MEMORIA DE PRÁCTICAS

Seguridad de Aplicaciones

UDC Curso 22-23

Carlos Torres Paz
Samuel Santos Espido
Ismael Verde Costas

Contenido

Endpoint de autenticación 2

Configuración de autorización y control de acceso 2

Control de acceso con roles de usuario 2

Autenticación con OAuth2 2

Endpoint de autenticación

Originalmente en lugar de generarse un nuevo token JWT solo se comprobaba si el usuario autenticado era el user1 y se le asignaba el mismo token siempre, en lugar de generar uno nuevo por cada acceso. Las medidas tomadas fueron eliminar esas líneas de código y sustituirlas por la recuperación de los datos del usuario que se intenta autenticar, para después usarlos para generar un token JWT nuevo, el cual devolveremos como respuesta de la petición.

Configuración de autorización y control de acceso

La configuración de este apartado la haremos toda desde la clase WebSecurityConfig, para empezar añadiremos como política de sesión *STATELESS* y añadiremos el filtro JwtAuthorizationFilter para gestionar el uso de los tokens JWT. Luego usaremos el apartado authorizeRequests para tener un listado de los endpoints y con que métodos HTTP los permitiremos. Deberemos tener en cuenta los archivos estáticos, la página de swagger, la página principal de la aplicación, las llamadas a la API y denegaremos todas las demás, ya que así si no hemos tenido en cuenta algún recurso no se podrá acceder igualmente. En el caso de los recursos estáticos los añadiremos en el *configure* del *WebSecurity* para que se permita recuperar dichos archivos que van a ser necesarios siempre, y que no sea necesario que pasen por la lista de endpoints permitidos.

Control de acceso con roles de usuario

Para hacer el uso del sistema de roles tendremos que modificar la lista de endpoints del WebSecurityConfig hecha en el apartado anterior. Para varios endpoints sustituiremos el *permitAll* por un *hasRole* que contendrá el parámetro *USER* o *ADMIN* según requiera el nivel de autorización del endpoint en cuestión. De este modo se comprobará el rol en el token JWT para permitir el acceso a ese endpoint, en caso de que no coincida se le denegará.

La siguiente cuestión que tenemos que abordar es comprobar que solo el propietario del recurso pueda modificarlo o eliminarlo según sea necesario. Para ello tendremos que modificar el propio código de cada función para que reciba como parámetro el usuario que está realizando la petición, esto lo obtenemos del objeto Principal que contiene los datos del usuario autenticado. Una vez sabemos qué usuario intenta lanzar dicha función, buscaremos quien es el propietario del proyecto o tarea que se intenta modificar o eliminar y comprobaremos que coincidan. Repetiremos este proceso para los distintos casos de uso que así lo requieran.

Autenticación con OAuth2

Para poder autenticarnos con OAuth2 haremos uso del flujo *implicit*, el cual comienza redirigiendo al cliente a */oauth-server/oauth/authorize*. En esta petición especificaremos que el tipo de respuesta es un token, el identificador del cliente como *tasks_app* y la url a la que queremos que nos redirija al finalizar el proceso de autenticación que es */loginOAuth*. De este modo podremos autenticarnos en el servidor de OAuth y al volver a la aplicación esta recibirá en el hash de la URI un token JWT que nos permitirá usar la aplicación.