



Estado de login 21/25



RECURSOS

APUNTES

La lógica de los Guards suelen necesitar de los datos de un usuario para determinar si tiene acceso o no a una ruta. Veamos como es posible obtener los datos del usuario utilizando **Observables** y **RxJS**.

Qué es el estado de la aplicación

Se lo conoce como “*estado*” al valor de los datos que la aplicación utiliza en un momento dado. A medida que el usuario interactúa con la app, el estado cambia.

Lectura recomendada: [Reactividad básica](#)

¿Cómo conocer el estado del usuario?

Una buena manera de conocer el estado del usuario en todo momento es creando un Observable que notifique a los interesados si hay algún cambio en los datos del mismo.

1. Observable para el estado del usuario

En el servicio de autenticación, crea el Observable que contendrá los datos del usuario para que los interesados puedan suscribirse y escuchar los cambios.

```
// modules/shared/services/auth.service.ts
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';

export interface User {
  email: string;
  name: string;
  role: string;
}

@Injectable({
  providedIn: 'root'
})
export class AuthService {

  private userData: User | null = null;
  private userSubject = new BehaviorSubject<User | null>(null);
  public user$ = this.userSubject.asObservable();

  setUser(u: User): void {
    this.userData = u;
    this.userSubject.next(this.userData);
  }

}
```

2. Guardando datos del usuario

Luego de un login exitoso, guarda el usuario en el Observable.

```
this.auth.setUser({  
  email: 'prueba@platzi.com',  
  name: 'Platzi',  
  role: 'admin'  
});
```

3. Acceder a los datos

De esta manera, podrás acceder a los datos del usuario en los Guards y permitir el acceso o no al mismo.

```
// modules/shared/guards/admin.guard.ts  
import { AuthService } from '../services/auth.service';  
  
@Injectable({  
  providedIn: 'root'  
})  
export class AdminGuard implements CanActivate {  
  
  constructor(  
    private router: Router,  
    private auth: AuthService  
  ) {}  
  
  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot):  
    Observable<boolean> | Promise<boolean> | boolean {  
  
    return this.auth.user$  
      .pipe(  
        map(user => {
```










```
        if (user)
            return true;
        else {
            this.router.navigate(['/home']);
            return false;
        }
    })
};

}
```

De esta manera, los Guards se convierten en asíncronos, que suele ser lo más apropiado para validar tokens y permisos de los usuarios para el ingreso o no a una ruta.

Contribución creada por: Kevin Fiorentino.

Archivos de la clase

 /	
 tsconfig.app.json	
 package.json	
 angular.json	
 .eslintrc.json	

 package-lock.json




 .browserslistrc




 .gitignore



 tsconfig.spec.json



 tsconfig.json



 netlify.toml




 README.md



 .editorconfig



 karma.conf.js



 src

Lecturas recomendadas

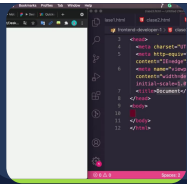


Reactividad básica

<https://platzi.com/clases/2486-angular-componentes/41217-reactividad-basica/>



Clases relacionadas



Login



Curso Práctico de Frontend Developer