Programación Web Full Stack

Manipulación de datos

Práctica integradora

Objetivo

En la práctica anterior vimos cómo conectar un proyecto de Node.js con Express a una base de datos y cómo podíamos listar y mostrar la información que teníamos almacenada en dicha base de datos. Ahora, practicaremos el resto de las operaciones para manipular datos con el ORM Sequelize.

¡Buena suerte! 😎👍✨

📌 Micro desafío - Paso 1:

Utilizando de base el ejercicio de la clase anterior, o en su defecto podemos usar el siguiente **proyecto creado con Express** (no hay que olvidarse de correr npm install, y si usamos el proyecto de la clase anterior, recordemos que debemos agregar url encoded, para que nos pueda llegar la información desde el formulario al req.body 😉), con la base de datos **movies_db**, debemos efectuar lo siguiente:

Para la construcción de esta versión del sitio web, el cliente espera contar con la posibilidad de acceso a las siguientes URLs:

- /movies/add (GET)
  - Muestra el formulario para la creación de una película.
- /movies/create (POST)
  - Recibe los datos del formulario anterior y escribe la información en la base de datos.
  - El controlador deberá utilizar la conexión a la base de datos y el modelo de Película ya creado. Luego, el método **create** permitirá la creación del registro. Finalmente, el controlador debe redirigir al listado de películas.
  - Idealmente, la información recibida debe estar validada con **express-validator**, pero por no tener tanto tiempo es algo que podemos hacer como un bonus en casa para seguir practicando.

Es importante que antes de comenzar la práctica verifiquemos y analicemos qué tenemos listo en el proyecto que nos entregan y qué es lo que nos están solicitando. 😉

Micro desafío - Paso 2:

Actualizar una película indicada en la URL según el ID:

- /movies/edit/:id (GET)
    - Muestra un formulario ya completo con los datos de la película según el id que indica la URL.
    - Previamente debe agregarse en el detalle de la película un botón de editar que envíe a esta URL. El controlador recupera el id de la URL mediante **req.params**.
- /movies/update/:id (POST) --- **Idealmente PUT**
    - Recibe información del formulario mencionado anteriormente y en conjunto con el id que indica la URL actualiza la información de la película.
    - Recibe información del formulario mencionado anteriormente recuperable mediante **req.body**. El controlador deberá utilizar la conexión a la base de datos y el modelo de Película ya creado. Mediante el método **update** se modifica la información de la base de datos. Luego, el controlador debe redirigir al listado de películas.
    - Idealmente, la información recibida debe estar validada con **express-validator**, pero por no tener tanto tiempo es algo que podemos hacer como un bonus en casa para seguir practicando.
    - Utilizar los paquetes y modificaciones necesarias para que el método que responda al pedido sea **PUT**.

Micro desafío - Paso 3:

Elimina la película indicada en la URL según el ID:

- /movies/delete/:id (POST) --- **Idealmente DELETE**
    - Previamente debe agregarse en el detalle de la película un botón de borrado que envíe a esta URL. El controlador recupera el id de la URL mediante **req.params**. El controlador deberá utilizar la conexión a la base de datos y el modelo de Película ya creado. Mediante el método **destroy** se elimina el registro y luego se redirige al listado de películas.
    - Probemos de eliminar varias películas, ¿funciona siempre bien?

Conclusión

Tenemos listo nuestro primer CRUD completo: creación de recursos, lectura de los mismos, actualización y borrado. Recomendamos armar una lista, con cada uno de los pasos que hemos realizado entre la práctica de la clase anterior y esta, para comprender cómo encajan cada una de las piezas a la hora de manipular datos de una base de datos relacional como lo es MySQL utilizando Sequelize.

¡Hasta la próxima!