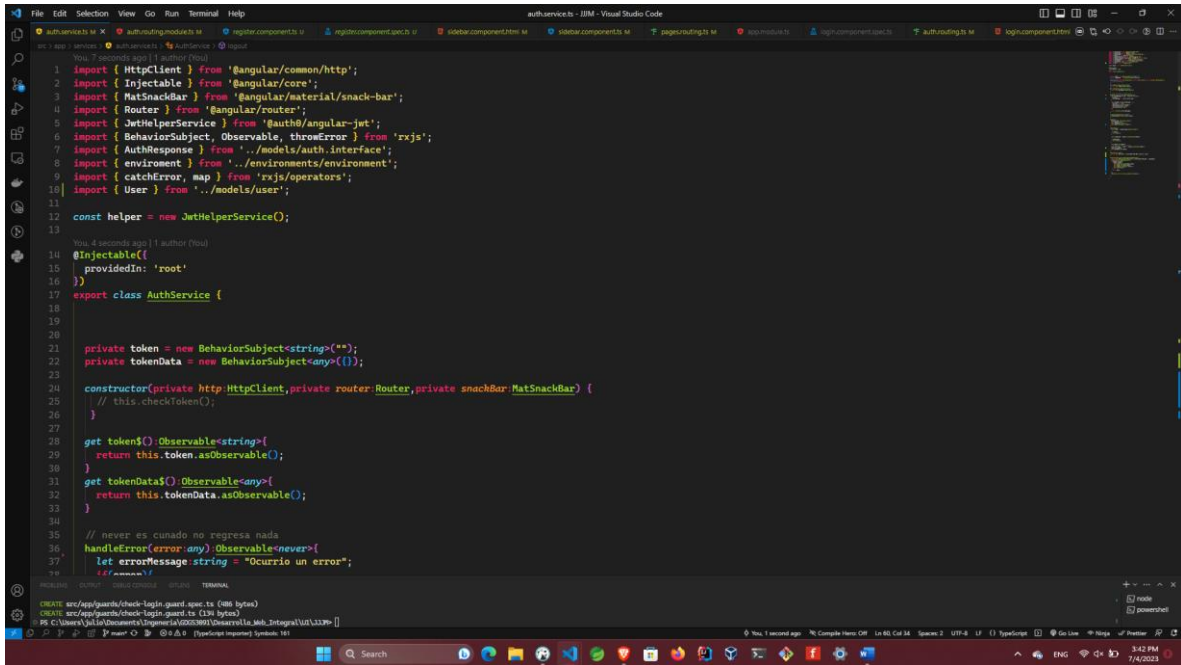
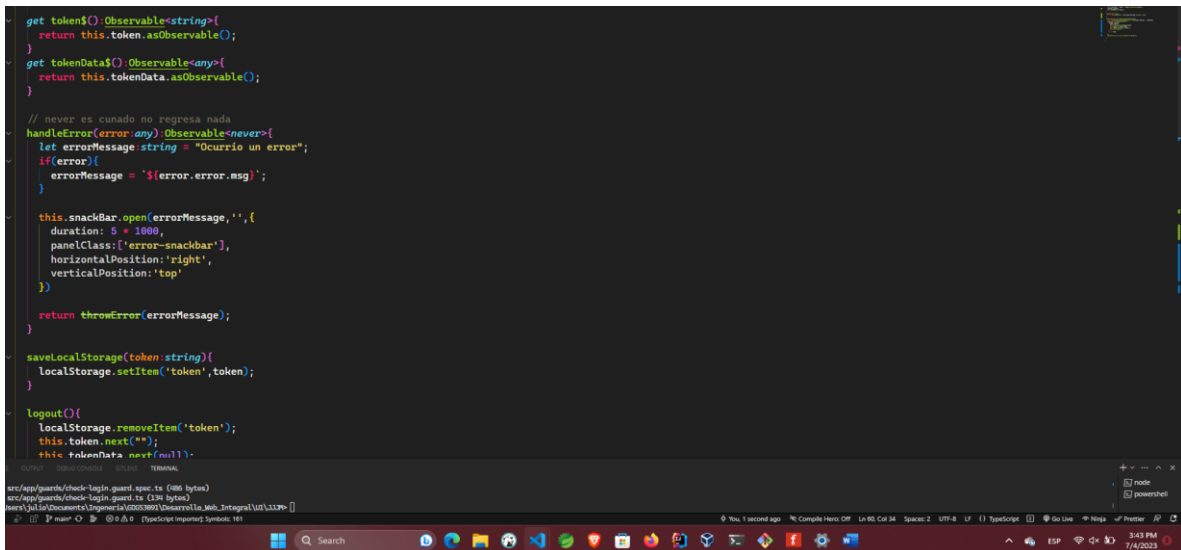


Creacion de los servicios del servicio de auth service



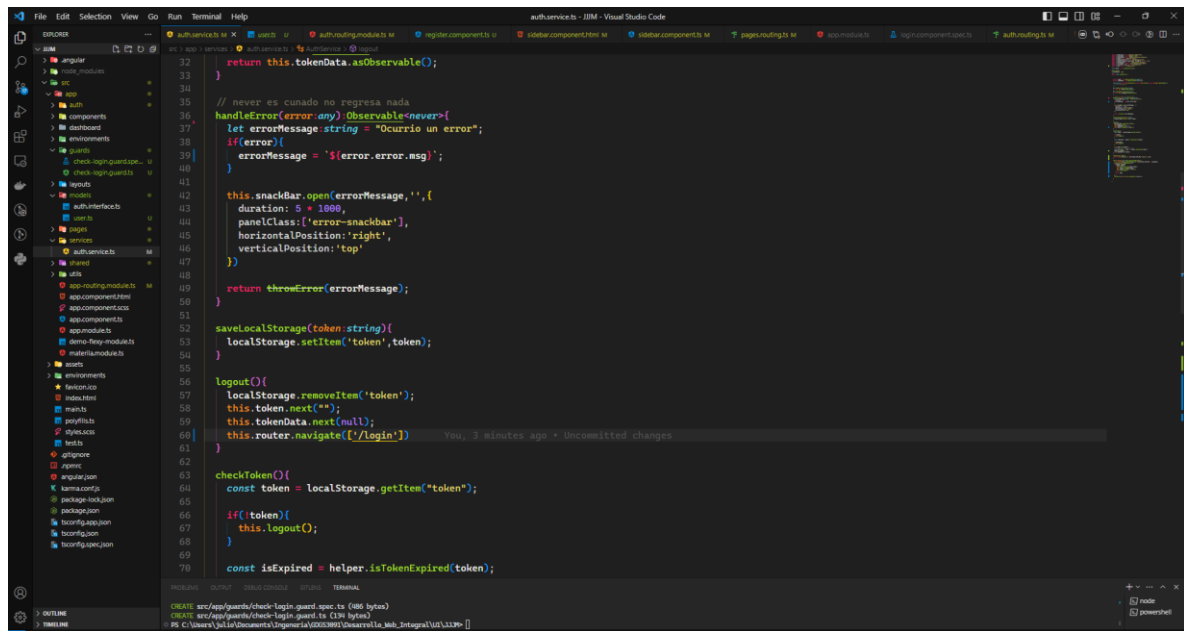
```
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { MatSnackBar } from '@angular/material/snack-bar';
4 import { Router } from '@angular/router';
5 import { JethelperService } from '@auth0/angular-jwt';
6 import { BehaviorSubject, Observable, throwError } from 'rxjs';
7 import { AuthResponse } from '../models/auth.interface';
8 import { environment } from '../environments/environment';
9 import { catchError, map } from 'rxjs/operators';
10 import { User } from '../models/user';
11
12 const helper = new JethelperService();
13
14 @Injectable({
15   providedIn: 'root'
16 })
17 export class AuthService {
18
19
20   private token = new BehaviorSubject<string>('');
21   private tokenData = new BehaviorSubject<any>({});
22
23   constructor(private http: HttpClient, private router: Router, private snackBar: MatSnackBar) {
24     // this.checkToken();
25   }
26
27   get token(): Observable<string> {
28     return this.token.asObservable();
29   }
30
31   get tokenData(): Observable<any> {
32     return this.tokenData.asObservable();
33   }
34
35   // never es cuando no regresa nada
36   handleError(error: any): Observable<never> {
37     let errorMessage string = "Ocurrio un error";
38     if (error) {
39       errorMessage = `${error.error.msg}`;
40     }
41
42     this.snackBar.open(errorMessage, '', {
43       duration: 5 * 1000,
44       panelClass: ['error-snackbar'],
45       horizontalPosition: 'right',
46       verticalPosition: 'top'
47     });
48
49     return throwError(errorMessage);
50   }
51
52   saveLocalStorage(token: string) {
53     localStorage.setItem('token', token);
54   }
55
56   logout() {
57     localStorage.removeItem('token');
58     this.token.next('');
59     this.tokenData.next(null);
60   }
61 }
```

Agregando propiedad para obtener el Token y obtener la información que contiene el token



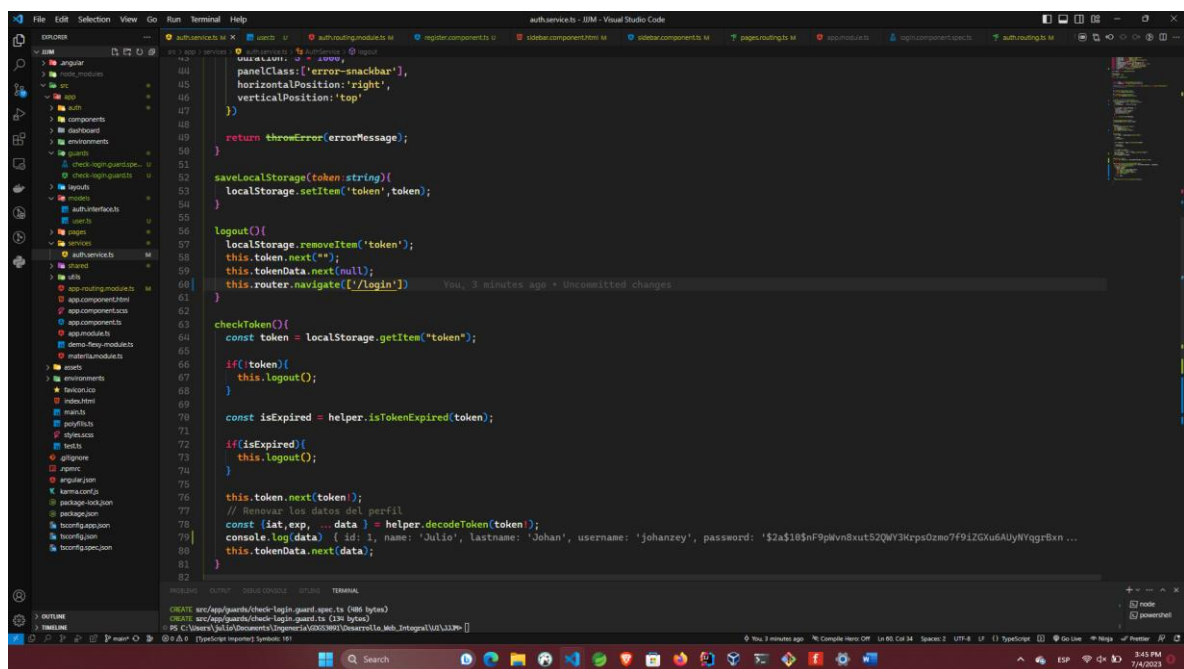
```
1 get token(): Observable<string> {
2   return this.token.asObservable();
3 }
4
5 get tokenData(): Observable<any> {
6   return this.tokenData.asObservable();
7 }
8
9 // never es cuando no regresa nada
10 handleError(error: any): Observable<never> {
11   let errorMessage string = "Ocurrio un error";
12   if (error) {
13     errorMessage = `${error.error.msg}`;
14   }
15
16   this.snackBar.open(errorMessage, '', {
17     duration: 5 * 1000,
18     panelClass: ['error-snackbar'],
19     horizontalPosition: 'right',
20     verticalPosition: 'top'
21   });
22
23   return throwError(errorMessage);
24 }
25
26 saveLocalStorage(token: string) {
27   localStorage.setItem('token', token);
28 }
29
30 logout() {
31   localStorage.removeItem('token');
32   this.token.next('');
33   this.tokenData.next(null);
34 }
```

Handle error es un metodo para la obtención de errores que contiene el back como los errores HTTP



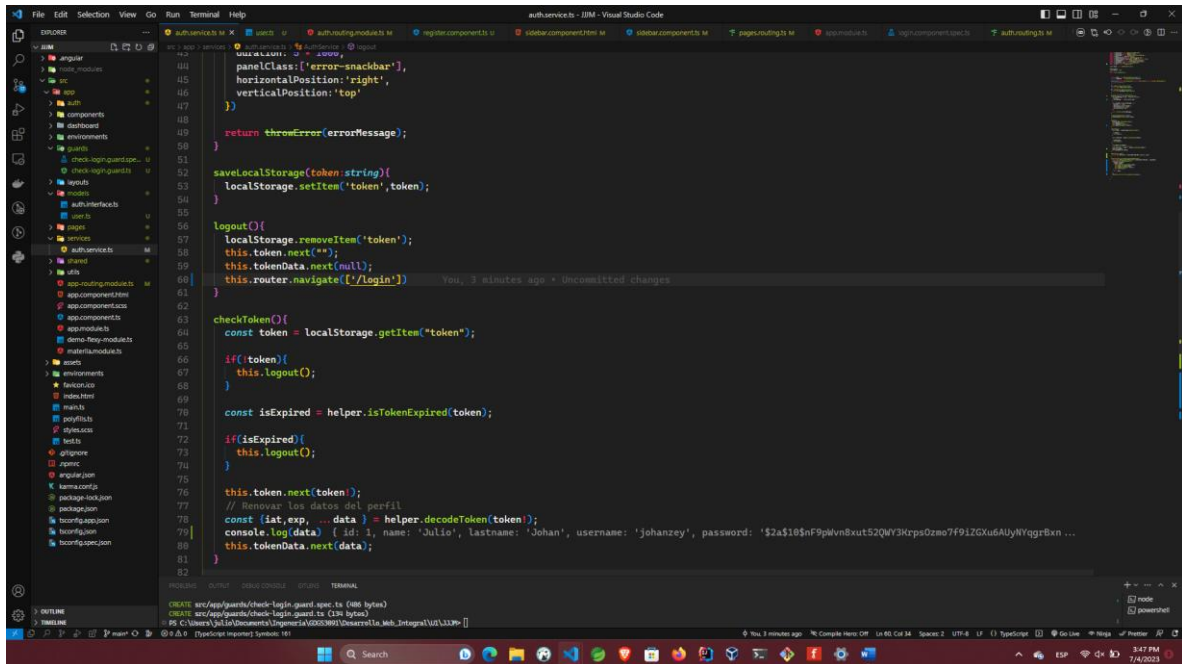
```
32 return this.tokenData.asObservable();
33 }
34
35 // never es cuando no regresa nada
36 handleError(error: any): Observable<never> {
37   let errorMessage string = "Ocurrio un error";
38   if (error) {
39     errorMessage = `${error.error.msg}`;
40   }
41
42   this.snackBar.open(errorMessage, '', {
43     duration: 5 * 1000,
44     panelClass: ['error-snackbar'],
45     horizontalPosition: 'right',
46     verticalPosition: 'top'
47   });
48
49   return throwError(errorMessage);
50 }
51
52 saveLocalStorage(token: string) {
53   localStorage.setItem('token', token);
54 }
55
56 logout() {
57   localStorage.removeItem('token');
58   this.token.next("");
59   this.tokenData.next(null);
60   this.router.navigate(['/login'])
61 }
62
63 checkToken() {
64   const token = localStorage.getItem("token");
65
66   if (token) {
67     this.logout();
68   }
69
70   const isExpired = helper.isTokenExpired(token);
```

Creamos un método para guardar el token en local storage además de la creación del método logout



```
44   panelClass: ['error-snackbar'],
45   horizontalPosition: 'right',
46   verticalPosition: 'top'
47 })
48
49 return throwError(errorMessage);
50 }
51
52 saveLocalStorage(token: string) {
53   localStorage.setItem('token', token);
54 }
55
56 logout() {
57   localStorage.removeItem('token');
58   this.token.next("");
59   this.tokenData.next(null);
60   this.router.navigate(['/login'])
61 }
62
63 checkToken() {
64   const token = localStorage.getItem("token");
65
66   if (token) {
67     this.logout();
68   }
69
70   const isExpired = helper.isTokenExpired(token);
71
72   if (isExpired) {
73     this.logout();
74   }
75
76   this.token.next(token);
77
78   // Renueva los datos del perfil
79   const {iat, exp, ...data} = helper.decodeToken(token);
80   console.log(data) { id: 1, name: 'Julio', lastname: 'Johan', username: 'johanzey', password: '$2a$16$nf9pWv8xut52QWY3Kp50zmo7f91ZGXu6AUyVqgrBxn ...
81   this.tokenData.next(data);
82 }
```

Verificación del token



```
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { LocalStorage } from '@ngx-pwa/local-storage';
import { Observable } from 'rxjs';
import { AuthResponse } from '../models/auth-response';
import { User } from '../models/user';
import { AuthService } from './auth-service';
import { Helper } from './helper';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  private router: Router;
  private localStorage: LocalStorage;
  private authService: AuthService;
  private helper: Helper;

  constructor(router: Router, localStorage: LocalStorage, authService: AuthService, helper: Helper) {
    this.router = router;
    this.localStorage = localStorage;
    this.authService = authService;
    this.helper = helper;
  }

  login(loginData: any): Observable<AuthResponse> {
    return this.authService.login(loginData);
  }

  register(userData: any): Observable<AuthResponse> {
    return this.authService.register(userData);
  }

  logout(): void {
    this.localStorage.removeItem('token');
    this.tokenData.next(null);
    this.router.navigate(['/login']);
  }

  checkToken(): void {
    const token = this.localStorage.getItem('token');

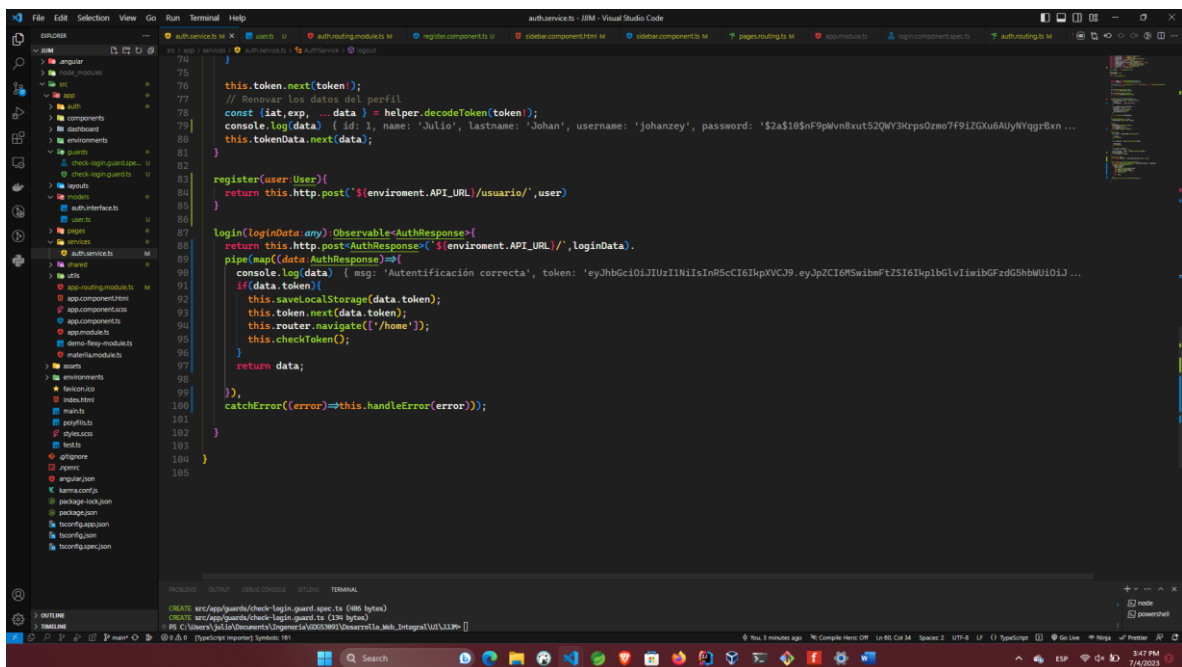
    if (!token) {
      this.logout();
    }

    const isExpired = this.helper.isTokenExpired(token);

    if (isExpired) {
      this.logout();
    }

    this.tokenData.next(token);
    // Renovar los datos del perfil
    const {iat, exp, ...data} = this.helper.decodeToken(token);
    console.log(data) { id: 1, name: 'Julio', lastname: 'Johan', username: 'johanzy', password: '$2a$10$9F9pWvN8xut52QWY3Krp50zmo7f91ZGXu6AuyWqgrBxn ... };
    this.tokenData.next(data);
  }
}
```

Método de login y register donde se harán las peticiones http al backend



```
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { LocalStorage } from '@ngx-pwa/local-storage';
import { Observable } from 'rxjs';
import { AuthResponse } from '../models/auth-response';
import { User } from '../models/user';
import { AuthService } from './auth-service';
import { Helper } from './helper';
import { axios } from 'axios';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  private router: Router;
  private localStorage: LocalStorage;
  private authService: AuthService;
  private helper: Helper;

  constructor(router: Router, localStorage: LocalStorage, authService: AuthService, helper: Helper) {
    this.router = router;
    this.localStorage = localStorage;
    this.authService = authService;
    this.helper = helper;
  }

  login(loginData: any): Observable<AuthResponse> {
    return this.http.post<AuthResponse>(`${environment.API_URL}/login`, loginData);
  }

  register(userData: any): Observable<AuthResponse> {
    return this.http.post<AuthResponse>(`${environment.API_URL}/register`, userData);
  }

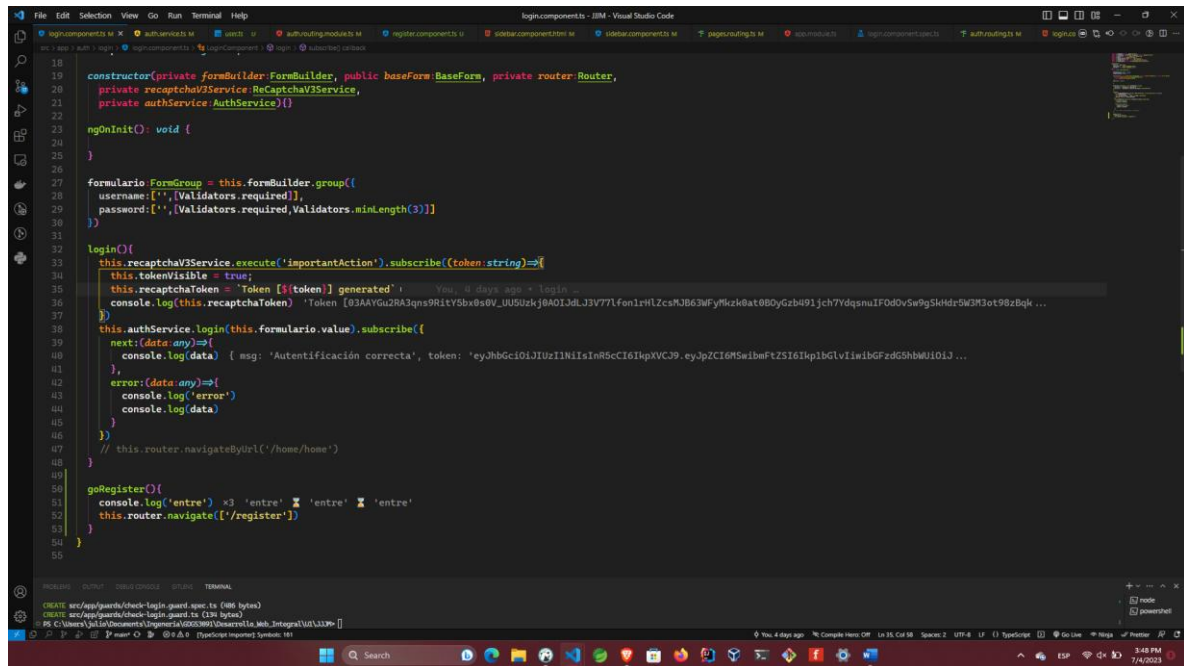
  login(loginData: any): Observable<AuthResponse> {
    return this.http.post<AuthResponse>(`${environment.API_URL}/login`, loginData);
  }

  register(userData: any): Observable<AuthResponse> {
    return this.http.post<AuthResponse>(`${environment.API_URL}/register`, userData);
  }

  login(loginData: any): Observable<AuthResponse> {
    return this.http.post<AuthResponse>(`${environment.API_URL}/login`, loginData);
  }

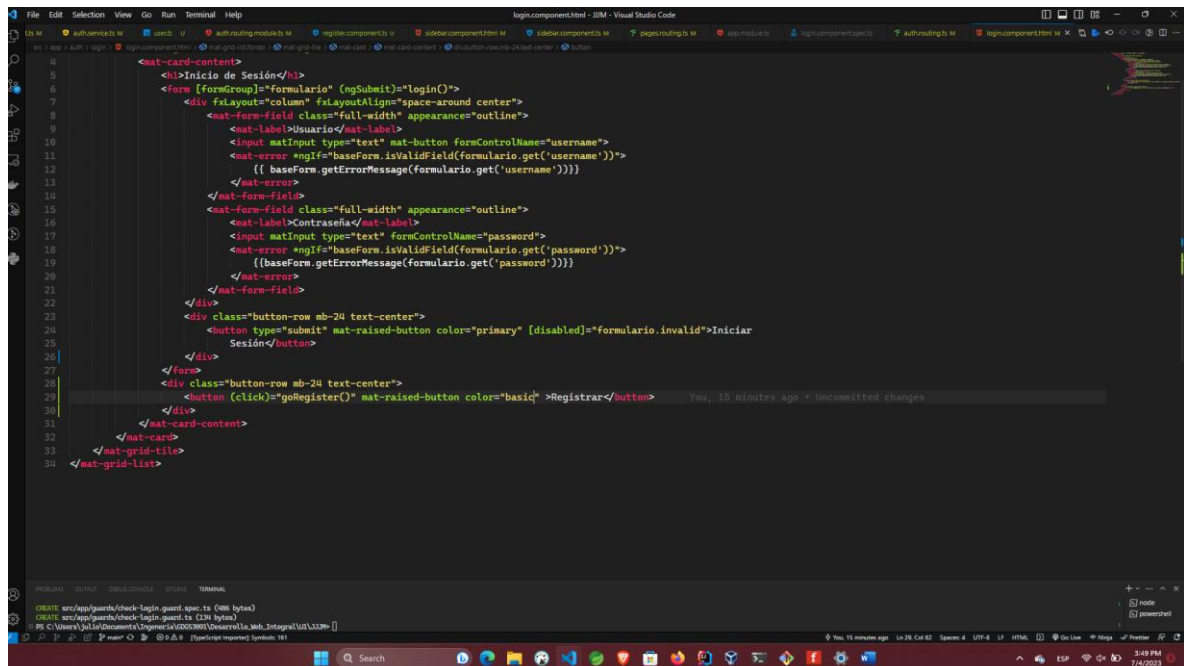
  register(userData: any): Observable<AuthResponse> {
    return this.http.post<AuthResponse>(`${environment.API_URL}/register`, userData);
  }
}
```

Ponemos en el constructor el auth service donde podremos utilizar los métodos como el login para hacer la petición



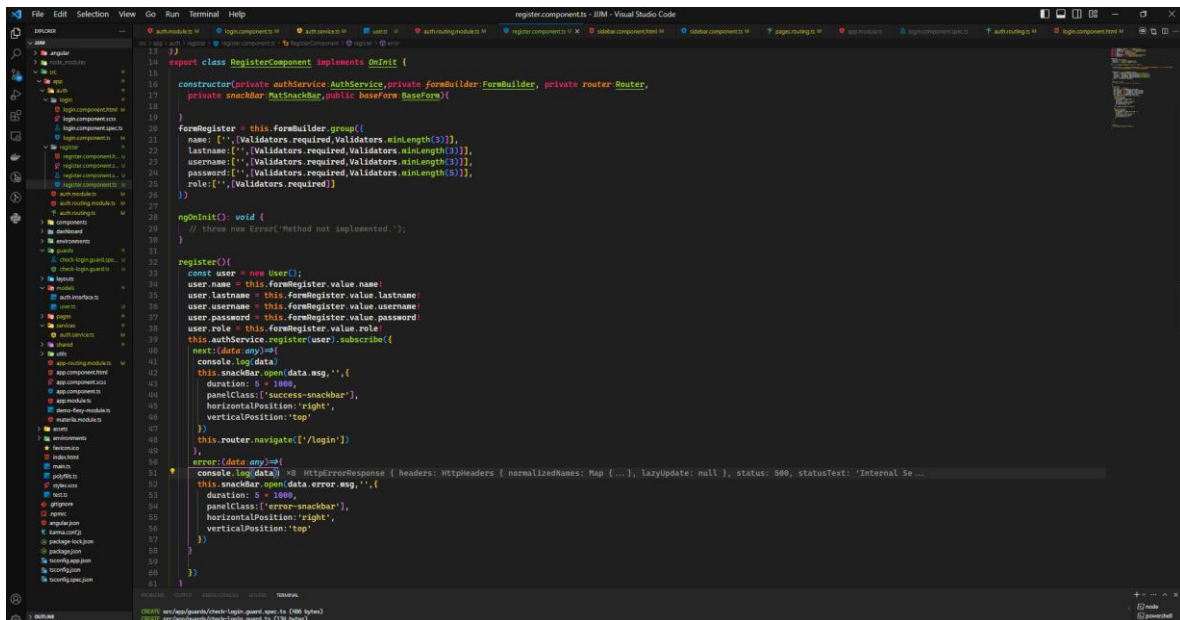
```
18 constructor(private FormBuilder: FormBuilder, public baseForm: BaseForm, private router: Router,  
19 private recaptchaV3Service: RecaptchaV3Service,  
20 private authService: AuthService) {}  
21  
22 ngOnInit(): void {  
23  
24 }  
25  
26  
27 formulario FormGroup = this.formBuilder.group({  
28   username: ['', Validators.required],  
29   password: ['', Validators.required, Validators.minLength(3)]  
30 })  
31  
32 login() {  
33   this.recaptchaV3Service.execute('importantAction').subscribe((token: string) => {  
34     this.tokenVisible = true;  
35     this.recaptchaToken = 'Token [' + token + '] generated';  
36     console.log(this.recaptchaToken);  
37     this.authService.login(this.formulario.value).subscribe({  
38       next: (data: any) => {  
39         console.log(data);  
40         console.log('mensaje: Autenticación correcta', token);  
41       },  
42       error: (data: any) => {  
43         console.log('error');  
44         console.log(data);  
45       }  
46     });  
47     // this.router.navigateByHref('/home/home')  
48   }  
49  
50 goRegister() {  
51     console.log('entre') * 3 'entre' * 2 'entre' * 1 'entre'  
52     this.router.navigate(['/register'])  
53   }  
54  
55 }
```

Ponemos el metodo para el inicio de sesión e ir al registro



```
6 <mat-card-content>  
7   <h3>Inicio de Sesión</h3>  
8   <form [formGroup]="formulario" (ngSubmit)="login()">  
9     <div fxLayout="column" fxLayoutAlign="space-around center">  
10      <mat-form-field class="full-width" appearance="outline">  
11        <input matInput type="text" mat-form-field="username">  
12        <mat-error *ngIf="baseForm.isValidField(formulario.get('username'))">  
13          {{ baseForm.getErrorMessage(formulario.get('username')) }}  
14      </mat-form-field>  
15      <mat-form-field class="full-width" appearance="outline">  
16        <mat-label>Contraseña</mat-label>  
17        <input matInput type="password" formControlName="password">  
18        <mat-error *ngIf="baseForm.isValidField(formulario.get('password'))">  
19          {{ baseForm.getErrorMessage(formulario.get('password')) }}  
20      </mat-form-field>  
21    </div>  
22    <div class="button-row mb-24 text-center">  
23      <button type="submit" mat-raised-button color="primary" [disabled]="formulario.invalid">Iniciar  
24        Sesión</button>  
25    </div>  
26    <div class="button-row mb-24 text-center">  
27      <button (click)="goRegister()" mat-raised-button color="basic">Registrar</button>  
28    </div>  
29  </mat-card-content>  
30 </mat-card>  
31 </mat-grid-tile>  
32 </mat-grid-list>
```

Creamos el método para registrarnos, aquí igual agregamos nuestro formulario para crear nuestro usuario, además de poner en el constructor las dependencias que requerimos



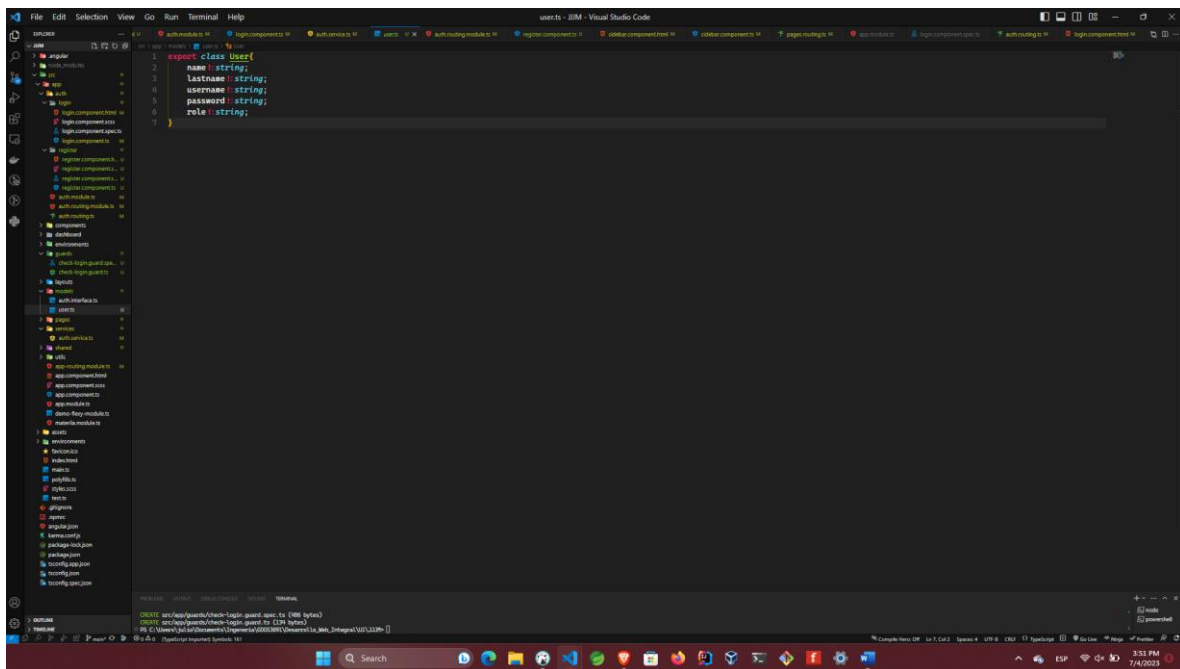
```
export class RegisterComponent implements OnInit {
  constructor(private authService: AuthService, private formBuilder: FormBuilder, private router: Router,
    private snackBar: MatSnackBar, public baseForm: BaseForm) {}

  formRegister = this.formBuilder.group({
    name: ['', Validators.required, Validators.minLength(3)],
    lastname: ['', Validators.required, Validators.minLength(3)],
    username: ['', Validators.required, Validators.minLength(3)],
    password: ['', Validators.required, Validators.minLength(5)],
    role: ['', Validators.required]
  });

  ngOnInit(): void {
    // show new Error(Method not implemented.);
  }

  register() {
    const user = new User();
    user.name = this.formRegister.value.name;
    user.lastname = this.formRegister.value.lastname;
    user.username = this.formRegister.value.username;
    user.password = this.formRegister.value.password;
    user.role = this.formRegister.value.role;
    this.authService.register(user).subscribe({
      next: (data: any) => {
        console.log(data);
        this.snackBar.open(data.msg, '', {
          duration: 5 * 1000,
          panelClass: ['success-snackbar'],
          horizontalPosition: 'right',
          verticalPosition: 'top'
        });
        this.router.navigate(['/login']);
      },
      error: (data: any) => {
        console.log(data);
        this.snackBar.open(data.msg, '', {
          duration: 5 * 1000,
          panelClass: ['error-snackbar'],
          horizontalPosition: 'right',
          verticalPosition: 'top'
        });
      }
    });
  }
}
```

Antes que nada, agregamos una clase donde estará estructurada nuestro clase de usuario



```
export class User {
  name: string;
  lastname: string;
  username: string;
  password: string;
  role: string;
}
```


Implementando todo el HTML para que el usuario se pueda registrar

En nuestra app routing importamos las rutas donde nuestro app podra cargar las rutas

```

File Edit Selection View Go Run Terminal Help
app-routing.module.ts - JSM - Visual Studio Code
// src/app/modules/auth/login/login.component.ts
import { LoginComponent } from './login.component';
import { Router } from '@angular/router';

const routes: Routes = [
  // path "",
  // component: FullComponent,
  // children: [
  //   {
  //     path: "",
  //     redirectTo: "/login",
  //     pathMatch: "full",
  //   },
  //   {
  //     path: "home",
  //     component: DashboardComponent,
  //   },
  //   {
  //     path: "alerts",
  //     component: AlertsComponent,
  //   },
  //   {
  //     path: "forms",
  //     component: FormsComponent,
  //   },
  //   {
  //     path: "table",
  //     component: ProductComponent,
  //   },
  //   {
  //     path: "grid-list",
  //     component: DetailsComponent,
  //   },
  //   {
  //     path: "menu",
  //     component: MenuComponent,
  //   },
  //   {
  //     path: "tabs",
  //     component: TabsComponent,
  //   },
  //   {
  //     path: "expansion",
  //     component: ExpansionComponent,
  //   },
  //   {
  //     path: "chips",
  //     component: ChipsComponent,
  //   },
  //   {
  //     path: "progress",
  //     component: ProgressComponent,
  //   },
  //   {
  //     path: "toolbar",
  //     component: ToolbarComponent,
  //   },
  //   {
  //     path: "progress-snapper",
  //     component: ProgressSnapperComponent,
  //   },
  //   {
  //     path: "snackbar",
  //     component: SnackbarComponent,
  //   },
  //   {
  //     path: "slider",
  //     component: SliderComponent,
  //   },
  //   {
  //     path: "live-toggle",
  //     component: LiveToggleComponent,
  //   },
  //   {
  //     path: "tooltip",
  //     component: TooltipComponent,
  //   },
  //   {
  //     path: "button",
  //     component: ButtonComponent,
  //   },
  // ],
  // ],
  {
    path: "register",
    component: RegisterComponent,
  },
  {
    path: "login",
    component: LoginComponent,
  },
  {
    path: "",
    redirectTo: "/login",
    pathMatch: "full",
  },
  {
    path: "",
    redirectTo: "/login",
    pathMatch: "full",
  },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule],
})
export class AppRoutingModule {}

```

Al hacer este desafio primero se elimino la tabla para poder agregar los campos que se requirieron

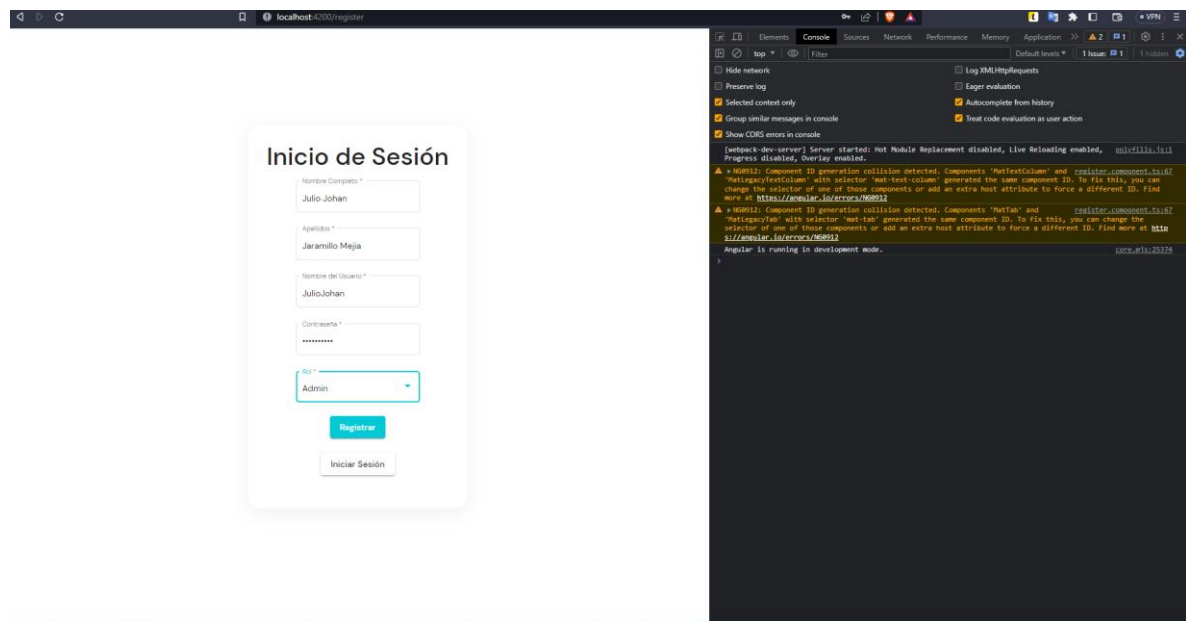
```
CREATE TABLE `usuario` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `name` varchar(100) NOT NULL,  
  `lastname` varchar(100) NOT NULL,  
  `username` varchar(50) NOT NULL,  
  `password` varchar(255) NOT NULL,  
  `rol` enum('admin','user') NOT NULL,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `username` (`username`)  
)
```

- Agregar más campos a la tabla tbl_usuario: nombre, apellidos.
- Una vez qué está logueado imprimir su nombre completo

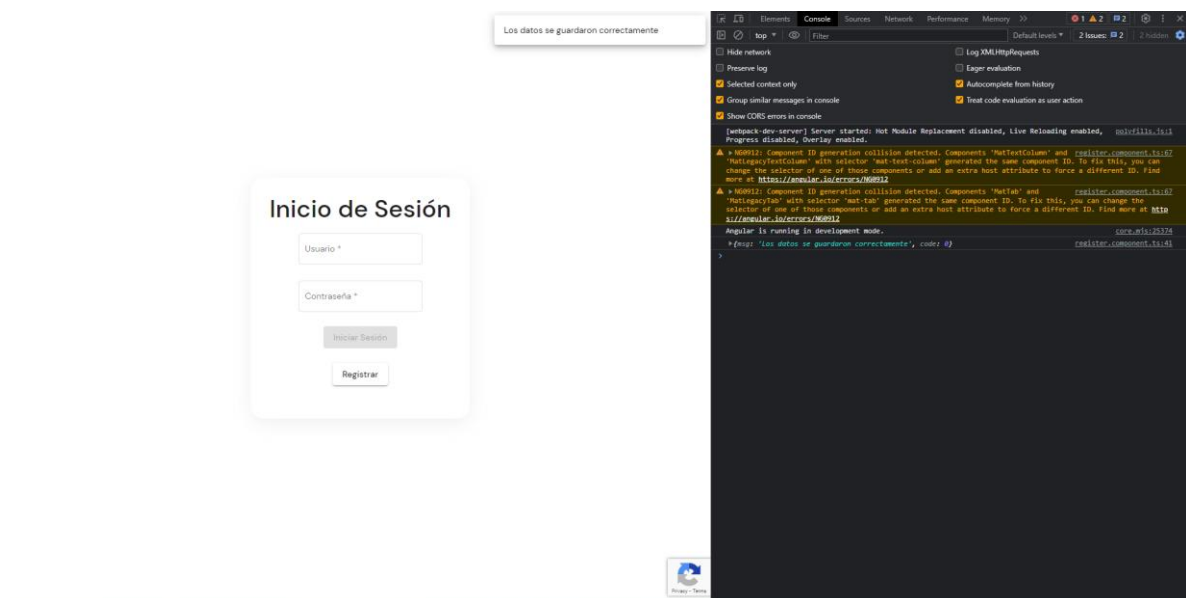
Pues al finalizar por el momento se agregó en el endpoint registro donde se esta construyendo en new user donde se guardará la información del usuario

```
public async insertar(req: Request, res: Response) {  
  try {  
    // se obtienen los datos del body  
    const usuario = req.body;  
    console.log(usuario);  
  
    // validar que los datos no sean nulos o indefinidos  
    if (!usuario.username || !usuario.password || !usuario.rol || !usuario.name || !usuario.lastname) {  
      return res.status(500).json({ msg: "Todos los datos son requeridos", code: 1 });  
    }  
  
    const usuarioDB: any[] = await model.getUserByUsername(usuario.username);  
    console.log(usuarioDB);  
  
    if (usuarioDB.length > 1) {  
      return res.status(400).json({ msg: "El nombre de usuario existe :(", code: 1 });  
    }  
  
    // encriptar nuestra contraseña  
    const encryptedText = await utils.hashPassword(usuario.password);  
    usuario.password = encryptedText;  
    console.log("Contraseña encriptada " + typeof usuario.password);  
  
    const newUser = {  
      name: usuario.name,  
      lastname: usuario.lastname,  
      username: usuario.username,  
      password: usuario.password,  
      rol: usuario.rol  
    };  
  
    console.log(newUser);  
  
    // inserción de los datos  
    // Agregar enunciado  
    const result = await model.insertar(newUser);  
  
    if (result.affectedRows > 0) {  
      return res.json({ msg: "Los datos se guardaron correctamente", code: 0 });  
    } else {  
      return res.status(500).json({ msg: "Error al guardar los datos", code: 1 });  
    }  
  } catch (error) {  
    console.log(error);  
    return res.status(500).json({ msg: "Error al insertar el usuario", code: 1 });  
  }  
}
```

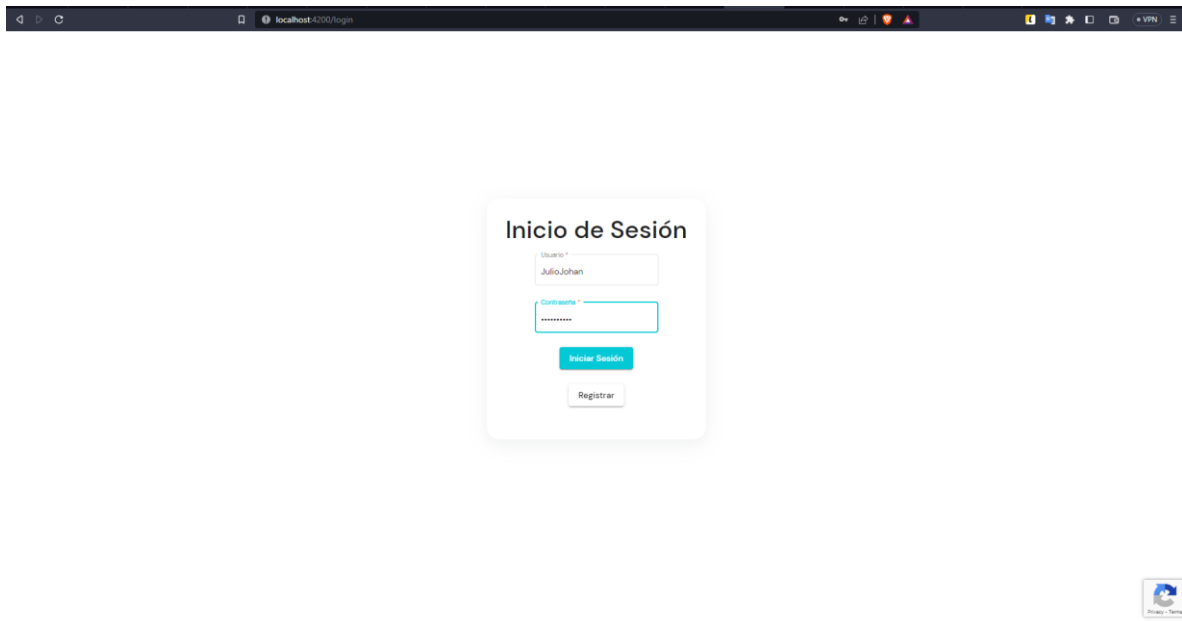
Regresando al Frontend agregamos nuestra información para registrar



Se dirige a la pagina de login para autenticarse



Iniciamos sesión



The screenshot shows a web browser window with the address bar displaying 'localhost:4200/login'. The main content area features a centered login form titled 'Inicio de Sesión'. The form has two input fields: 'Usuario' with the value 'JulioJohan' and 'Contraseña' with masked characters '*****'. Below the password field are two buttons: 'Iniciar Sesión' (highlighted in blue) and 'Registrar'. A small 'Recursos' icon is visible in the bottom right corner of the browser window.

Al inicio de sesión en la parte derecha se muestra nuestro nombre y apellidos

