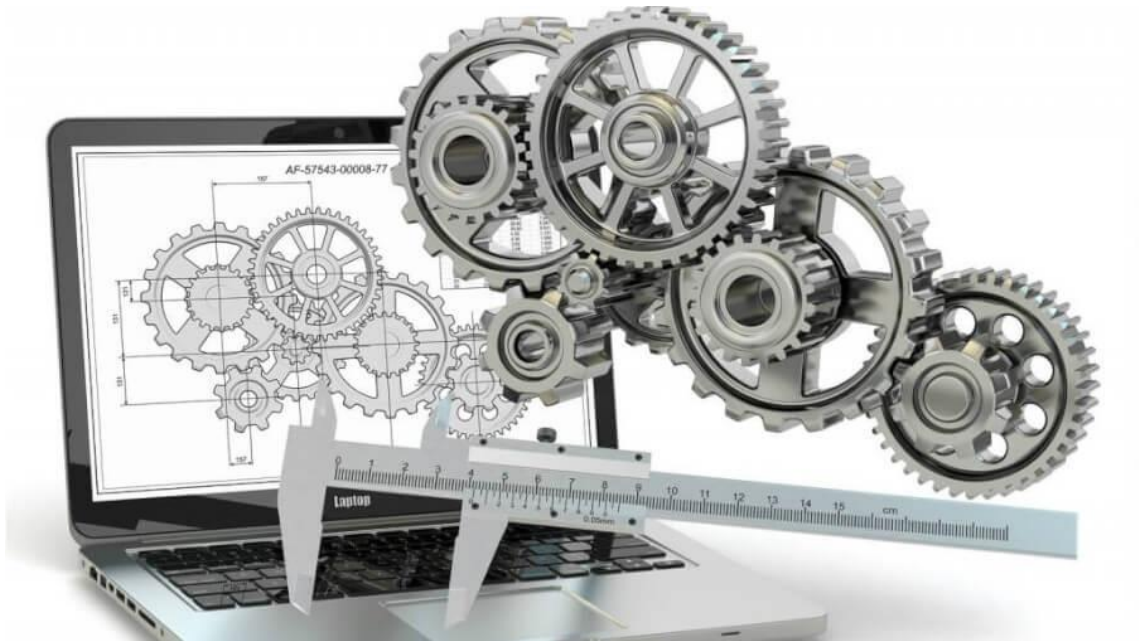




---

# INGENIERÍA INVERSA

---



©DANILO RIVERO PÉREZ

GRADO EN INGENIERÍA INFORMÁTICA: DAW2 A 02/05/2022

## ÍNDICE

1 INTRODUCCIÓN.....	2
2 REQUISITOS PREVIOS .....	2
3 ESTRUCTURA DEL PROYECTO BASE .....	3
4 DESCRIPCIÓN DE LOS COMPONENTES DE LA CARPETA SRC .....	5
5 DESCRIPCIÓN DE LOS GUARDS .....	7
6 DESCRIPCIÓN DE LOS MÓDULOS .....	7
7 DESCRIPCIÓN DEL SERVICIO DE AUTENTICACIÓN .....	12
8 FICHERO PRINCIPAL DE ENRUTAMIENTO .....	13
9 CONCLUSIONES.....	13
10 BIBLIOGRAFÍA.....	14

## 1 INTRODUCCIÓN

En esta memoria se documentará una aplicación que ha sido entregada a través del campus virtual de la ULPGC, con el objetivo de que a partir de ella se diseñe una aplicación para gestionar una plantilla de un equipo deportivo, haciendo uso de la ingeniería inversa.

La *ingeniería inversa* o *retroingeniería* es el proceso llevado a cabo con el objetivo de obtener información o un diseño a partir de un producto, con el fin de determinar cuáles son sus componentes y de qué manera interactúan entre sí y cuál fue el proceso de fabricación.

Por lo cual, a continuación se procederá a documentar la aplicación base con el fin de entenderla y estudiarla a la perfección para llevar a cabo la gestión de una plantilla de un equipo deportivo.

## 2 REQUISITOS PREVIOS

El proyecto a analizar necesita hacer uso tanto de *Bootstrap* como de *Fontawesome* para su correcto funcionamiento. Por lo tanto, antes de comenzar a trabajar con él se deben ejecutar esta secuencia de comandos para que este proyecto base se pueda ejecutar sin problemas:

```
ng add @ng-bootstrap/ng-bootstrap
```

```
ng add @fortawesome/angular-fontawesome
```

```
npm install
```

## 3 ESTRUCTURA DEL PROYECTO BASE

La estructura del proyecto base es la siguiente (Ver *Ilustración 1*):

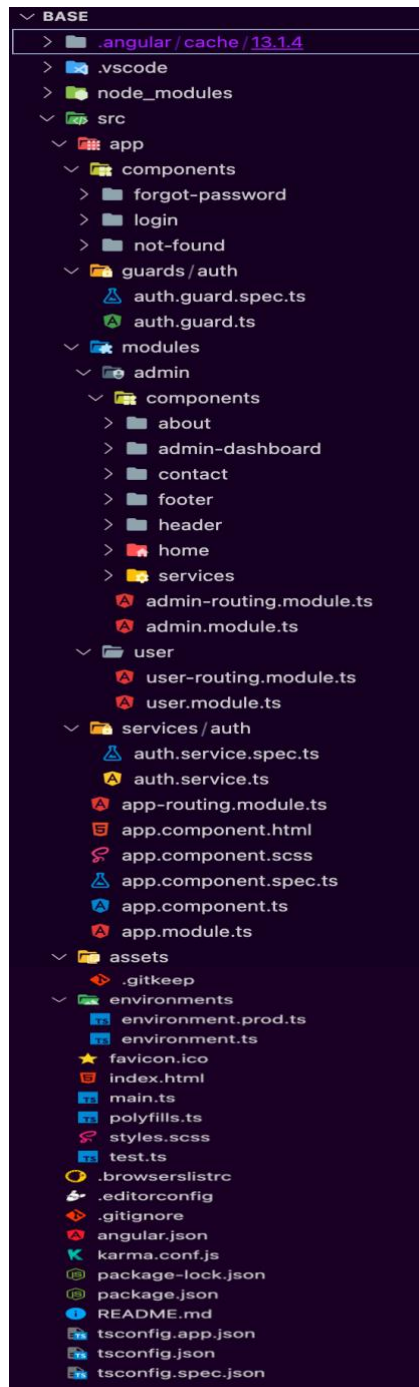


Ilustración 1: Estructura del proyecto

Para comenzar, analizaremos la estructura de carpetas básica de este proyecto, y que se divide en:

**Carpeta src:** Es la carpeta con el código fuente del proyecto. En el interior de esta carpeta tres subcarpetas muy utilizadas en Angular, como son: carpeta *App*, *environments* y *assets*.

**Carpeta App:** Carpeta en donde se ubica toda la implementación de los componentes principales, junto a su template html, archivos de estilos css y los archivos TS. En este proyecto, la carpeta App alberga en su interior el servicio de autenticación para el inicio de sesión, los módulos de la aplicación para admin y user y además hace uso de un *guard* para proteger las rutas y saber si se permite la navegación o no.

**Carpeta assets:** El directorio *assets* es el encargado de almacenar los elementos estáticos (que no cambian de la aplicación de Angular) como imágenes, pdfs, documentos de word, archivos mp3, etc.

**Carpeta environments:** Donde se encuentra las configuraciones y variables de entorno para poner el proyecto tanto en desarrollo como en producción.

Además, en el interior de la carpeta **src** se encuentran algunos archivos como:

**Archivo index.html:** Archivo de la página principal del proyecto.

**Archivo main.ts:** Es el archivo TypeScript inicial del proyecto donde podrás configurar todas las configuraciones globales del proyecto.

**Archivo favicon.ico:** Es el archivo de los iconos del proyecto.

**Archivo styles.scss:** Archivo principal para los estilos de la aplicación.

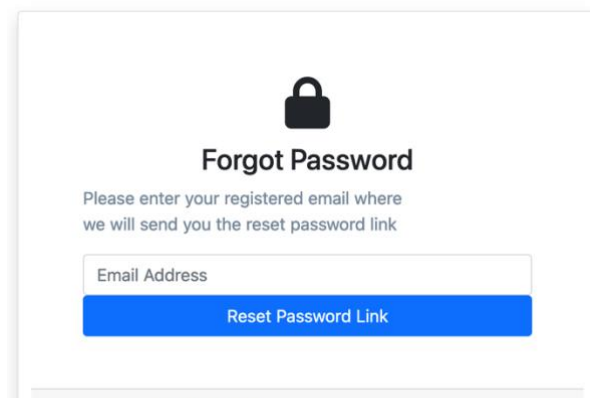
**NOTA:** En este proyecto se usa SCSS como procesador de estilos en lugar de CSS puro.

Por último, el proyecto contiene varios ficheros de configuración como son: **angular.json**, **package.json**, **tsconfig.json**, etc. Y los ficheros con las rutas.

## 4 DESCRIPCIÓN DE LOS COMPONENTES DE LA CARPETA SRC

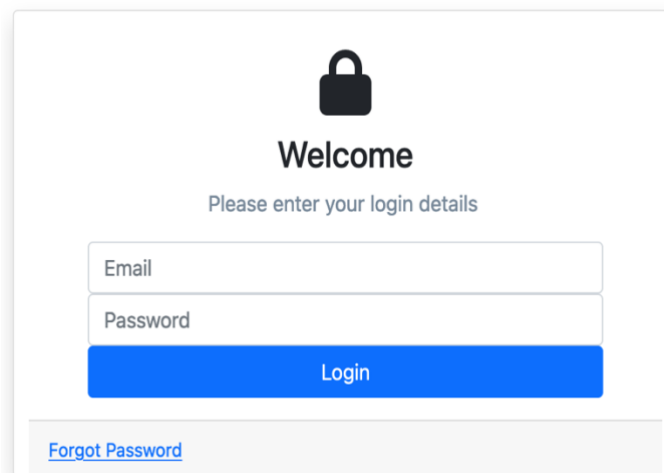
En el interior de la carpeta **src**, se cuenta con una carpeta denominada **components** que cuenta con tres componentes: **forgot-password**, **login** y **not-found**.

**Forgot-password:** Este componente contiene la lógica y el template dedicados a la página de “Has olvidado la contraseña”. Este componente entra en acción cuando el usuario hace clic en “Forgot-password” en el inicio de sesión.



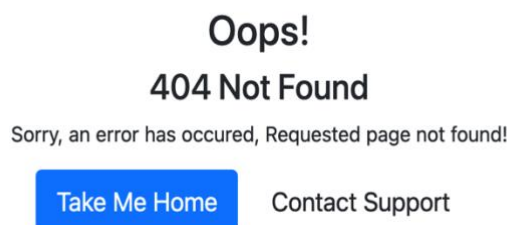
*Ilustración 2: Componente Forgot Password*

**Login**: Componente dedicado al inicio de sesión. Este componente contiene la lógica y el template dedicados al inicio de sesión en la aplicación. Este componente hace uso del servicio de autenticación para navegar hasta la página de administración al iniciar sesión.

A login form illustration. At the top is a black padlock icon. Below it is the word "Welcome" in bold. Underneath is the text "Please enter your login details". There are two input fields: "Email" and "Password". Below these is a blue "Login" button. At the bottom is a link that says "Forgot Password".

*Ilustración 3: Componente Login.*

***Not-found***: Componente que entra en juego cuando una página no existe.

A not-found page illustration. It starts with "Oops!" in large bold text, followed by "404 Not Found" in bold. Below that is the message "Sorry, an error has occurred, Requested page not found!". At the bottom are two buttons: a blue "Take Me Home" button and a "Contact Support" link.

*Ilustración 4: Componente not-found.*

## 5 DESCRIPCIÓN DE LOS GUARDS

Los guards en Angular son interfaces que permiten proteger las rutas e indican al enrutador si se permitirá la navegación a una ruta o no. En este caso se utiliza el guard **CanActivate**, para redirigir al usuario a la página de administración o a la de login, según corresponda. Además, se hace uso del servicio de autenticación para llevar a cabo la funcionalidad anterior.

```
export class AuthGuard implements CanActivate {
  constructor(private router: Router, private auth: AuthService) {}
  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ): boolean {
    if (!this.auth.isLoggedIn()) {
      this.router.navigate(['/login']);
    }
    return this.auth.isLoggedIn();
  }
}
```

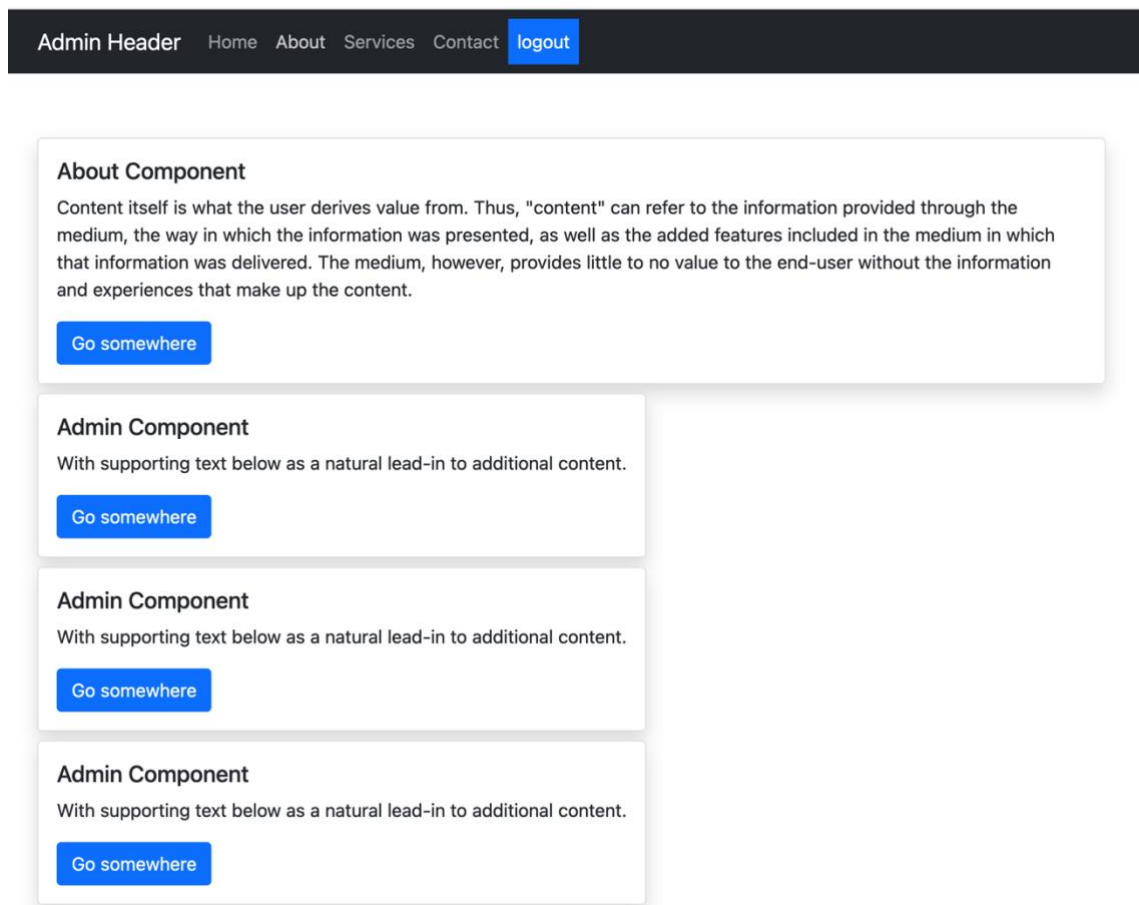
*Ilustración 5: Guard para permitir la navegación o no.*

## 6 DESCRIPCIÓN DE LOS MÓDULOS

La carpeta /src/app/modules, contiene los módulos referentes al usuario y al administrador. A su vez, la carpeta admin (correspondiente al módulo de administrador) contiene una serie de componentes tales:

**Componente about:** Componente que contiene la lógica y el template relativo a la página de About. Este componente en su template alberga diferentes tarjetas bootstrap.





*Ilustración 6: Componente About.*

**Componente Admin Dashboard:** Componente que contiene la lógica y el template relativo a la página de administración. Este componente en su template alberga el header y el footer, haciendo uso de enrutamiento (router-outlet). Este componente actúa como padre de los componentes header y footer.

**Componente Contact:** Componente que contiene la lógica y el template relativo a la página de Contacto. Este componente en su template alberga diferentes tarjetas bootstrap.

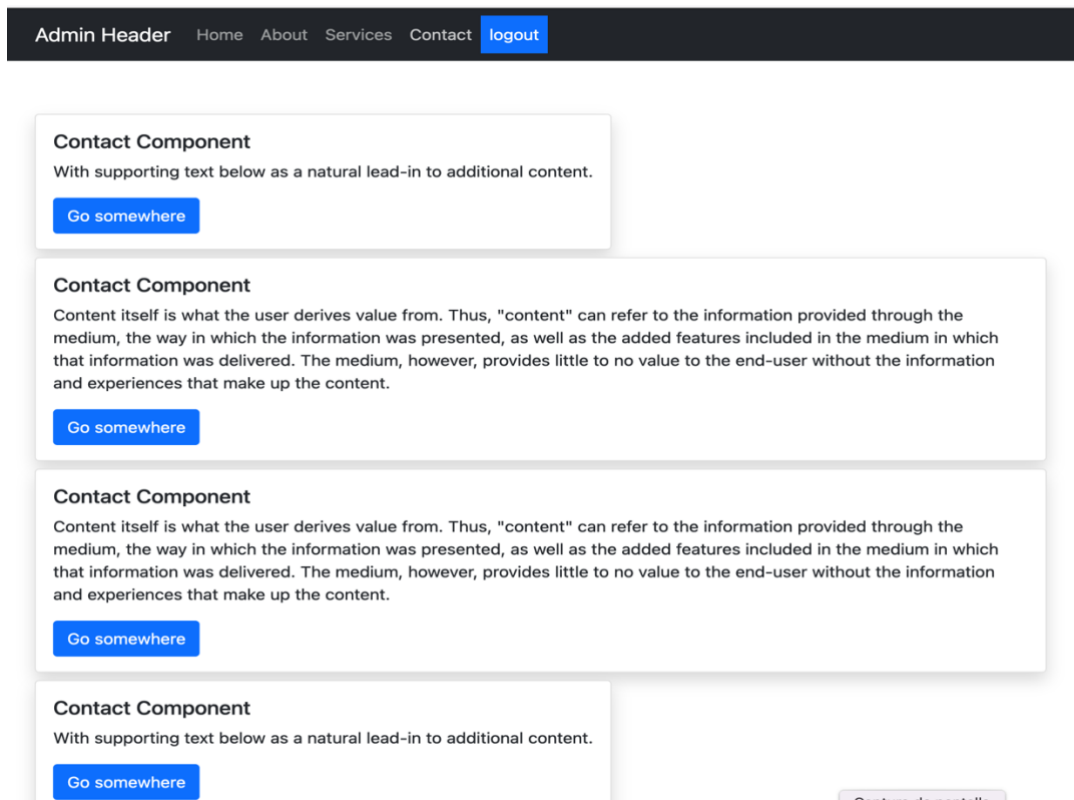


Ilustración 7: Componente contact.

**Componentes Header y footer:** Componentes que contienen la lógica y los templates relativos a la barra de navegación y al footer de las páginas de la aplicación. Este componente es declarado en su componente padre AdminDashboard. Además, hacen uso de enrutamiento y navegación para navegar a las diferentes páginas del sitio web haciendo uso de sus ítems.

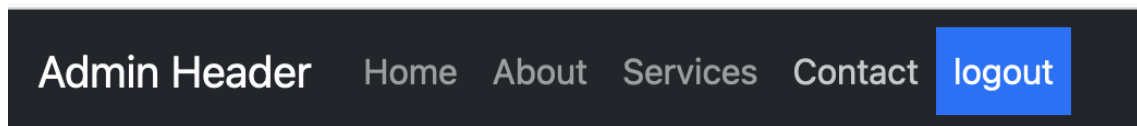
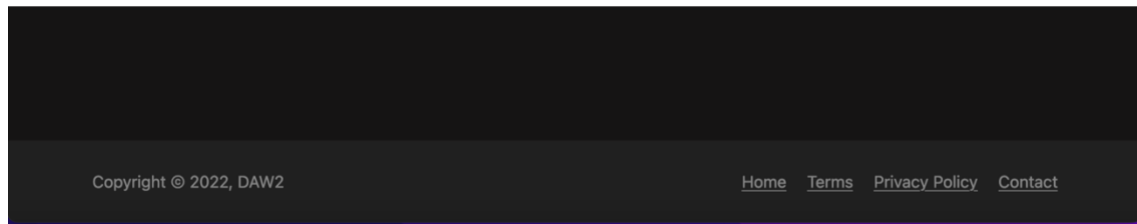
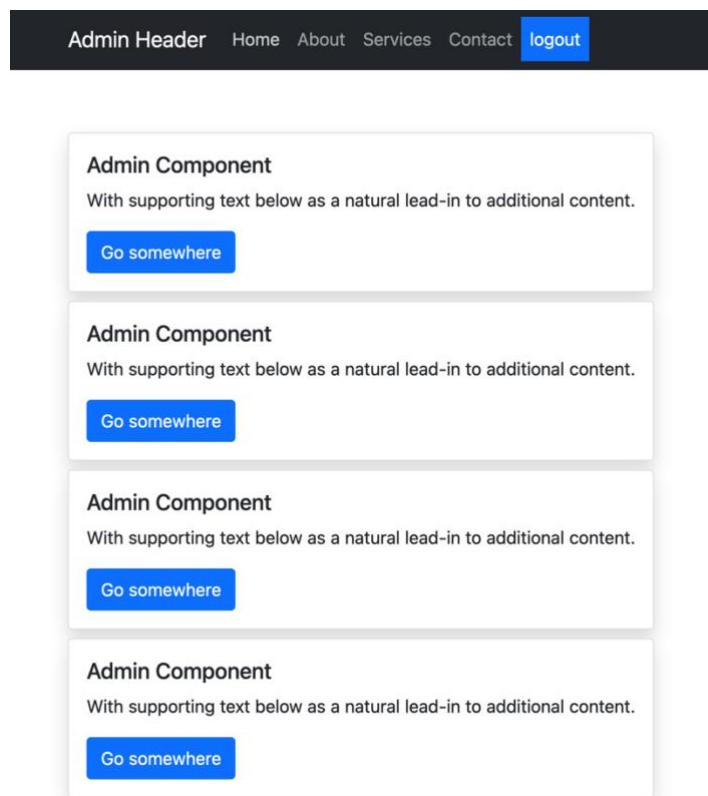


Ilustración 8: Componente Header.



*Ilustración 9: Componente footer.*

**Componentse Home y Service:** Componentes que contienen la lógica y los templates relativos a la páginas de Home y de Servicios. Estos componentes en sus templates albergan diferentes tarjetas bootstrap.



*Ilustración 10: Componentes Home y Services.*

Este módulo referente al **admin**, hace uso de enrutamiento, basándose en el componente padre AdminDashboard y varias rutas hijas que navegan a los componentes de Home, Services, Contact, etc. De esta manera y tras su importación en el archivo de enrutamiento principal, se podrá conseguir cargar los diferentes módulos de forma perezosa (lazy load).

```
const routes: Routes = [  
  {  
    path: '',  
    component: AdminDashboardComponent,  
    children: [  
      { path: 'home', component: HomeComponent },  
      { path: 'about', component: AboutComponent },  
      { path: 'services', component: ServicesComponent },  
      { path: 'contact', component: ContactComponent },  
      { path: '', redirectTo: '/admin/home', pathMatch: 'full' },  
    ],  
  },  
];
```

*Ilustración 11: Enrutamiento para el módulo admin.*

Además, en el fichero **admin.module.ts**, se encuentran declarados todos los componentes que se usan en el módulo admin.

Por último, el módulo **user**, contiene los ficheros para el enrutamiento de un usuario no registrado que no posea privilegios de administrador.

## 7 DESCRIPCIÓN DEL SERVICIO DE AUTENTICACIÓN

En este proyecto base, se hace uso de un servicio que controla el inicio de sesión. Este servicio contiene varios métodos para iniciar sesión, cerrar sesión, comprobar si un usuario está logueado, establecer y obtener un token en el almacenamiento local (localStorage), etc. Este servicio es utilizado en el componente de login y en el guard de autenticación.

```
export class AuthService {
  constructor(private router: Router) {}

  setToken(token: string): void {
    localStorage.setItem('token', token);
  }

  getToken(): string | null {
    return localStorage.getItem('token');
  }

  isLoggedIn() {
    return this.getToken() !== null;
  }

  logout() {
    localStorage.removeItem('token');
    this.router.navigate(['login']);
  }

  login({ email, password }: any): Observable<any> {
    if (email === 'admin@gmail.com' && password === 'admindaw2') {
      this.setToken('abcdefghijklmnopqrstuvwxyz');
      return of({ name: 'Marilola', email: 'admin@gmail.com' });
    }
    return throwError(new Error('Failed to login'));
  }
}
```

Ilustración 12: Servicio de autenticación.

## 8 FICHERO PRINCIPAL DE ENRUTAMIENTO

En este fichero se declaran las rutas generales de la aplicación y se importa de manera perezosa el módulo de admin, para así conseguir cargar la aplicación haciendo uso de lazy load, mejorando su rendimiento/performance.

```
const routes: Routes = [
  { path: '', redirectTo: '/login', pathMatch: 'full' },
  { path: 'login', component: LoginComponent },
  { path: 'forgot-password', component: ForgotPasswordComponent },
  {
    path: 'admin',
    canActivate: [AuthGuard],
    loadChildren: () =>
      import('./modules/admin/admin.module').then((m) => m.AdminModule),
  },
  { path: '**', component: NotFoundComponent },
];
```

Ilustración 13: Fichero app-routing.module.ts

## 9 CONCLUSIONES

Una vez analizado el proyecto base en su totalidad, me ha servido para entender cómo funciona la aplicación, que características tiene y como está organizada. De esta manera, será mucho más sencillo aplicar la ingeniería inversa para gestionar la plantilla de un equipo deportivo, la cuál es la actividad planteada.

## 10 Bibliografía

[1] “Proyecto base”, ULPGC, 2022. [En línea].

Disponible en: <https://aep22.ulpgc.es/mod/folder/view.php?id=1712293> .

[Accedido: 02/05/2022]