MANUAL DE USO DE API REST

PROYECTO: API PELICULAS

EMPRESA: NEXO SOLUCIONES

DESARROLLADOR: NICOLAS ORTIZ

noviembre de 2023

Tabla de contenidos

1. Introducción

Introducción

El proyecto se centra en la manipulación de datos de películas utilizando una API externa. La API se conecta a una fuente de información sobre películas para extraer detalles como títulos, años de lanzamiento, descripciones y más. El objetivo principal es proporcionar a los usuarios la capacidad de gestionar estos datos, permitiendo operaciones como subir nuevos registros a la base de datos, editar información existente y eliminar películas.

Objetivo y Finalidad

La finalidad principal es facilitar la gestión de información cinematográfica, brindando a los usuarios la capacidad de:

- **Explorar Películas:** Obtener detalles de películas desde una fuente externa.
- **Almacenar en Base de Datos:** Subir información de películas a una base de datos local.
- **Actualizar Datos:** Permitir a los usuarios actualizar información existente en la base de datos.
- **Eliminar Películas:** Ofrecer la posibilidad de eliminar registros de películas de la base de datos.

2. Instalación

A continuación, se detallan los pasos para la instalación del proyecto, que está desarrollado en Laravel y disponible en GitHub.

Requisitos Previos

Antes de comenzar, asegúrate de tener instalados los siguientes componentes en tu sistema:

- Composer
- PHP (versión 7.4 o superior)
- Git
- Postman

Pasos de Instalación

1. **Clonar el Repositorio:**

git clone https://github.com/ nicoortiz1/nexoapi.git

2. **Acceder al Directorio del Proyecto:**

cd nexoapi

3. **Instalar Dependencias:**

composer install

3. Configuración

Configurar el Archivo de Entorno:

Copia el archivo **.env.example** y crea un nuevo archivo llamado **.env**.
Configura la conexión a la base de datos y agrega las credenciales de la API externa.

Ejecutar Migraciones:

php artisan migrate

Iniciar el Servidor:

php artisan serve

Acceder a la Aplicación:

http://localhost:8000.

4. Estructura

Estructura del Proyecto

El proyecto sigue la estructura estándar de Laravel, organizado de la siguiente manera:

app: Contiene la lógica principal de la aplicación, incluyendo controladores, modelos, y otros componentes.

Http/Controllers: Controladores que manejan las solicitudes HTTP.

MovieController.php: Controlador para gestionar las operaciones relacionadas con las películas.

Models: Modelos que representan las entidades de la base de datos.

Movie.php: Modelo para la entidad de películas.

database: Contiene las migraciones y semillas de la base de datos.

migrations: Archivos de migración que definen la estructura de la base de datos.

- 2023_11_16_135038_create_movies_table.php: Migración para la tabla de películas.

public: Contiene los recursos públicos accesibles desde el navegador.

resources: Recursos que no son de código, como vistas y archivos de traducción.

routes: Define las rutas de la aplicación.

api.php: Define las rutas de uso de cada api.

tests: Contiene pruebas automatizadas.

vendor: Dependencias de Composer.

5. API y Endpoints

1. Obtener Lista de Películas:
   - **Ruta:** GET http://localhost:8000/api/movies
   - **Controlador y Método:** MovieController@index
   - **Nombre de la Ruta:** movies.index
   - **Descripción:** Esta ruta devuelve la lista de películas obtenidas de una API externa. Proporciona detalles como título, año de lanzamiento, sinopsis, etc.
   - **Respuesta existosa:** 200


```
"movies": {
        "page": 1,
        "results": [
            {
                "adult": false,
                "backdrop_path": "/rLb2cwF3Pazuxaj0sRXQ037tGI1.jpg",
                "genre_ids": [
                    18,
                    36
                ],
                "id": 872585,
                "original_language": "en",
                "original_title": "Oppenheimer",
                "overview": "The story of J. Robert Oppenheimer's role in the development of the atomic bomb during World War II.",
                "popularity": 2601.168,
                "poster_path": "/8Gxv8gSFCU0XGDykEGv7zR1n2ua.jpg",
                "release_date": "2023-07-19",
                "title": "Oppenheimer",
                "video": false,
                "vote_average": 8.2,
                "vote_count": 4410
            },
```

- **Respuesta errónea:** 500


```
{
    "error": "Client error: `GET https://api.themoviedb.org/3/discover/movie?api_keyc00a452f935459d11de849bba8d6bc9e&page=1` resulted in a `401 Unauthorized` response:\n{\"status_code\":7,\"status_message\":\"Invalid API key: You must be granted a valid key.\",\"success\":false}\n\n"
}
```


2. **Agregar Películas a la base de datos:**

- **Ruta:** POST http://localhost:8000/api//movies
- **Controlador y Método:** MovieController@store
- **Nombre de la Ruta:** movies.store
- **Descripción:** Permite a los usuarios agregar películas a la base de datos. La información de la película se obtiene de una API externa y se almacena localmente para su referencia futura.
- **Respuesta existosa:** 200


```
{
    "message": "Peliculas subidas correctamente"
}
```


- **Respuesta errónea:** 500


```
{
    "error": "Client error: `GET https://api.themoviedb.org/3/discover/movie?api_keyc00a452f935459d11de849bba8d6bc9e&page=1` resulted in a `401 Unauthorized` response:\n{\"status_code\":7,\"status_message\":\"Invalid API key: You must be granted a valid key.\",\"success\":false}\n\n"
}
```


3. **Actualizar Película por ID:**

- **Ruta: PUT** http://localhost:8000/api/movies/{id}
- **Controlador y Método:** MovieController@update
- **Nombre de la Ruta:** movies.update
- **Descripción:** Permite la modificación de los detalles de una película existente en la base de datos. El ID de la película se proporciona en la URL, y los datos actualizados se envían en el cuerpo de la solicitud que deben cumplir con los requisitos estimados.
- **Cuerpo de la solicitud:**


```
{
  "title": "Nueva película",
  "year": 2022,
  "overview": "Nueva descripción de la película."
}
```

- **Respuesta exitosa:** 200

```
{
    "message": "Pelicula actualizada correctamente"
}
```

- **Respuesta errónea:** 400

```
{
    "error": {
        "title": [
            "The title field must be a string."
        ]
    }
}
```

500

```
{
    "error": "Undefined variable $id"
}
```

4. Eliminar Película por ID:

- **Ruta:** DELETE http://localhost:8000/movies/{id}
- **Controlador y Método:** MovieController@destroy
- **Nombre de la Ruta:** movies.destroy
- **Descripción:** Elimina una película específica de la base de datos. Se identifica por su ID único, que se proporciona en la URL. La película eliminada ya no estará disponible en los registros de la base de datos.
- **Respuesta exitosa:** 200

```
{
    "message": "Pelicula eliminada correctamente"
}
```

- **Respuesta errónea:** 400

```
{
    "error": "Movie not found"
}
```

- **Respuesta errónea:** 500

```
    "error": "Undefined variable $id"
}
```

6. Enlaces

Enlace al Repositorio (GitHub):
El código fuente de este proyecto está disponible en GitHub.

https://github.com/nicoortiz1/nexoapi.git

Enlace a la Documentación en Postman:

Se puede usar Postman para probar la API de forma interactiva.

https://www.postman.com/orbital-module-physicist-92341442/workspace/my-workspace/collection/26505721-c98a1dbe-374e-4458-8580-107c90e4e272?action=share&creator=26505721