# Informe de Ejecución y Pruebas

Integrantes:

- Diana Lorena Balanta
- George Trujillo
- Samuel Soto
- Geovanni Quintero

Este informe está vinculado al Taller III del curso "Computación en Internet II" en la Universidad Icesi. El taller tiene como objetivo la aplicación práctica de los conocimientos adquiridos en Spring y Spring Boot para desarrollar una aplicación que ofrezca una API REST para la gestión de datos relacionados con libros y autores. Además, se introduce un sistema de autenticación basado en JSON Web Tokens (JWT).

Objetivos específicos:

1. Crear una aplicación Spring Boot que proporcione una API REST para la gestión de libros y autores.

2. Desarrollar las operaciones CRUD para libros y autores, que incluyan la creación, lectura, actualización y eliminación de registros.

3. Implementar un sistema de autenticación basado en JSON Web Tokens (JWT) para asegurar que solo los usuarios autorizados puedan acceder a la API.

***Desarrollo:***
Se diseñó un modelo de datos que consta de dos entidades principales: Películas y Directores

Para gestionar eficazmente los datos y la lógica de negocio de la aplicación, se han creado repositorios, servicios y controladores dedicados a las entidades "Pelicula" y "Director". Estos componentes desempeñan roles clave en la estructura de la aplicación:

1. Repositorios: Se han implementado repositorios que se encargan de interactuar con la base de datos y proporcionar métodos para realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar) en las entidades "Pelicula" y "Director". Los repositorios permiten acceder y manipular los datos almacenados de manera eficiente.

2. Servicios: Los servicios contienen la lógica de negocio de la aplicación. Se encargan de procesar solicitudes de los controladores, aplicar la lógica necesaria y coordinar las operaciones de los repositorios. Los servicios son responsables de garantizar la integridad de los datos y aplicar reglas específicas.

3. Controladores: Los controladores definen los puntos de entrada de la API. Cada endpoint mencionado en el informe, como la lista de películas, detalles de una película, creación de una película, etc., está implementado en un controlador específico. Los controladores se comunican con los servicios para procesar las solicitudes y devolver las respuestas adecuadas.

*Endpoints:*

Películas:

- GET /peliculas: Este endpoint permite listar todas las películas registradas en el repositorio.
- GET /pelicula/{id}: Permite obtener detalles de una película específica identificada por su ID.
- POST /pelicula: Permite crear una nueva película en el repositorio. Los datos de la película se envían en el cuerpo de la solicitud.
- PUT /pelicula/{id}: Permite actualizar una película existente en el repositorio. Se utiliza el ID para identificar la película a ser modificada. Los datos de la película a actualizar se envían en el cuerpo de la solicitud.
- DELETE /pelicula/{id}: Permite eliminar una película específica identificada por su ID.
- GET /directores/{id}/peliculas: Permite listar las películas asociadas a un director específico identificado por su ID.

Directores:

- GET /directores: Este endpoint permite listar todos los directores registrados en el repositorio.
- GET /director/{id}: Permite obtener detalles de un director específico identificado por su ID.
- POST /director: Permite crear un nuevo director en el repositorio. Los datos del director se envían en el cuerpo de la solicitud.
- PUT /director/{id}: Permite actualizar un director existente en el repositorio. Se utiliza el ID para identificar el director a ser modificado. Los datos del director a actualizar se envían en el cuerpo de la solicitud.
- DELETE /director/{id}: Permite eliminar un director específico identificado por su ID.

Autenticación (Auth):
Se implementó un sistema de autenticación utilizando JSON Web Tokens (JWT). Para ello, se creó un endpoint:

POST /auth: Este endpoint recibe un nombre de usuario y contraseña en el cuerpo de la solicitud y devuelve un token JWT si las credenciales son válidas. El token JWT se utilizará posteriormente para autorizar el acceso a otros endpoints protegidos de la API.

***Ejecución:***

Herramientas necesarias: Para lograr la correcta ejecución y visualización del proyecto se debe tener lo siguiente:

Se debe asegurar tener instalado en el entorno de desarrollo lo siguiente:

- JDK 17 (Java)
- IDE compatible con Spring Boot y Maven

*Pasos para la ejecución:*

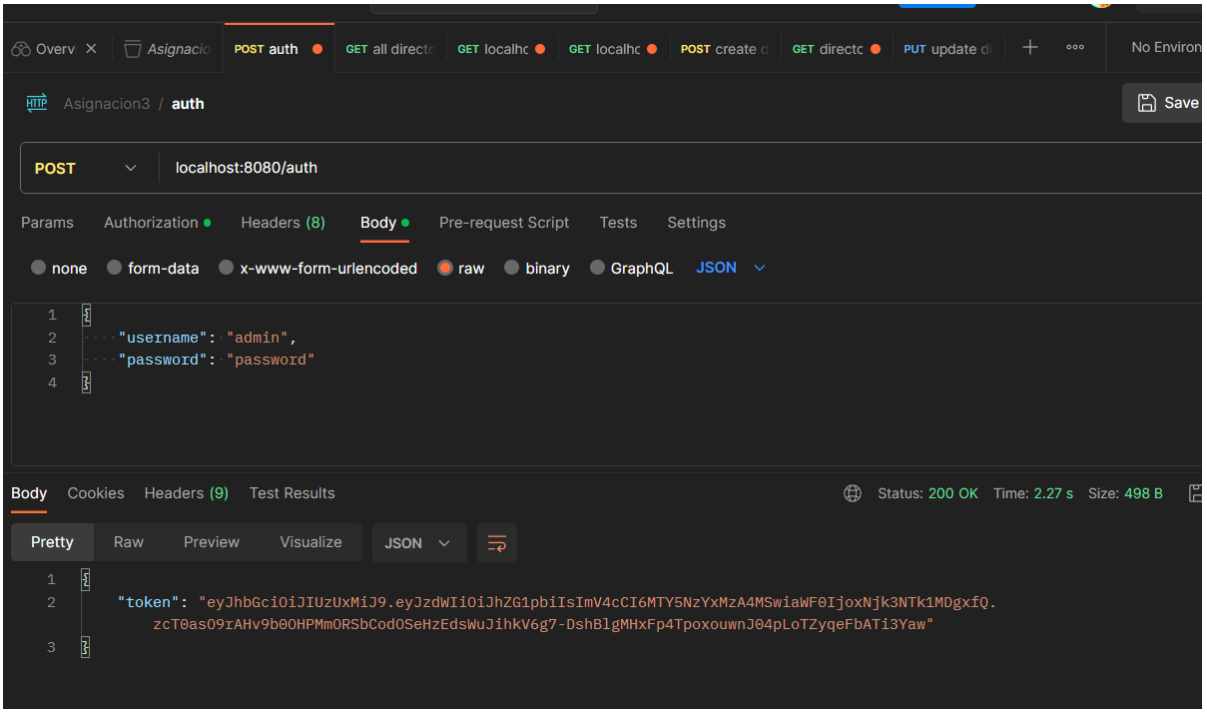
1. Clonar el Repositorio:

Link: https://github.com/GeorgeU2030/Asignacion3_SpringBoot.git

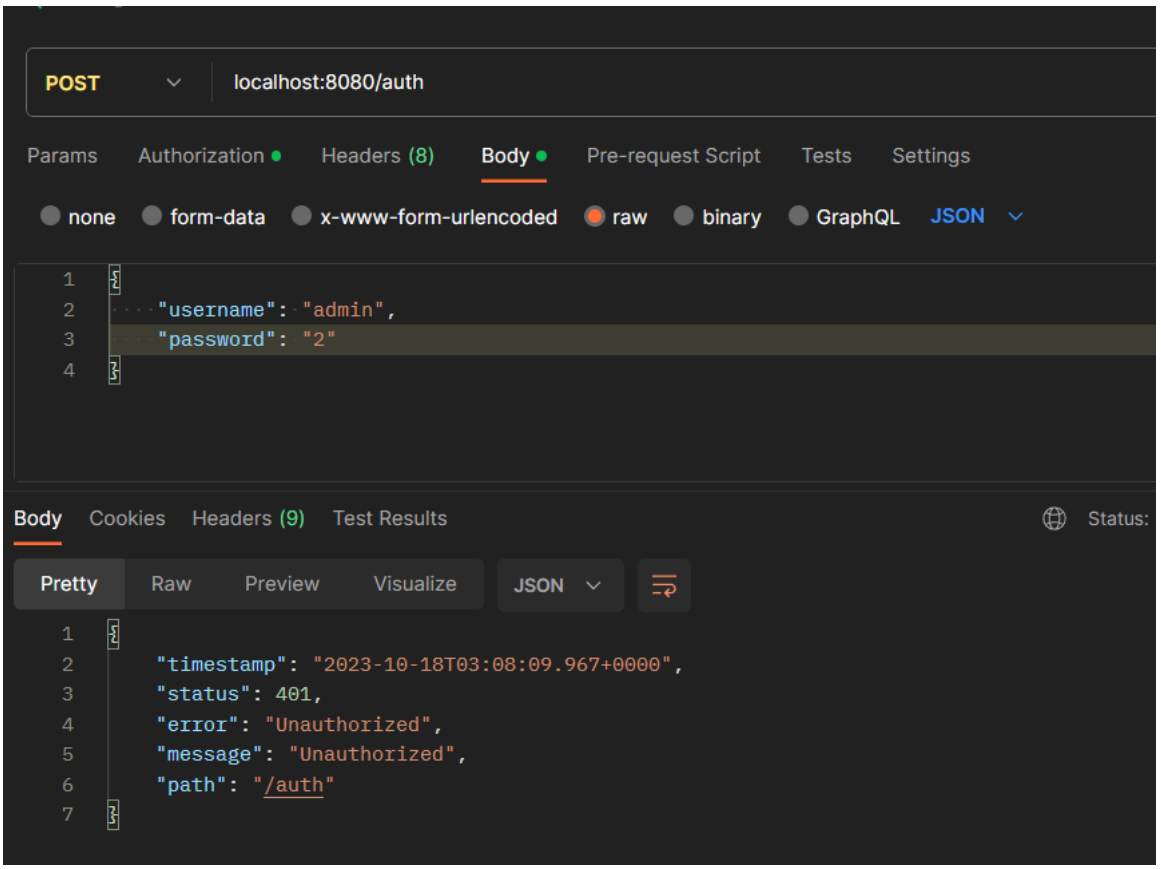
2. Importe el proyecto en el IDE.
3. Ejecute la aplicación Spring Boot haciendo click en el botón "Run"
3.La aplicación se iniciará en un servidor embebido y estará en el puerto predeterminado 8080.

**Pruebas de API REST con POSTMAN:**

- **Autentificación:**

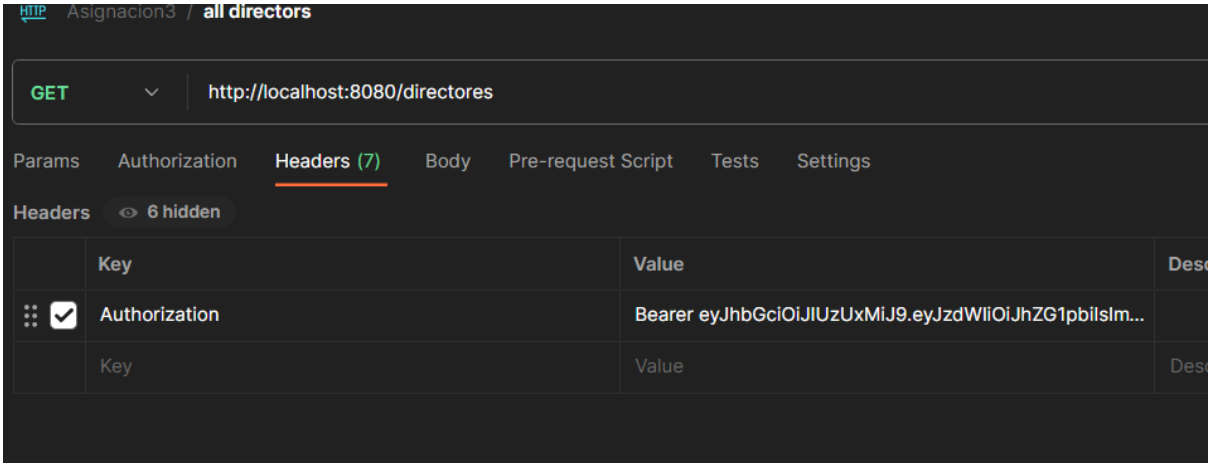



- Observamos que al enviar las credenciales username=”admin” y password=”password” predeterminadas en la aplicación se realiza la validación del usuario generando un token.

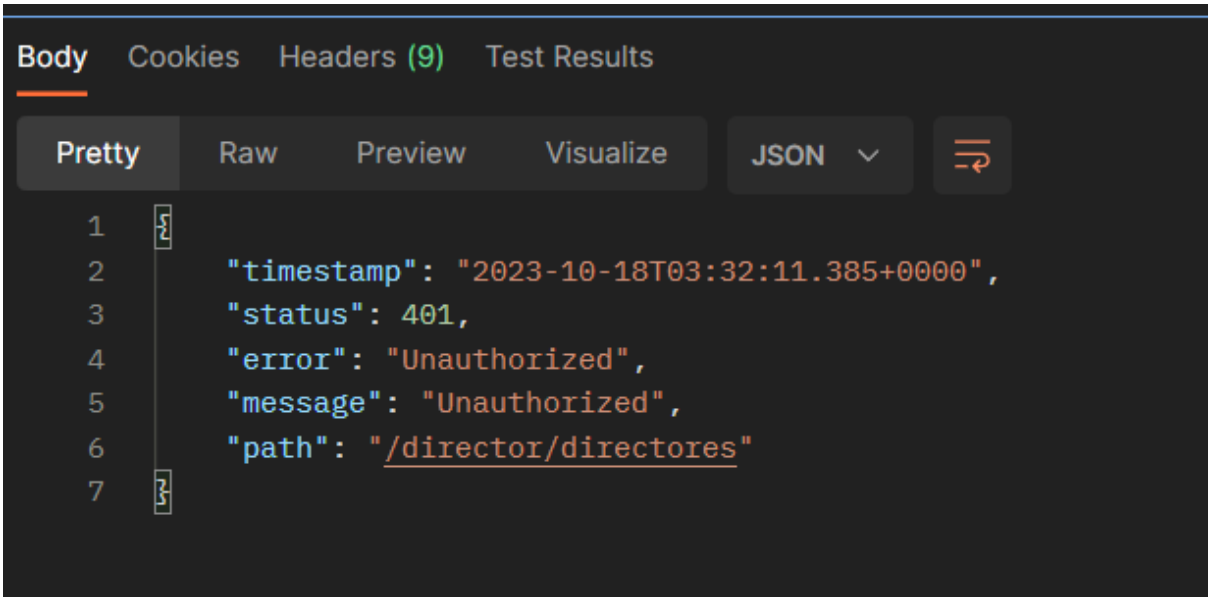



- En caso de enviar otras credenciales no validadas en la aplicación entonces no se genera el token. De esta manera observamos que funciona correctamente la autentificación.

- Ahora, para validar los demás ENDPOINTS, debemos configurar el token generado en cada ENDPOINT a probar, de la siguiente manera:

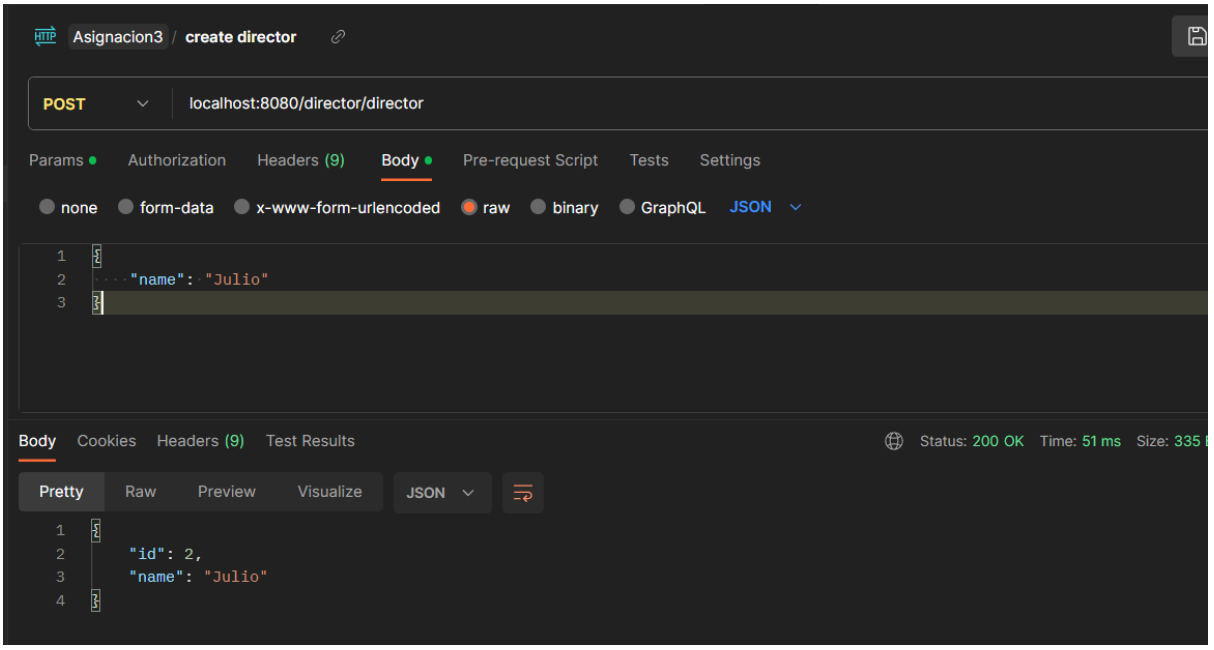



- Así nos permitirá tener acceso al ENDPOINT  ya que de otra manera nos saldría el siguiente error al no estar autentificados:



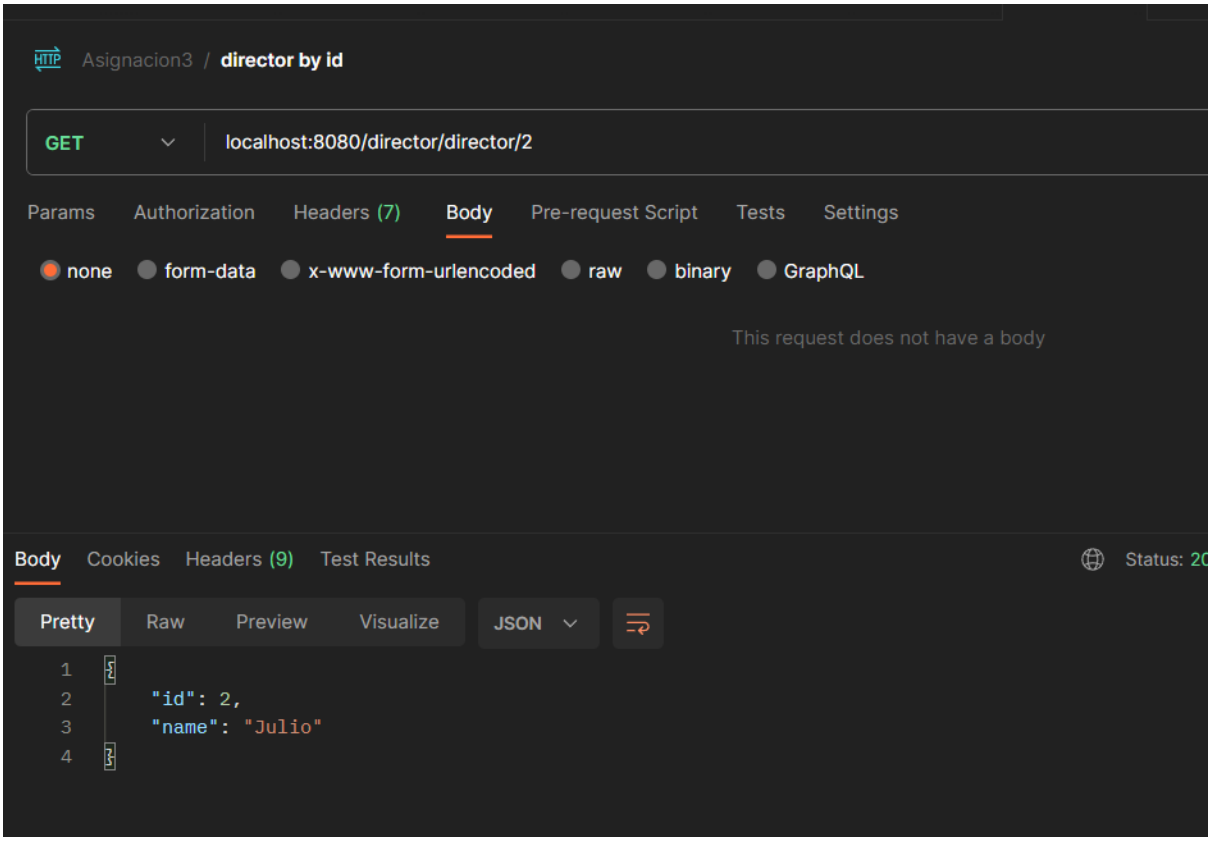

- **Pruebas ENDPOINTS para la entidad Director:**

**POST /director: Crear un nuevo director:**
URL: *http://localhost:8080/director/director*



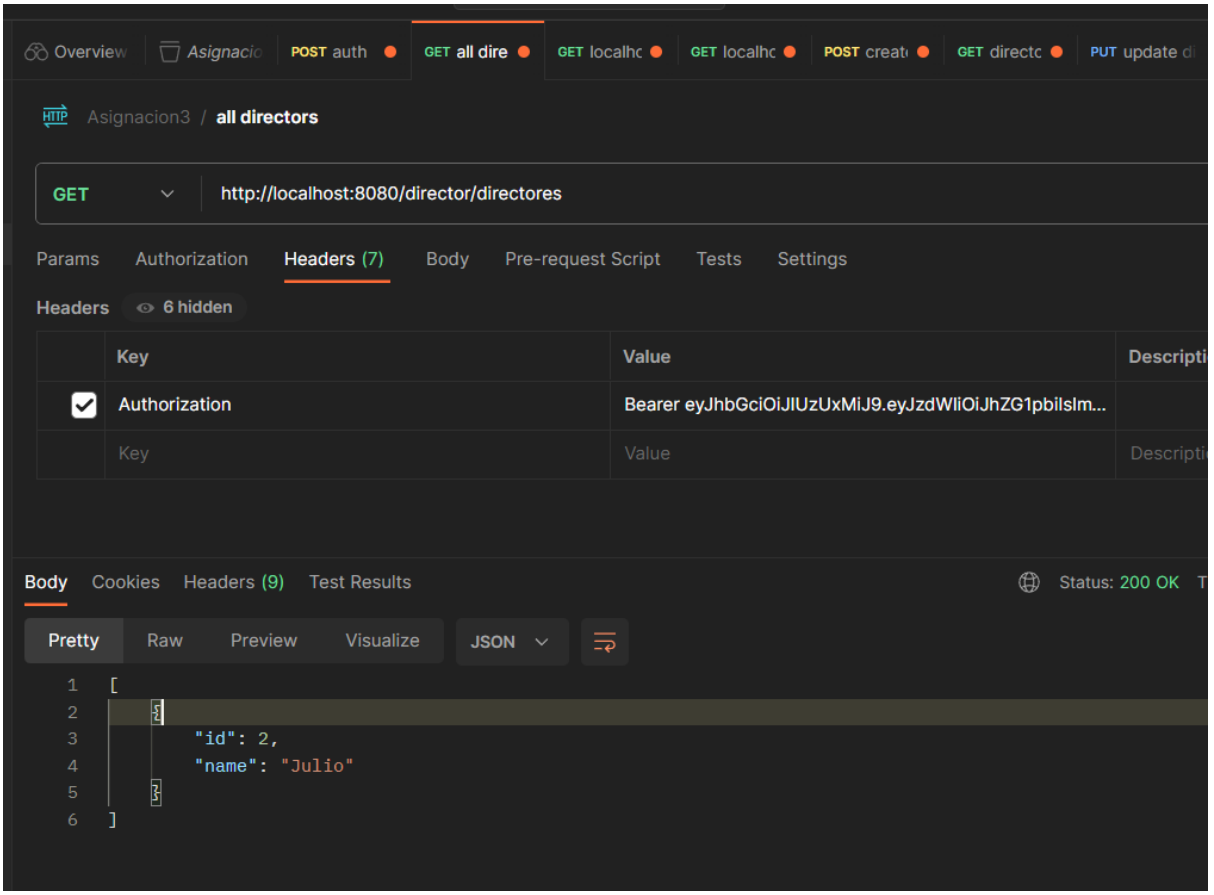

En este caso se crea el nuevo director con el nombre “Julio” y la aplicación le asigna un id autogenerado.

**GET /director/{id}: Obtener detalles de un director específico:**
URL:*localhost:8080/director/director/2*



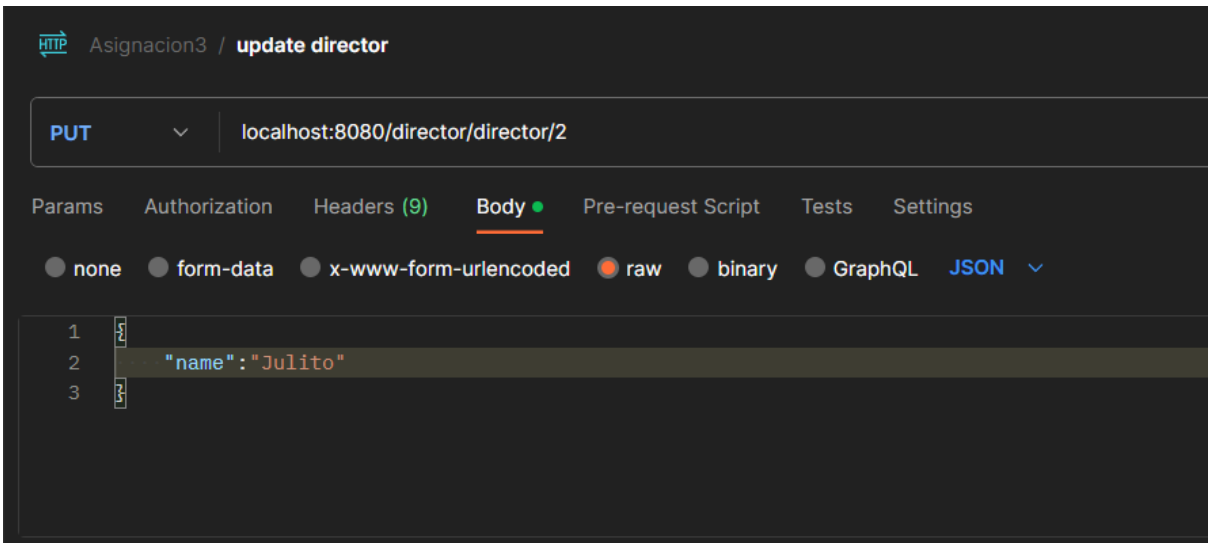

Observamos como al buscar al director recién agregado por su id nos retorna el objeto encontrado.

**GET /directores: Listar todos los directores:**
URL:*localhost:8080/director/directores*



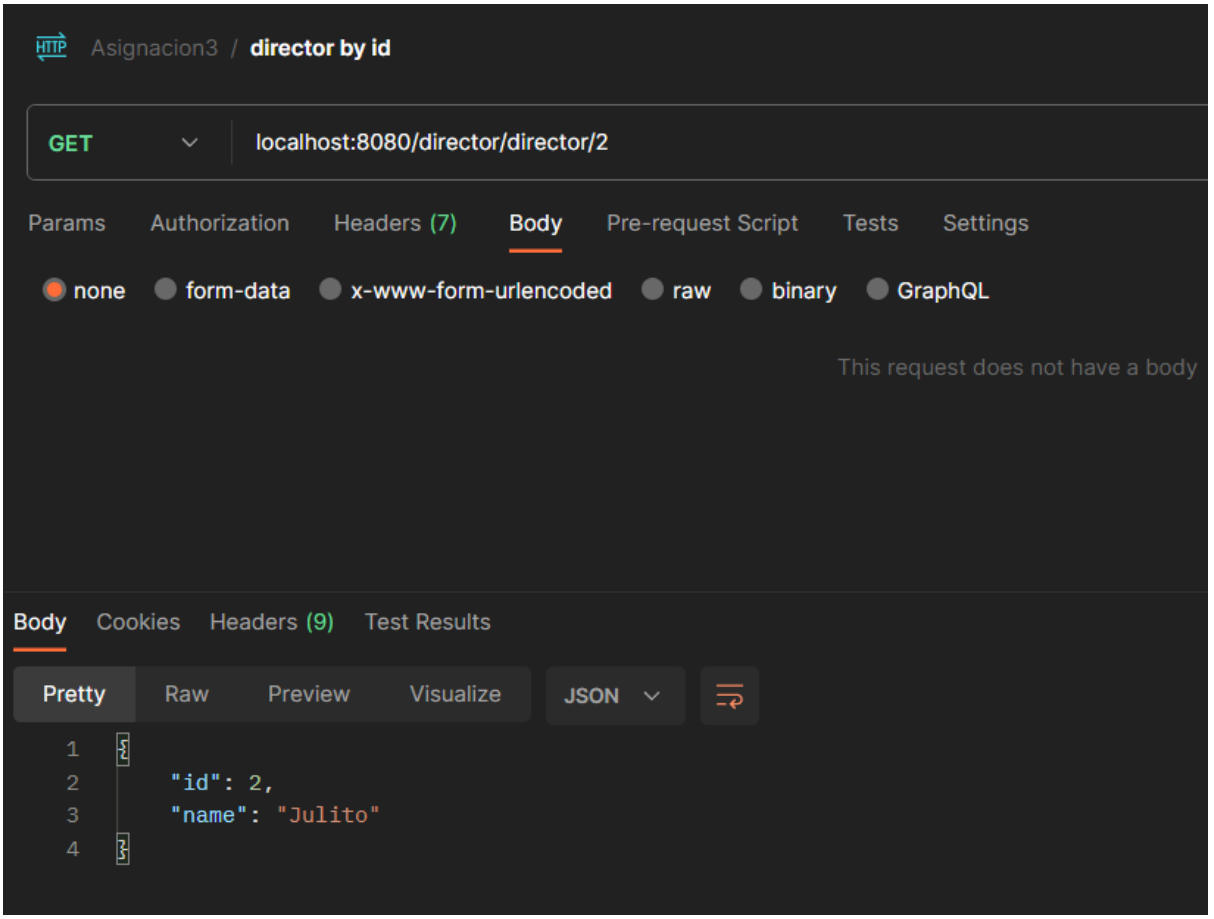

El método nos devuelve la lista de directores, en este caso el único agregado ha sido Julio.

**PUT /director/{id}: Actualizar un director existente:**
URL:*localhost:8080/director/director/2*

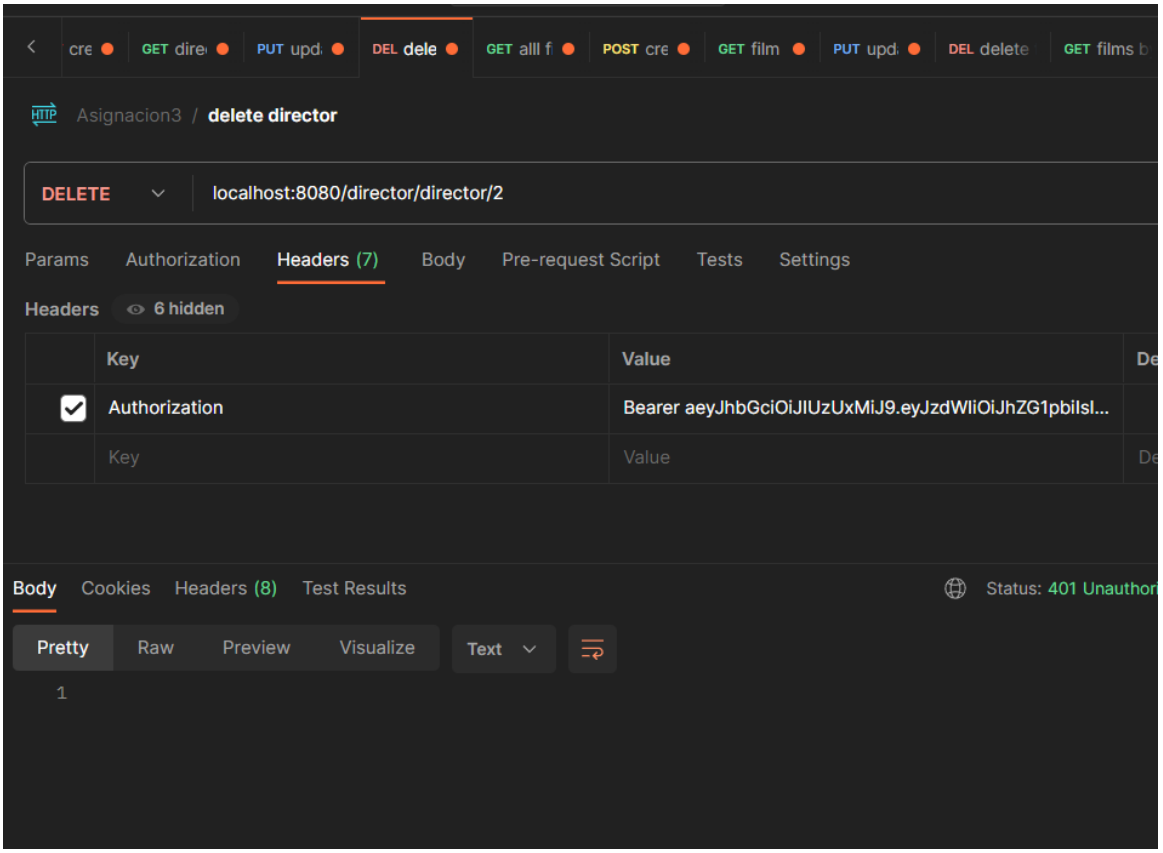



En este caso pasamos en la url el id del director a modificar, y en el Body escribimos el nuevo nombre.

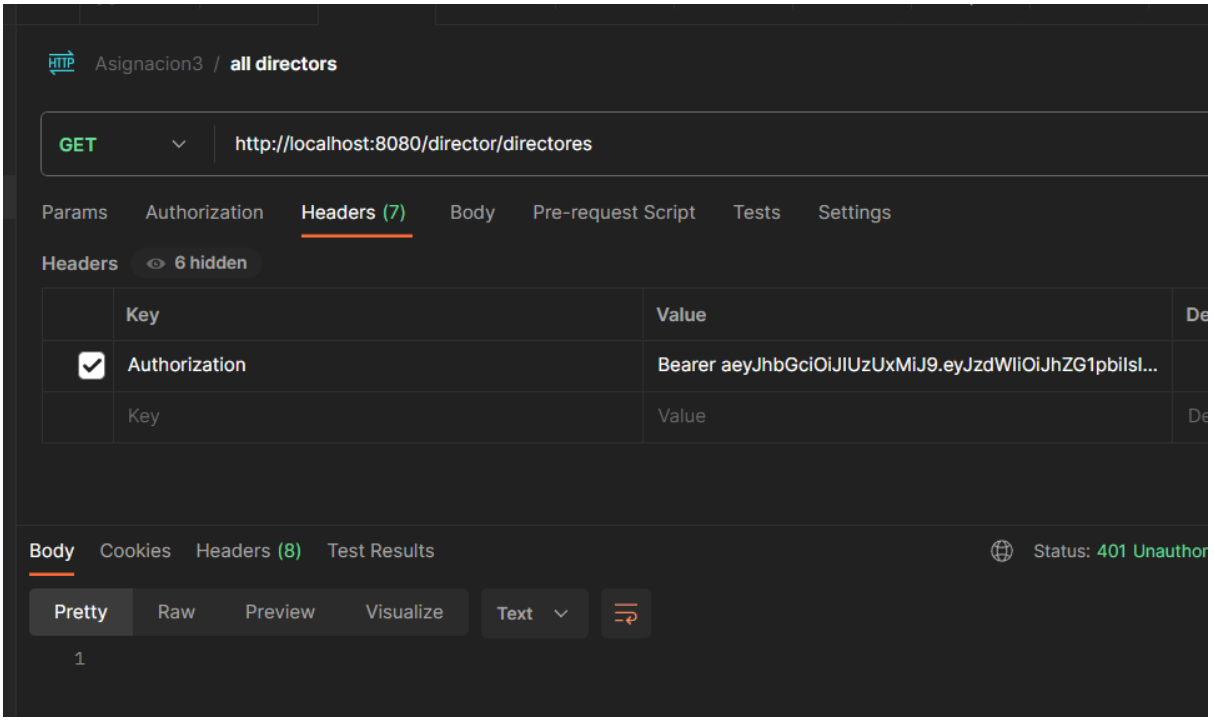
Al enviar, volvemos a ejecutar el director by id=2 y encontramos el objeto ya actualizado:



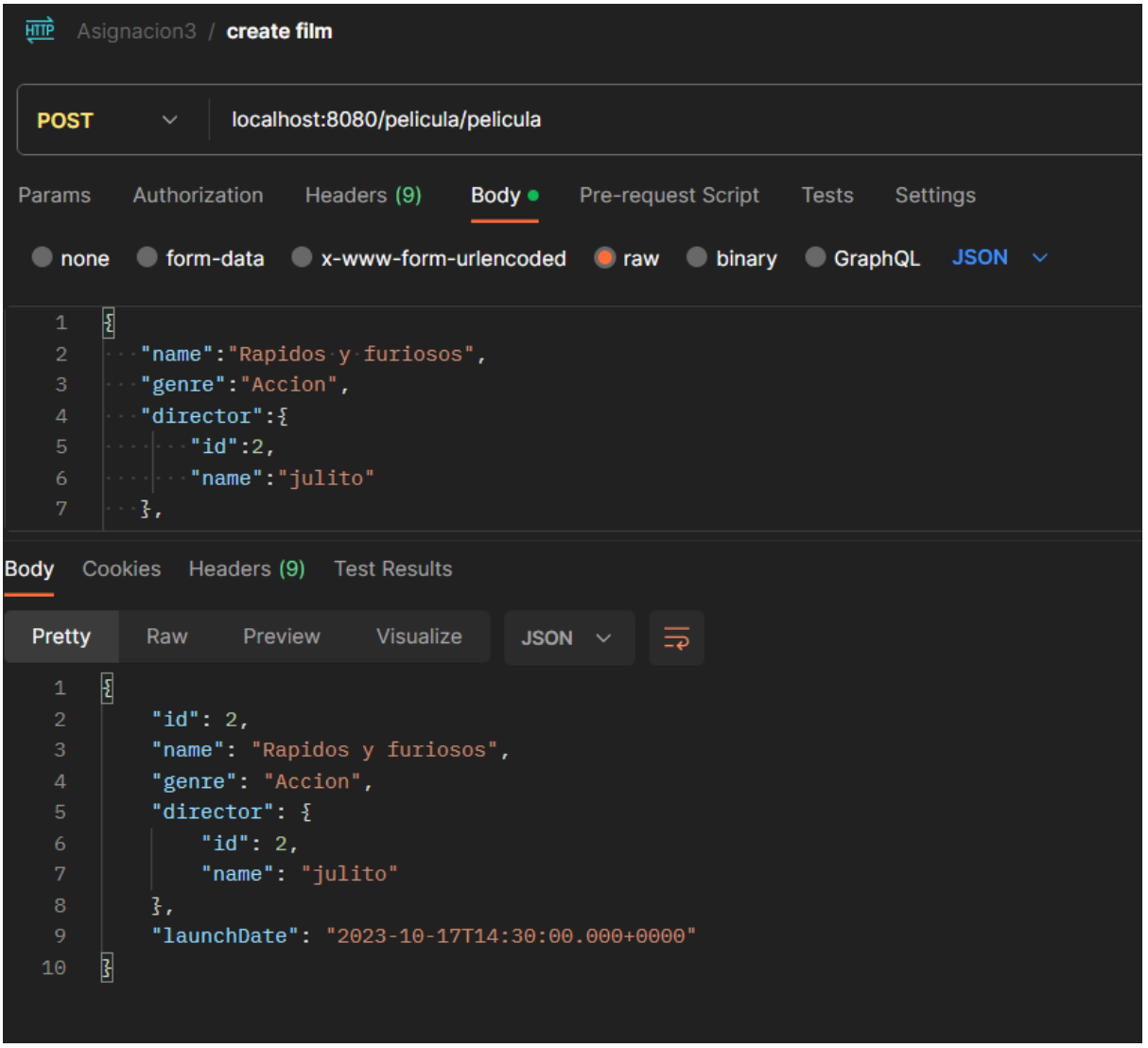

**DELETE /director/{id}: Eliminar un director:**
URL:*localhost:8080/pelicula/pelicula*




Eliminamos el director correspondiente al id pasado por la URL.
Listamos los directores para verificar la eliminación y vemos que ya no se encuentra:

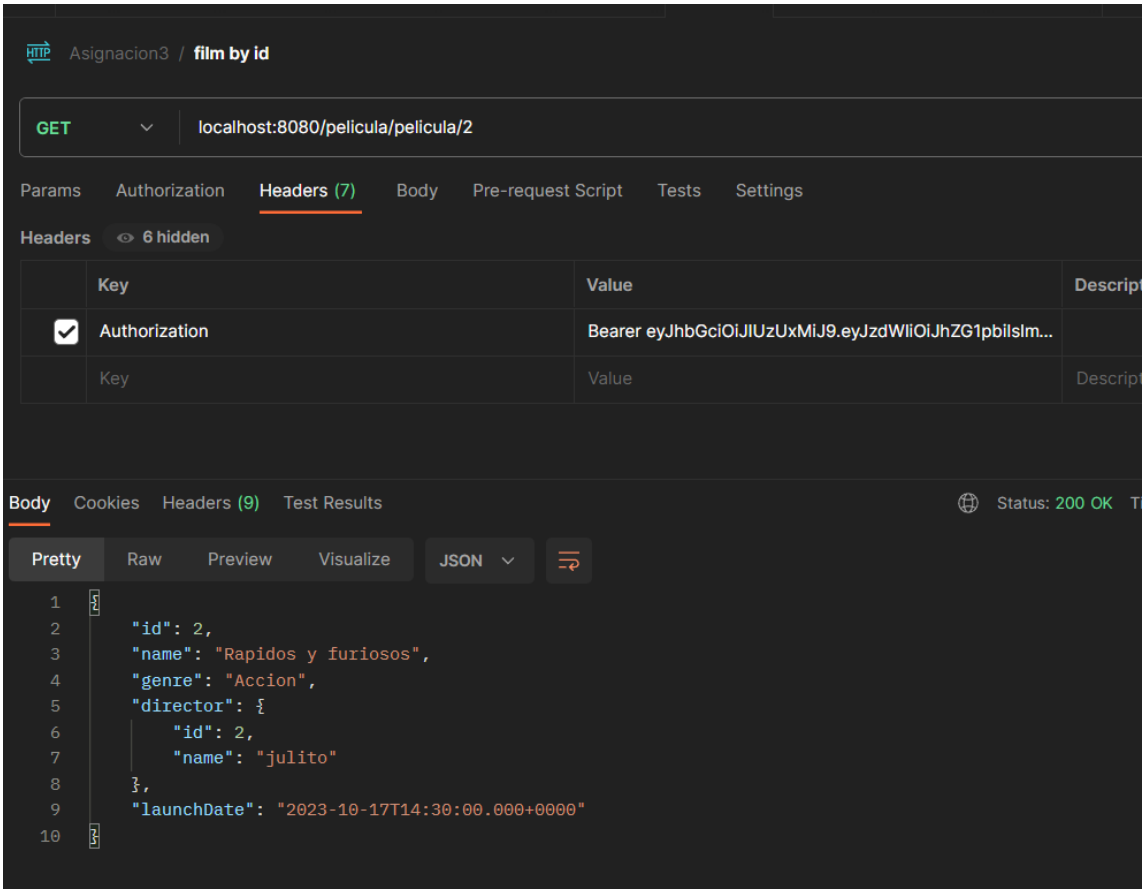


- **Pruebas ENDPOINTS para la entidad Peliculas:**
  **POST /pelicula: Crear ua nueva película:**
  URL:*localhost:8080/pelicula/pelicula*



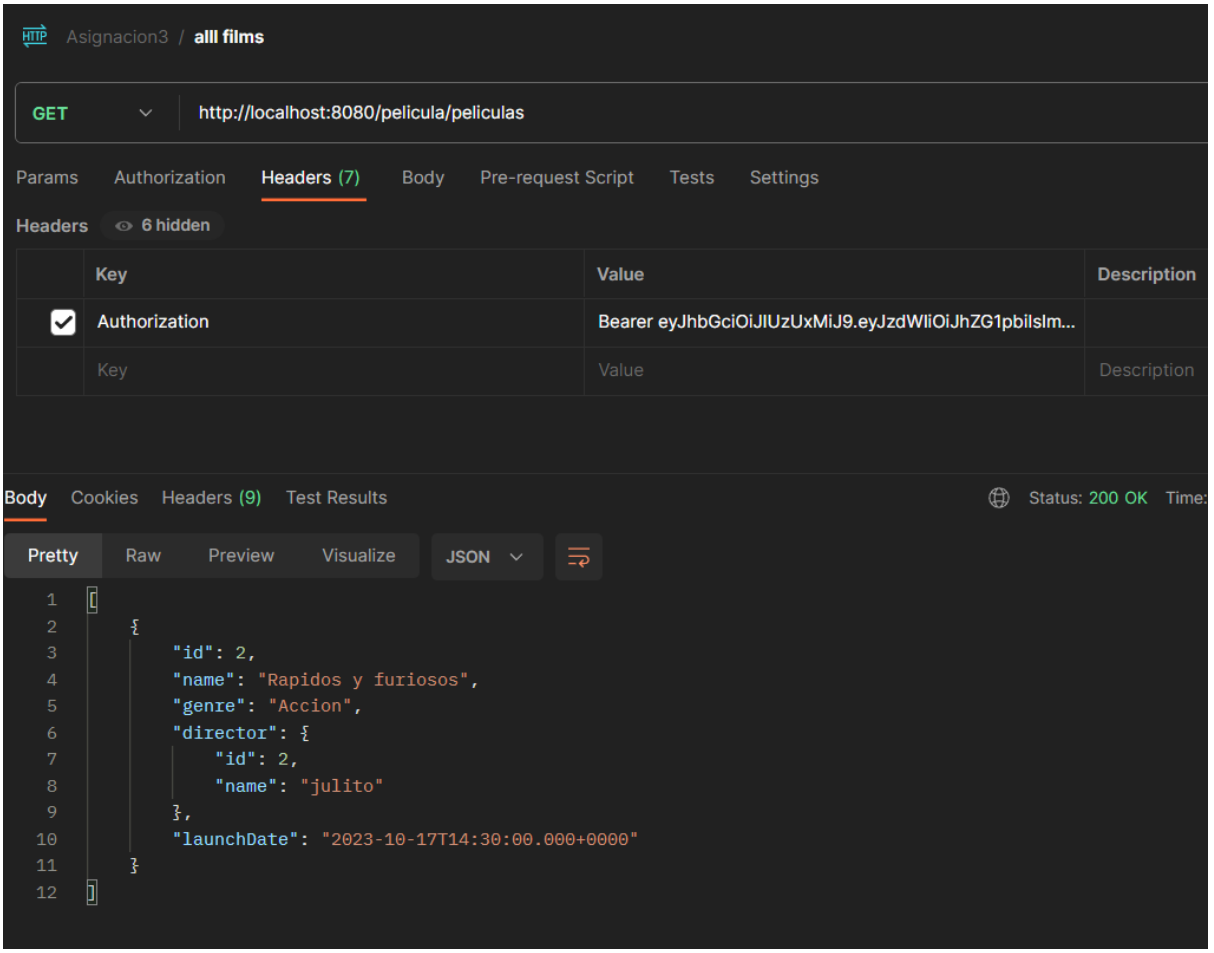

Enviamos los datos para crear una nueva pelicula y nos devuelve el objeto creado con el id asignado.

**GET /pelicula/{id}: Obtener detalles de una pelicula específica:**
URL:*localhost:8080/pelicula/pelicula/2*



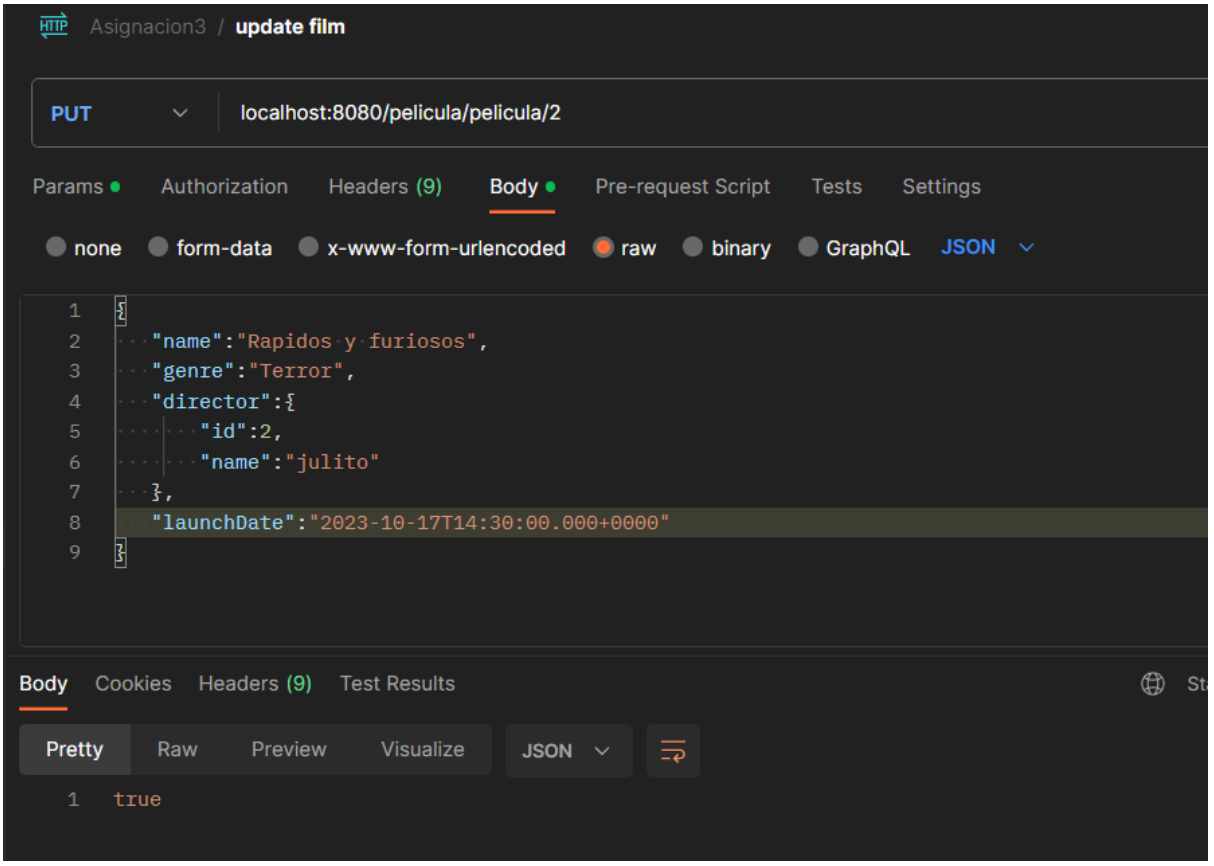

Se prueba buscando la película recién registrada de id=2, la encuentra y nos devuelve el objeto.

**GET /peliculas: Listar todas las peliculas:**
URL:*localhost:8080/pelicula/peliculas*



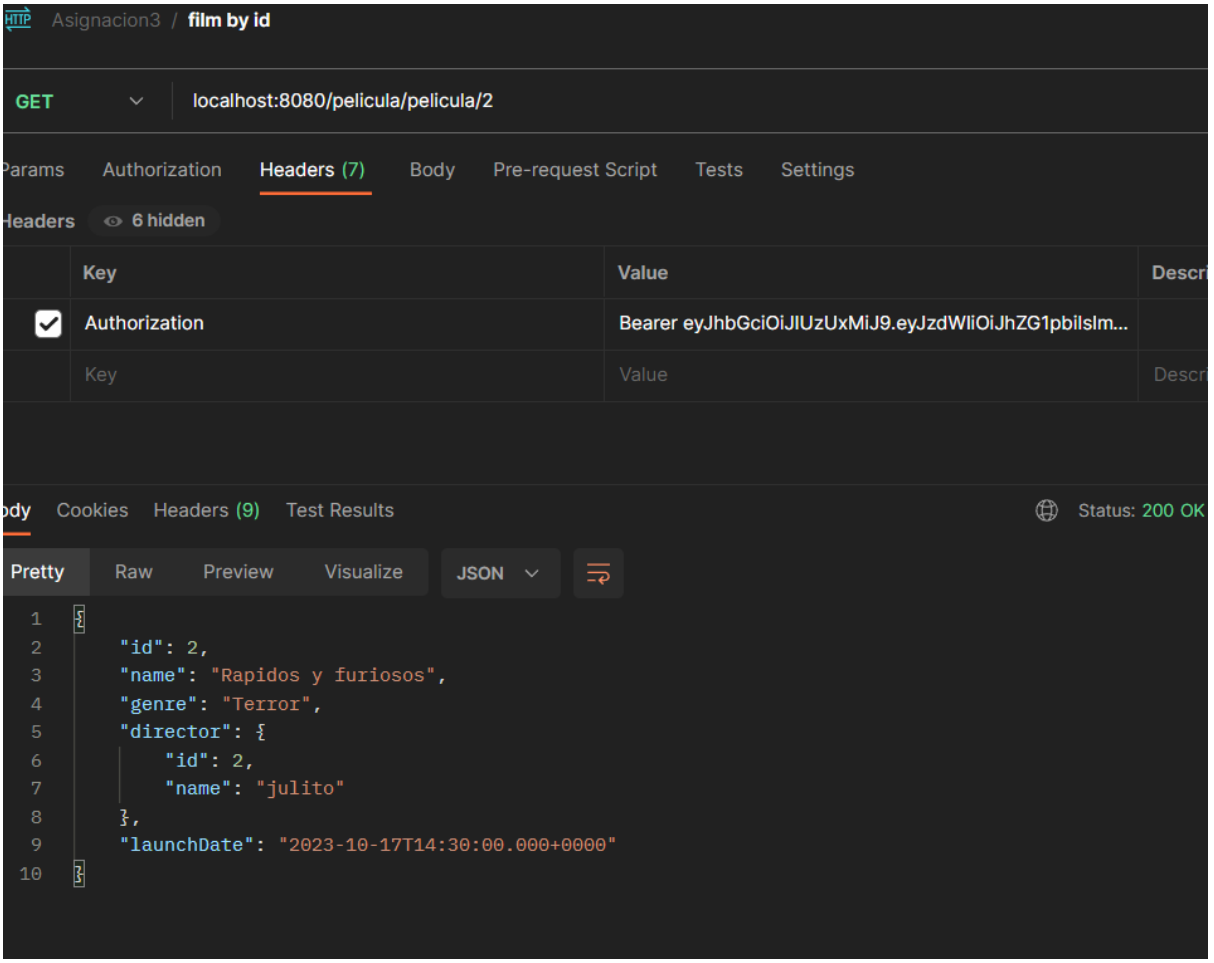

Retorna la lista de películas, en este caso solo se ha agregado una película.

**PUT /pelicula/{id}: Actualizar una pelicula existente:**
URL:*localhost:8080/pelicula/pelicula/2*



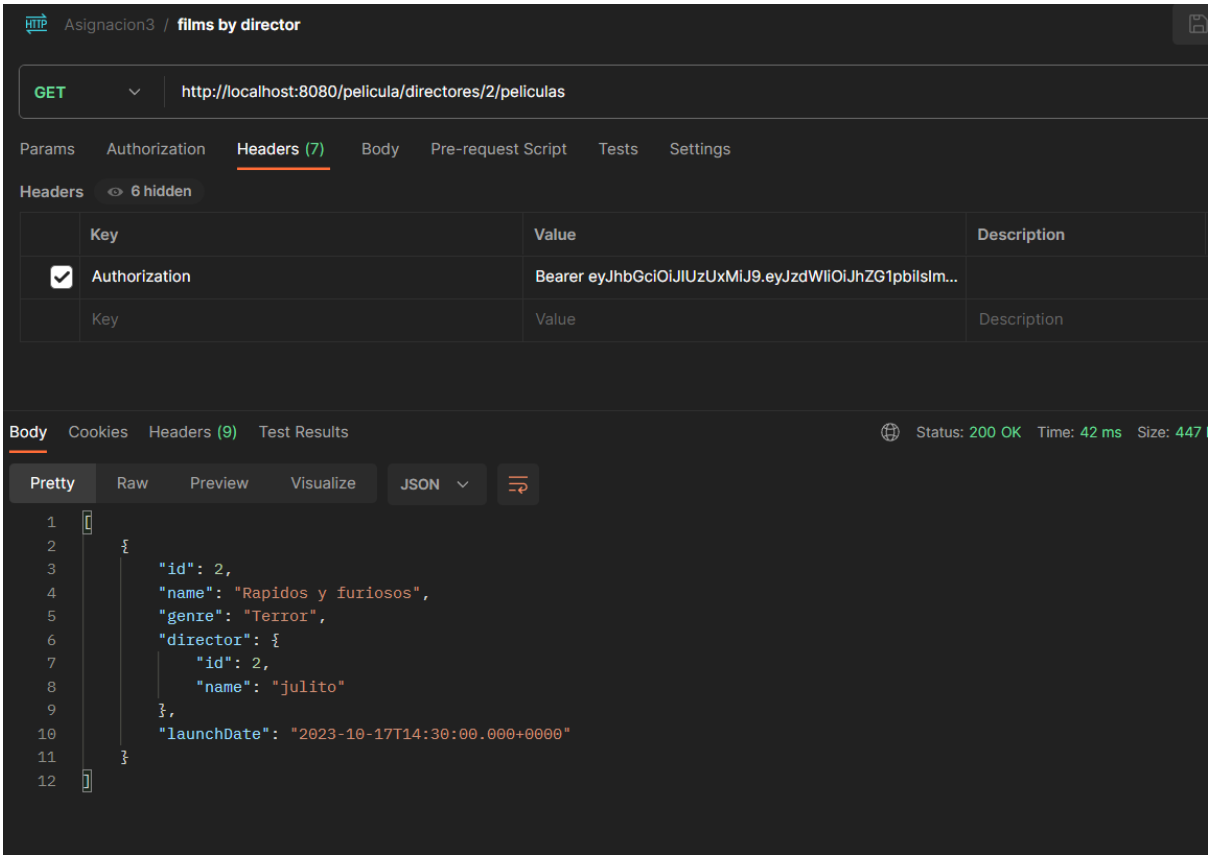

En este caso cambiamos la película de id=2 , se modificó el género obteniendo el nuevo valor terror. true indica que la película se actualizó.

Verificamos buscándola por el id y se observa la actualización:



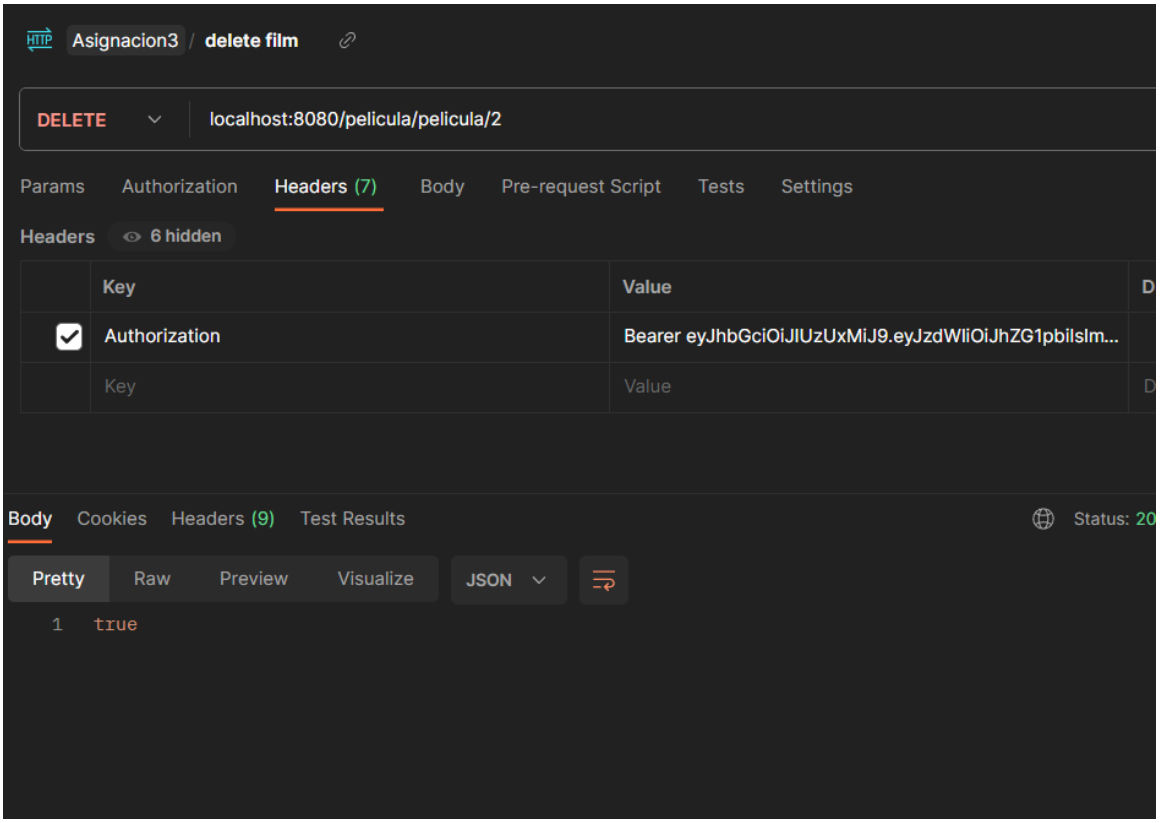

**GET /directores/{id}/peliculas: Listar las peliculas de un director específico.**

URL:*localhost:8080/pelicula/directores/2/peliculas*



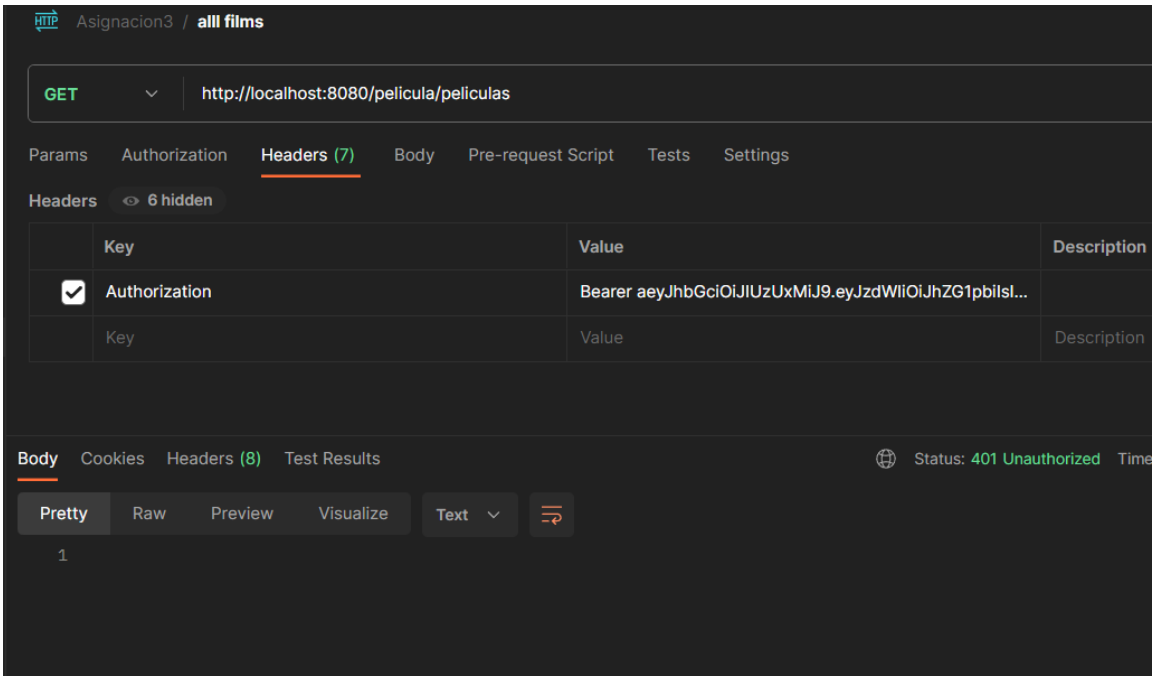

Pasando el id del director como parámetro en la URL, obtenemos las películas que ha dirigido, en este caso solo Rapidos y furiosos.

**DELETE /pelicula/{id}: Eliminar una pelicula:**
URL:*localhost:8080/pelicula/pelicula/2*




Eliminamos la película correspondiente al id pasado en la URL. Recibimos un true indicando que la operación se realizó con éxito

Listamos las películas para verificar la validación y notamos que no se encuentra:



## Conclusiones:

Este proyecto de desarrollo de una aplicación basada en Spring Boot, destinada a la gestión de películas y directores a través de una API REST, junto con la implementación de un sólido sistema de autenticación mediante JSON Web Tokens (JWT), representa un avance en nuestro proceso de aprendizaje. A lo largo de este proceso, se ha logrado diseñar un modelo de datos eficiente, implementar una serie de endpoints funcionales que permiten una gestión completa de películas y directores, y garantizar la seguridad y autorización de los usuarios a través del sistema de autenticación JWT.

Las pruebas llevadas a cabo en Postman durante el desarrollo han asegurado la calidad y la fiabilidad del sistema, verificando la funcionalidad de nuestros servicios en los endpoints, así como para garantizar la integridad y confidencialidad de la información. Este proyecto representa una base sólida para futuros desarrollos y su potencial implementación en aplicaciones del mundo real, ya que brinda la flexibilidad necesaria para adaptarse a diversos escenarios y requisitos.