UNIR

Manual Técnico

Aplicación web de gestión de Autos y Películas. (Softlutions v2)

Mario Alberto Cortés Muñoz (2do semestre)
MASTER EN DIRECCIÓN E INGENIERÍA DE SITIOS WEB

Asignatura	Datos del alumno	Fecha
Ingeniería y desarrollo en la web	Apellidos: Cortés Muñoz	23 de enero de 2021
	Nombre: Mario Alberto	

Contenido

Asignatura	Datos del alumno	Fecha
Ingeniería y desarrollo en la web	Apellidos: Cortés Muñoz Nombre: Mario Alberto	23 de enero de 2021

Introducción

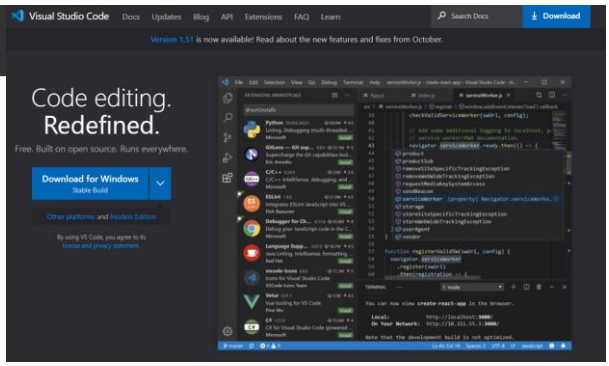
Con un manual técnico podemos conocer más a fondo la forma como se desarrolló un sistema, las tecnologías que lo conforman, las librerías, utilidades y demás componentes que fueron necesarios para poder lograr que el sistema funcione como se esperaba.

Objetivo

La elaboración de este documento tiene como principal objetivo, presentar un informe lo más detallado posible del trabajo realizado, se explicarán las decisiones de diseño e implementación que se tomaron conforme se fue realizando la construcción de la aplicación.

Construcción de la aplicación

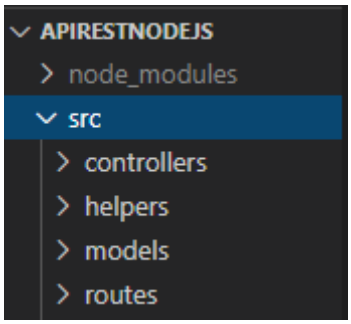
Al momento de iniciar con el desarrollo de la aplicación, ya se cuenta con las siguientes tecnologías:



Configuración del entorno de desarrollo

Con estas dos herramientas ya instaladas, dimos inicio al proyecto que sería un sistema para la creación, consulta, modificación y eliminación (CRUD) en la que una de las entidades consumiría recursos de una ApiRest externa. (Películas). La aplicación se denominó como Softlutions v2.

Asignatura	Datos del alumno	Fecha
Ingeniería y desarrollo en la web	Apellidos: Cortés Muñoz	23 de enero de 2021
	Nombre: Mario Alberto	



Creamos la estructura del árbol de carpetas que debe tener el proyecto:

» **Src**. – Ésta carpeta es la carpeta raíz de la aplicación, en ella se encuentran contenidas las demás carpetas y archivos.

» **Controllers**. – En esta carpeta están definidos los controladores de la aplicación.

» **Helpers**.- En esta carpeta se encuentran los archivos con funciones auxiliares para la aplicación.

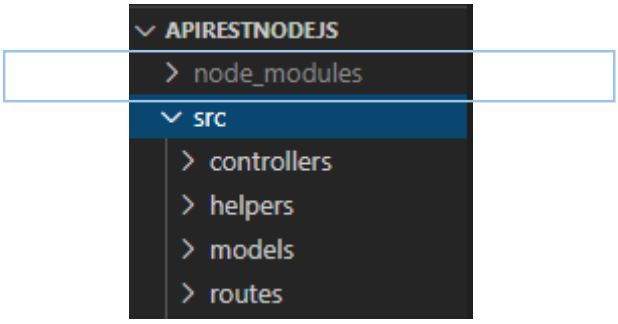
» **Models**. – En esta carpeta están definidos los modelos que tienen las tablas de las entidades guardadas en la base de datos.

» **Routes**. – En esta carpeta están definidas las rutas de acceso a las diferentes páginas de la aplicación web.

Siguiente paso:

Desde la terminal del Visual Studio Code, usamos la instrucción `npm install <nombre del módulo> --save` para instalar los módulos necesarios para que la aplicación pueda funcionar.

La instalación de estos módulos crea en el árbol de carpetas una más que se llama node_modules que contiene todos los módulos anteriormente instalados.



Después de instalar todos los módulos y dependencias necesarias, probamos la conexión al servidor. En el archivo index.js tenemos el código que se encarga de escuchar las peticiones que le llegan al servidor:

```
// Servidor que escucha las peticiones
app.listen(app.get('port'), () => {
    console.log("Server on port ", app.get('port'));
});
```

Y para probarlo en el ambiente de desarrollo ejecutamos el siguiente comando:

```
    npm run dev
```

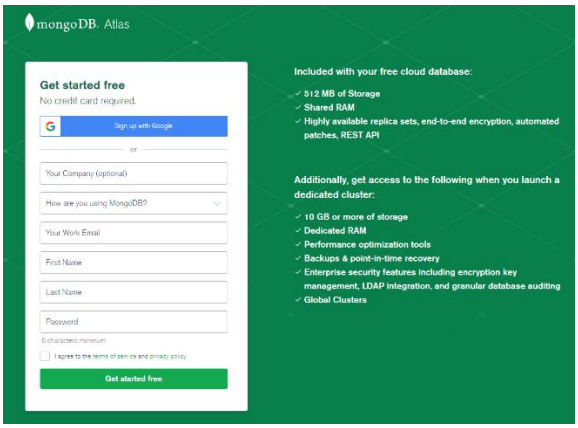
La respuesta:

```
> apirestnodejs@1.0.0 dev C:\Users\Mario Cortés\Desktop\ApiRestNodeJs
> nodemon src/index.js

[nodemon] 2.0.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node src/index.js`
Server on port  3000
base de datos conectada a :  cluster0-shard-00-00.bx0cv.mongodb.net
```

Siguiente paso:

Para la base de datos se optó por el servicio en la nube de MongoDB Atlas, mismo que se utilizó para la actividad anterior.

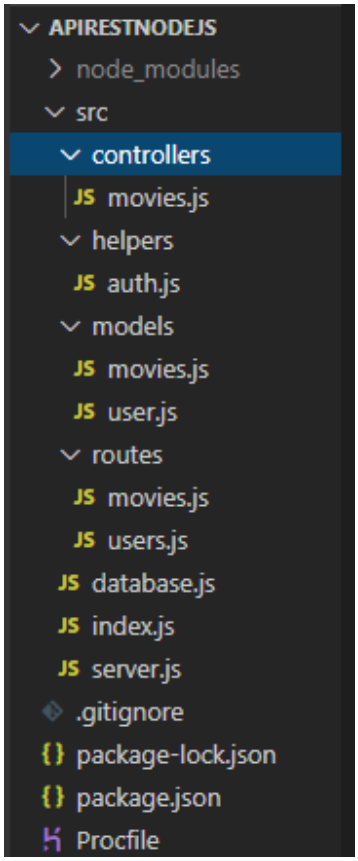


Ahora ya tenemos listo el servidor y la conexión a la base de datos.

Desarrollo de la aplicación

La aplicación se desarrolló desde cero, iniciando con la descarga de las tecnologías, y posteriormente con la preparación y configuración del ambiente de desarrollo en general.

Al final, la estructura del proyecto y de las carpetas quedó de la siguiente manera:



Controllers:
Se encuentra el controlador que contienen las funciones que acceden a la base de datos, y que realizan las acciones del crud.

Helpers: Se encuentra una función auxiliar para la autenticación de las peticiones recibidas.

Models:
Aquí están definidas las estructuras que tienen las tablas de Películas y Usuarios en la base de datos del MongoDb Atlas.

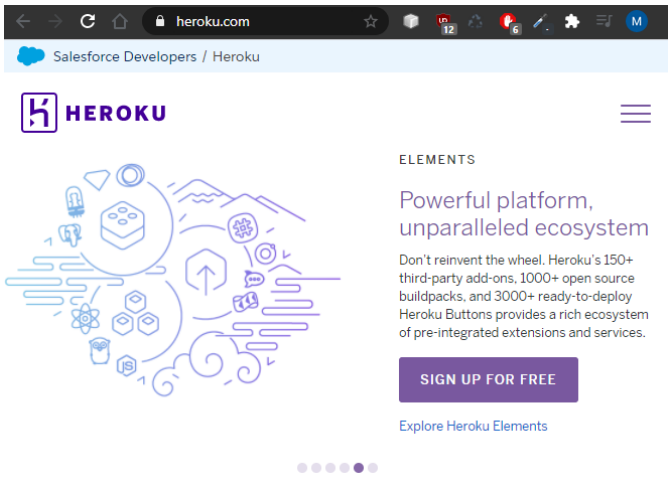
Routes:
Aquí están definidas las diferentes rutas que siguen las acciones de ambos módulos (usuarios y películas)

Fuera de éstas carpetas, pero siempre dentro de la carpeta src, están los archivos para la conexión a la bd y para el inicio del servidor.

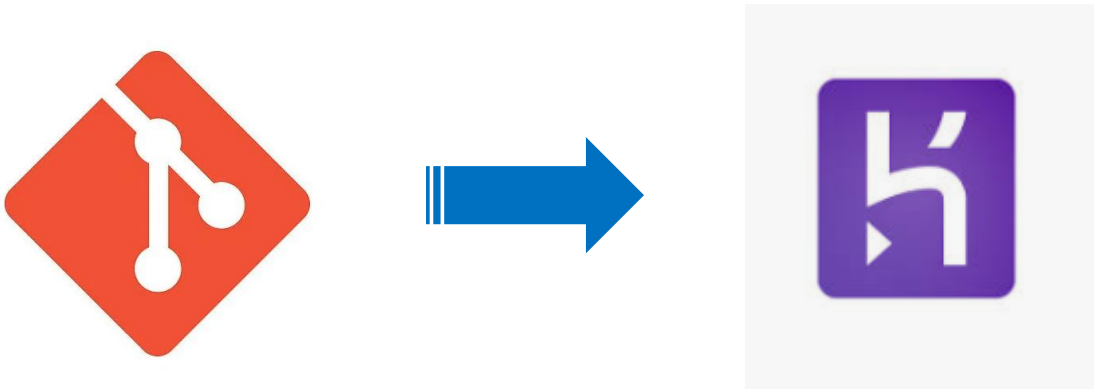
Por último está el archivo package.json que contiene la lista de módulos y dependencias instaladas en el proyecto.

Asignatura	Datos del alumno	Fecha
Ingeniería y desarrollo en la web	Apellidos: Cortés Muñoz	23 de enero de 2021
	Nombre: Mario Alberto	

Ya concluido el desarrollo de la aplicación, necesitamos un espacio para desplegarlo. Para ello utilizamos un servicio en la nube de Heroku que nos provee de dicho servicio. Se creó una cuenta gratuita para acceder al servicio.

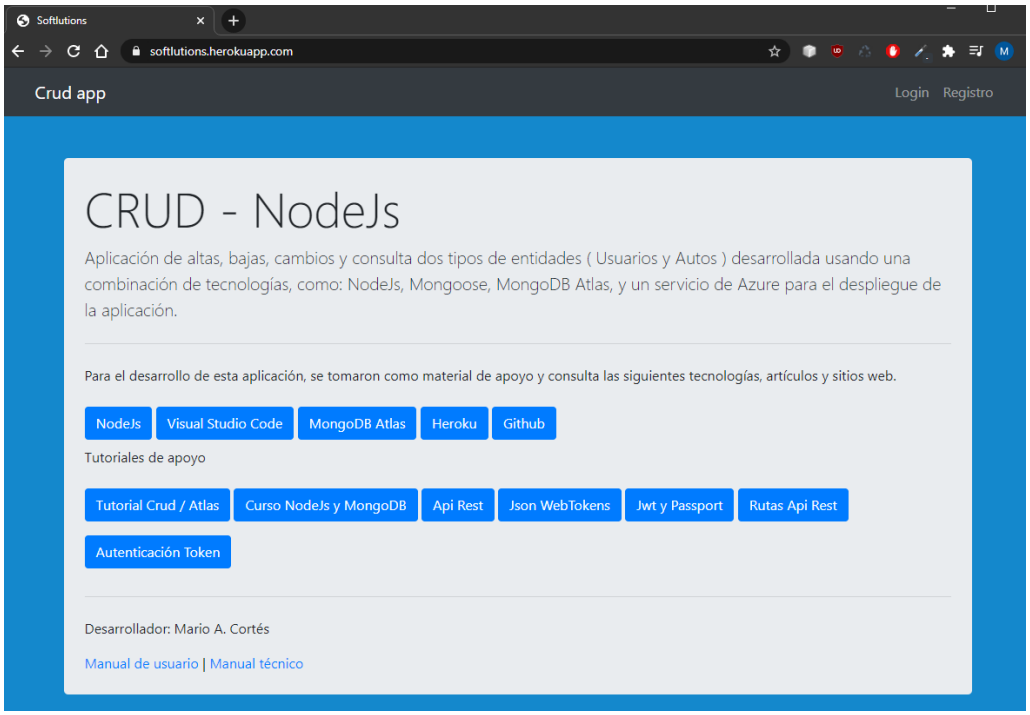


En este servicio se crearon dos espacios, uno para la app “cliente” y otro para la apiRest de películas. Ambos se subieron utilizando la herramienta Git.



Finalmente obtenemos el enlace en donde se encuentra publicada la aplicación.
https://softlutions.herokuapp.com/

Asignatura	Datos del alumno	Fecha
Ingeniería y desarrollo en la web	Apellidos: Cortés Muñoz	23 de enero de 2021
	Nombre: Mario Alberto	



Uso del código de la app

Para usar/instalar el código de ambas aplicaciones, se puede clonar desde Github a la máquina local como sigue:

git clone
https://github.com/ismariocortes/softlutionsNodejs (cliente)
https://github.com/ismariocortes/ApiRestNodeJs (apiRest-movies)

Después de clonarlo, abrirlo con el Visual Studio Code (recomendado), y desde la terminal del editor se ejecuta el comando:

npm i

Con este comando se instalan todos los módulos y dependencias que necesita el proyecto.
Cuando termine de instalar, ejecutar el comando:

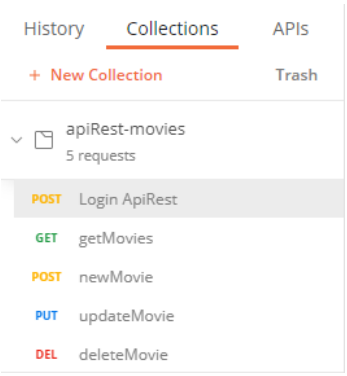
npm run dev

¡Y eso es todo!

Asignatura	Datos del alumno	Fecha
Ingeniería y desarrollo en la web	Apellidos: Cortés Muñoz	23 de enero de 2021
	Nombre: Mario Alberto	

Pruebas Postman

Para realizar las pruebas de funcionamiento de la aspiRest de películas se utilizó la herramienta Postman.

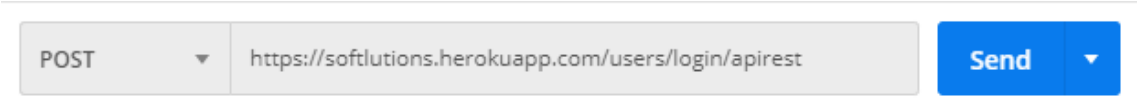


Se creó una colección de peticiones para probar los 4 verbos que se usan para el CRUD y para el logueo en la app cliente.

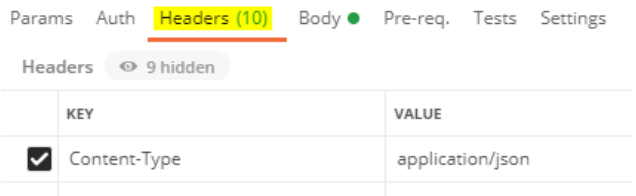
Login ApiRest

La primera petición es para realizar el logueo en la aplicación cliente y nos devuelva la información del usuario que está accediendo, principalmente el token de seguridad que se generó.

La petición es de tipo POST, la url es https://softlutions.herokuapp.com/users/login/apirest

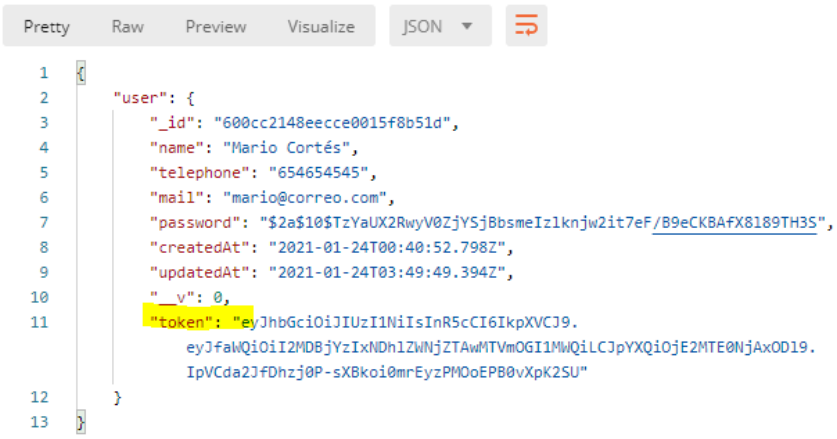


En la cabecera se especifica lo siguiente:



En el Body se pasan las credenciales del usuario previamente registrado en la app cliente:

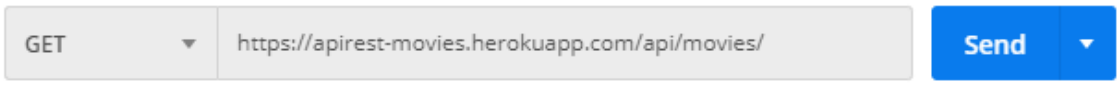
Como respuesta obtenemos la información del usuario logueado y el token de seguridad que se le asignó, mismo que usaremos para las pruebas del CRUD.



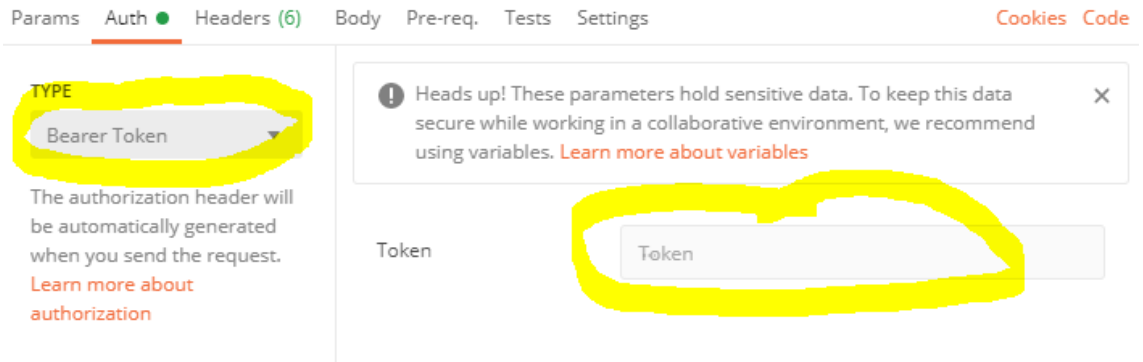
Get Movies

La siguiente petición será de tipo GET, y se hará a la url en donde se encuentra alojada la ApiRest de películas:

https://apirest-movies.herokuapp.com/api/movies/

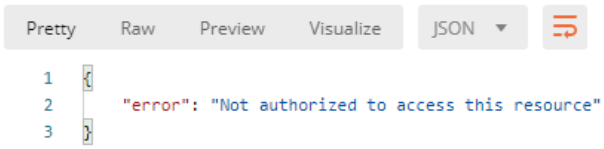


En las pestañas de los Headers y del body no enviamos nada. En este caso la pestaña que nos interesa es la de Auth.

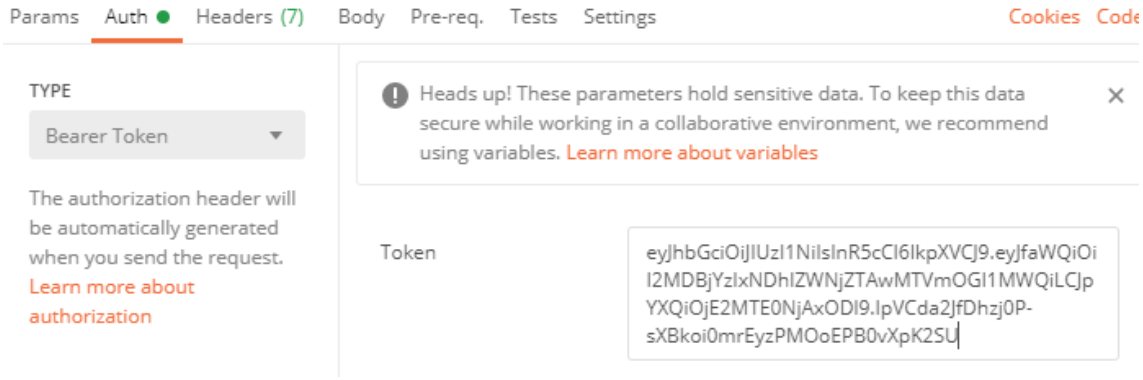


Se selecciona el tipo de autenticación Bearer Token. Y en el campo token se debe colocar el token que se generó para el usuario logueado.

Si enviamos la petición sin el token, nos devuelve lo siguiente:

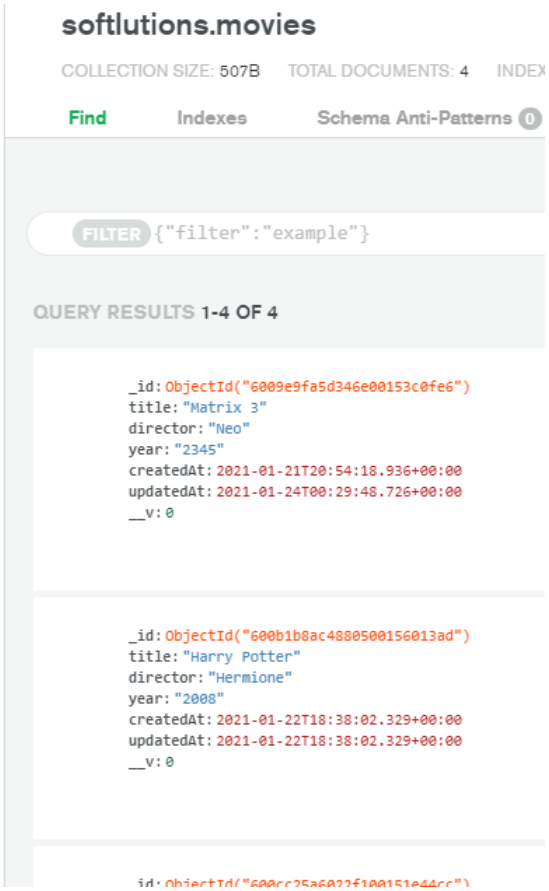


Pero si ahora la enviamos con el token,



Nos devuelve lo siguiente:

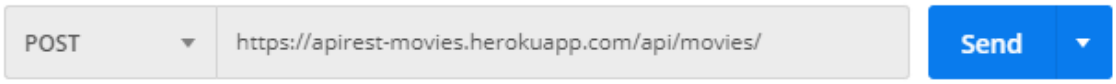
Un Json con todas las películas que se encuentran guardadas en la base de dato de MongoDb Atlas.



New Movie

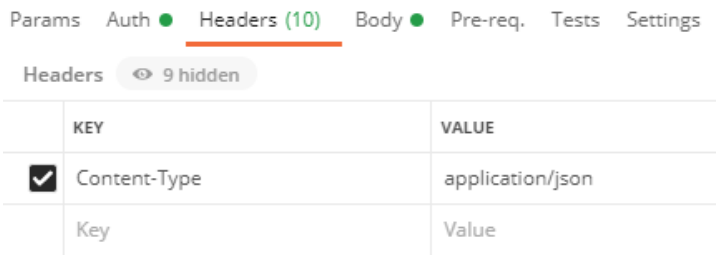
La siguiente petición es también de tipo POST y se hace a la siguiente url:

https://apirest-movies.herokuapp.com/api/movies/

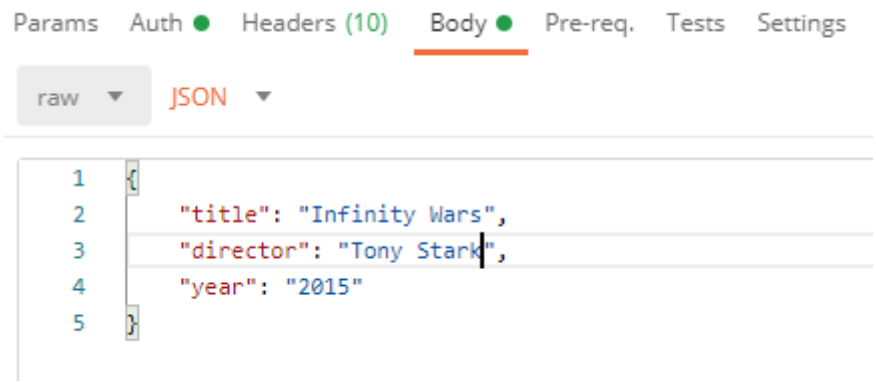


Para esta petición usaremos nuevamente la pestaña de Headers y Body, además de la pestaña Auth.

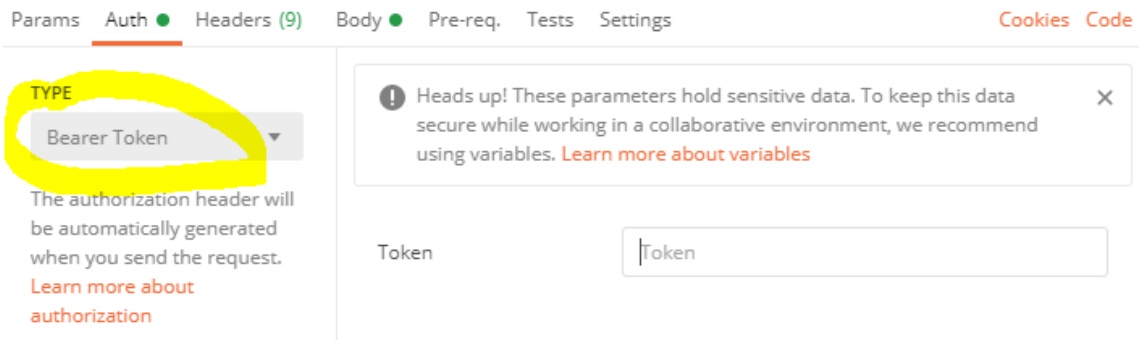
Para el header colocamos lo siguiente:



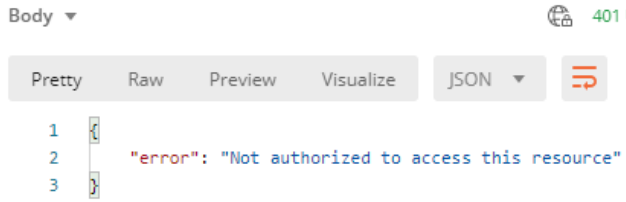
Y en el body enviamos un json con los datos de la película que queremos crear.



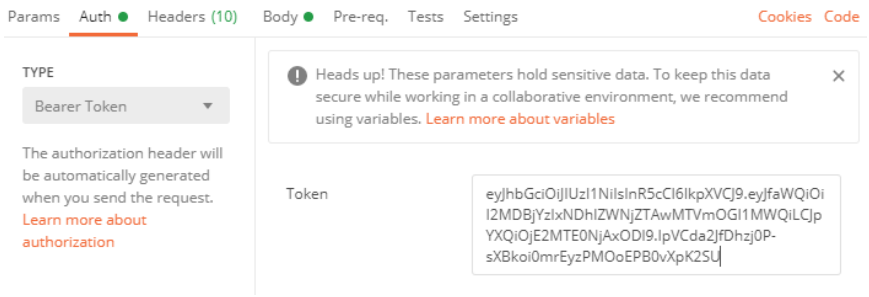
De la misma manera en la pestaña Auth debemos seleccionar el tipo Bearer Token



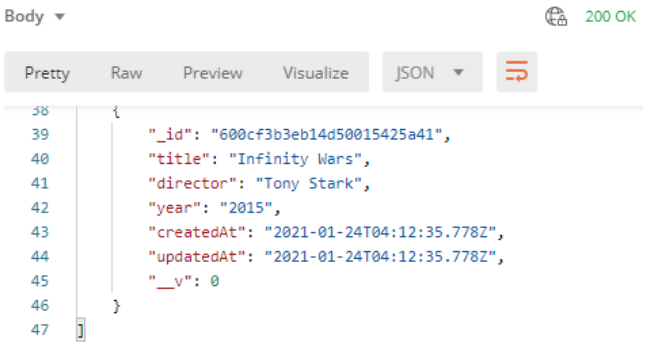
Y si enviamos la petición sin el token, nos devuelve lo siguiente:



Y ahora con el token



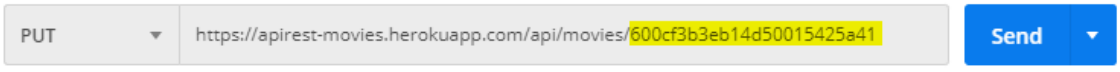
Nos devuelve toda la lista de películas en la bd, y al final la película que creamos.



Update Movie

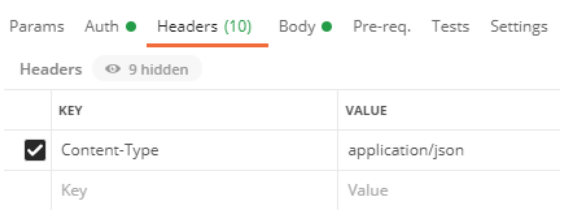
Esta petición es similar a la anterior en cuestión de parámetros, a diferencia de que es de tipo PUT y lleva un parámetro adicional en la url:

https://apirest-movies.herokuapp.com/api/movies/600cf3b3eb14d50015425a41



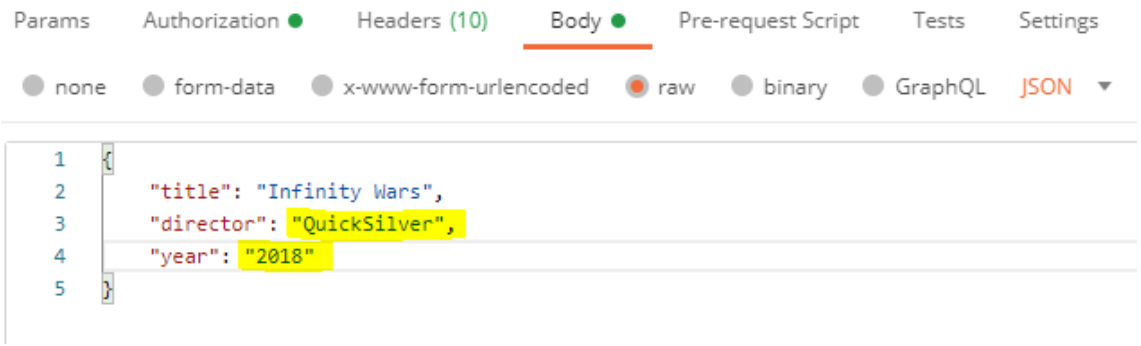
Ese parámetro corresponde al id de la película que queremos editar, en este caso es el id de la película que creamos en la petición anterior.

Las cabeceras son similares, en el header va el content-type.

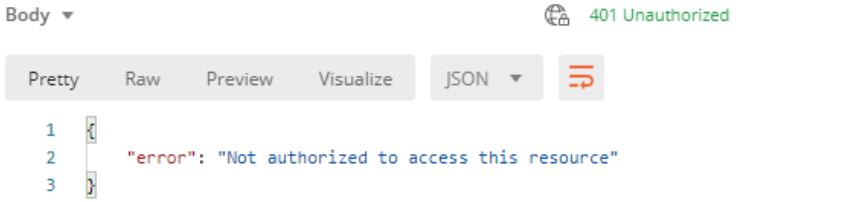


Y en el body va el json con la nueva información de la película que se va a editar.

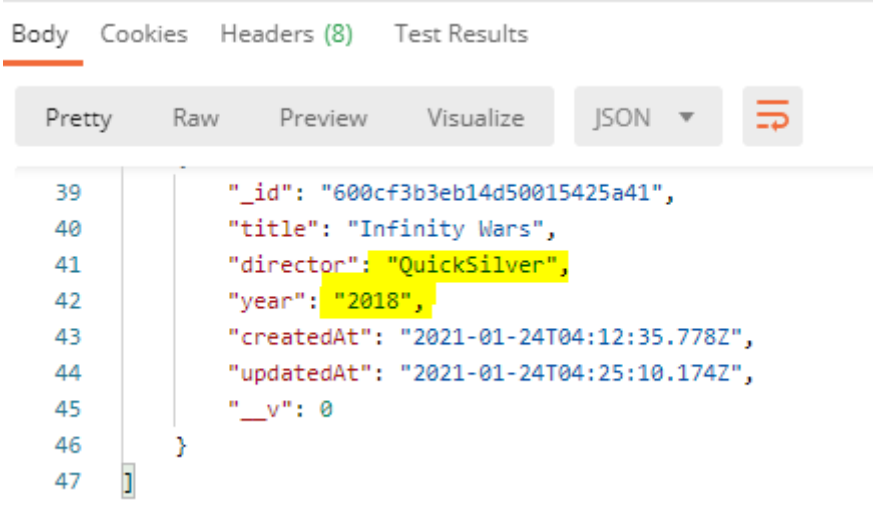
Modificaremos el campo director y year.



Nuevamente si no pasamos el token nos devuelve el mensaje de error.



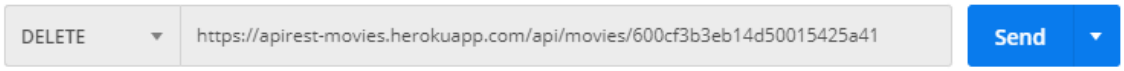
Y con el token nos devuelve la película actualizada.



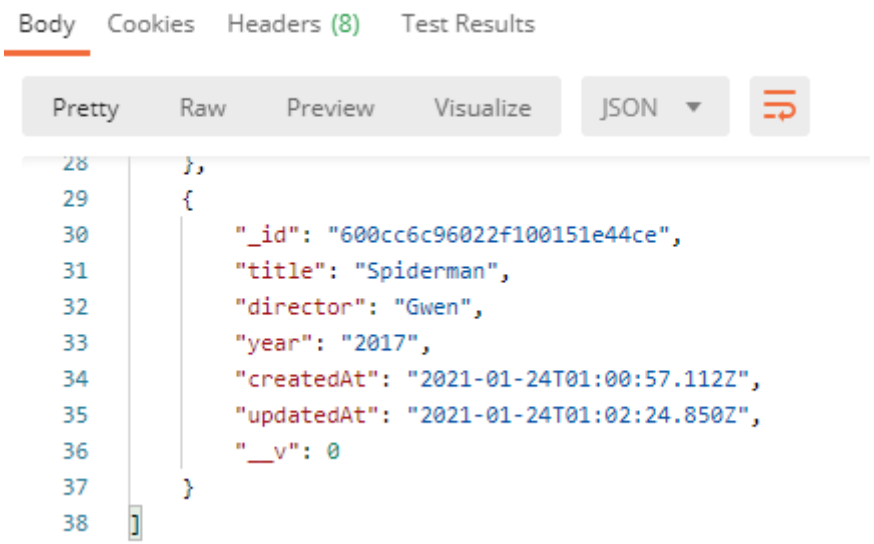
Delete Movie

Por último, tenemos la petición de tipo DELETE que lleva como parámetro en la url el id de la película que queremos eliminar, en este caso le pasaremos el mismo id de la película que creamos.

https://apirest-movies.herokuapp.com/api/movies/600cf3b3eb14d50015425a41



Para esta petición no necesitamos nada en el header ni en el body, solamente el mismo caso en la pestaña Auth, el cual nos devuelve mensaje de error si no le pasamos en token de seguridad, o de lo contrario nos devuelve la lista de películas guardadas ahora sin la película que acabamos de eliminar.



Asignatura	Datos del alumno	Fecha
Ingeniería y desarrollo en la web	Apellidos: Cortés Muñoz	23 de enero de 2021
	Nombre: Mario Alberto	

Conclusión

Un verdadero reto, me costó mucho trabajo implementar la parte del token, el CRUD salió relativamente rápido, pero encontrar la forma de generar el token me llevó varios días. Y después de que logré que se genere el token al momento de que el usuario se loguee, otro problema, ¿cómo conecto mi aplicación cliente con mi apirest?, ¿cómo le envío los parámetros?, ¿cómo valido que se envíe el token y que se reciba en la apirest, y además que sea válido?, bueno, un caos.
Por un momento pensé en dejarlo por la paz, ya me tenía muy estresado y me estaba llevando demasiado tiempo. Pero no estaba tranquilo, no podía dejarlo pasar y quedarme "derrotado" por un reto. Así que seguí buscando tutoriales, videos, incluso me topé con uno que me parece que estaba hablado en un idioma muy extraño que no era inglés, pero al final después de tantas cosas que le moví al código, por fin funcionó. Fue muy cansado, pero al final también muy satisfactorio al ver que todo funcionó a la perfección. ;)

Referencias

NodeJs
https://nodejs.org/es/

Visual Studio Code
https://code.visualstudio.com/

MongoDb Atlas
https://www.mongodb.com/cloud/atlas/signup

Heroku
https://www.heroku.com/

Git
https://git-scm.com/

***Tutoriales para el desarrollo de la aplicación
https://www.youtube.com/watch?v=bK3AJfs7qNY&t=912s&ab_channel=FaztCode
https://www.youtube.com/watch?v=qckBlIfOnlA&t=1912s&ab_channel=FaztCode
https://www.youtube.com/watch?v=EcCIlxfxc4g&t=430s&ab_channel=PractiDev
https://www.youtube.com/watch?v=w8It1NHeGps&ab_channel=CarlosAzaustre
https://leonvalen.wordpress.com/2020/02/12/autenticacion-y-autorizacion-de-json-web-token-en-nodejs-express-y-mongodb-rest-api/