Universidad ORT Uruguay

Facultad de ingeniería

Diseño de aplicaciones 2
Obligatorio 1

https://github.com/ORT-DA2/SobralGoni

210361 - José Pablo Goñi
192247 - Juan Pablo Sobral

Criterios seguidos 3

Mecanismo de autenticación 4

Códigos de error 5

Descripción de los recursos 6

Criterios seguidos

Antes de comenzar a diseñar la API nos tomamos el tiempo para leer sobre cómo crear APIs REST que fuesen de buena calidad. Tomamos como referencia la documentación de Microsoft sobre buenas prácticas a la hora de diseñar APIs y el ebook **Web Api Design** de **apigee.**

Siguiendo las sugerencias que encontramos en ambos documentos, decidimos organizar nuestra API en torno a los recursos que iba a manejar, en este caso *Usuarios, Archivos y Carpetas*.

Como resultado surgieron los siguientes endpoints:

- http://localhost:$puerto/api/users
- http://localhost:$puerto/api/files
- http://localhost:$puerto/api/folders

En cuanto a las operaciones implementadas, nos limitamos a cuatro de los verbos disponibles: *GET, POST, PUT y DELETE*.

Nuestras operaciones GET retornan código 200 y una o varias entidades del sistema, 404 en caso de que no exista la entidad buscada y 501 en caso de que el usuario no esté autorizado a acceder a la misma.

Nuestras operaciones POST retornan código 201 y la entidad creada en caso de ser válida la operación, en otro caso (por ejemplo creación de un usuario con un username repetido) se envía un 400 que contiene el mensaje de error producido por el sistema en formato de excepción, y en caso de que sea un usuario no administrador se retorna un 501.

Nuestras operaciones PUT se utilizan únicamente para la actualización de entidades, por lo cual retornan 200 cuando se actualiza una entidad y 404 cuando la entidad no existe.

Nuestras operaciones DELETE retornan un 200 cuando se borra una entidad, 501 en caso de que el usuario autenticado no sea administrador y 404 si la entidad que se desea borrar no existe en el sistema.

Mecanismo de autenticación

Para el mecanismo de autenticación, decidimos utilizar bearer tokens de tipo JWT, lo cual permite que el sistema identifique qué usuario es el que está intentando operar sobre la API tan solo manejando ese token que va en el header de las request.

Este token se emite cuando el usuario se autentica contra la aplicación y tiene una validez de 15 minutos, lo cual lleva a que el token válido esté almacenado en memoria en el servidor web.

Decidimos implementar dos roles(User y Admin), aunque a futuro se puede ampliar a más en caso de ser necesario sin ningún tipo de problema.

El funcionamiento de este mecanismo es el siguiente:

- Por defecto todos los endpoint de la aplicación están protegidos, es decir, no pueden ser accedidos si el usuario que intenta acceder no se identifica mediante un token.
- Hay una excepción a la regla anterior: el endpoint de autenticación. Para obtener el token, los usuarios deben postear su username y password en el body de una request POST en el endpoint */users/authenticate*. En caso de ser válidas las credenciales, se les provee de un token en la respuesta de su request, el cual usará para identificarse en las request que le sigan.
- Más allá de eso, hay varias validaciones internas que la aplicación lleva a cabo cuando un usuario intenta acceder a un endpoint. Se pueden resumir de la siguiente manera: un usuario puede acceder a toda su información, todos sus archivos y todas sus carpetas, además de las carpetas y/o archivos que le fueron compartidos. Un administrador, puede acceder a toda su información como un usuario cualquiera, pero además tiene permisos para acceder al contenido de todos los archivos y carpetas del sistema, al igual que potestad para crear, actualizar y borrar usuarios y archivos ajenos.

Códigos de error

Como ya se mencionó anteriormente, no utilizamos demasiados códigos en nuestra API porque no lo vimos como necesario. Los utilizados son los siguientes:

- **200** para requests correctas, por ejemplo cuando un usuario se autentica y se le retorna su bearer token.
- **201** para requests correctas que crean una entidad, por ejemplo cuando un administrador crea un usuario, se le retorna el código 201 y la info del usuario.
- **400** para requests incorrectas, por ejemplo si intento de crear un usuario con un username ya existente en mi sistema.
- **404** para requests que no tienen respuesta, por ejemplo si tengo 10 usuarios en mi sistema y un admin hace una request GET al endpoint /users/30.
- **500** para errores genéricos del servidor.
- **401** para operaciones no autorizadas, por ejemplo si un usuario común intenta de crear otro usuario.

Descripción de los recursos

La documentación de la API se encuentra según el estándar de OpenAPI, en el siguiente sitio web accesible públicamente: https://app.swaggerhub.com/apis-docs/juanpaSobral/TwoDrive-APIGoniSobral/1

Por otro lado, el YAML que esa página utiliza como base se encuentra disponible en el repositorio.