## Creando un contexto a interceptor 20/23

RECURSOS APUNTES

Suele ocurrir que un front-end consume más de un backend diferente, con diferentes URLs, que eventualmente podrían utilizar diferentes tokens o necesitar cada uno de una manipulación diferente de la request por parte del interceptor.

### Manipulando múltiples tipos de solicitudes

Es posible crear un contexto para los interceptores para indicar si queremos que X endpoint sea interceptado o no.

#### 1. Importando clases necesarias

Para esto, importando `HttpContextToken` y `HttpContext` desde `@angular/common/http` vamos a crear una función para validar si el endpoint necesita o no la inyección de un token.

```
// interceptors/token-interceptor.interceptor.ts
import { HttpContextToken, HttpContext } from '@angular/common/http';

const ADD_TOKEN = new HttpContextToken<boolean>(() => true);

export function addToken() {
```

```
  return new HttpContext().set(ADD_TOKEN, false);
}
```

Creamos la constante **ADD_TOKEN** que indicará si el endpoint necesita de un token o no. Por defecto, todos los endpoints necesitan un token. La función `addToken()` hará que no se inyecte el token cuando no se requiere.
También puedes utilizar la lógica inversa, que ningún endpoint por defecto utilice el token y solo habilitar manualmente los que si lo requieren.

### 2. Creando el contexto para las solicitudes

En la lógica del interceptor, colocamos un **IF** para validar si la petición necesita o no del token.

```
// interceptors/token-interceptor.interceptor.ts
intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<unknown>> {
  if (request.context.get(ADD_TOKEN)) {
    request = this.addHeaders(request);
    return next.handle(request);
  } else
    return next.handle(request);
}
```

### 3. Denegar la utilización del token

Ahora, importamos dicha función y la utilizamos para que un endpoint no requiera del token.

```
// services/auth.service.ts
import { addToken } from '../interceptors/token-interceptor.interceptor';

@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, { context:
addToken() });
  }
}
```

Agregamos como argumento `context` a las opciones de la solicitud. El endpoint de Login no suele necesitar de un token para funcionar, es el único que excluimos del uso del interceptor.

---

***Contribución creada por:*** *Kevin Fiorentino.*

## Archivos de la clase

/

tsconfig.app.json

package.json