

Informe Taller 3

- David Esteban Peñaranda
- Ana Sofia Londoño
- Samuel Hernández

Ejecución proyecto:

Para poner en funcionamiento este proyecto, primero hay que ejecutarlo y después se realizan las solicitudes de prueba con Postman. Los métodos y endpoints implementados se ajustan a las especificaciones mencionadas en el enunciado. Una vez que la aplicación esté en ejecución, puedes utilizar Postman para acceder a los endpoints, crear, actualizar, eliminar autores y libros, y obtener detalles de autores y libros específicos.

Introducción:

Este proyecto se centra en el desarrollo de una aplicación web utilizando el framework Spring Boot. La aplicación gestiona datos de autores y libros, incluyendo operaciones de creación, lectura, actualización y eliminación (CRUD) para ambas entidades, y requiere autenticación mediante tokens JWT para acceder a los endpoints protegidos.

Modelo de Datos

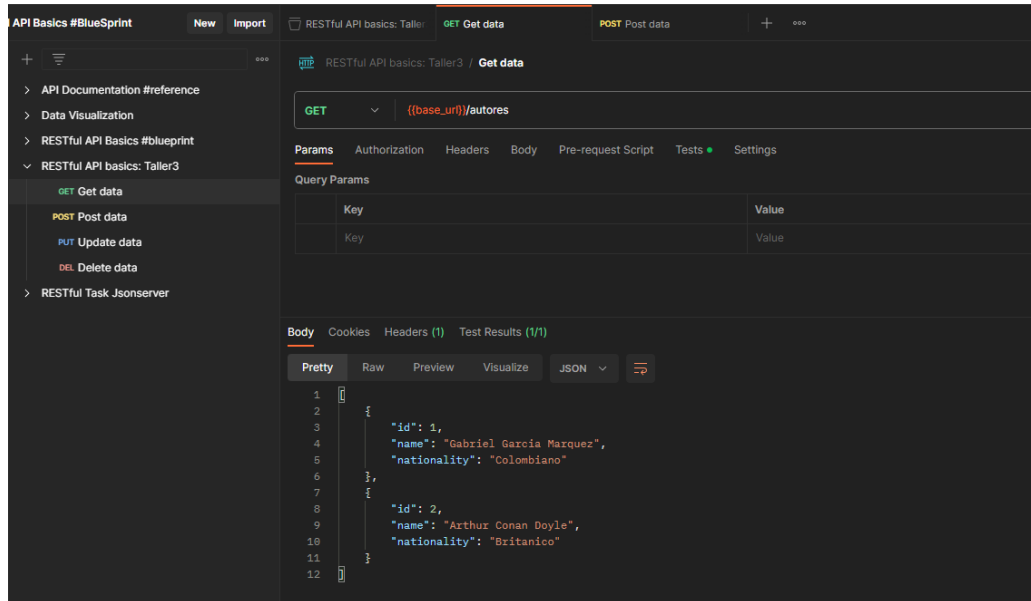
- Autor (Author): Cada instancia de la entidad "Autor" se caracteriza por tener tres atributos: un identificador único (id), un nombre (name), y la nacionalidad del autor (nationality).
- Libro (Book): La entidad "Libro" contiene información detallada sobre los libros. Cada libro posee un identificador único (id), un título (title), una fecha de publicación (releaseDate), y una referencia al autor que escribió el libro (author), estableciendo así una relación entre libros y autores.
- Usuario (User): La clase "Usuario" almacena información de autenticación. Cada instancia de esta clase contiene dos atributos: un nombre de usuario (username) y una contraseña (password), que se utilizan para la autenticación de usuarios en el sistema.

Endpoints

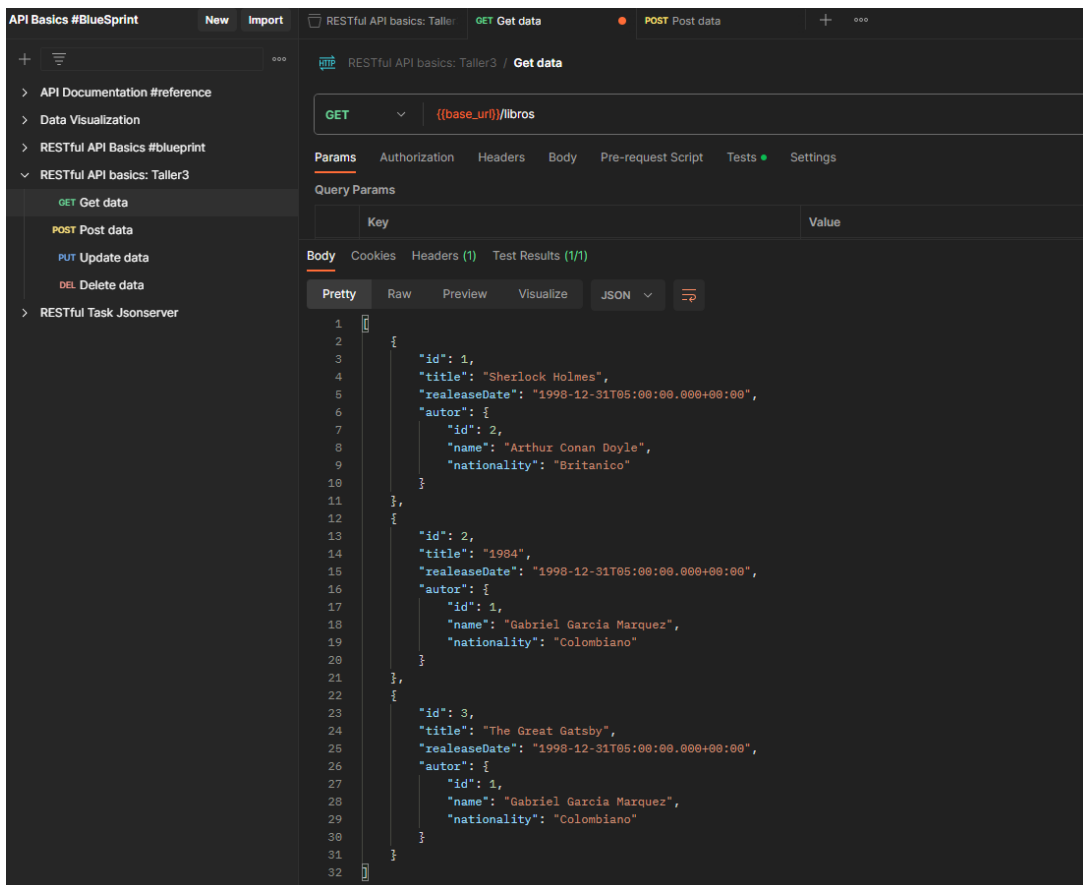
Los endpoints definidos en el proyecto siguen siendo los mismos y se prueban utilizando la herramienta Postman. Los endpoints incluyen:

Comprobación con Postman sin autenticación

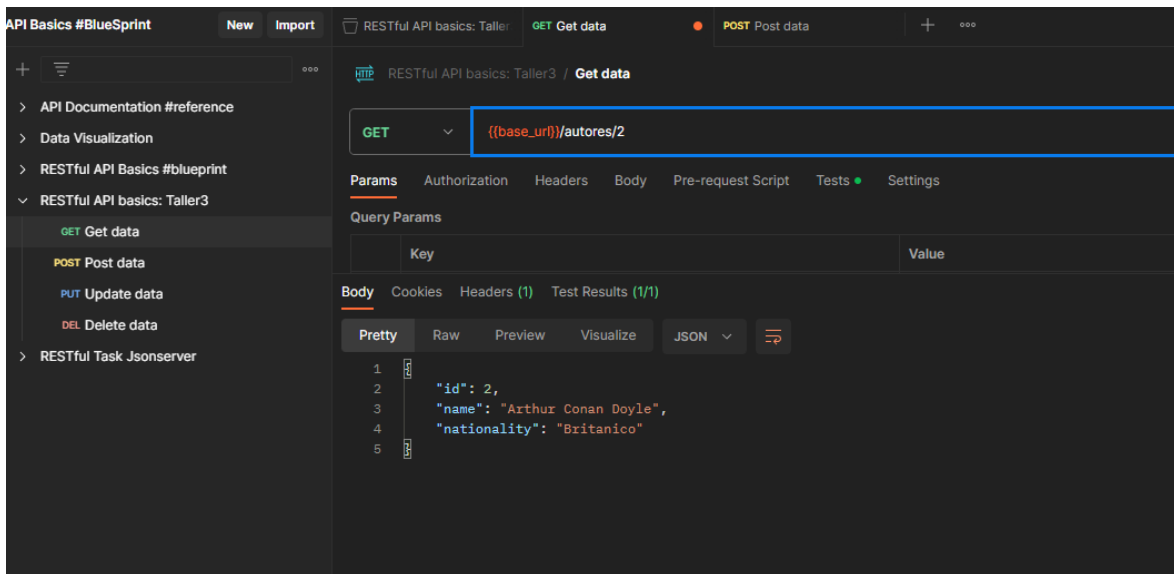
Autores



Libros



Autor por ID



RESTful API basics: Taller3 / Get data

GET `{{base_url}}/autores/2`

Params Authorization Headers Body Pre-request Script Tests Settings

Query Params

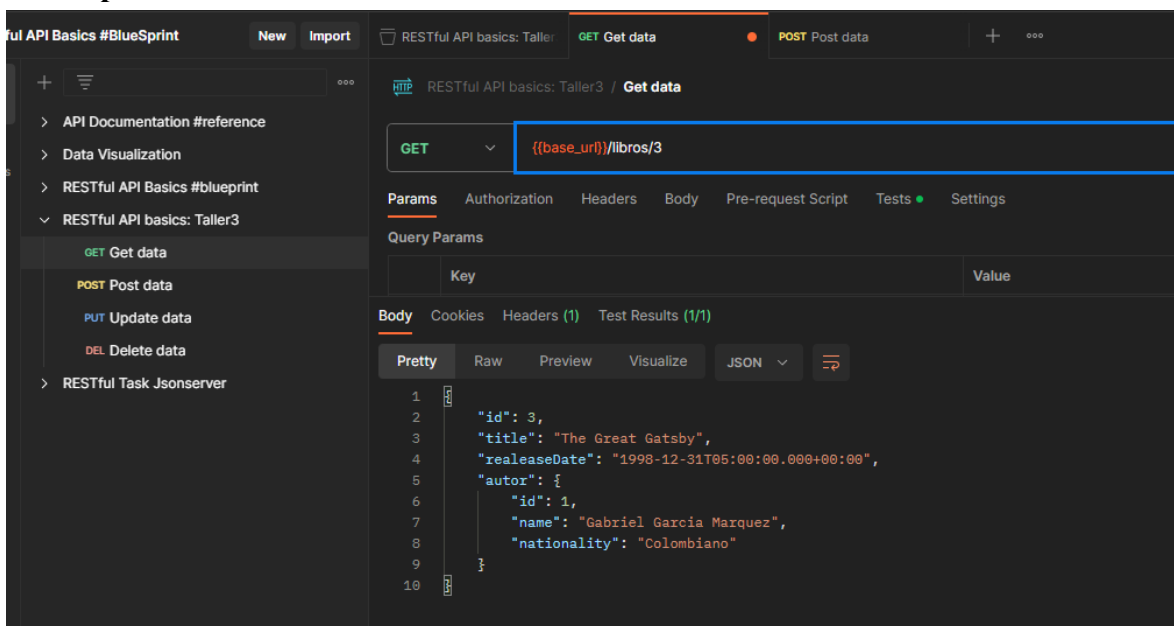
Key	Value
-----	-------

Body Cookies Headers (1) Test Results (1/1)

Pretty Raw Preview Visualize JSON

```
1  {
2    "id": 2,
3    "name": "Arthur Conan Doyle",
4    "nationality": "Britanico"
5  }
```

Libros por ID



RESTful API basics: Taller3 / Get data

GET `{{base_url}}/libros/3`

Params Authorization Headers Body Pre-request Script Tests Settings

Query Params

Key	Value
-----	-------

Body Cookies Headers (1) Test Results (1/1)

Pretty Raw Preview Visualize JSON

```
1  {
2    "id": 3,
3    "title": "The Great Gatsby",
4    "releaseDate": "1998-12-31T05:00:00.000+00:00",
5    "autor": {
6      "id": 1,
7      "name": "Gabriel Garcia Marquez",
8      "nationality": "Colombiano"
9    }
10 }
```

Añadir autor

The screenshot shows the Postman interface with a REST client project named "RESTful API basics: Taller3". The left sidebar lists the API endpoints: "GET Get data", "POST Post data", "PUT Update data", and "DELETE Delete data". The main panel displays the "POST" request to the endpoint `{{base_url}}/autores`. The "Body" tab is selected, showing a JSON payload:

```
1 {
2   "id": "3",
3   "name": "George R. R. Martin",
4   "nationality": "Estadounidense"
5 }
```

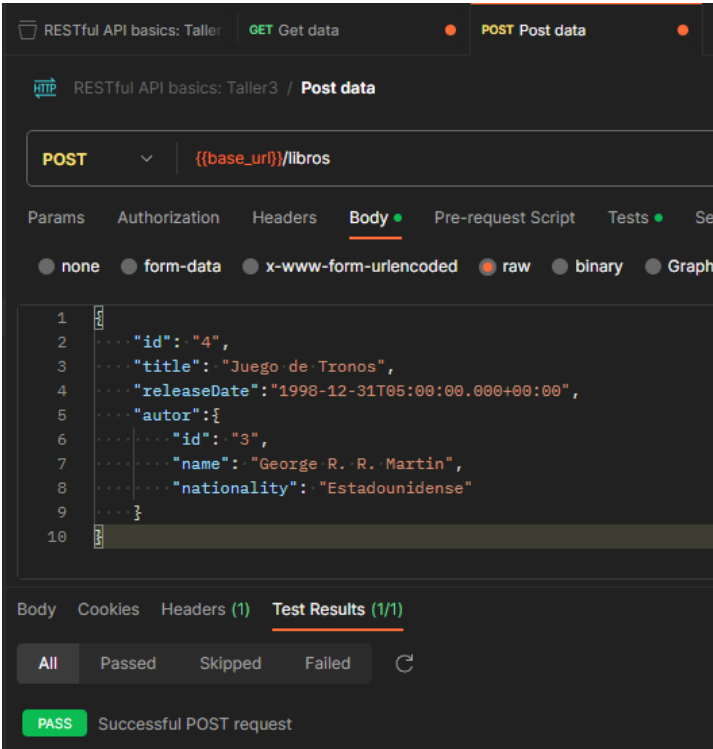
Below the body, the "Test Results (1/1)" section shows a "PASS" status with the message "Successful POST request".

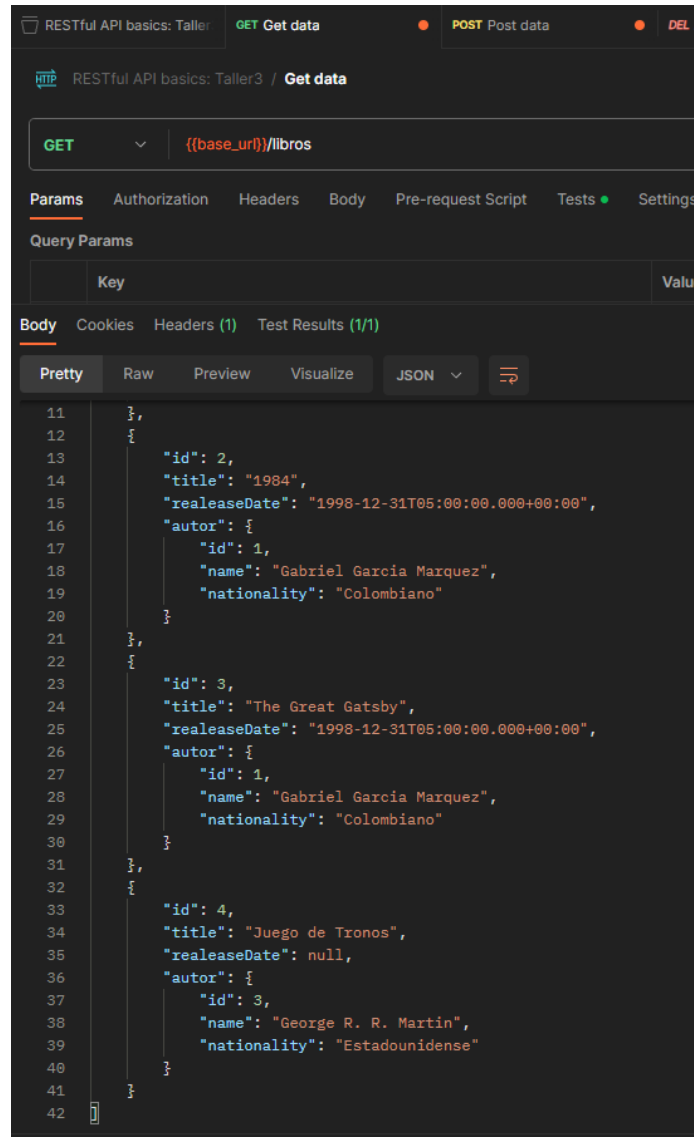
The screenshot shows the Postman interface with the same REST client project. The main panel displays the "GET" request to the endpoint `{{base_url}}/autores`. The "Body" tab is selected, showing a JSON array of three authors:

```
1 [
2   {
3     "id": 1,
4     "name": "Gabriel Garcia Marquez",
5     "nationality": "Colombiano"
6   },
7   {
8     "id": 2,
9     "name": "Arthur Conan Doyle",
10    "nationality": "Britanico"
11  },
12  {
13    "id": 3,
14    "name": "George R. R. Martin",
15    "nationality": "Estadounidense"
16  }
17 ]
```

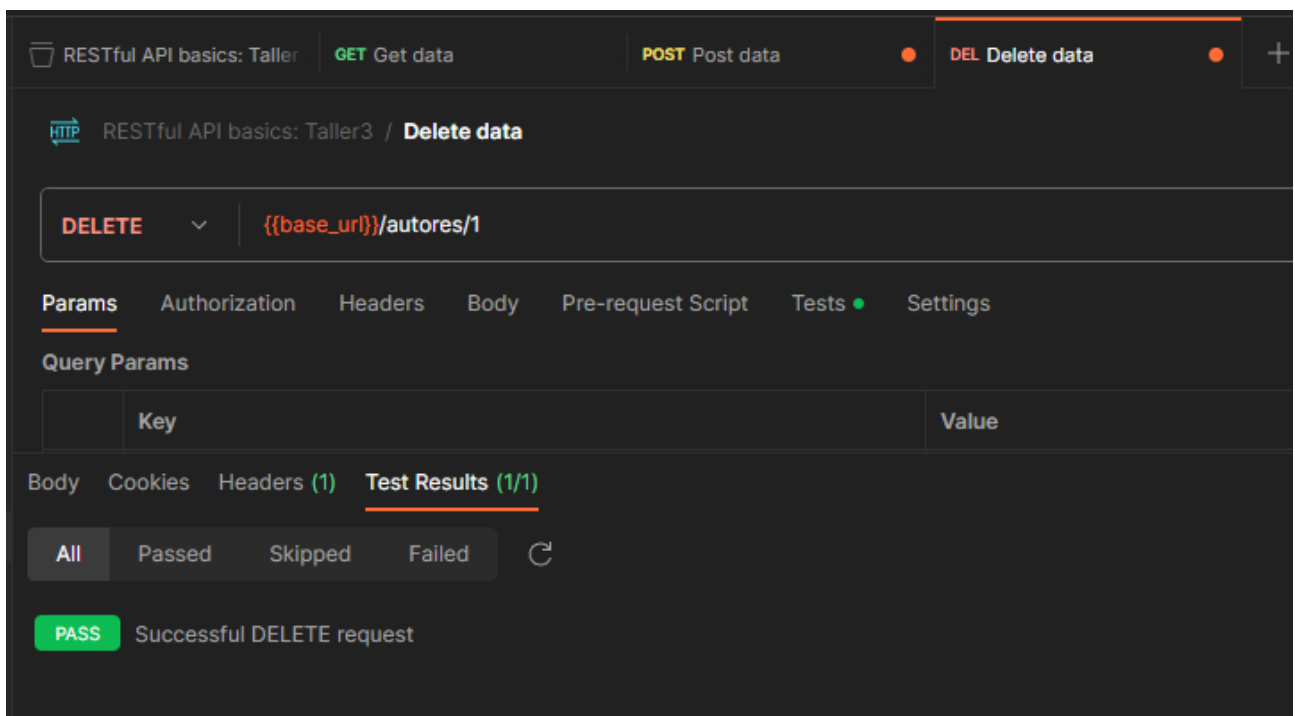
The "Test Results (1/1)" section shows a "PASS" status with the message "Successful GET request".

Añadir libro





Eliminar autor



RESTful API basics: Taller3

GET Get data

POST Post data

DEL Delete data

+

RESTful API basics: Taller3 / Get data

GET

{{base_url}}/autores

ParamsAuthorizationHeadersBodyPre-request ScriptTestsSettings

Query Params

	Key	Value
--	-----	-------

BodyCookiesHeaders (1)Test Results (1/1)

PrettyRawPreviewVisualizeJSON

```
1 {
2   {
3     "id": 2,
4     "name": "Arthur Conan Doyle",
5     "nationality": "Britanico"
6   },
7   {
8     "id": 3,
9     "name": "George R. R. Martin",
10    "nationality": "Estadounidense"
11  }
12 }
```

Eliminar libro

RESTful API basics: Taller3

GET Get data

POST Post data

DEL Delete data

+

RESTful API basics: Taller3 / Delete data

DELETE

{{base_url}}/libros/2

ParamsAuthorizationHeadersBodyPre-request ScriptTestsSettings

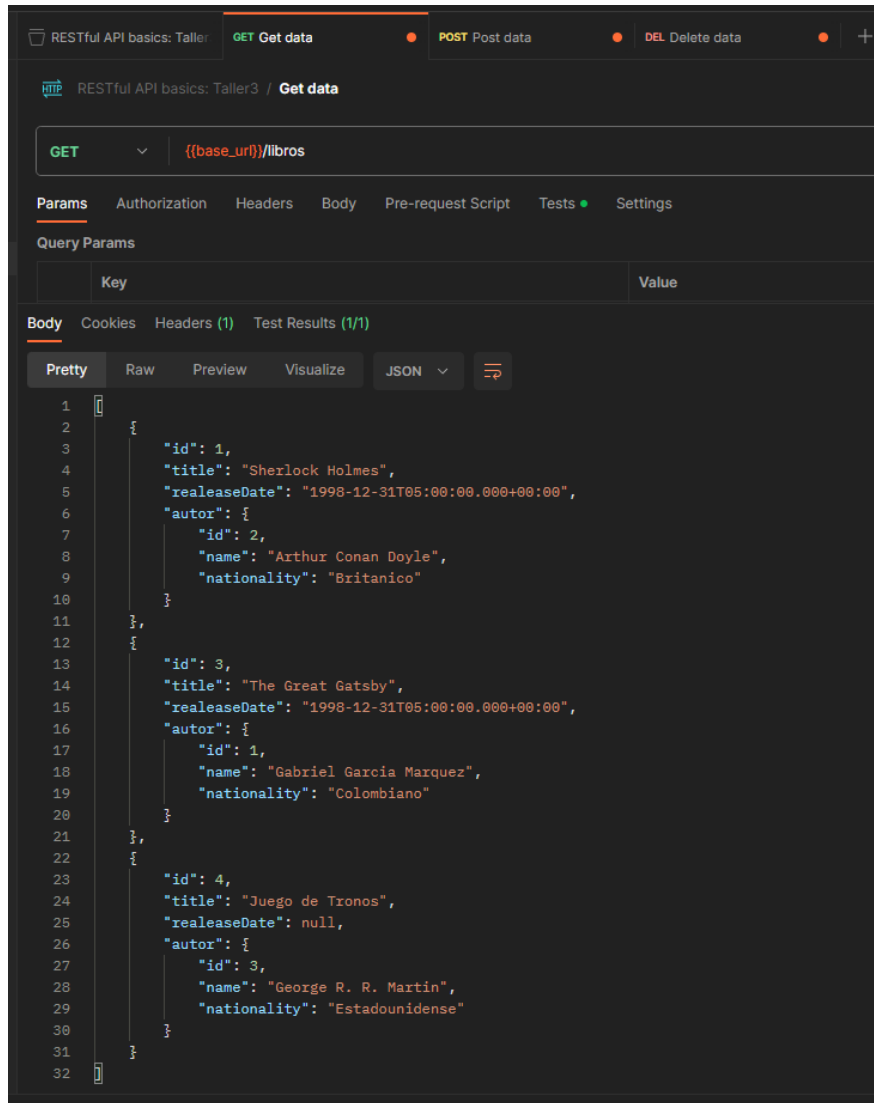
Query Params

	Key	Value
--	-----	-------

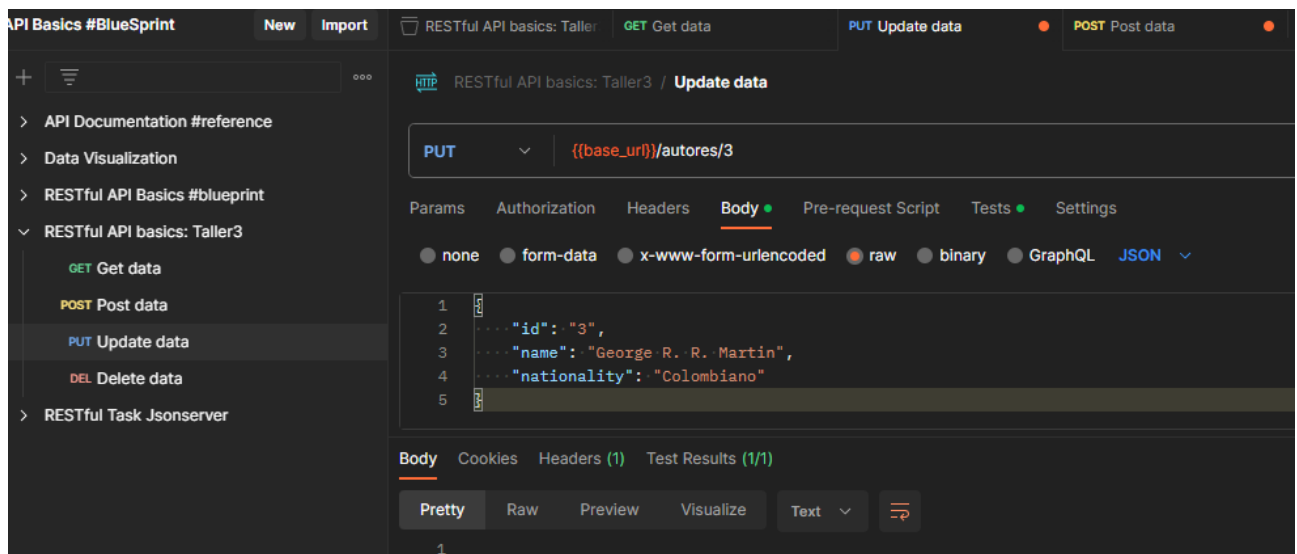
BodyCookiesHeaders (1)Test Results (1/1)

AllPassedSkippedFailed

PASS Successful DELETE request



Editar autor



RESTful API basics: Taller

GET Get data

PUT Update data

POST Post data

DELETE Delete data

RESTful API basics: Taller3 / Get data

GET

▼

{{base_url}}/autores

ParamsAuthorizationHeadersBodyPre-request ScriptTests●Settings

Query Params

	Key	Value
--	-----	-------

BodyCookiesHeaders (1)Test Results (1/1)

PrettyRawPreviewVisualizeJSON▼↻

1

2

3

4

5

6

7

8

9

10

11

12

{

"id": 2,

"name": "Arthur Conan Doyle",

"nationality": "Britanico"

},

{

"id": 3,

"name": "George R. R. Martin",

"nationality": "Colombiano"

}

}

Editar libro

RESTful API basics: Taller

GET Get data

PUT Update data

POST Post data

DEL Del

RESTful API basics: Taller3 / Update data

PUT

{{base_url}}/libros/3

Params

Authorization

Headers

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {
2   ...."id": 3,
3   ...."title": "Cien años de soledad",
4   ...."releaseDate": "1998-12-31T05:00:00.000+00:00",
5   ...."autor": {
6     ...."id": 1,
7     ...."name": "Gabriel Garcia Marquez",
8     ...."nationality": "Colombiano"
9   }
10 }
```

Body

Cookies

Headers (1)

Test Results (1/1)

Pretty

Raw

Preview

Visualize

Text

1

RESTful API basics: Taller

GET Get data

PUT Update data

POST Post data

DEL Delete

RESTful API basics: Taller3 / Get data

GET

{{base_url}}/libros

ParamsAuthorizationHeadersBodyPre-request ScriptTestsSettings

Query Params

Key	Value
-----	-------

BodyCookiesHeaders (1)Test Results (1/1)

PrettyRawPreviewVisualizeJSON

```
11  },
12  {
13    "id": 2,
14    "title": "1984",
15    "releaseDate": "1998-12-31T05:00:00.000+00:00",
16    "autor": {
17      "id": 1,
18      "name": "Gabriel Garcia Marquez",
19      "nationality": "Colombiano"
20    }
21  },
22  {
23    "id": 4,
24    "title": "Juego de Tronos",
25    "releaseDate": null,
26    "autor": {
27      "id": 3,
28      "name": "George R. R. Martin",
29      "nationality": "Colombiano"
30    }
31  },
32  {
33    "id": 3,
34    "title": "Cien años de soledad",
35    "releaseDate": "1998-12-31T05:00:00.000+00:00",
36    "autor": {
37      "id": 1,
38      "name": "Gabriel Garcia Marquez",
39      "nationality": "Colombiano"
40    }
41  }
42 }
```

Libros por autor

RESTful API basics: Taller

GET Get data

PUT Update data

POST Post data

DEL Delete

RESTful API basics: Taller3 / Get data

GET

{{base_url}}/autores/1/libros

Params

Authorization

Headers

Body

Pre-request Script

Tests

Settings

Query Params

Key	Value
-----	-------

Body

Cookies

Headers (1)

Test Results (1/1)

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": 2,
3   "title": "1984",
4   "releaseDate": "1998-12-31T05:00:00.000+00:00",
5   "autor": {
6     "id": 1,
7     "name": "Gabriel Garcia Marquez",
8     "nationality": "Colombiano"
9   }
10 },
11 {
12   "id": 3,
13   "title": "Cien años de soledad",
14   "releaseDate": "1998-12-31T05:00:00.000+00:00",
15   "autor": {
16     "id": 1,
17     "name": "Gabriel Garcia Marquez",
18     "nationality": "Colombiano"
19   }
20 }
21 }
22 }
```

Autenticación JWT

RESTful API Basics #blueprint / Get data

GET

http://localhost:8082/autores

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Query Params

Key	Value
Key	Value

Body

Cookies

Headers (10)

Test Results (0/1)

All

Passed

Skipped

Failed

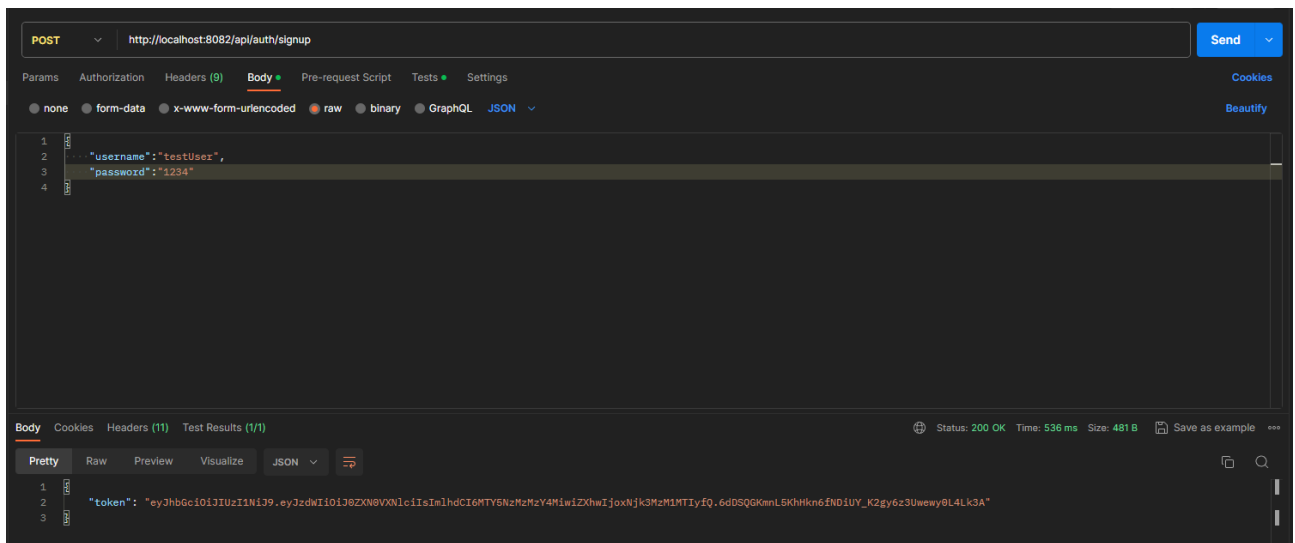
FAIL

Status code is 200 | AssertionError: expected response to have status code 200 but got 401

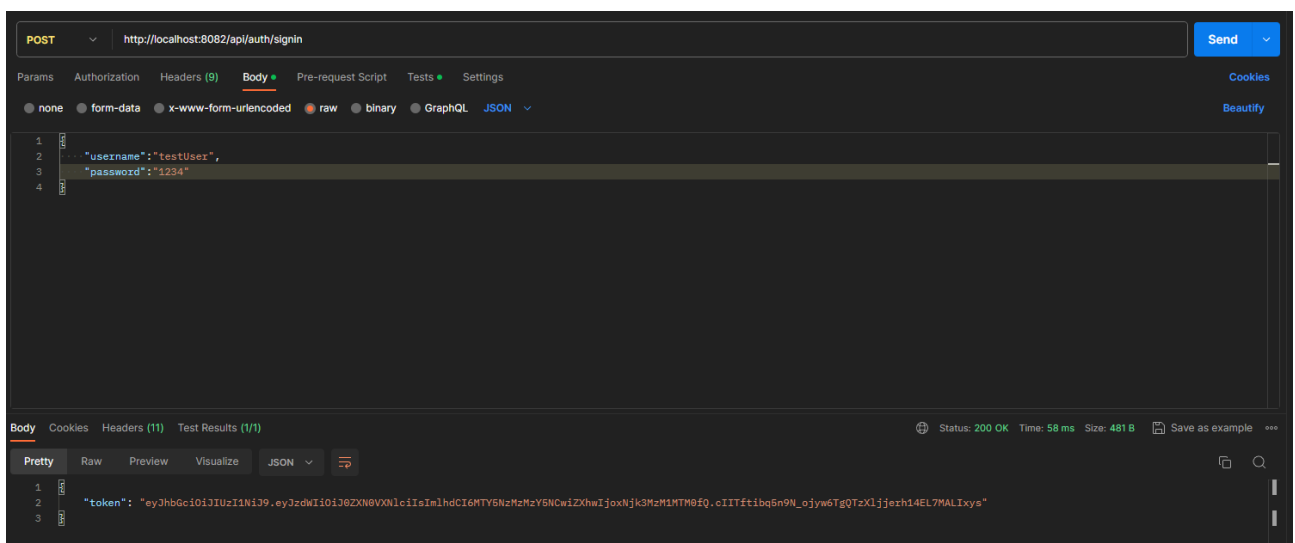
Autenticación JWT

Para fortalecer la seguridad de la aplicación, hemos implementado un sistema de autenticación basado en tokens JWT (JSON Web Tokens). Se ha creado un endpoint POST /auth que permite a los usuarios autenticarse proporcionando un nombre de usuario y una contraseña. Si las credenciales son válidas, el sistema genera un token JWT que se utiliza para acceder a los endpoints protegidos. En otras palabras, todos los endpoints, excepto /auth, se protegen mediante autenticación. Para acceder a los recursos protegidos, los usuarios deben proporcionar un token JWT válido en la cabecera de sus solicitudes. Esto garantiza que solo los usuarios autorizados puedan acceder a los datos y realizar operaciones CRUD.

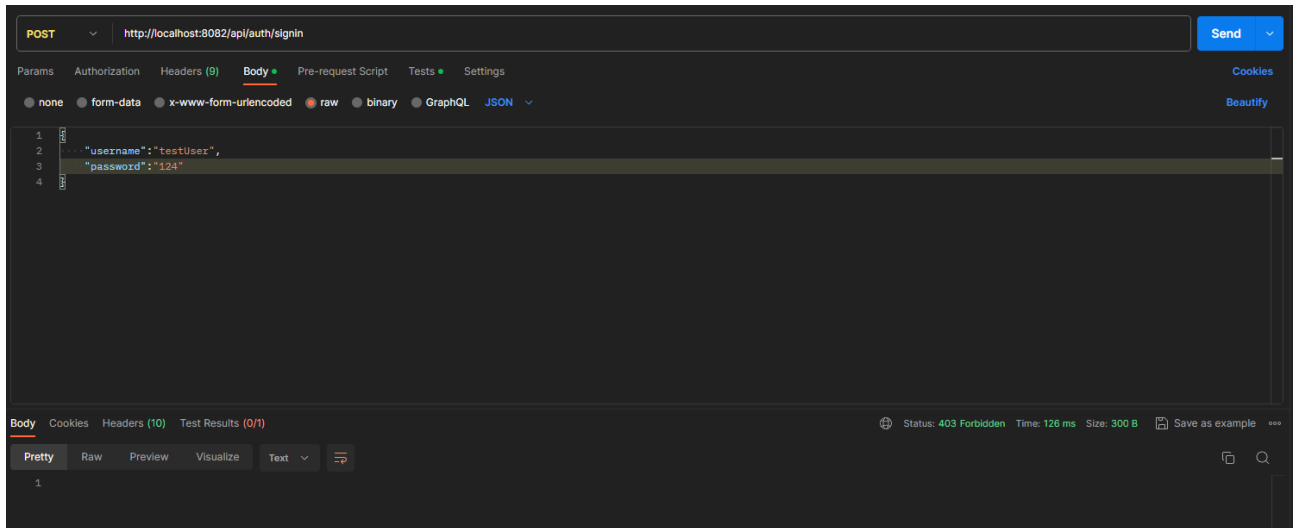
Para empezar, se puede registrar un nuevo usuario en `api/auth/signup`:



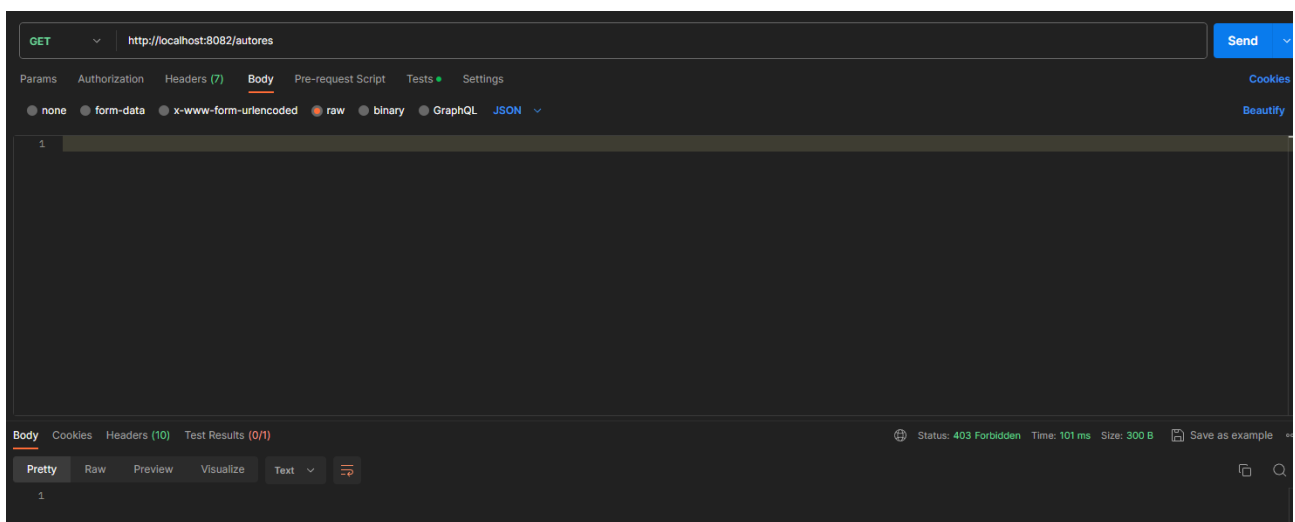
La correcta ejecución del registro retorna un token de usuario. Ahora se puede iniciar sesión con el usuario en `api/auth/signin`:



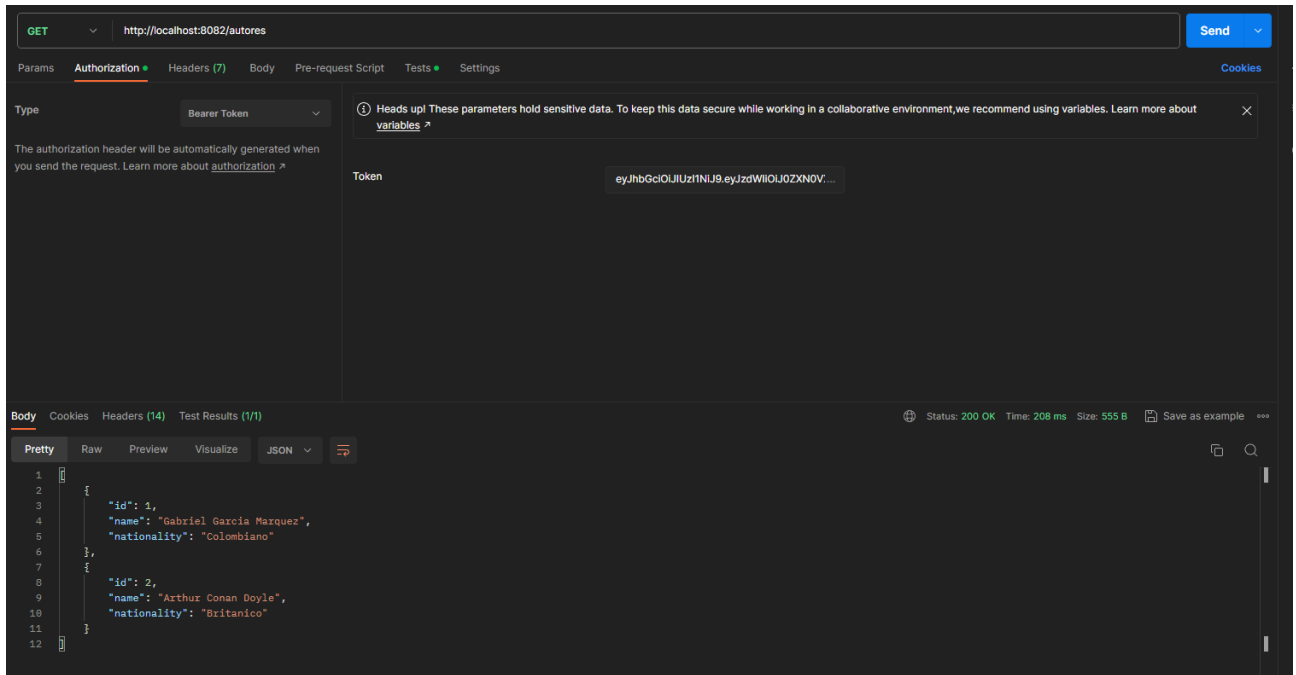
Nótese que el token obtenido no es siempre el mismo, es decir, que se generan dinámicamente. Si se ingresa una contraseña incorrecta no se obtiene token, sino un error 403 (que es lo esperado):



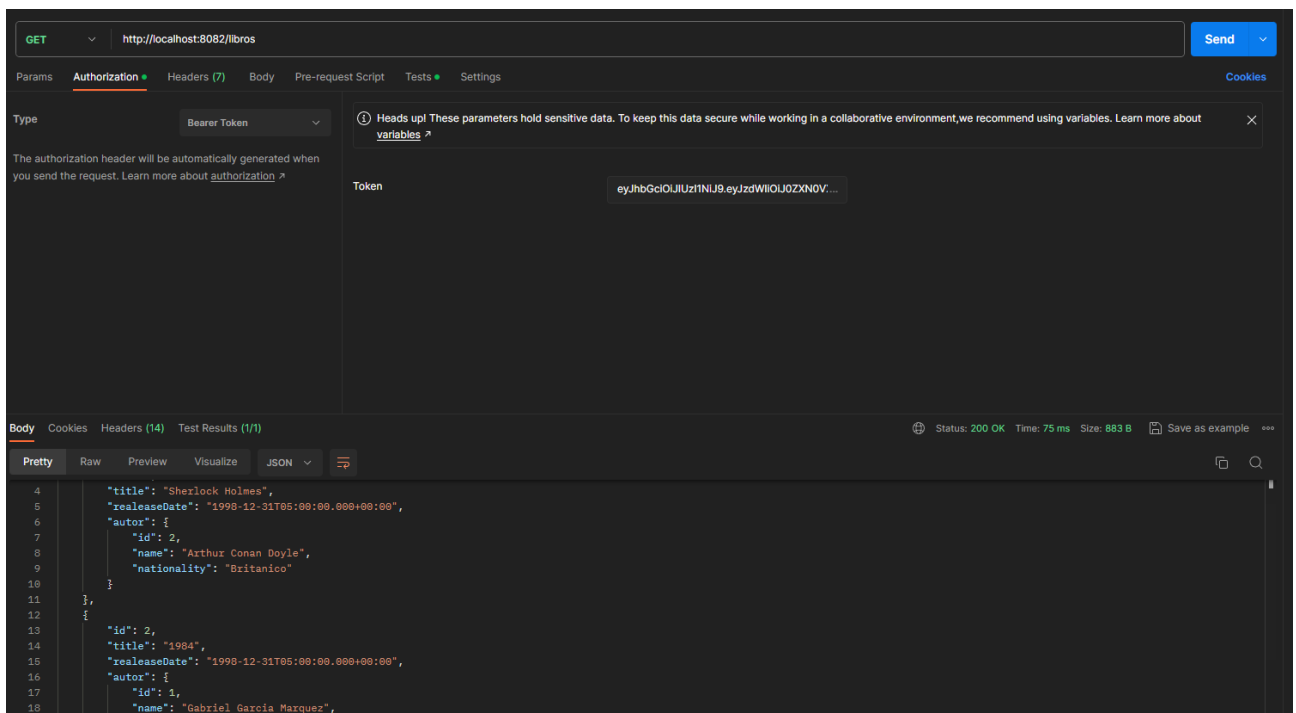
Las rutas anteriores son las únicas que no requieren de previa autenticación para ser accedidas. Es decir, que el usuario puede obtener un token válido para acceder a la api bien sea registrándose o iniciando sesión correctamente. Cualquier otra ruta requerirá un token en su cabecera. Por ejemplo, para solicitar la lista de autores:



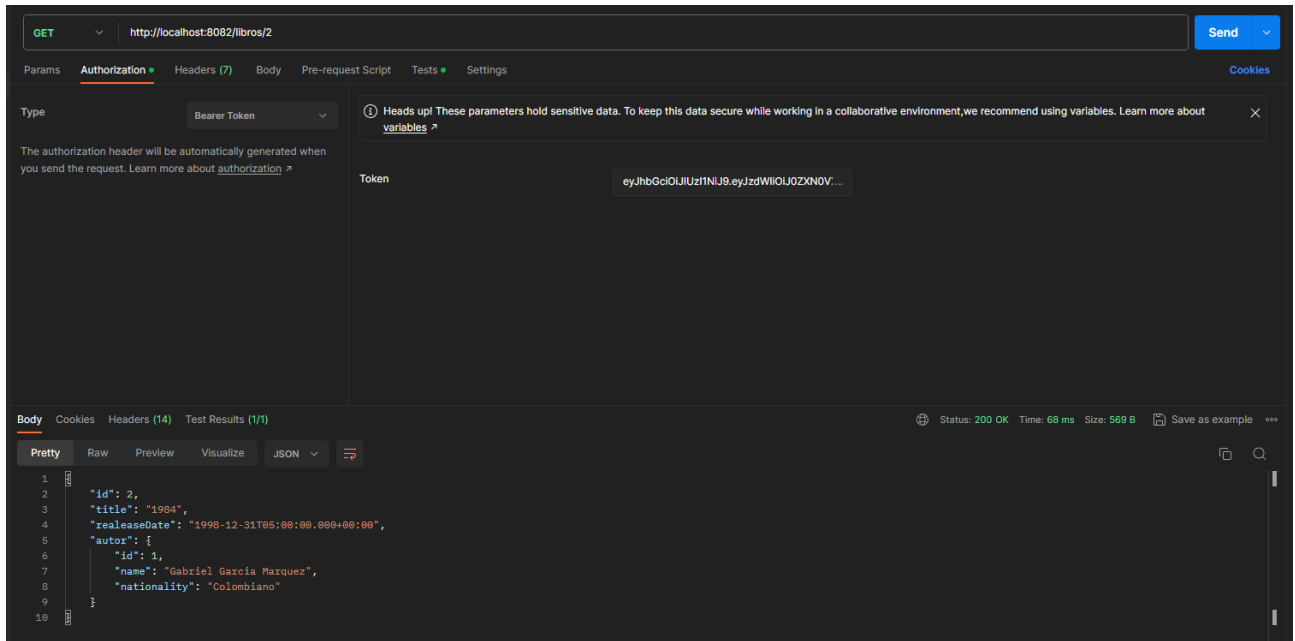
Se obtiene un 403 en vez de la lista de autores, porque la ruta está protegida y es necesario un token JWT válido para poder acceder a ella. Si en la la cabecera del GET se añade uno de los tokens obtenidos vía signup o signin entonces sí es posible acceder al recurso:



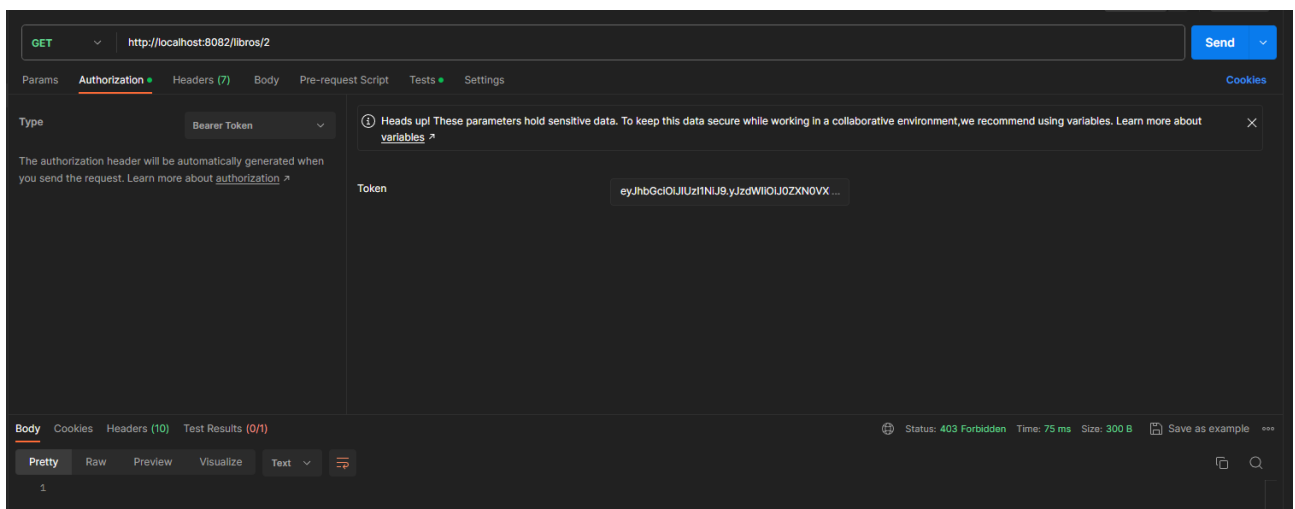
Tras añadir el token, el JwtFilter lo valida y concede el acceso. Lo mismo sucede con el endpoint de libros:



Y en consultas particulares (porque las rutas están anidadas en endpoints bloqueados).



Lo mismo aplica para cada ruta que no esté en auth (no tendría sentido que el usuario ya contara con credenciales de acceso si apenas va a registrarse o a iniciar sesión para tener unas, aparte de esas rutas especiales todo sí requerirá de una credencial). Si por alguna razón se usa un token inválido el Filter no le concederá acceso:



Tras borrar un solo char se invalida completamente el token y se obtiene un erro 403.

Conclusión

Este proyecto nos proporciona una aplicación robusta para gestionar datos de autores y libros, con un sistema de autenticación JWT para garantizar la seguridad de los recursos. Los controladores, servicios y repositorios se encargan de la gestión de datos, y la autenticación JWT asegura que solo los usuarios autorizados puedan interactuar con la aplicación. La realización de pruebas con Postman garantiza el correcto funcionamiento de los endpoints y la seguridad de la aplicación.