Instituto Politécnico Nacional

Omar Montoya Romero

7CM1

WEB CLIENT AND BACKEND DEVELOPMENT FRAMEWORKS

Sistemas Computacionales

Ejercicio 04.- API de la Tabla de Peliculas

Mtro. Efraín Arredondo Morales

30/09/2023

Primero inicializamos el proyecto con el comando npm init -y, después instalamos npm install express, una vez instaladas las librerías podemos seguir configurando el servidor, el puerto lo configuramos de la siguiente manera `process.env.port` el cual nos sirve para que al momento de subirlo a render o algún otro servicio, el servidor tome como endpoint la liga que da al subirlo, al mismo tiempo se pone que si no esta ese endpoint se prenda en el puerto 3000.

```
const express = require("express");
const app = express();
const port = process.env.port || 3000;
app.use(express.json()); //Habilitacion para recibir datos por medio de solicitud (insomnia)
```

Creamos el arreglo de las peliculas donde cada uno tiene su id, titulo, director, año de lanzamiento, genero y calificacion. Esto para poder hacer las consultas con los métodos GET, POST, PUT, DELETE

```
// Arreglo de objetos de categorias
let peliculas = [
    {
        id: 1,
        titulo: "Casablanca",
        director: "Michael Curtiz",
        añoLanzamiento: 1942,
        genero: "Drama/Romance",
        calificacion: 9.2,
    },
    {
        id: 2,
        titulo: "El Padrino",
        director: "Francis Ford Coppola",
        añoLanzamiento: 1972,
        genero: "Crimen/Drama",
        calificacion: 9.2,
    },
    {
        id: 3,
        titulo: "Lo que el viento se llevó",
        director: "Victor Fleming",
        añoLanzamiento: 1939,
        genero: "Drama/Romance",
        calificacion: 8.1,
    },
];
```

Empezamos a implementar los diferentes métodos:

- Obtener la lista de todos las películas (GET).
- Obtener una película por su ID (GET).
- Agregar una nueva película por su ID (POST).
- Actualizar una película por su ID (PUT).
- Eliminar una película por su ID (DELETE).

Primero tenemos el método de consulta GET para obtener todas las peliculas, el cual hace una comprobación para saber si el arreglo esta lleno o vacío, en el caso de que este lleno muestra la lista de las peliculas, mientras que si esta vacío arroja el mensaje de "No existen peliculas"

```
// Obtener la lista de todas las peliculas(GET).
app.get("/socios/v1/peliculas", (req, res) => {
    if (peliculas.length > 0) {
        res.status(200).json({
            estado: 1,
            mensaje: "Existen peliculas",
            peliculas: peliculas,
        });
    } else {
        res.status(404).json({
            estado: 0,
            mensaje: "No existen peliculas",
        });
    }
});
```

El segundo método es usando el GET pero por id, donde el id lo consigue de los params, pero al mismo tiempo hace la comprobación de que el id este en la lista de películas, en caso de que si este muestra la película más un mensaje que dice "Película encontrada" mientras si el id ingresado no se encuentra lanza un mensaje "No existe película".

```
// Obtener un pelicula por su ID (GET).
app.get("/socios/v1/peliculas/:id", (req, res) => {
    const id = req.params.id;
    const pelicula = peliculas.find((peliculas) => peliculas.id == id);
    if (pelicula) {
        res.status(200).json({
            estado: 1,
            mensaje: "Pelicula encontrada",
            pelicula: pelicula,
        });
    } else {
        res.status(404).json({
            estado: 0,
            mensaje: "No existen pelicula",
```

```
        });
    }
});
```

En el apartado de agregar una nueva película se pide en el body el titulo, director, genero y calificación, se comprueba que estos no sean nulos, si no son nulos se les asigna a una constante llamada película la cual se encarga de almacenar cada variable por medio de referencia, calculamos la longitud inicial del arreglo antes de agregar la nueva, se hace push para agregarla y se comprara si el arreglo de películas es mayor a longitud inicial s se muestra el mensaje de se agrego correctamente. En caso de que no sea así manda el Mensaje de Película no agregada.

```
// Agregar una nueva pelicula (POST).
app.post("/socios/v1/peliculas", (req, res) => {
    const { titulo, director, genero, calificacion } = req.body;
    const id = Math.round(Math.random() * 1000);
    if (
        (titulo == undefined || director == undefined || calificacion == undefined,
        genero == undefined)
    ) {
        res.status(400).json({
            estado: 0,
            mensaje: "BAD REQUEST Faltan parametros en la solicitud",
        });
    } else {
        const pelicula = {
            id: id,
            titulo: titulo,
            director: director,
            genero: genero,
            calificacion: calificacion,
        };
        const longitudInicial = peliculas.length;
        peliculas.push(pelicula);
        if (peliculas.length > longitudInicial) {
            res.status(201).json({
                estado: 1,
                mensaje: "Pelicula creada correctamente",
                pelicula: pelicula,
            });
        } else {
            res.status(500).json({
                estado: 0,
                mensaje: "No se agrego correctamente",
            });
        }
    }
```

```
});
```

En este caso vamos a actualizar, tiene el mismo funcionamiento que el anterior la única diferencia es que sobreescribimos lo que ya esta en una película y para encontrar la película es por id el cual se encuentra en los params.

```
// Actualizar un pelicula por su ID (PUT).
app.put("/socios/v1/peliculas/:id", (req, res) => {
    const { id } = req.params;
    const { titulo, director, genero, calificacion } = req.body;
    if (
        (titulo == undefined || director == undefined || calificacion == undefined,
        genero == undefined)
    ) {
        res.status(400).json({
            estado: 0,
            mensaje: "BAD REQUEST Faltan parametros en la solicitud",
        });
    } else {
        const posActualizar = peliculas.findIndex((pelicula) => pelicula.id == id);
        if (posActualizar != -1) {
            peliculas[posActualizar].titulo = titulo;
            peliculas[posActualizar].director = director;
            peliculas[posActualizar].calificacion = calificacion;
            peliculas[posActualizar].genero = genero;
            res.status(200).json({
                estado: 1,
                mensaje: "pelicula actualizada correctamente",
                pelicula: peliculas[posActualizar],
            });
        } else {
            res.status(404).json({
                estado: 0,
                mensaje: "No se actualizo",
            });
        }
    }
});
```

Para la parte de eliminar se busca la película por id, una vez encontrada se compara si es diferente a -1 en caso de que sea diferente se elimina y manda mensaje de eliminada mientras si no es econtrada se manda el mensaje de Película no encontrada.

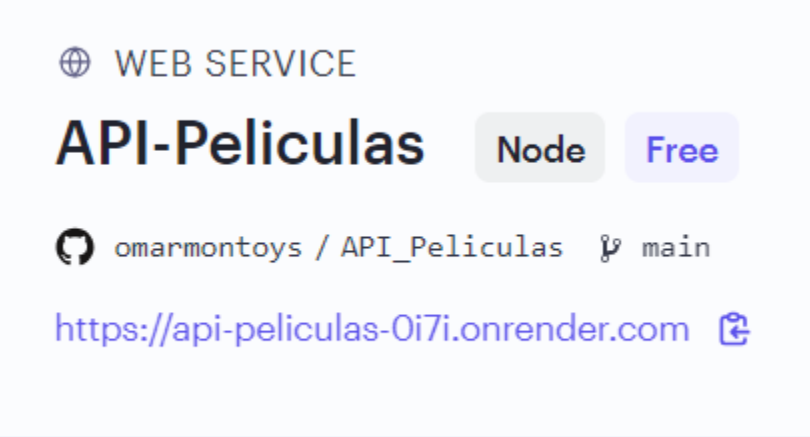
```
// Eliminar un pelicula por su ID (DELETE).
app.delete("/socios/v1/peliculas/:id", (req, res) => {
    const { id } = req.params;
    const indiceEliminar = peliculas.findIndex((pelicula) => pelicula.id == id);
    if (indiceEliminar != -1) {
```

```
        peliculas.splice(indiceEliminar, 1);
        res.status(201).json({
            estado: 1,
            mensaje: "pelicula eliminada correctamente",
        });
    } else {
        res.status(404).json({
            estado: 0,
            mensaje: "No se elimino",
        });
    }
});
```

Cuando encendemos el servidor desde la terminal el listen lo usamos para saber en que puerto se encendio y ver si esta jalando correctamente.

```
app.listen(port, () => {
    console.log("Ejecutandose en el servidor: ", port);
});
```

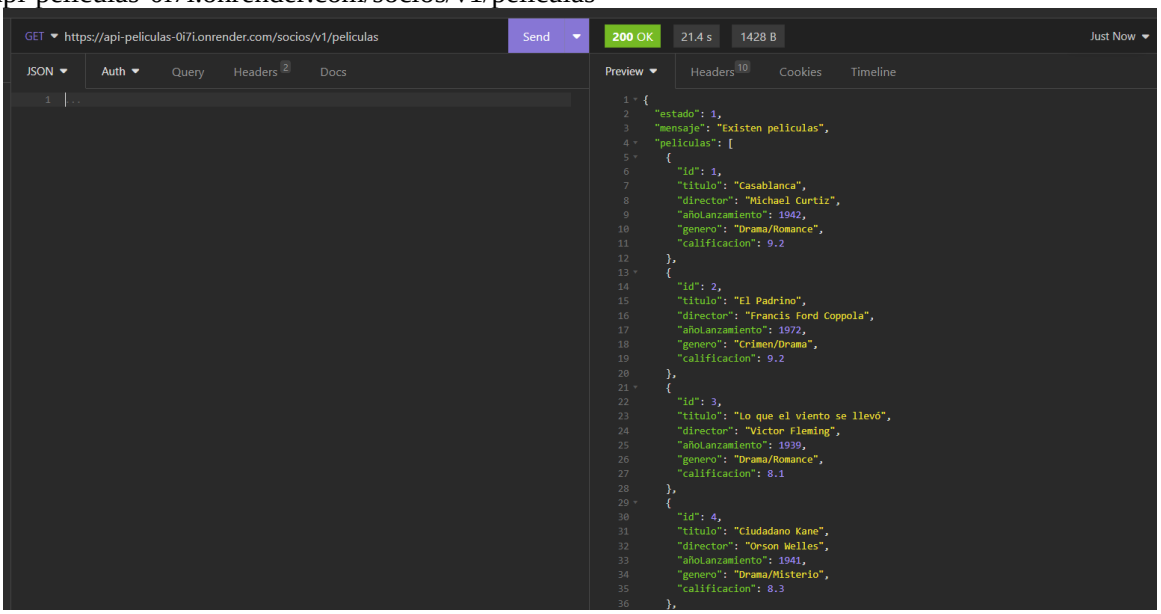
Despues de hacerlo jalar en la terminal pasamos a implementarlo en Render



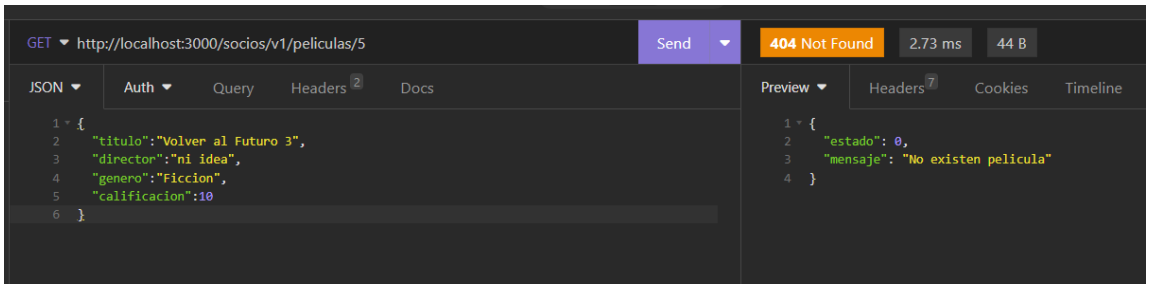

```
Sep 29 12:10:44 AM  ==> Using Node version 14.17.0 (default)
Sep 29 12:10:44 AM  ==> Docs on specifying a Node version: https://render.com/docs/node-version
Sep 29 12:10:45 AM  ==> Running 'node app.js'
Sep 29 12:10:45 AM  Ejecutandose en el servidor:  3000
Sep 29 12:10:56 AM  ==> Detected service running on port 3000
Sep 29 12:10:56 AM  ==> Docs on specifying a port: https://render.com/docs/web-services#port-detection
Sep 29 12:11:14 AM  ==> Using Node version 14.17.0 (default)
Sep 29 12:11:14 AM  ==> Docs on specifying a Node version: https://render.com/docs/node-version
Sep 29 12:11:14 AM  ==> Running 'node app.js'
Sep 29 12:11:14 AM  Ejecutandose en el servidor:  3000
```

Al tener el servidor en linea gracias a render procedemos a usarlo en insomnia y al mismo tiempo se puede visualizar en el navegador

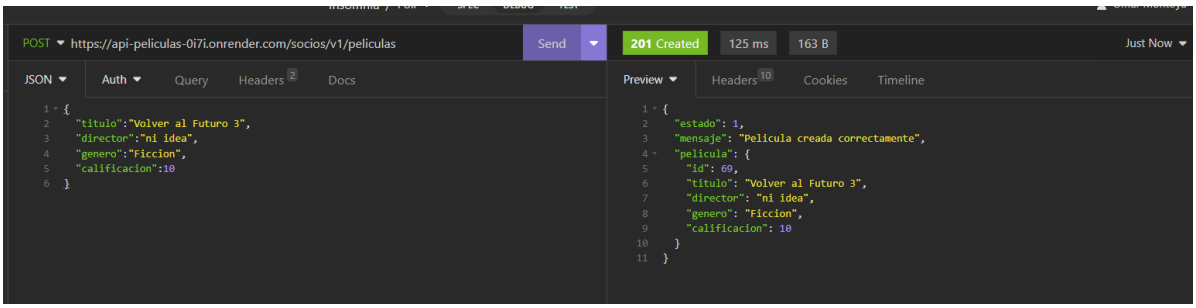
Utilizando el método GET
https://api-peliculas-0i7i.onrender.com/socios/v1/peliculas



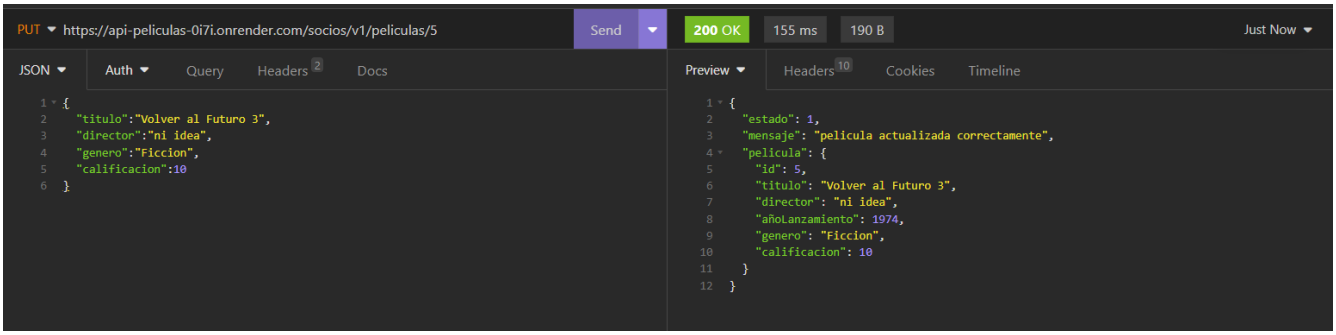
Utilizando el método GET por ID
https://api-peliculas-0i7i.onrender.com/socios/v1/peliculas/5



Utilizando el método POST
https://api-peliculas-0i7i.onrender.com/socios/v1/peliculas

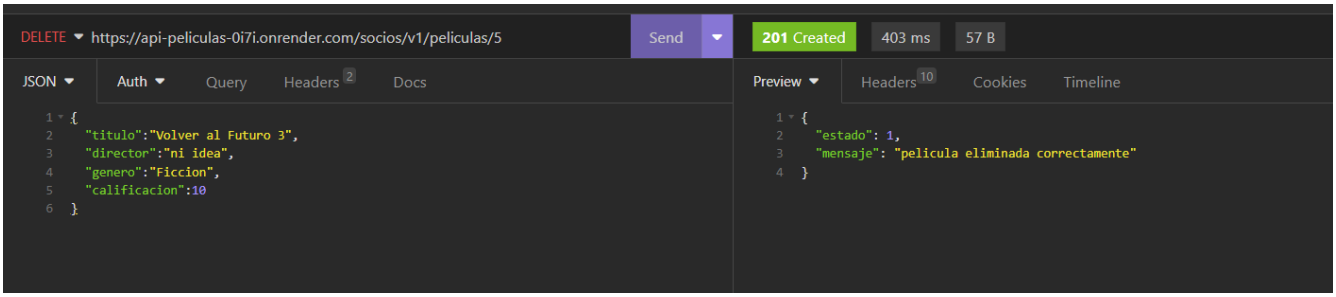


Utilizando el método PUT
https://api-peliculas-0i7i.onrender.com/socios/v1/peliculas/5



Utilizando el método DELETE, Eliminamos la película 5

https://api-peliculas-0i7i.onrender.com/socios/v1/peliculas/5



Link de GitHub. - https://github.com/omarmontoys/API_Peliculas
Link de Render. - https://api-peliculas-0i7i.onrender.com