DigitalHouse > Coding School

Desarrollo Web Full Stack Node

Ejercitación - M09C22

Mis películas - API

Introducción

Si bien es probable que ya tengas armado un backend que extraiga información de la base de datos de películas y las muestre en un bonito frontend, se acerca un nuevo cliente a la mesa...un cliente externo.

Este cliente está maravillado con tu sitio web y quiere utilizar toda esa valiosa información e integrarla en otros sitios y aplicaciones.

Dado esto, te requiere que armes una API para poder extraer la información de tu sistema

Requisitos

- **Paquete Sequelize instalado:** Para lograr el objetivo es fundamental lograr instalar Sequelize y los paquetes relacionados necesarios.
- **Conexión con base de datos:** El paquete sequelize no solamente precisa ser instalado sino que también precisa ser inicializado y configurado correctamente.
- **Modelos creados**: El paquete de Sequelize requiere la creación y configuración de modelos para esclarecer la forma de nuestra base de datos.

- **Ecosistema Express:** Naturalmente montaremos la aplicación en Express. Podés usar una que ya tenías o una nueva. Recomendamos utilizar una ya conocida para aprovechar todas las configuraciones a la base de datos que seguramente ya tengas escritas.
- **Postman:** Para probar la API usaremos Postman. Si no lo tenés instalado, es un gran momento.

Objetivo

Para la construcción de esta API del sitio web, el cliente espera contar con la posibilidad de acceso al recursos

- /api/movies

En una segunda instancia agregar acceso para los recursos:

- /api/genres
- /api/actors

Todos estos recursos se estipulan siguiendo el estándar **REST** que permitirá para cada recurso:

- Listar
- Ver detalles
- Crear
- Actualizar
- Borrar

Como adicional deseado el cliente pide:

- Incluir tests en postman
- Realizar un endpoint de login que envié un JWT. El resto de los endpoints deben estar protegidos y autenticados.

Consignas

1. Sobra la inicialización del proyecto

Recomendamos utilizar algún proyecto que ya cuente con un backend armado y tenga ya configurado todo lo que concierne a la base de datos (Dato: Si hiciste la ejercitación de Sequelize, es un GRAN punto de partida!)

Si aún no contas con un proyecto armado recomendamos crear uno con express-generator ¡No olvides aclarar que usaremos ejs como template engine!

Utilizaremos la base de datos **movies_db** y más aún, será importante instalar y configurar sequelize (Importante: Si no sabes como hacer esto, además de pedir ayuda, es una buena idea volver a la clase de Sequelize)

2. Rutas y Controllers

Te recomendamos que para cada uno de los recursos crees un archivo de ruta y uno controlador para organizar mejor la aplicación.

Empecemos por los archivos de ruta y controlador para **películas** pero ya pensando que más adelante se vendrán archivos para más recursos...

Es una buena idea crear una subcarpeta dentro de las rutas llamada **api** para agrupar todos los archivos que respondan a apis. Lo mismo aplica para los controladores.

3. Sobre los endpoints y cómo probarlos

El recurso **películas** tendrá 5 endpoints. Te invitamos a que pienses cuales serán siguiendo el estándar REST. Recordá que debes proveer endpoints para:

1. Listado
2. Detalle de un elemento
3. Creación
4. Actualización
5. Borrado

Para hacerte las cosas más fáciles, te recomendamos que tal vez empieces utilizando los métodos GET y POST de HTTP. Si bien es cierto que no es estándar, y no es la mejor manera de hacerlo puede ayudarte a empezar a hacer las cosas más rápidamente.

Cada uno de estos endpoints se verá en el archivo de rutas y tendrá un método controlador que lo represente. Si ya tenes un backend completo hecho con sequelize encontrarás que la lógica de cada método ya la tengas escrita en otros lados de tu sitio ahora… ¿Cuál será la diferencia?

En vez de utilizar el método **res.render** utilizaremos **res.send**. No olvides que el formato esperado de respuesta es JSON.

No olvides tener siempre a mano Postman para probar cada uno de los endpoints que vayas creando y asegurar que la respuesta sea la esperada así como verificar que en la base de datos veas los cambios estipulados.

4. Bonus Track

Te dejamos algunas ideas de mejoras para esta API:

1. Verbos HTTP: Si utilizaste solo GET y POST...es hora de mejorar esto. Asegurate que los endpoints de actualización usen PUT o PATCH y que el de eliminación utilice el verbo DELETE
2. Agreguemos más recursos: Hagamos los endpoints para acceder a géneros y actores.
3. Autenticación mediante JWT:
    a. En una primera versión asumamos que el usuario no precisa un registro. Asegurate que haya un usuario en la base de datos para poder probar.
    b. Deberás crear un endpoint /api/login que haga chequee el login y en caso de éxito retorna un JWT indicando el id del usuario logueado como payload.
    c. Luego deberás agregar un middleware para todos las rutas de tu API que chequee que llegue un JWT válido.
    d. Recordá al probarlo en Postman enviar el JWT en las cabeceras de los pedidos.
    e. Ahora que ya hiciste todo esto… ¿Por qué no agregamos un endpoint para la registración también?
4. Tests en Postman: Algo muy prolijo para entregarle al cliente son los tests hechos en Postman para que también los pueda probar ¿Te animás a hacerlo?

¡Mucha suerte!