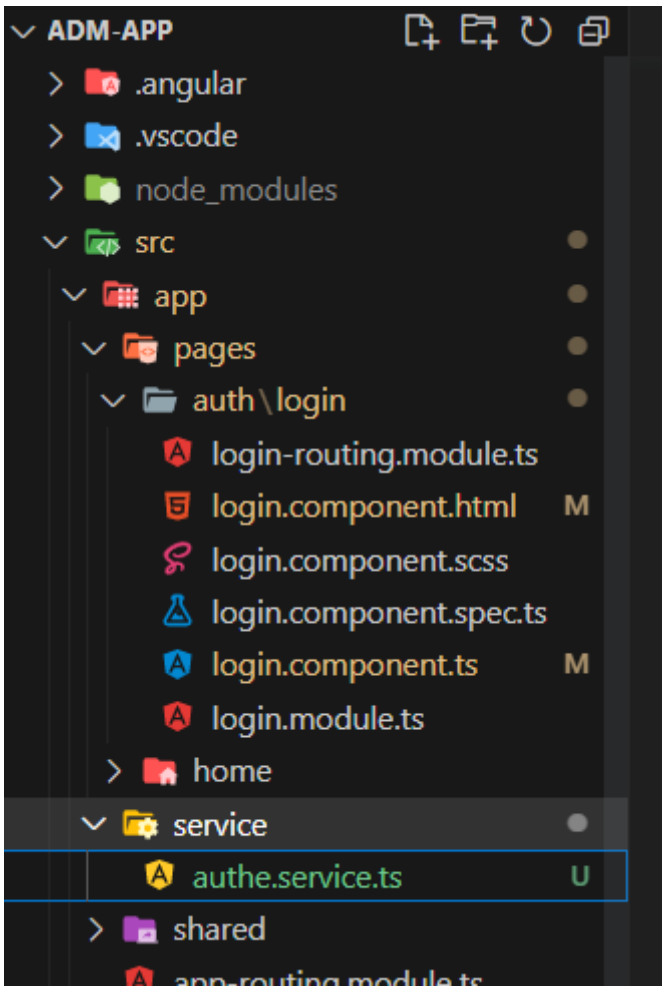
Crear servicio de Autenticación

Primeramente ,agregar módulo **MatSnackBarModule** al archivo **material.module.ts** para usar el servicio de mensajes; modificar el archivo **auth.service.ts**

Abrir terminal y ejecutar el siguiente comando para crear el servicio de autenticación: **ng g s shared/services/auth**

De hecho la ruta la definen de acuerdo a su experiencia



Agregamos la siguiente instrucción para ocupar elementos a utilizar para el manejo del Token y datos de usuario

```
import { BehaviorSubject, Observable, catchError, map, throwError }
from 'rxjs';
```

Agregar las propiedades a la clase AuthService:

token Es el token encriptado
tokenData Datos de usuario encriptado qué es el PayLoad

Agregar las siguientes instrucciones al archivo **auth.service.ts** con el objetivo de definir las propiedades **token** y **tokenData** como Observables y permitir suscripciones y saber de sus valores

```
import { Injectable } from '@angular/core';
import { BehaviorSubject, Observable, catchError, map, throwError } from 'r

@Injectable({
  providedIn: 'root',
})
export class AutheService {
  private token = new BehaviorSubject<string>('');
  private tokenData = new BehaviorSubject<any>({});
  constructor() {}

  get token$(): Observable<string> {
    return this.token.asObservable();
  }

  get tokenData$():Observable<any>{
    return this.tokenData.asObservable();

  }
}

```

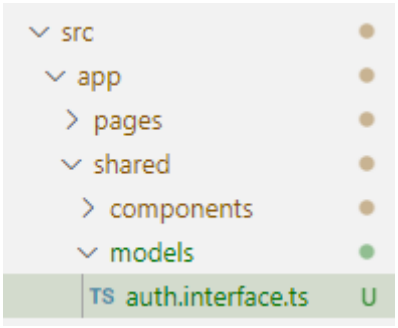
el utilizar environments en tu aplicación es con el objetivo de utilizar API KEY para entornos de pruebas y de producción durante el desarrollo de la aplicación.

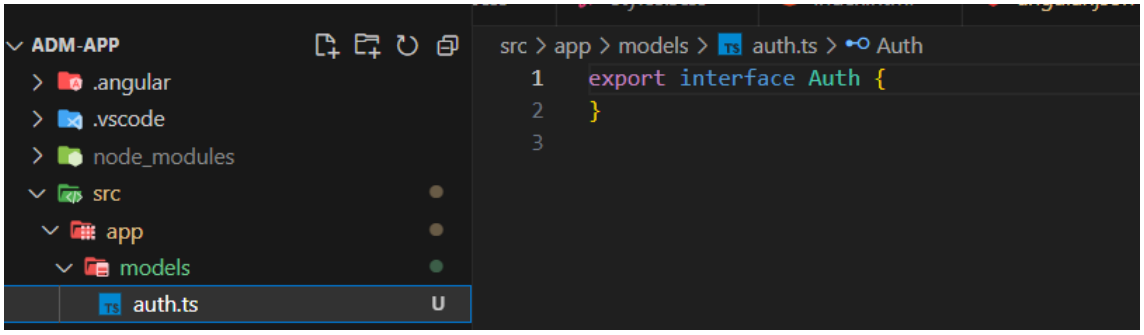
En la terminal ejecutar el siguiente comando: **ng g environments**

Modificar los archivos environments de la siguiente forma

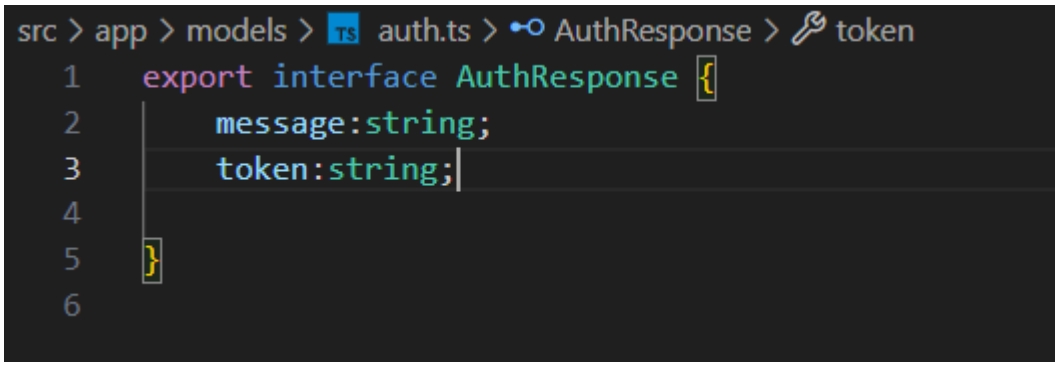
```
export const environment = {
    production: false,
    recaptcha:{
        siteKey:'6LcCRnUmAAAAAB7HgUSKzZNUVN_PZp
    },

    API_URL:"http://localhost:3000/api"
};

```

En carpeta **shared/models** agregar un archivo con el nombre **auth.interface.ts,** en caso qué no exista se deberá de crear





Modificar el archivo **auth.interface.ts**



Agregar el siguiente método handleError al archivo **auth.service.ts**

```
/**
   * Método para el maneno de errorres desplegados en el SnackBar
   * @param error Error enviado al método
   * @returns Lanza error en caso que el mensaje no esté nulo o
vacío
   */
  handleError(error:any):Observable<never>{
    let message = "Ha ocurrido un error";

    if (error) {
      message = `${error.error.message}`;
    }
    console.log('Handle Error ' + message);

    this.snackBar.open(message, '', {
      duration: 5*1000,
```

```
      panelClass: ['error-snackbar'],
      horizontalPosition: 'right',
      verticalPosition: 'top'
    });
    return throwError(message);
  }
```

Guardado de Sesión

Instalar el módulo @auth0/angular-jwt

Installation

```
# installation with npm
npm install @auth0/angular-jwt

# installation with yarn
yarn add @auth0/angular-jwt
```

```
added 1 package, and audited 994 packages in 13s

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

36 vulnerabilities (33 moderate, 3 high)

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.
PS C:\Users\52418\Documents\utng\Materias\desarrollo web integral\Paracticas\adm-app>
```

Importar clase JwtHelperService

```
import { JwtHelperService } from '@auth0/angular-jwt';
```

Agregar una variable llamada helper en auth.service.ts

```
const helper = new JwtHelperService();
@Injectable({
  providedIn: 'root'
})
export class AuthService {
```

```
import { JwtHelperService } from '@auth0/angular-jwt';

const helper = new JwtHelperService();
```

LocalStorage

El LocalStorage permite almacenar datos en el navegador web. Y que estos persistan y estén disponibles durante la navegación en la aplicación web, hasta que esta información sea borrada del navegador.

Agregar el método **saveLocalStorage** en archivo **auth.service.ts**

Agregar el método **logout** en archivo **auth.service.ts**

```
49      */
50
51      seveLocalStarage(token: string) {
52        localStorage.setItem('token', token);
53      }
54
55      /**
56       * Cierre de sesión y la eliminacio de local storage
57       */
58
59      logout() {
60        localStorage.removeItem('token');
61        this.token.next('');
62        this.tokenData.next(null);
63        this.router.navigateByUrl('/');
64      }
65
```

Agregar el método **checklogin** en archivo **auth.service.ts**

```
/**
   * Cada vez que se realice una petición verifica que Token
   */
  checkToken() {
    let token = localStorage.getItem("token");

    if(token) {
      //Checa expiración de token traido 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();
    }
```

```
  }

  /**
   * Cada vez que se realice una petición verifica que Token
   */
  checkToken() {
    let token = localStorage.getItem("token");

    if(token) {
      //Checa expiración de token traido 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();
    }
  }

}
```

Modificar método login de la siguiente forma:

```
/**
   * Hace la petición  al BackEnd
   * @param loginData Usuario y Contraseña de usuario
   * @returns
   */
  login(loginData:any):Observable<AuthResponse|void> {

    return
this.http.post<AuthResponse>(`${environment.API_URL}/`,loginData)
      .pipe(map((data:AuthResponse) => {

        if(data.token) { //No hay errores y contiene información
          this.saveLocalStorage(data.token);
          this.token.next(data.token); //Notifica a suscriptores
          this.router.navigate(['/home']);

          this.checkToken();
        }
        return data;
```

```
      }),
      catchError((error) => this.handleError(error)));
  }
```

```
  login(loginData: any): Observable<AuthResponse | void> {
    return this.http
      .post<AuthResponse>('http://localhost:3000/api/login', loginData)
      .pipe(
        map((data: AuthResponse) => {
          if (data.token) {
            console.log(data)
            //No hay errores y contiene información
            this.seveLocalStarage(data.token);
            this.token.next(data.token); //Notifica a suscriptores
            this.router.navigate(['/home']);

            this.checkToken();
          }
          return data;
        }),
        catchError((error) => this.handleError(error))
      );
  }
}
```

Modificar el archivo **material.module.ts** y agregar **MatToolbarModule**

```
8   import { MatToolbarModule } from '@angular/material/toolbar';
9
10  const myModules :any = [
11      MatButtonModule,
12      MatCardModule,
13      MatInputModule,
14      MatIconModule,
15      MatGridListModule,
16      MatSnackBarModule,
17      MatToolbarModule
18  ];
```

```
You, 36 minutes ago | 2 authors (Yop789 and others)
import { NgModule } from '@angular/core';
import { MatFormFieldModule } from '@angular/material/form-fiel
import { MatCardModule } from '@angular/material/card';
import { MatInputModule } from '@angular/material/input';
import { MatIconModule } from '@angular/material/icon';
import { MatGridListModule } from '@angular/material/grid-list'
import { MatButtonModule } from '@angular/material/button';
import {MatToolbarModule} from '@angular/material/toolbar';
import {MatSnackBarModule} from '@angular/material/snack-bar';

const myModules: any = [
  MatButtonModule,
  MatCardModule,
  MatInputModule,
  MatIconModule,
  MatGridListModule,
  MatFormFieldModule,
  MatToolbarModule,        You, 36 minutes ago • Uncommitted cha
  MatSnackBarModule
```

Modificar el archivo **header.component.scss**

```
.spacer {
    flex: 1 1 auto;
}
```

```
.example-spacer {
  flex: 1 1 auto;
}       Yop789, last month • p
```

Modificar el archivo **header.component.html**

```
<mat-toolbar>
    <span>Desarrollo Web Integral</span>
    <span class="spacer"></span>

    <button mat-button>{{data?.username }}</button>
    <button mat-icon-button (click)="onLogout()">
      <mat-icon >login</mat-icon>       You, 20 seconds ago • Uncommi
    </button>
</mat-toolbar>
```

```
<div class="header">
  <mat-toolbar style="background-color: transparent;">       You, 1

    <span>App-David</span>
    <span class="example-spacer"></span>
    <button mat-button *ngIf="data"
    >
      {{data?.username}}
    </button>
    <button *ngIf="data"
      mat-icon-button
      class="example-icon"
      aria-label="Example icon-button with share icon"
      (click)=" onLogout()"
    >
      <mat-icon>login</mat-icon>
    </button>
  </mat-toolbar>
</div>
```

Modificar el archivo **header.component.ts**

```
 8  export class HeaderComponent implements OnInit {
 9
10    constructor() { }
11
12    ngOnInit(): void { }
13
14    onLogout() { }
15
16  }
```

Modificar el archivo **auth.service.ts** y agregar al constructor la siguiente línea:

```
this.checkToken();
```

```
constructor(private snackBar: MatSnackBar,
  private http:HttpClient,private router: Router) {
    this.checkToken()
  }
```

Modificar el archivo header.component.ts

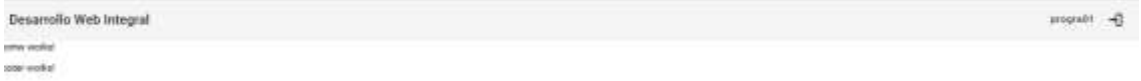
```
export class HeaderComponent implements OnInit {

  data: any = {}; //Datos de Usuario

  constructor(private authService: AuthService) {
  }

  /**
   * Se suscribe para notificación cuando cambie el token de usuario.
   */
  ngOnInit(): void {
    this.authService.getTokenData$().subscribe((data:any)=> {
      this.data = data;
    });
  }

  /**      You, 25 seconds ago • Uncommitted changes
   * Cerrar sesión
   */
  onLogout() {
    this.authService.logout();
    this.data = null;
  }
```

```
export class HeaderComponent implements OnInit {
  data: any = [];

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

  onLogout() {
    this.authService.logout();
    this.data = null;       You, 1 second ago • Uncommitt
  }
}
```

Modificar el archivo **header.component.html**

```
<mat-toolbar>
    <span>Desarrollo Web Integral</span>
    <span class="spacer"></span>

    <button mat-button>{{data?.username }}</button>
    <button mat-icon-button (click)="onLogout()">
      <mat-icon >login</mat-icon>
    </button>
</mat-toolbar>
```

Al finalizar se mostrará de la siguiente manera



Desafío

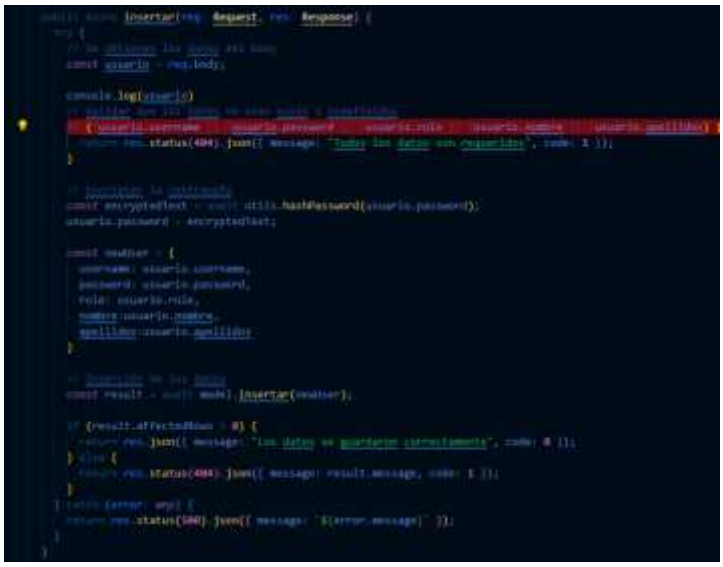
- Agregar más campos a la tabla tbl_usuario: nombre, apellidos.

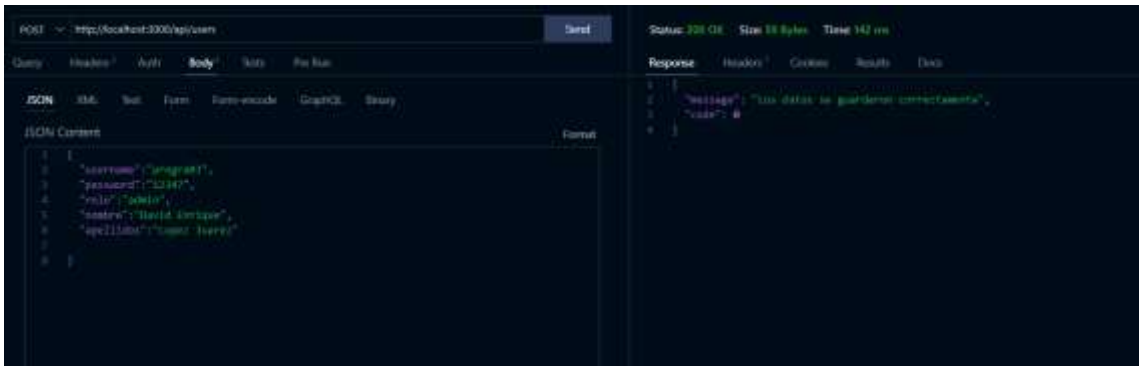
Agregar columnas a la tabla de la base de datos

```
ALTER TABLE Usuarios
ADD nombre VARCHAR(30) not null ;

ALTER TABLE Usuarios
add apellidos VARCHAR(30) not null;
```

	username	password	role	nombre	apellidos
▶	progra01	$2a$10$HDHZsiaqD7XNmKCUa4sjOee0mXLH6A...	admin		
	progra02	$2a$10$9LAcdWlzxx297JNaSZCcd.5XlTP3H24s...	hsdjhs		

Back end
Incertar



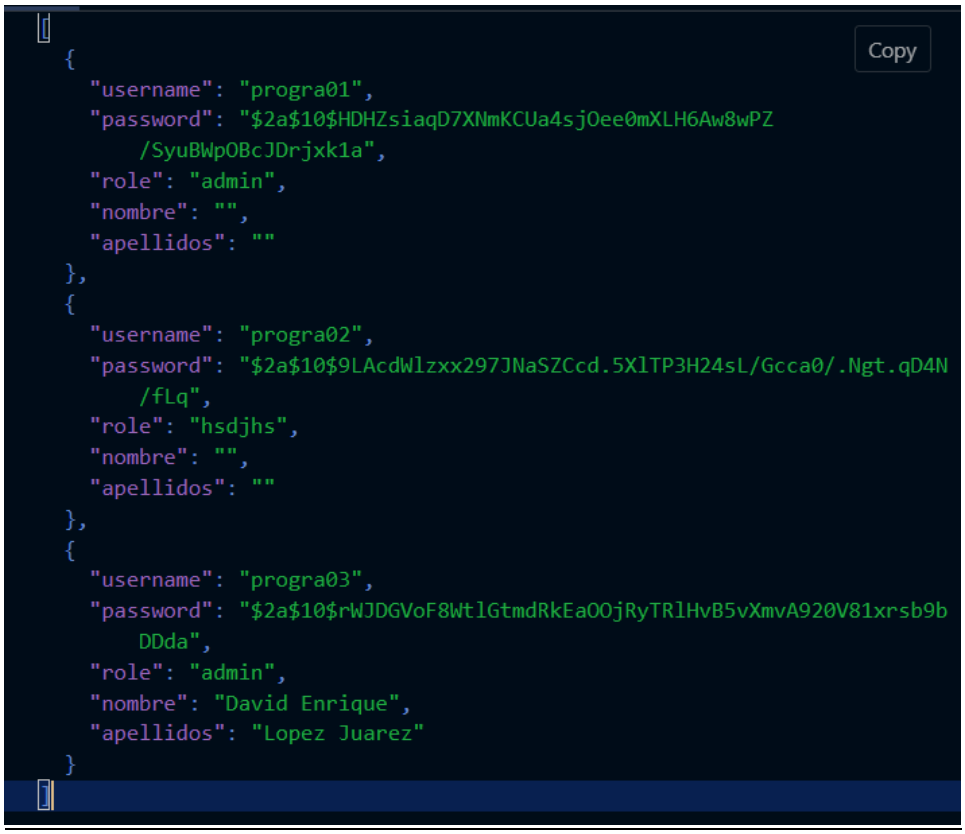


Actualizar

```
public async actualizar(req: Request, res: Response) {
  try {
    // Se obtienen los datos del body
    const usuario = req.body;

    // Validar que los datos no sean nulos o indefinidos
    if (!usuario.username || !usuario.password || !usuario.role || !usuario.nombre || !usuario.apellidos) {
      return res.status(404).json({ message: "Todos los datos son requeridos", code: 1 });
    }

    // Encriptar la contraseña
    const encryptedText = await utils.hashPassword(usuario.password);
    usuario.password = encryptedText;

    const newUser = {
      password: usuario.password
    }

    // Actualización de los datos
    const result = await model.actualizar(newUser, usuario.username);

    if (result.affectedRows > 0) {
      return res.json({ message: "Los datos se actualizaron correctamente", code: 0 });
    } else {
      return res.status(404).json({ message: result.message, code: 1 });
    }
  } catch (error: any) {
    return res.status(500).json({ message: `${error.message}` });
  }
}
```

Listar user modelo

```
public async listar() {
    const result = await pool.then( async (connection) => {
        return await connection.query(
            " SELECT u.username, u.password, u.role, u.nombre, u.apellidos"
            + " FROM Usuarios u ")  });
    return result;
}
```



Iniciar sesión authController

```
public async iniciarSesion(req: Request, res: Response) {
  try {
    const { username, password } = req.body;
    console.log(username)
    // verificar que los datos no esten vacios
    if (
      validator.isEmpty(username.trim()) ||
      validator.isEmpty(password.trim())
    ) {
      return res
        .status(400)
        .json({ message: "Los campos son requeridos", code: 1 });
    }
    const lstUsers = await model.getuserByusername(username);
    if (lstUsers.length <= 0) {
      return res
        .status(404)
        .json({
          message: "El usuario y/o contraseña es incorrecto",
          code: 1,
        });
    }
    console.log(lstUsers[0].username,lstUsers[0].password )
    const newUser = {
      username:lstUsers[0].username,
      password:lstUsers[0].password,
      role:lstUsers[0].role,
      nombre:lstUsers[0].nombre,
      apellidos:lstUsers[0].apellidos

    }
    let token =jwt.sign(newUser,db.keys.secret,{expiresIn:'1h'})
    return res.json({ message: "Autentificación correcta",token, code: 0 });

  } catch (error: any) {
    return res.status(500).json({ message: `${error.message}` });
  }
 }
}
export const authController = new AuthController();
```

Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey
J1c2VybmFtZSI6InByb2dyYTAzIiwicGFzc3dvc
mQiOiIkMmEkMTAkcldKREdWb0Y4V3RsR3RtZFJr
RWFPT2pSeVRSbEh2QjV2WG12QTkyMFY4MXhyc2I
5YkREZGEiLCJyb2xlIjoiYWRtaW4iLCJub21icm
UiOiJEYXZpZCBFbnJpcXVlIiwiYXBlbGxpZG9zI
joiTG9wZXogSnVhcmV6IiwiaWF0IjoxNjg4MzQx
NjQzLCJleHAiOjE2ODgzNDUyNDN9.cRuw77DCid
TwxMWjngjClj6gaeq4wIVNAtf8QGZjj5Y
```

Decoded

```
HEADER: ALGORITHM & TOKEN TYPE

{
  "alg": "HS256",
  "typ": "JWT"
}

PAYLOAD: DATA

{
  "username": "progra03",
  "password":
"$2a$10$WLdOGVcF8Rt1GtwdRsEaOOjRyTR1HvB5vXmvA92BVdlarsh
9bDDda",
  "role": "admin",
  "nombre": "David Enrique",
  "apellidos": "Lopez Juarez",
  "iat": 1688341643,
  "exp": 1688345243
}
```

- Una vez qué está logueado imprimir su nombre completo

```html
<div class="header">
  <mat-toolbar style="background-color: transparent;">

    <span>App-David</span>
    <span class="example-spacer"></span>
    <button mat-button *ngIf="data"
    >
      {{data?.nombre+' '+data?.apellidos}}
    </button>
    <button *ngIf="data"
      mat-icon-button
      class="example-icon"
      aria-label="Example icon-button with share icon"
      (click)=" onLogout()"
    >
      <mat-icon>login</mat-icon>
    </button>
  </mat-toolbar>
</div>
```