Trabajo práctico 2

Introducción

Habiendo comprendido la creación de un backend api restful utilizando NodeJS y Express y logrando la conexión a una base de datos de MongoDB, continuamos con el trabajo práctico integrador de backend.

Alcance

El objetivo es realizar un backend que sería el encargado de proporcionar a un sitio web de películas o series **(no es necesario desarrollar frontend)** o puede ser también animé. Es meramente para dar un contexto a nuestra aplicación.
En los puntos asumiré que van a crear una backend para una web de animé, pero es válido para cualquiera de las opciones anteriores (películas o series).

Definición de entidades de la base de datos (es decir, que *Collections* deben existir y que datos almacenará cada C*ollection*).

Modelo de datos a crear

Users: Debe contener email, password, registerDate, un listado de id's de animés FAVORITOS.
Animes: titulo, descripción, url de imagen de portada, categoría, un listado de ID's de los capítulos que contiene cada animé.
Capítulos: id, titulo, descripción, url video del capítulo.

Se detallan cada uno de los servicios a crear y sus funcionalidades mínimas.
La nomenclatura de request y responses de los servicios queda a criterio del alumno.

Endpoints para la Entidad Users:

Login: Endpoint para loguearse utilizando email y password.
Registro: Endpoint para registrarse enviando email y password.
Agregar favorito: Endpoint que reciba el ID del usuario y el ID del animé a agregar al listado de favoritos.

Endpoints para la Entidad ANIME:

GET – Obtener listado animés

Este endpoint debe retornar todos los animes guardados en la base de datos.
Sólo debe retornar el id, el título, una descripción y una URL a una imagen de portada.

GET – Obtener el detalle de un animé

Este endpoint, recibirá el id de un animé y retornará los campos: id, titulo, descripción, URL imagen de portada y una LISTA de todos los capítulos que tiene este animé.

DELETE: Eliminar un Animé por ID: Este endpoint eliminar un animé por ID

PUT o PATCH: Actualizar un Anime: Actualiza un Animé por ID para todos sus campos excepto el campo ID.

POST Crear animé: Agregar un nuevo animé a la base de datos, en este endpoint no se insertan capítulos, simplemente se recibirán los datos: titulo, descripción, url de imagen de portada, categoría.

Endpoints para la Entidad Capitulos:

GET – Obtener listado capitulos para un anime: Este endpoint recibe el ID del animé del que se quieren obtener los capítulos.

POST - Crear capitulo para anime: Este endpoint, recibe **título, descripción, url video del capítulo** de un animé y lo inserta en la base de datos en la entidad para los capítulos correspondiente.

DELETE– Elimina un capítulo de la base de datos: Este endpoint recibe el ID del animé del que se quieren obtener los capítulos.

PUT o PATCH – Actualiza un capítulo: Actualiza un registro de la entidad de la base de datos donde se guardan los capítulos.

Aclaraciones

- Se deben validar todos los campos como requeridos tanto en las inserciones como en las actualizaciones.
- Se valora la nomenclatura en el código y en la llamada a servicios (url, parámetros y respuestas) en inglés.
- Se valorará la correcta identación y prolijidad del código.
- Se valorará el correcto uso de path params, body params, query params según corresponde para cada caso según lo visto en las clases.
- La clave de usuario debe estar encriptada tal cual vimos en las clases.
- El punto resaltado en AMARILLO es OPCIONAL, pero se valorará su entrega.
- Se debe utilizar un archivo .end utilizando dotenv para variables de entorno, como por ejemplo, el puerto, la conexión a la base de datos y la clave secreta para generar el JWT.
- Se valorará el correcto uso de los status code según corresponda para los diferentes tipos de éxito o error.

- El código debe ser subido a un repositorio **PUBLICO** de GIT para su entrega y en el archivo Readme deben figurar los miembros (hasta 4 máximo) que participaron en el proyecto.

Recomendaciones:

Utilicen todo el código que tienen disponible de cada clase, pueden utilizar el código de las clases como base para empezar, lo que resuelve la estructura de archivos y carpetas básicas sobre la cual pueden agregar la lógica del ejercicio.

¡Éxitos!