  
}
```

```
.nga-card-step {  
  box-shadow: 0 1px 1px rgba(0, 0, 0, .04), 0 3px 3px rgba(0, 0, 0, .09);  
  margin: 4px;  
  cursor: pointer;  
  text-decoration: none;  
}
```

```
.nga-card-step:hover {  
  box-shadow: 0 4px 4px rgba(0, 0, 0, .09), 0 4px 4px rgba(0, 0, 0, .13);
```

```

border-radius: 5px;
color: #0d6efd;

}

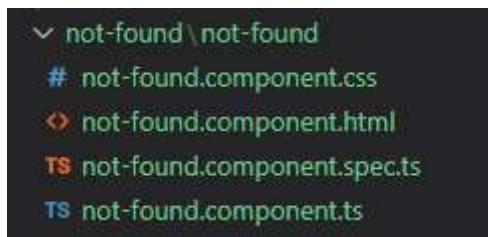
.nga-card-step a {
  color: black;
  text-decoration: none;
}

.nga-card-step a:hover {
  color: #0d6efd;
  text-decoration: none;
}

```

## b. NotFoundComponent

Este es el componente que cuando una vista no este presente, o no se ha construido, tendrá la tarea de mostrar una página de error.



### not-found.component.ts

```

import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-not-found',
  templateUrl: './not-found.component.html',
  styleUrls: ['./not-found.component.css']
})

```

```
export class NotFoundComponent implements OnInit {
  constructor() {}
  ngOnInit() {
  }
}
```

### **not-found.component.html**

```
<div class="container pt-5">
  <div class="container h-100 d-flex justify-content-center align-items-center">
    <div class="row">
      <div class="col-md-12 wow fadeIn mb-3">
        <div class="card card-body rgba-primary-slight text-center primary-text">
          <ul class="list-unstyled py-5 mx-lg-5">
            <li>
              <h1 class="display-1 mt-5 mx-5 mt-lg-0 font-weight-bold primary-text wow
fadeIn"
              data-wow-delay="0.3s">
                <strong>Error 404</strong>
              </h1>
            </li>
            <li>
              <h4 class="primary-text">Parece que esa página no existe</h4>
              <p class="description">Por favor, seleccione un enlace a continuación o en la
navegación de arriba para volver al camino correcto
              <p>
                <p class="description">
                  <a class="nga-btn-notfound" routerLink="/news">Regresando</a>
                </p>
              </li>
            </ul>
          </div>
```



```
</div>
</div>
</div>
</div>
```

#### **not-found.component.css**

```
.nga-btn-notfound {
  font-family: inherit;
  color: inherit;
  display: inline-block;
  line-height: 32px;
  line-height: 3.2rem;
  padding: 0 16px;
  font-size: 14px;
  font-size: 1.4rem;
  font-weight: 400;
  border-radius: 3px;
  text-decoration: none;
  overflow: hidden;
  border: none;
  color: white;
}
```

```
.nga-btn-notfound {
  background-color: black;
  padding: 2px 34px 0;
  font-size: 16px;
  font-size: 1.6rem;
  font-weight: 600;
```

```

line-height: 30px;

line-height: 3rem;

border-radius: 48px;

box-shadow: 0 2px 5px 0 rgba(0, 0, 0, .26);

box-sizing: border-box;

cursor: pointer
}

```

```

.nga-btn-notfound:hover {

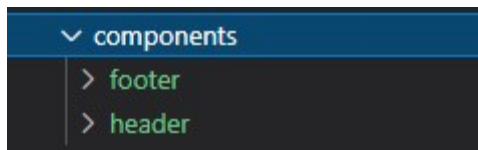
background-color: #09238d;

color: white;

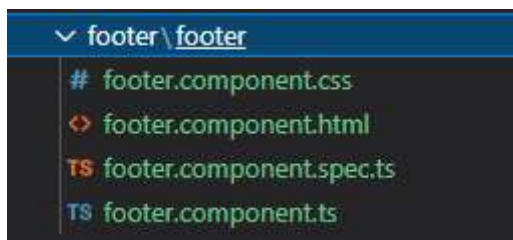
}

```

A continuación, se mostrarán la cabecera y pie de página de la aplicación Angular en desarrollo. Por ello, se creará la carpeta **components**.



#### a. FooterComponent



#### footer.component.ts

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

```

@Component({

selector: 'app-footer',

templateUrl: './footer.component.html',

```

```

    styleUrls: ['./footer.component.css']
  })
  export class FooterComponent implements OnInit {
    constructor() { }
    ngOnInit() {
    }
  }
}

```

### footer.component.html

```

<footer class="nga-footer">
  <div class="hint">
    <div class="container text-center text-md-start">
      <div class="row py-4">
        <div class="col">
          <div style="font-size:.90em" class="h6 mb-0 text-white">
            El poder de la inteligencia emana de nuestra vasta diversidad, no de un único y
            perfecto principio
          </div>
        </div>
      </div>
    </div>
  </div>
  <div class="container py-5 text-center text-lg-start">
    <div class="row">
      <div class="col-12 col-lg-5 mb-3">
        <h2 class="h5">Instructor TIC</h2>

```

---

Aplicacion Web : Angular 12, Bootstrap 5

Routing, Lazy Loading, SSR, PWA, SEO

## Herramientas

---

```

<ul class="list-unstyled">
  <li class="mb-2"><a href="https://angular.io/">Angular</a></li>
  <li class="mb-2"><a href="https://getbootstrap.com/">Bootstrap</a></li>
  <li class="mb-2"><a href="https://fontawesome.com/">Font Awesome</a></li>
</ul>
</div>

<div class="col-6 col-lg-3 mb-3">
  <h2 class="h5">Aprender</h2>
  <hr class="text-primary mt-0 d-inline-block" style="width: 70px;">
  <ul class="list-unstyled">
    <li class="mb-2"><a href="https://internetofthings.ibmcloud.com/#/">Internet de
las Cosas</a></li>
    <li class="mb-2"><a href="https://www.esmartcity.es/ciudades-
inteligentes">Smart Cities</a></li>
  </ul>
</div>
</div>
</div>

<div class="py-3 text-center" style="background-color: black;">
  <div class="container">
    2021 :<a href="https://publons.com/researcher/3425011/jorge-guerra-guerra/">
Msc Jorge Guerra</a>
  </div>
</div>
</footer>

```

### footer.component.css

```

.nga-footer {
  background-color: #212121;
  color: white;
}

```

```
}
```

```
.nga-footer a {  
  color: white;  
  text-decoration: none  
}
```

```
.nga-footer a:hover,  
.nga-footer a:focus {  
  color: yellow;  
  text-decoration: underline;  
}
```

```
.nga-footer .hint {  
  background-color: #1976d2;  
}
```

```
.nga-footer .hint:hover {  
  opacity: 0.8;  
}
```

```
.nga-btn-social {  
  position: relative;  
  z-index: 1;  
  display: inline-block;  
  padding: 0;  
  margin: 10px;  
  overflow: hidden;  
  vertical-align: middle;
```

```
cursor: pointer;

border-radius: 50%;

-webkit-box-shadow: 0 5px 11px 0 rgba(0, 0, 0, 0.18), 0 4px 15px 0 rgba(0, 0, 0, 0.15);
box-shadow: 0 5px 11px 0 rgba(0, 0, 0, 0.18), 0 4px 15px 0 rgba(0, 0, 0, 0.15);

-webkit-transition: all 0.2s ease-in-out;
transition: all 0.2s ease-in-out;

width: 47px;
height: 47px
}
```

```
.nga-btn-social i {

font-size: 1.25rem;

line-height: 47px
}
```

```
.nga-btn-social i {

display: inline-block;

width: inherit;

color: white;

text-align: center
}
```

```
.nga-btn-social:hover {

-webkit-box-shadow: 0 8px 17px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19);
box-shadow: 0 8px 17px 0 rgba(0, 0, 0, 0.2), 0 6px 20px 0 rgba(0, 0, 0, 0.19)
}
```

```
.nga-btn-social i:hover {

color: black;
}
```

```
}
```

```
.nga-btn-github {  
  background-color: #333;  
}
```

```
.nga-btn-gitlab {  
  background-color: #ff4500;  
}
```

```
.nga-btn-bitbucket {  
  background-color: #4c75a3;  
}
```

```
.nga-btn-linkedin {  
  background-color: #0082ca;  
}
```

```
.nga-btn-twitter {  
  background-color: #55acee;  
}
```

## **b. HeaderComponent**

### **head.component.ts**

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

```
@Component({  
  selector: 'app-header',  
  templateUrl: './header.component.html',  
  styleUrls: ['./header.component.css']  
})
```



```

}))

export class HeaderComponent implements OnInit {

  constructor() {}

  ngOnInit() {

  }

}

```

### header.component.html

```

<header class="navbar navbar-expand-lg navbar-dark fixed-top nga-navbar">

  <nav class="container" aria-label="Main navigation">

    <a routerLink="/" class="navbar-brand" alt="Accueil" aria-label="TIC">

      <span class="nga-navbar-logo mx-1">Instructor TIC</span>

    </a>

    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarCollapse"

  aria-controls="navbarCollapse"      aria-expanded="false"      aria-label="Toggle
navigation">

      <span class="navbar-toggler-icon"></span>

    </button>

    <div class="collapse navbar-collapse" id="navbarCollapse">

      <ul class="navbar-nav mx-auto">

        <li class="nav-item">

          <a class="nav-link" aria-current="page" routerLink="/angular">

            <i class="fab fa-angular me-1"></i>Angular</a>

```

```

</li>
<li class="nav-item">
  <a class="nav-link" aria-current="page" routerLink="/bootstrap">
    <i class="fab fa-bootstrap me-1"></i>Bootstrap</a>
  </li>
</ul>
<ul class="navbar-nav me-auto">
  <li class="nav-item">
    <a class="nav-link" aria-current="page" routerLink="/news">
      <i class="fas fa-photo-video me-1"></i>Noticias</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" aria-current="page" routerLink="/boxoffice">
        <i class="fas fa-film me-1"></i>Boxoffice</a>
      </li>
    </ul>
    <ul class="navbar-nav me-auto">
      <li class="nav-item">
        <a class="nav-link" aria-current="page" routerLink="/about">
          <i class="far fa-question-circle me-1"></i>Acerca</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" aria-current="page" routerLink="/contact">
            <i class="fas fa-envelope me-1"></i>Contacto</a>
          </li>
        </ul>
        <form class="d-flex">
          <button type="button" class="btn btn-sm nga-btn-navbar me-2"
routerLink="/signup"><i
          class="fas fa-user-plus me-2"></i>Sign up</button>

```

```

        <button type="button" class="btn btn-sm btn-outline-light me-2"
routerLink="/login"><i
        class="fas fa-sign-in-alt me-2"></i>Login</button>
    </form>
</div>
</nav>
</header>

```

### header.component.css

```

.navbar.navbar-dark .navbar-nav .nav-item .nav-link {
    color: white;
    font-weight: 500;
    border-top: 1px solid #09238d;
    border-bottom: 1px solid #09238d;
}

.navbar.navbar-dark .navbar-nav .nav-item .nav-link:hover {
    color: yellow;
    border-top: 1px solid yellow;
    border-bottom: 1px solid yellow;
}

.nga-navbar {
    -webkit-box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.16), 0 11px 10px 0 rgba(0, 0, 0, 0.12);
    box-shadow: 0 2px 5px 0 rgba(0, 0, 0, 0.16), 0 11px 10px 0 rgba(0, 0, 0, 0.12);
    background-color: #09238d;
}

.nga-navbar-logo {
    font-weight: 700;
}

```

```
}
```

```
.nga-navbar-logo:hover {  
  color: rgba(255, 255, 255, 0.75);  
}
```

```
.nga-btn-navbar {  
  color: #fff;  
  background-color: #1976d2;  
  border-color: #0d6efd;  
}
```

```
.nga-btn-navbar:hover {  
  color: white;  
  background-color: #0b5ed7;  
  border-color: #0a58ca;  
}
```

## Configuración y Navegación Final

Para la configuración final del proyecto se definirá el código para los archivos ***app.module.ts*** y ***app.routing.module.ts***

### **app.module.ts**

```
import { NgModule } from '@angular/core';  
import { BrowserModule } from '@angular/platform-browser';  
  
import { AppRoutingModuleModule } from './app-routing.module';  
import { AppComponent } from './app.component';  
import { HttpClientModule } from '@angular/common/http';
```

```
import { FooterComponent } from './components/footer/footer/footer.component';
import { HeaderComponent } from './components/header/header/header.component';
import { HomeComponent } from './operaciones/home/home/home.component';
import { NotFoundComponent } from './operaciones/not-found/not-found/not-found.component';
```

```
@NgModule({
  declarations: [
    AppComponent,
    FooterComponent,
    HeaderComponent,
    HomeComponent,
    NotFoundComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

#### **app.routing.module.ts**

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HomeComponent } from './operaciones/home/home/home.component';
import { NotFoundComponent } from './operaciones/not-found/not-found/not-found.component';
```

```
const routes: Routes = [
  { path: '', component: HomeComponent, },
  { path: '**', component: NotFoundComponent }
];
```

```
@NgModule({
  imports: [RouterModule.forRoot(routes, {
    initialNavigation: 'enabledBlocking',
    scrollPositionRestoration: 'enabled'
  })],
  exports: [RouterModule]
})

export class AppRoutingModule { }
```

Como se observa en este código, hay 2 atributos definidos en el método `forRoot`, estos atributos son:

#### **a. *InitialNavigation***

Valores permitidos en un objeto `ExtraOptions` que configuran cuando el router realiza la operación de navegación inicial.

```
type InitialNavigation = 'disabled' | 'enabled' | 'enabledBlocking' |
'enabledNonBlocking';
```

*'enabledBlocking'* - La navegación inicial comienza antes de que se cree el componente raíz. El bootstrap se bloquea hasta que la navegación inicial se completa. Este valor es necesario para que el renderizado del lado del servidor funcione.

*'enabledNonBlocking'* - (por defecto) La navegación inicial comienza después de la creación del componente raíz. El bootstrap no se bloquea al finalizar la navegación inicial.

*'disabled'* - No se realiza la navegación inicial. El receptor de ubicación se configura antes de que se cree el componente raíz. Utilizar si hay una razón para tener más control sobre cuando el router comienza su navegación inicial debido a alguna lógica de inicialización compleja.

*'enabled'* - Esta opción es 1:1 sustituible por `enabledBlocking`. Este valor está obsoleto desde la versión 11 y no debe utilizarse en las nuevas aplicaciones

## **b. *scrollPositionRestoration***

Configura si la posición de desplazamiento debe ser restaurada cuando se navega hacia atrás.

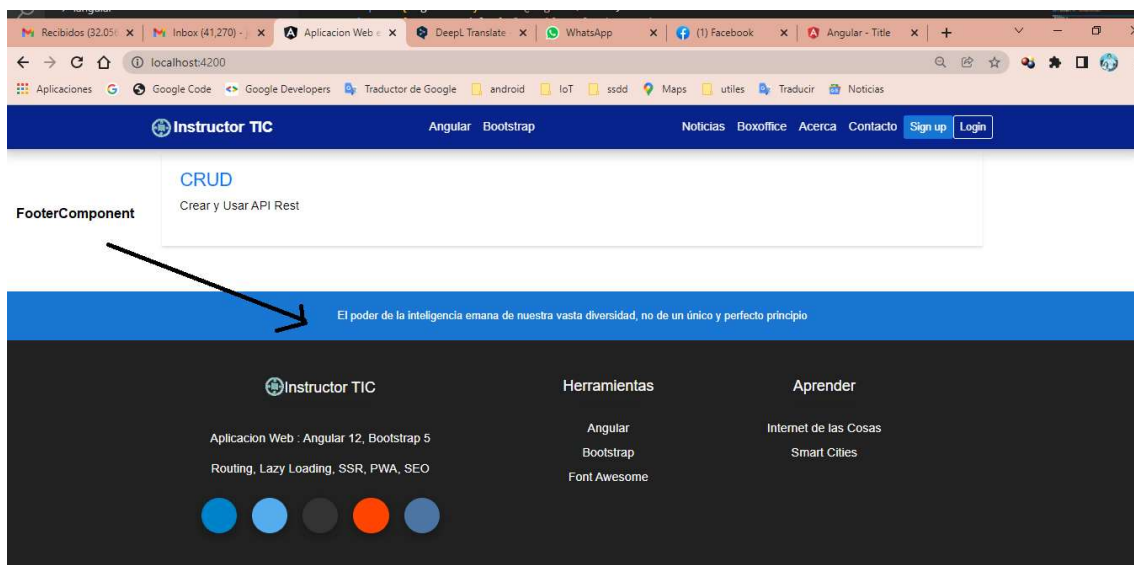
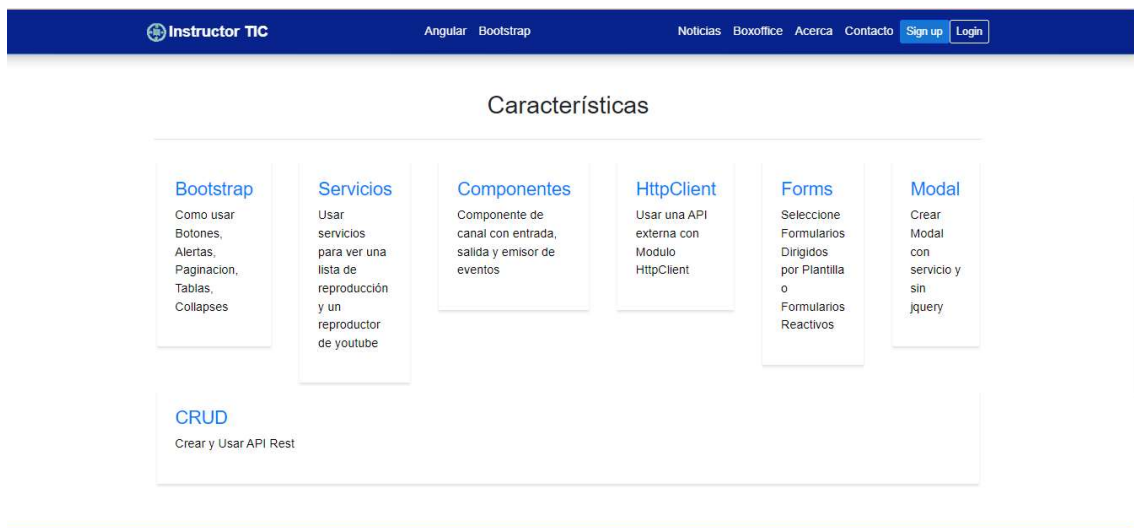
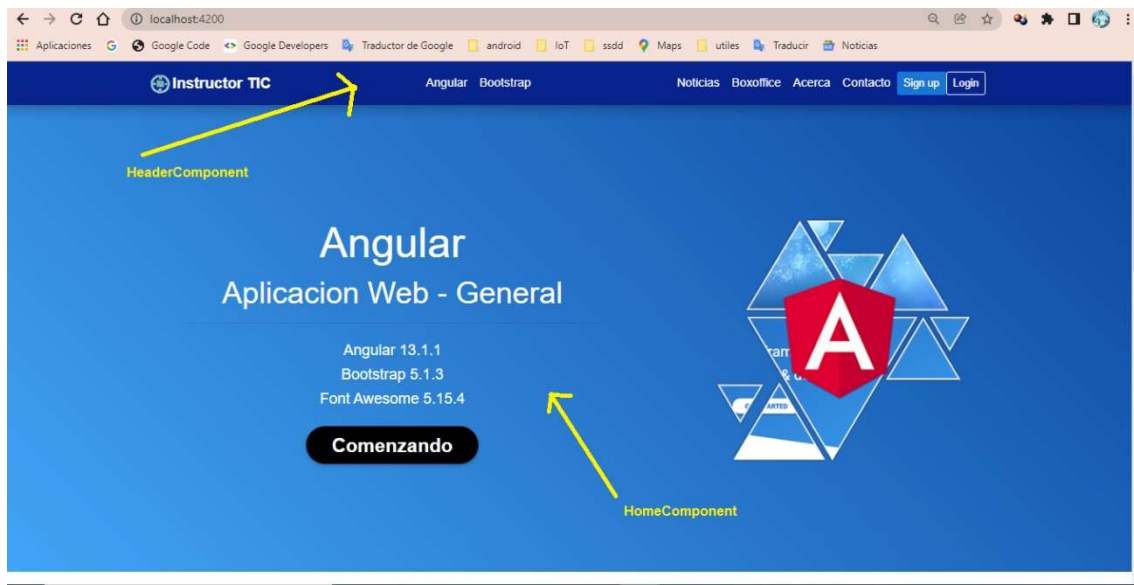
'disabled'--no hace nada (por defecto). La posición de desplazamiento se mantendrá en la navegación.

'top'-pone la posición de desplazamiento en  $x = 0$ ,  $y = 0$  en toda la navegación.

'enabled'--restaura la posición de desplazamiento anterior en la navegación hacia atrás, o establece la posición al ancla si se proporciona una, o establece la posición de desplazamiento a  $[0, 0]$  (navegación hacia adelante). Esta opción será la predeterminada en el futuro.

En general, las opciones que se pueden usar en el método `forRoot()` son las siguientes:

- *enableTracing* Alterna si el router debe registrar todos los eventos de navegación en la consola.
- *useHash* Activa la estrategia de localización que utiliza el fragmento de URL en lugar de la API del historial.
- *initialNavigation* Desactiva la navegación inicial.
- *errorHandler* Define un manejador de errores personalizado para las navegaciones fallidas.
- *preloadingStrategy* Configura una estrategia de precarga. Ver `PreloadAllModules`.
- *onSameUrlNavigation* Define lo que el router debe hacer si recibe una petición de navegación a la URL actual.
- *scrollPositionRestoration* Configura si la posición de desplazamiento debe ser restaurada cuando se navega hacia atrás.
- *anchorScrolling* Configura si el router debe desplazarse hacia el elemento cuando la url tiene un fragmento.
- *scrollOffset* Configura el desplazamiento del router cuando se desplaza hacia un elemento.
- *paramsInheritanceStrategy* Define cómo el enrutador fusiona params, datos y datos resueltos de las rutas padre a las rutas hijo.
- *malformedUriErrorHandler* Define una función personalizada de gestión de errores de uri malformada. Este manejador se invoca cuando el `encodedURI` contiene secuencias de caracteres no válidas.
- *urlUpdateStrategy* Define cuándo el enrutador actualiza la URL del navegador. El comportamiento por defecto es actualizar después de una navegación exitosa.
- *relativeLinkResolution* Habilita la correcta resolución de enlaces relativos en componentes con rutas vacías.





## Vista del componente NotFoundComponent

