Tercer proyecto integrador

Para finalizar la **Etapa 2** del curso de programación Backend de Argentina programa, deberás desarrollar una aplicación CRUD integrando MongoDB en tu proyecto Node.js.

El proyecto final debe incluir:

1. Desarrollo de una bb.dd MySQL con el modelo relacional aplicado de acuerdo a la información almacenada en trailerflix.json
    a. **catalogo** (almacenar datos de películas y series)
    b. **categorias** (serie, película, documental, etc...)
    c. **actricesyactores** (datos de reparto)
    d. **genero** (Sci-Fi, Acción, Drama, Terror, Comedia)
    e. ***N* tablas intermedias** (catalogo_reparto, catalogo_tags, etc...)

Para el desarrollo de este punto, puedes aprovechar la información brindada en la **Presentación - Desarrollar un modelo relacional**. La herramienta allí sugerida te ayudará a crear el modelo relacional de la bb.dd. y las diferentes tablas, y luego exportar el código a .SQL para poder crear la bb.dd., tablas y relaciones de forma mucho más fácil.

Además tienes TIPs de cómo implementar SCRIPTS de inserción de datos en las diferentes tablas, aplicando las herramientas de Inteligencia Artificial que tenemos disponibles a mano.

2. Crear una vista SQL que muestre todos los datos, tal como se presentan en el archivo .json original.

3. Crea un API REST que acceda a los datos de trailerflix
    a. servidor Express JS
    b. conexión con MySQL
    c. genera al menos 5 endpoints (puedes crear más si quieres)
    d. **/categorias** (servirá información de todas las categorías existentes)
    e. **/catalogo** (servirá el catálogo completo 'la vista SQL')
        i. **/catalogo/:id** (filtrar por código de la película/serie)
        ii. **/catalogo/nombre/:nombre** (filtrar por nombre o parte del nombre)
        iii. **/catalogo/genero/:genero** (filtrar por género del contenido)
        iv. **/catalogo/categoria/:categoria** (filtrar por serie - película o cualquier otra categoría que pueda existir)
    f. y otros endpoint que consideres interesante crear…

Consideraciones

Cuando se sirve la información del endpoint **/catalogo**, la información del campo **'poster'** donde se muestra la ruta a una hipotética imagen, debe enviar en la respuesta la ruta absoluta a la imagen, dependiendo por supuesto de cuál es la ruta del endpoint al momento de ejecutarse.

4. Archivo .sql de creación de la base de datos. Debe incluir la creación de las tablas y la carga de las entidades que figuran en el archivo trailerflix.json.
5. Documentación en formato MarkDown que explique cómo utilizar cada endpoint creado, que muestre un ejemplo de código, y el listado de endpoints en una tabla.

Rúbricas

Para aprobar el proyecto, considerar:

A. **aprobación perfecta:**

a. La bb.dd. creada con al menos 6 tablas correctamente relacionadas
b. La aplicación backend funcional con al menos 5 endpoints sugeridos que sirven datos de la bb.dd. solicitada
c. El cuidado necesario para responder a una petición que solicite datos erróneos
d. La documentación en formato Markdown asociada al proyecto donde se explica el mismo y representa al menos en una tabla, cómo utilizar cada endpoint y qué información retorna el mismo

B. **aprobación parcial 1:**

a. La bb.dd. creada con al menos 6 tablas correctamente relacionadas
b. La aplicación backend funcional con al menos 5 endpoints sugeridos que sirven datos de la bb.dd. solicitada
c. El cuidado necesario para responder a una petición que solicite datos erróneos

C. **aprobación parcial 2:**

a. La bb.dd. creada con al menos 6 tablas correctamente relacionadas
b. La aplicación backend funcional con al menos 5 endpoints sugeridos que sirven datos de la bb.dd. solicitada

D. **desaprobación / re-entregar:**

a. Si obvia alguno de los puntos (a ó b) listados en (A - aprobación perfecta) y envía o no en su lugar el punto (d) listado.

No es necesario integrar JWT en el proceso de CRUD, por una cuestión de practicidad al momento de corregir.