



## Login y manejo de Auth 16/23



### RECURSOS

### APUNTES

Muchas aplicaciones hoy en día requieren de un *Login* para identificar al usuario y que pueda interactuar con la plataforma.

## Login de Usuarios

Veamos a continuación buenas prácticas para el manejo de sesiones con Angular.

### 1. Interfaces para tipado de datos

Comencemos creando algunas interfaces para conocer y tipar la estructura de los datos.

```
// interfaces/user.interface.ts
export interface Credentials {
  email: string;
  password: string;
}
export interface Login {
  access_token: string;
}
export interface User {
  id: string;
  email: string;
  password: string;
```

```
name: string;  
}
```

Hemos creado tres interfaces. \*

- **Credentials** para tomar los datos necesarios para registrar a un usuario
- **Login** que contiene la respuesta al registrar exitosamente un usuario
- **User** la interfaz que contiene la estructura de datos de un usuario

## 2. Servicio para el manejo del Login

Crea un simple servicio que manejará todo lo relacionado con inicios de sesión de los usuarios.

```
// services/auth.service.ts  
import { Credentials, Login, User } from 'src/app/interfaces/user.interface';  
  
@Injectable({  
  providedIn: 'root'  
})  
export class AuthService {  
  
  constructor(private http: HttpClient) { }  
  
  loginUser(credentials: Credentials): Observable<Login> {  
    return this.http.post<Login>(`https://example.com/api/login`, credentials);  
  }  
}
```

### 3. Inicio de sesión desde el componente

Desde el componente que contiene el formulario de login, realizamos el inicio de sesión del usuario.

```
// components/login/login.components.ts
import { Component } from '@angular/core';
import { AuthService } from 'src/app/services/auth.service';
import { Credentials, Login, User } from 'src/app/interfaces/user.interface';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.scss']
})
export class LoginComponent {

  constructor(private auth: AuthService) { }







  loginUser(): void {
    const credentials: Credentials = {
      email: 'user@gmail.com',
      password: '123456',
    };
    this.auth.loginUser(credentials)
      .subscribe((res: Login) => {
        localStorage.setItem('platzi_token', res.access_token);
      });
  }
}
```


En este ejemplo, efectuamos el *login* de un usuario con su *email* y *password*. El backend suele devolver el token de autenticación del usuario o un *access\_token*. Lo guardamos en **Local Storage** o en el *store* que sea de tu preferencia.

*NOTA: Existen diferentes tipos de Tokens para autenticar usuarios. Uno de los más comunes es [Json Web Token](#) o JWT.*


**Contribución creada por:** Kevin Fiorentino.

## Archivos de la clase

 /	
 tsconfig.app.json	
 package.json	
 angular.json	
 .eslintrc.json	
 package-lock.json	
 .browserslistrc	
 .gitignore	

 obs-promises.js




 tsconfig.spec.json



 proxy.config.json




 tsconfig.json




 README.md



 .editorconfig



 karma.conf.js



 src

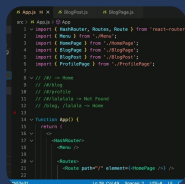
## Lecturas recomendadas



Insomnia Docs  
<https://docs.insomnia.rest/>



## Clases relacionadas



## useAuth: login y logout



Curso de React.js: Navegación con React Router