Uso de interceptores 18/23

RECURSOS APUNTES

Los **Interceptores** de Angular llegan para facilitar la manipulación de las peticiones que tu aplicación realiza.

Mi Primer Interceptor

Un interceptor, como su nombre indica, interceptará las solicitudes HTTP antes de que se envíen al servidor, para agregar información a las *request*, manipular datos, entre otras utilidades.

1. Crea el interceptor

Con el CLI de Angular, puedes crear un interceptor con el comando `ng generate interceptor <interceptro_name>` . En este ejemplo, generaremos un interceptor para manipular los errores HTTP en toda tu aplicación.

```
// interceptors/errors.interceptor.ts
import { Injectable } from '@angular/core';
import { HttpRequest, HttpHandler, HttpEvent, HttpInterceptor } from
'@angular/common/http';
import { Observable } from 'rxjs';

@Injectable()
export class ErrorsInterceptor implements HttpInterceptor {
```

```
  constructor() {}

  intercept(request: HttpRequest<unknown>, next: HttpHandler):
Observable<HttpEvent<unknown>> {
    return next.handle(request)
      );
  }
}
```

2. Inyección del interceptor

Lo interceptores son muy similares a los Servicios, pero se inyectan en el módulo de tu app de una forma diferente.

```
// app.module.ts
import { HTTP_INTERCEPTORS } from '@angular/common/http';
import { ErrorsInterceptor } from './interceptors/errors.interceptor';

@NgModule({
  // ...
  providers: [
    {
      provide: HTTP_INTERCEPTORS,
      useClass: ErrorsInterceptor
    }
  ]
})
export class AppModule { }
```

Importando primeramente `HTTP_INTERCEPTORS` desde `@angular/common/http`, puedes importar tu propio interceptor. Ahora, todas las solicitudes que salgan de tu aplicación, se interceptarán por el mismo.

Manejo de errores con un interceptor

Un buen uso para los interceptores es el manejo de errores. Como casi todo en Angular utiliza Observables, los interceptores no son la excepción y puedes apoyarte de **RxJS** para manipular la emisión de los datos del observable.

```
// interceptors/errors.interceptor.ts
intercept(request: HttpRequest<unknown>, next: HttpHandler):
Observable<HttpEvent<unknown>> {
  return next.handle(request)
    .pipe(
      catchError((error: HttpErrorResponse) => {

        if (error.status === 401)
          return throwError('No posee permisos suficientes.');
        else if (error.status === 403)
          return throwError('Acceso prohibido.');
        else if (error.status === 404)
          return throwError('Registro no encontrado.');
        else
          return throwError('Ha ocurrido un error inesperado.');

      }),
    );
}
```