Universidad ORT Uruguay

Diseño de Aplicaciones 2

Obligatorio 1

Link al repositorio:
https://github.com/IngSoft-DA2/278311_227286_266889.git

Docentes:
Nicolás Fierro
Alexander Wieler
Máximo Halty

Alumnos:
Nicolás Russo 227286
Matias Hirschfeld 266889
Victoria Chappuis 278311

DOCUMENTO: Especificación de la API

Especificación API

Criterios REST

Para la creación de nuestra API seguimos los criterios REST. Estos criterios, además de tener importancia por sí mismos, son el estándar de la industria.

Recursos Identificables

Cada recurso dentro de un sistema qué utilice REST debe tener una identificación única, es decir una URI. Cada URI debe ser única y consistente, por ejemplo, para acceder a la información de un usuario con ID 1 la URI podría ser /usuarios/1. La URI solo representa al recurso y es independiente al método. Entonces, utilizando el ejemplo anterior, si yo quisiera eliminar el usuario 1 la URI sería la misma, únicamente cambiando el método de GET a DELETE.

Interfaz Uniforme

REST utiliza un conjunto de operaciones estándar para interactuar con los recursos. Estos incluyen

- GET: Recuperar un recurso
- POST: Crear un recurso nuevo
- DELETE: Eliminar un recurso
- PATCH: Modificar parcialmente un recurso
- PUT: Modificar totalmente un recurso

Stateless

Todas las peticiones del cliente al servidor deben tener toda la información necesaria para poder realizar la petición. El servidor no guarda información sobre el estado del cliente entre peticiones. Es decir, si un Usuario se registra en una petición, en la siguiente petición debe enviar datos qué le permitan ingresar nuevamente.

Cacheable

Las respuestas a las solicitudes REST pueden ser cacheadas para mejorar la eficiencia y reducir la carga del servidor. Las respuestas a ciertas solicitudes, especialmente las que usan el método GET, pueden almacenarse en caché, permitiendo que los clientes reutilicen los datos sin necesidad de volver a consultar el servidor.

Sistema de capas

Un sistema REST puede estar compuesto por capas intermedias entre el cliente y el servidor, como servidores proxy o gateways, lo que permite escalar y mejorar la seguridad.

Paginación

En esta implementación usamos los parámetros skip y take para controlar cuántos registros omitimos y cuántos devolvemos. Skip indica el número de registros a saltar desde el inicio de la colección, mientras que take define la cantidad de registros que se deben tomar después del salto. Esto permite manejar grandes conjuntos de datos dividiéndolos en partes más pequeñas optimizando la eficiencia de las consultas.

Además, aplicamos filtros del lado del servidor. Estos se aplican antes de la paginación garantizando que solo los registros relevantes se incluyan en la operación de skip y take, lo que reduce la carga en la base de datos y mejora el rendimiento.

Esta lógica de paginación está del lado de los servicios y no en los controladores porque los servicios son responsables de la lógica de negocio. En este caso eso incluye las operaciones de filtrado y paginación. Los controladores deben enfocarse en recibir las solicitudes del cliente y entregar las respuestas. Centralizar esta lógica en los servicios mejora la separación de responsabilidades y asegura que el código sea más modular.

Ventajas

La utilización de estos criterios tiene varias ventajas. Por ejemplo, al proveer una separación clara entre cliente y servidor, REST permite a los sistemas qué lo utilicen una mayor escalabilidad, como permiten qué el servidor y el cliente evolucionen de manera independiente, sin qué uno dependa del otro.

También, al utilizar el protocolo HTTP como protocolo base, qué es un protocolo ya conocido, provocando qué los desarrolladores no necesiten aprender un nuevo protocolo,

REST es fácil de implementar y simple de utilizar. Esto produce una interfaz predecible, cómo se utilizan los métodos de HTTP (GET, POST, PUT, DELETE, PATCH).

Además, REST es independiente a las plataformas o lenguajes qué lo utilicen. Para qué un sistema pueda implementar REST, basta con qué sea capaz de hacer solicitudes HTTP. Esto hace que REST sea una solución versátil y flexible para conectar diferentes tipos de tecnologías y plataformas, facilitando la integración entre sistemas.

Mecanismo de autenticación de los requests

Para la autenticación de las request utilizamos un mecanismo de autenticación por token. Para esto, el usuario deberá iniciar sesión en un endpoint qué creamos, denominado auth. En este endpoint, el usuario genera un token qué tiene cierta duración, en nuestro caso de una hora. Este token es único e identifica al usuario, permitiéndole acceder a los recursos qué precisan autenticación, poniendo dicho token en el header de la request http.
Todas las sesiones son almacenadas en la base de datos en la tabla Sessions. Es el mismo servicio qué se encarga de crear y validar los tokens, como también de eliminar las sesiones qué ya hayan caducado.

Códigos de error utilizados

200 (OK): Usado cuando una operación se realiza exitosamente, como asociar un dispositivo a un hogar o crear una empresa.
401 (Unauthorized): Usado cuando el usuario no está autenticado o no tiene permisos para realizar la acción, como intentar asociar un dispositivo sin un token válido.
400 (BadRequest): Usado cuando los datos enviados son inválidos o no existen, como intentar asociar un dispositivo inexistente o crear una empresa con datos incorrectos.
415 (UnsupportedMediaType): Usado cuando la solicitud no tiene un body o el formato no es el adecuado, como al asociar un dispositivo sin body.
204 (No content): Usado cuando un administrador elimina correctamente a otro usuario, sin contenido en la respuesta.

Descripción de los resources de la api

/api/homes: Gestiona la creación y administración de hogares, dispositivos asociados, miembros, y notificaciones, permitiendo a los usuarios autorizados (como propietarios de hogares) agregar miembros, dispositivos, y configurar notificaciones para el hogar.

/api/users: Maneja la creación, autenticación, y administración de usuarios, incluyendo roles como administradores.

/api/auth: Gestiona la autenticación de usuarios, permitiendo el inicio de sesión y la verificación de credenciales.

/api/companies: Gestiona la creación y consulta de empresas, permitiendo a los propietarios de empresas agregar nuevas compañías y a los administradores consultar las empresas existentes con filtros como nombre de la empresa o nombre del propietario.

/api/devices: Gestiona los dispositivos inteligentes, permitiendo consultar dispositivos existentes con filtros como nombre, modelo o tipo, listar los tipos de dispositivos, y permitir a los propietarios de empresas agregar nuevos dispositivos.