

Creando servicio de auth.

```
src > app > shared > services > auth.service.ts > AuthService
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { MatSnackBar } from '@angular/material/snack-bar';
4 import { Router } from '@angular/router';
5 import { BehaviorSubject, Observable, catchError, map, throwError } from 'rxjs';
6
7
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class AuthService {
12
13   private token= new BehaviorSubject<string>("");
14   private tokenData= new BehaviorSubject<any>("");
15
16   constructor(private http:HttpClient,
17     private router:Router, private snackBar:MatSnackBar) { }
18
19   get token$(): Observable<string> {
20     return this.token.asObservable();
21   }
22
23   get tokenData$(): Observable<any> {
24     return this.tokenData.asObservable();
25   }
26
27   /**
28    * Método para el manejo de errores desplegados en el SnackBar
29    * @param error Error enviado al método
30    * @returns Lanza error en caso que el mensaje no esté nulo o vacío
31    */
32   handleError(error:any):Observable<never>{
33     let message = "Ha ocurrido un error";
34
35
36     if (error) {
37       message = `${error.error.message}`;
38     }
39     console.log('Handle Error ' + message);
40
41     this.snackBar.open(message, '', {
42       duration: 5*1000,
43       panelClass: ['error-snackbar'],
44       horizontalPosition: 'right',
45       verticalPosition: 'top'
46     });
47     return throwError(message);
48   }
49 }
50
51
52
53
```

```
src > app > shared > services > auth.service.ts > AuthService
31 */
32 handleError(error:any):Observable<never>{
33   let message = "Ha ocurrido un error";
34
35
36   if (error) {
37     message = `${error.error.message}`;
38   }
39   console.log('Handle Error ' + message);
40
41   this.snackBar.open(message, '', {
42     duration: 5*1000,
43     panelClass: ['error-snackbar'],
44     horizontalPosition: 'right',
45     verticalPosition: 'top'
46   });
47   return throwError(message);
48 }
49
50
51
52 }
53
```

Api url.

```
},  
  API_URL: "https://localhost:3000"  
};
```

Interface de AuthResponse:

```
src > app > shared > models > TS auth.interface.ts > AuthResponse >  
1  export interface AuthResponse {  
2    message:string,  
3    token:string  
4  }  
5
```

Guardando sesión.


Instalado jwt

```
Alejandro Guerrero@LAPTOP-EEKAKGFK MINGW64 /d/Alejandro/Documentos/UTNG/9. Noveno/Aplicaciones web integrales/Unidad 1/App/edm-app (main)  
$ npm i @auth0/angular-jwt --force  
npm WARN using --force Recommended protections disabled.  
npm WARN ERESOLVE overriding peer dependency  
npm WARN While resolving: @angular/material@15.2.9  
npm WARN Found: @angular/cdk@16.0.4  
npm WARN node_modules/@angular/cdk  
npm WARN @angular/cdk@"^16.0.4" from the root project  
npm WARN  
npm WARN Could not resolve dependency:  
npm WARN peer @angular/cdk@"15.2.9" from @angular/material@15.2.9  
main* 01:11 % 0 0 0 {} 55 tabnine starter TypeScript @ Go Live Prettier
```

Importando librería.

```
import { BehaviorSubject, Observable, catchError, map, throwError } from 'rxjs';  
import { JwtHelperService } from '@auth0/angular-jwt';  
  
const helper = new JwtHelperService();  
  
@Injectable({  
  providedIn: 'root'  
})  
export class AuthService {
```

LocalStorage.



The screenshot shows the Visual Studio Code editor with a file named `auth.service.ts` open. The breadcrumb navigation at the top indicates the file path: `src > app > shared > services > auth.service.ts > AuthService > saveLocalStorage`. The code editor displays the following TypeScript code:

```
51 }  
52  
53 saveLocalStorage(token:string){  
54     localStorage.setItem('token', token);  
55 }  
56  
57  
58 }  
59
```

Logout



The screenshot shows a code editor with the following TypeScript code for the `logout()` function:

```
logout(){  
    localStorage.removeItem('token');  
    this.token.next(""); //Notofica a suscirptores que es vacio  
    this.tokenData.next(null); //Notifica a los suscriptores.  
    this.router.navigate(['/']);  
}
```

Checklogin

```
checkToken() {  
  let token = localStorage.getItem("token");  
  
  if(token) {  
    //Checa expiración de token traído del BackEnd  
    const isExpired = helper.isTokenExpired(token);  
  
    if(isExpired) {  
      this.logout(); //Cierra sesión  
    } else {  
      this.token.next(token); //Notifica token a suscriptores  
  
      //Decodifica token JWT Encriptado  
      const {iat, exp, ...data} = helper.decodeToken(token);  
      this.tokenData.next(data); //Notifica token a suscriptores  
    }  
  } else {  
    this.logout();  
  }  
}
```

Login

```
7  */
8  login(loginData:any):Observable<AuthResponse|void> {
9
10
11  return this.http.post<AuthResponse>(`${environment.API_URL}/`, loginData)
12    .pipe(map((data:AuthResponse) => {
13
14      if(data.token) { //No hay errores y contiene información
15        this.saveLocalStorage(data.token);
16        this.token.next(data.token); //Notifica a suscriptores
17        this.router.navigate(['/home']);
18
19        this.checkToken();
20      }
21      return data;
22    })),
23    catchError((error) => this.handleError(error)));
24 }
```

Header.

```
export class HeaderComponent implements OnInit{

  data: any = {}; //Datos del usuario

  constructor(private router:Router,
    private authService:AuthService){ }

  ngOnInit(): void {
    this.authService.tokenData$.subscribe((data:any)=>{
      this.data = data;
    })
  }

  //Cerrar sesion
  onLogout(){
    this.authService.logout();
    this.data= null;
  }

}
```

Desafío

- Agregar más campos a la tabla tbl_usuario: nombre, apellidos.

Modificando base de datos.

```
MariaDB [web_integral]> ALTER TABLE tbl_usuario ADD nombre VARCHAR(200);
Query OK, 0 rows affected (0.014 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [web_integral]> ALTER TABLE tbl_usuario ADD apellido VARCHAR(200);
Query OK, 0 rows affected (0.031 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Modificar el envío del Token se agregan los nuevos campos.

```
5         }
6         console.log(password)
7         if(await utils.checkPassword(password, lstUsers[0].password)){
8             const newUser = {
9                 username: lstUsers[0].username,
10                password: lstUsers[0].password,
11                role: lstUsers[0].role,
12                name: lstUsers[0].nombre,
13                lastName: lstUsers[0].apellido
14            }
15            console.log(newUser)
16            let token= jwt.sign(newUser, conn.keys.secret, {expiresIn: '1h'});
```

Token decodificado.

PAYLOAD: DATA
<pre>{ "username": "progra03", "password": "\$2a\$10\$cmrMn6sgyWoNG9j7XjUrDuIiVwtDqgLwhCaPuHu0TIEMDrS 15jhEu", "role": "admin", "name": "Miguel", "lastName": "Mares", "iat": 1688510083, "exp": 1688513683 }</pre>
VERIFY SIGNATURE

- Una vez que está logueado imprimir su nombre completo

Imprimiendo valores.

Se utiliza el código creado anteriormente para obtener el valor del token.

```

data: any = {}; // Datos del usuario

constructor(private router: Router,
             private authService: AuthService) { }

ngOnInit(): void {
  this.authService.tokenData$.subscribe((data: any) => {
    this.data = data;
  })
}

```

Se agrega el nombre y apellido al html.

```

<div class="col titulo">
  <h2>Bienvenido {{data?.name}} {{data?.lastName}}</h2>
</div>

<div class="col">
  <div class="row">
    <div class="col">

```

Evidencia del funcionamiento.

The screenshot shows a web application titled "Desarrollo web Integral" with a sub-header "progra01". The main content area displays a welcome message "Bienvenido Pedro Miguel Ortega" and a grid of car listings. The first row includes three cars: Mitsubishi Lancer Evolution, Nissan Skyline, and Mazda 6. The second row shows three more cars: a silver sedan, an orange sports car, and a classic muscle car. The right side of the image shows the browser's developer console with two successful login messages for the user 'progra01' with password '12345'.