Universidad ORT Uruguay

Facultad de Ingeniería

Bernard Wand Polak

Obligatorio

Diseño de aplicaciones 2

Integrantes

- Juan Pablo Sobral - 192247
- Chiara Di Marco - 186017

Links

- GitHub: https://github.com/ORT-DA2/BetterCalm-DiMarco-Sobral

Docentes

- Ignacio Valle
- Gabriel Piffaretti
- Nicolás Fierro

Criterios REST

Nomenclatura de recursos y uso de verbos

A la hora de dar nombre a los endpoints que permitiesen acceso a recursos de la aplicación, tomamos en cuenta los criterios discutidos en clase y leídos en las guías de diseño de APIs de Apigee.

Siguiendo las sugerencias previamente mencionadas, surgen las siguientes URIs con el objetivo de permitir a los usuarios interactuar con los mismos de forma intuitiva:

- /api/administrators
- /api/categories
- /api/consultations
- /api/playablecontents
- /api/playlists
- /api/psychologists

Como es de esperarse, se pueden crear nuevas entidades del tipo correspondiente enviando un POST a esas URIs y obtener todas con un GET. Si queremos acceder a una en particular, simplemente agregamos /{id} y enviamos un GET, o un DELETE para borrar, o un PUT a la hora de actualizar algún dato de la entidad.

Para ilustrar mejor, supongamos que soy un administrador que quiere dar de alta a un psicólogo , chequear que haya sido dado de alta, actualizar su nombre y finalmente borrarlo.

Las operaciones a realizar serían las siguientes:

1. POST a /api/psychologists con la información del psicólogo en el cuerpo.
2. GET a /api/psychologists/{id}, siendo este el obtenido en la respuesta de la consulta anterior.
3. PUT a /api/psychologists/{id} con la información actualizada del psicólogo en el cuerpo.
4. DELETE a /api/psychologists/{id}

Este flujo resulta intuitivo a usuarios que ya hayan interactuado con otras APIs RESTful, lo cual es un atributo logrado mediante el apego a las sugerencias propuestas. Otra cosa a notar es el hecho de que no están implementadas todas las operaciones que un usuario podría esperar poder realizar sobre un recurso. Esto es por diseño, dado que no fue requerido como parte de la solución, pero se la implementó de tal forma que si el día de mañana se quiere agregar esas funcionalidades faltantes, será sencillo e intuitivo para el usuario acostumbrarse al cambio.

Nomenclatura de consultas

Otro punto en el que se implementaron las sugerencias, fue en el de consultas más específicas.

Supongamos que como administrador quiero autenticarme. No quiero enviar la información de autenticación en la URL de mi consulta, y esto lo resolvemos permitiendo esa operación mediante un POST a /api/administrators/login con la información en el cuerpo de la consulta.

Como usuario puedo querer acceder directamente a los contenidos de una playlist una vez que la descubrí y se que me gusta. Es por eso que implementamos un endpoint particular, /api/playlists/{id}/contents, para evitar el envío de información innecesaria.

Otro ejemplo puede ser a la hora de agregar una canción a una playlist, operación que se lleva a cabo enviando un POST a /api/playlists/{id}/contents?contentId={contentId}. No hay problemas con que la información de los Ids viaje en la URL, y es la forma más utilizada de parametrizar una operación de este estilo.

Mecanismo de autenticación

Como mecanismo de autenticación nos decantamos por el uso de JWT, dado que es una tecnología relativamente madura y con bastante adopción dentro de la industria. Para implementarlo nos apoyamos en librerías de Microsoft y de sistema como IdentityModel y Security.

Uno de los motivos por el que decidimos utilizar este mecanismo es el grado de información que puede llegar a manejar el token. Más allá de que en esta iteración el único interés es saber si el usuario está autenticado o no (lo cual implica que sea administrador), es fácil imaginarse que a futuro tal vez los usuarios normales quieran autenticarse, en cuyo caso habría que distinguir entre los dos. Además, si los usuarios pueden autenticarse seguramente tengan posesión de ciertas entidades en nuestro sistema, a las cuales solo deberían poder acceder ellos o administradores.

El manejo de Claims en JWT nos permite almacenar la información necesaria para tener ese grado de control y su implementación utilizando librerías es relativamente sencilla, presentando potencialmente mucha ganancia a bajo costo.

Uso de códigos HTTP

Intentamos apegarnos lo más posible al propósito original que se le dió a cada código de forma que el usuario se familiarice rápidamente con la API.

- 404
    - Se utiliza cuando el usuario intenta de acceder a un recurso inexistente o que fue borrado.
- 200
    - Se utiliza como respuesta a una operación exitosa como puede ser un GET o un PUT, y en alguna ocasión con POST como el de agregar un contenido a una playlist.

- 201
    - Se utiliza como respuesta cuando un usuario crea de manera una entidad en el sistema.
- 202
    - Se utiliza como respuesta para la autenticación.
- 204
    - Se utiliza como respuesta a un borrado exitoso.
- 400
    - Se utiliza cuando el usuario envía información mal formada.
    - También se utiliza para enviar al usuario cuando se genera una excepción interna fruto de los datos enviados.
- 403
    - Se utiliza cuando un usuario no administrador intenta acceder a algun recurso al que no tiene permisos.
- 401
    - Se utiliza como respuesta cuando las credenciales provistas al autenticarse no son válidas.

Link a la documentación formato swagger

El yaml que generó el swagger se puede encontrar en el repositorio en la carpeta Documentacion y esta publicado en el siguiente link.

https://app.swaggerhub.com/apis-docs/juanpaSobral/BetterCalmAPI/1.1