

ACCESO A DATOS – DOCUMENTACIÓN PRUEBAS GRAN DAO



Índice

Descripción del proyecto.....	3
Diagrama de clases.....	3
Funcionamiento de la aplicación.....	4
Ventana de selección de jugadores.....	4
Ventana de inicio de sesión.....	5
Ventana de recuperación de contraseña.....	5
Ventana de registro.....	6
Tablero.....	8
Menú de fin de partida.....	8

Descripción del proyecto

El siguiente proyecto simula una aplicación de gestión de una biblioteca. Está realizado con SpringBoot. Las fuentes de datos son las siguientes:

Fuentes de datos

MariaDB

Una de las entidades con las que se trabaja es la de libros. Está almacenada en una base de datos relacional, concretamente en MariaDB. El script SQL es el siguiente.

```
-- Crear base de datos
CREATE DATABASE Biblioteca;
USE Biblioteca;

-- Tabla de Libros
CREATE TABLE Libros (
  libro_id INT PRIMARY KEY AUTO_INCREMENT,
  titulo VARCHAR(255) NOT NULL,
  genero VARCHAR(100),
  isbn VARCHAR(20) UNIQUE,
  cantidad INT
);

-- Insertar datos en la tabla Libros
INSERT INTO Libros (titulo, genero, isbn, cantidad) VALUES
  ( titulo 'El Quijote', genero 'Ficción', isbn '978-3-16-148410-0', cantidad 5),
  ( titulo 'Cien Años de Soledad', genero 'Realismo mágico', isbn '978-0-06-088328-7', cantidad 3),
  ( titulo '1984', genero 'Distopía', isbn '978-0-452-28423-4', cantidad 4),
  ( titulo 'Orgullo y Prejuicio', genero 'Romántico', isbn '978-1-59308-201-1', cantidad 2),
  ( titulo 'Crimen y Castigo', genero 'Psicológico', isbn '978-0-14-044913-6', cantidad 3),
  ( titulo 'Diálogos de Platón', genero 'Filosofía', isbn '978-0-19-283617-3', cantidad 4);
```

Script de la base de datos de MariaDB.

MongoDB

Otra de las entidades con las que se trabaja es la de autores. Está almacenada en una base de datos no relacional, concretamente en MongoDB. El fichero JSON posee el siguiente contenido:

```
{
  "_id": {
    "$oid": "67c4ade0d119f143d70a93b4"
  },
  "nombre": "Miguel de Cervantes",
  "pais_origen": "España"
},
{
  "_id": {
    "$oid": "67c4ade0d119f143d70a93b5"
  },
  "nombre": "Gabriel García Márquez",
  "pais_origen": "Colombia"
},
{
  "_id": {
    "$oid": "67c4ade0d119f143d70a93b6"
  },
  "nombre": "George Orwell",
  "pais_origen": "Reino Unido"
},
{
  "_id": {
    "$oid": "67c4ade0d119f143d70a93b7"
  },
  "nombre": "Jane Austen",
  "pais_origen": "Reino Unido"
},
{
  "_id": {
    "$oid": "67c4ade0d119f143d70a93b8"
  },
  "nombre": "Fiódor Dostoyevski",
  "pais_origen": "Rusia"
},
{
  "_id": {
    "$oid": "67c4ade0d119f143d70a93b9"
  },
  "nombre": "Platón",
  "pais_origen": "Grecia"
}
}
```

Script de la base de datos de MongoDB.

Ficheros de texto

Otra de las entidades con las que se trabaja es la de categorías. Está almacenada en un fichero de texto, cuyo contenido es el siguiente.

```
categoria_id,nombre_categoria
1,Ficción
2,Realismo mágico
3,Distopía
4,Romántico
5,Psicológico
6,Filosofía
```

Contenido del fichero de texto.

Ficheros XML

La última entidad con la que se trabaja es la de usuarios. Está almacenada en un fichero XML, cuyo contenido es el siguiente.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<usuarios>
  <usuario>
    <correo>juan.perez@example.com</correo>
    <nombre_usuario>Juan Pérez</nombre_usuario>
    <telefono>123456789</telefono>
    <usuario_id>1</usuario_id>
  </usuario>
  <usuario>
    <correo>ana.garcia@example.com</correo>
    <nombre_usuario>Ana García</nombre_usuario>
    <telefono>987654321</telefono>
    <usuario_id>2</usuario_id>
  </usuario>
  <usuario>
    <correo>carlos.lopez@example.com</correo>
    <nombre_usuario>Carlos López</nombre_usuario>
    <telefono>555666777</telefono>
    <usuario_id>3</usuario_id>
  </usuario>
  <usuario>
    <correo>maria.rodriguez@example.com</correo>
    <nombre_usuario>María Rodríguez</nombre_usuario>
    <telefono>444333222</telefono>
    <usuario_id>4</usuario_id>
  </usuario>
</usuarios>
```

Contenido del fichero XML.

Pruebas de funcionamiento

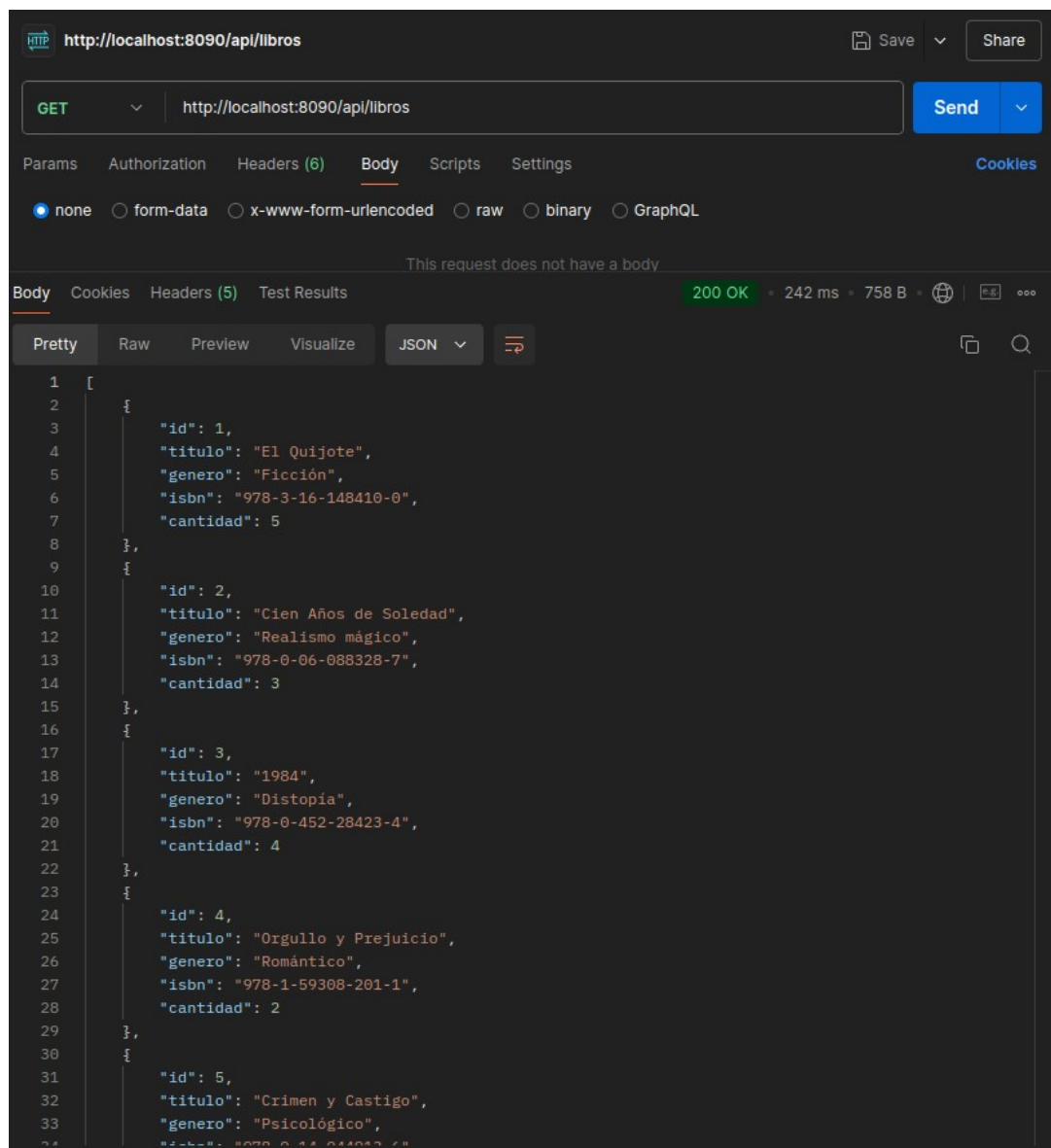
A continuación, se probará el funcionamiento de la aplicación, mediante la herramienta Postman.

Libros

Petición GET (a la lista)

Se ha realizado una petición GET vacía.

Podemos ver que devuelve la lista de libros, con código 200 OK.

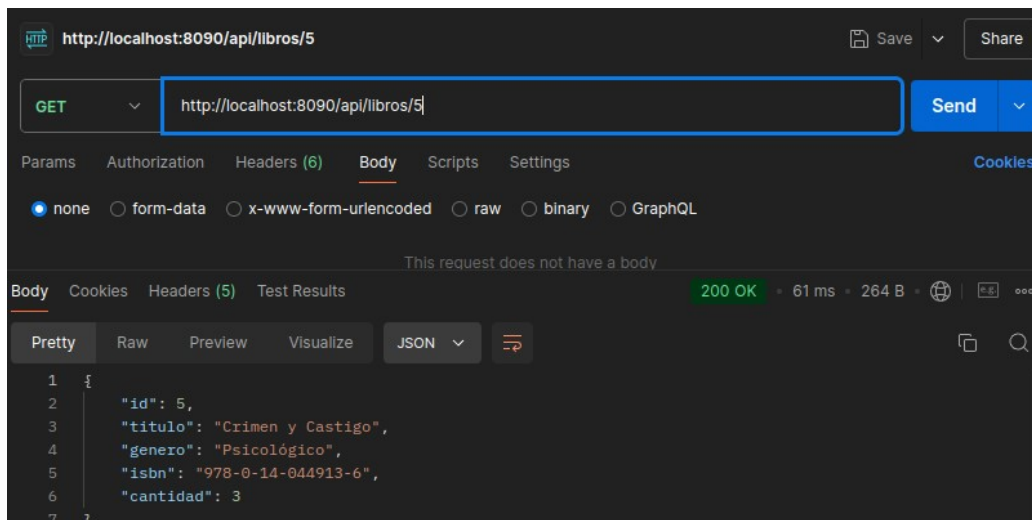


Petición GET para Libros.

Petición GET (a un elemento)

Se ha realizado una petición GET a un elemento.

Podemos ver que devuelve dicho libro, con código 200 OK.

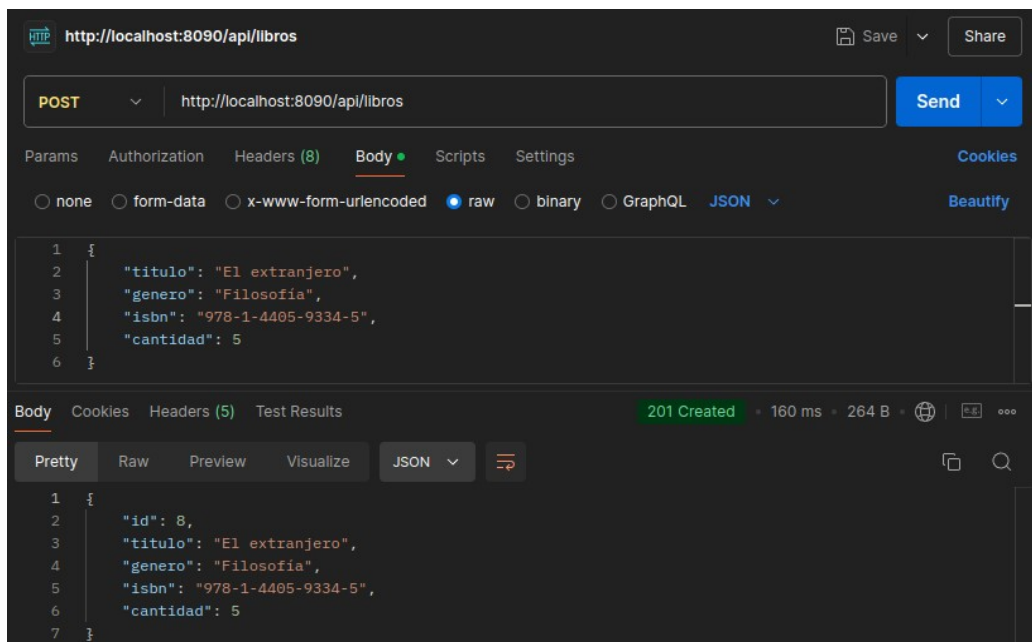


Petición GET para un elemento de Libros.

Petición POST

Se ha realizado una petición POST con un JSON con los datos del libro.

Podemos ver que se ha creado, pues devuelve una respuesta con el código 201 CREATED.

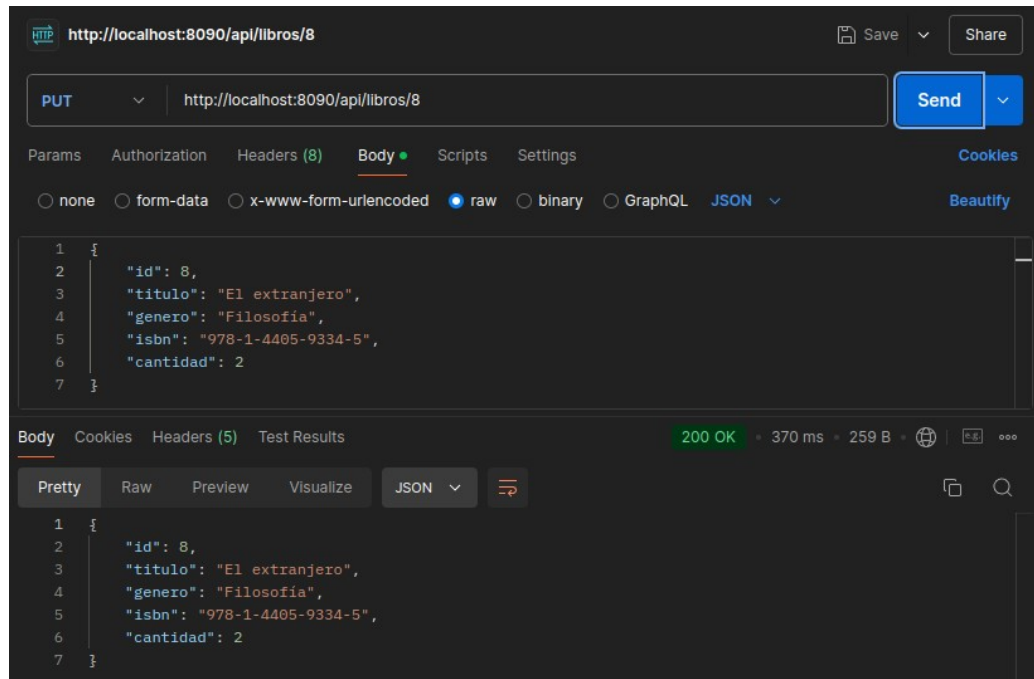


Petición POST para un elemento de Libros.

Petición PUT

Se ha realizado una petición PUT con un JSON con los datos del libro.

Podemos ver que se ha actualizado, pues devuelve una respuesta con el código 200 OK.

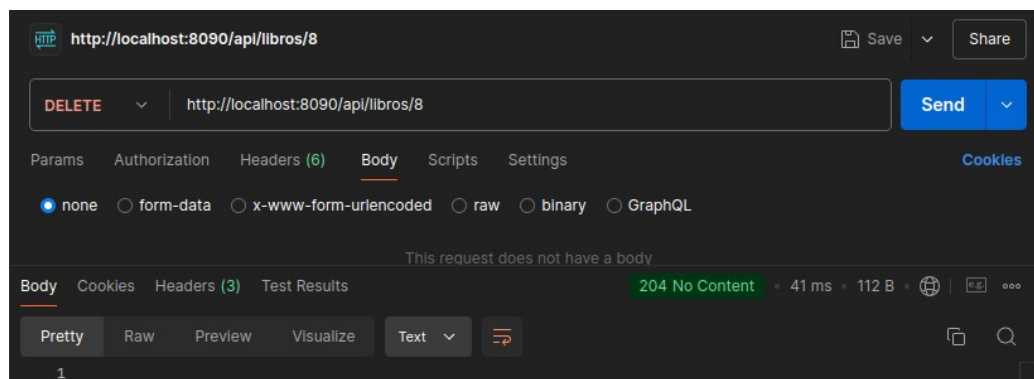


Petición PUT para un elemento de Libros.

Petición DELETE

Se ha realizado una petición DELETE vacía.

Podemos ver que se ha borrado, pues devuelve una respuesta con el código 204 NO CONTENT.



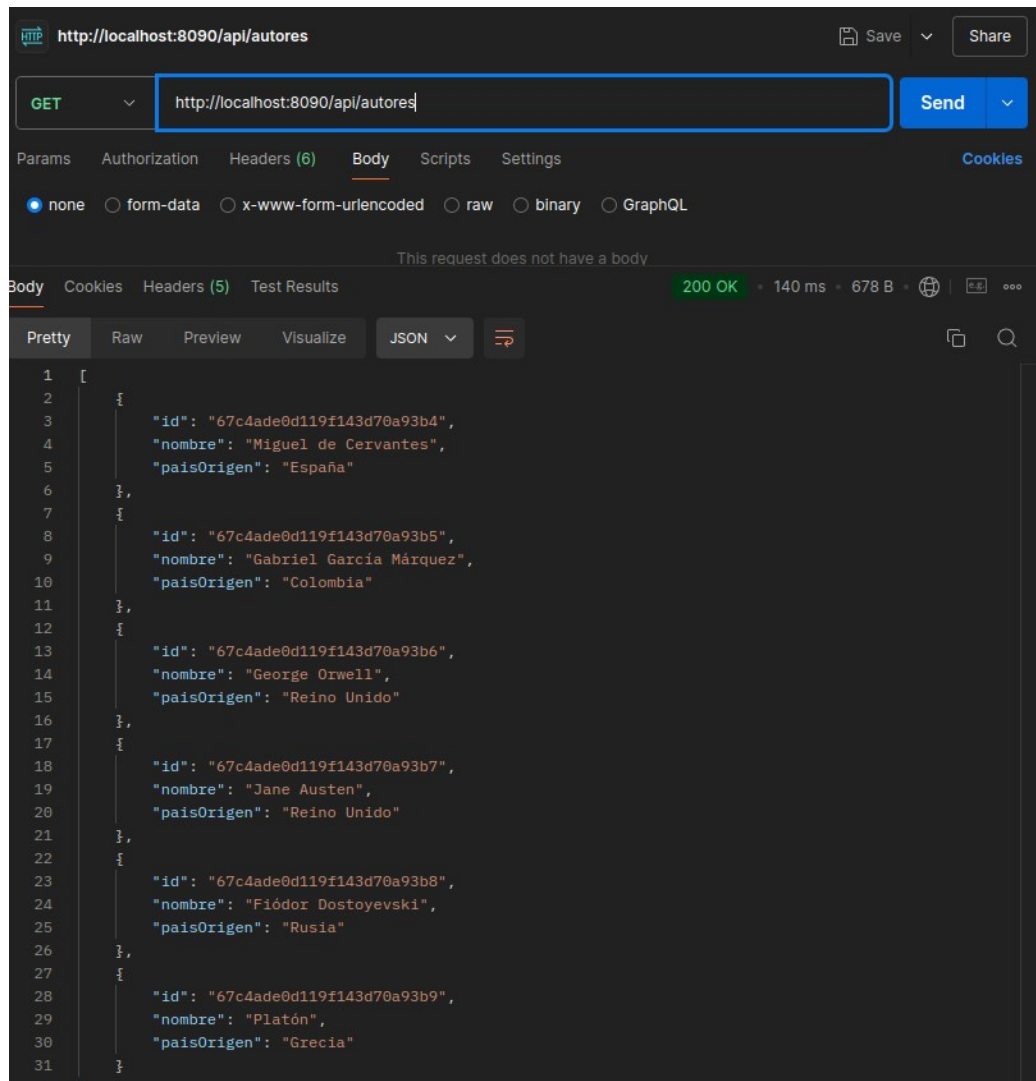
Petición DELETE para un elemento de Libros.

Autores

Petición GET (a la lista)

Se ha realizado una petición GET vacía.

Podemos ver que devuelve la lista de autores, con código 200 OK.

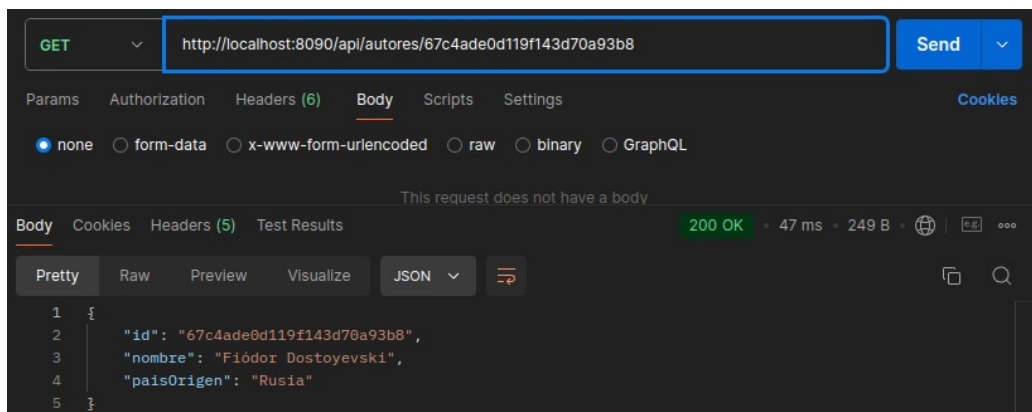


Petición GET para Autores.

Petición GET (a un elemento)

Se ha realizado una petición GET a un elemento.

Podemos ver que devuelve dicho autor, con código 200 OK.

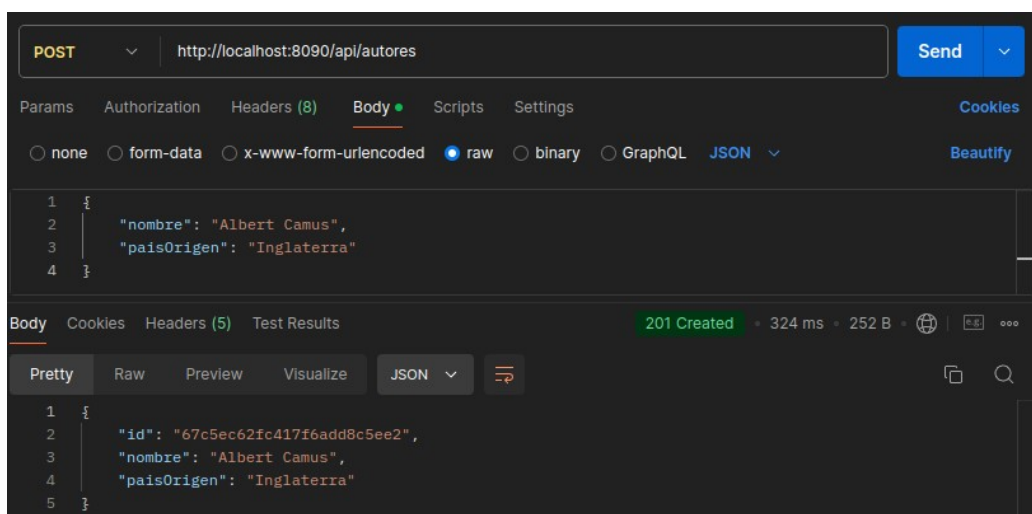


Petición GET para un elemento de Autores.

Petición POST

Se ha realizado una petición POST con un JSON con los datos del autor.

Podemos ver que se ha creado, pues devuelve una respuesta con el código 201 CREATED.

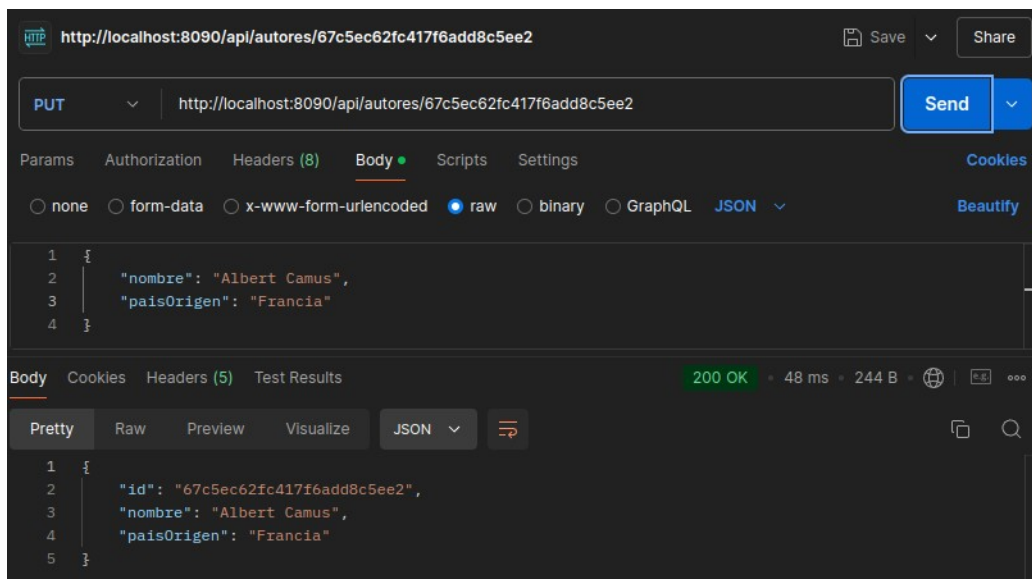


Petición POST para un elemento de Autores.

Petición PUT

Se ha realizado una petición PUT con un JSON con los datos del autor.

Podemos ver que se ha actualizado, pues devuelve una respuesta con el código 200 OK.

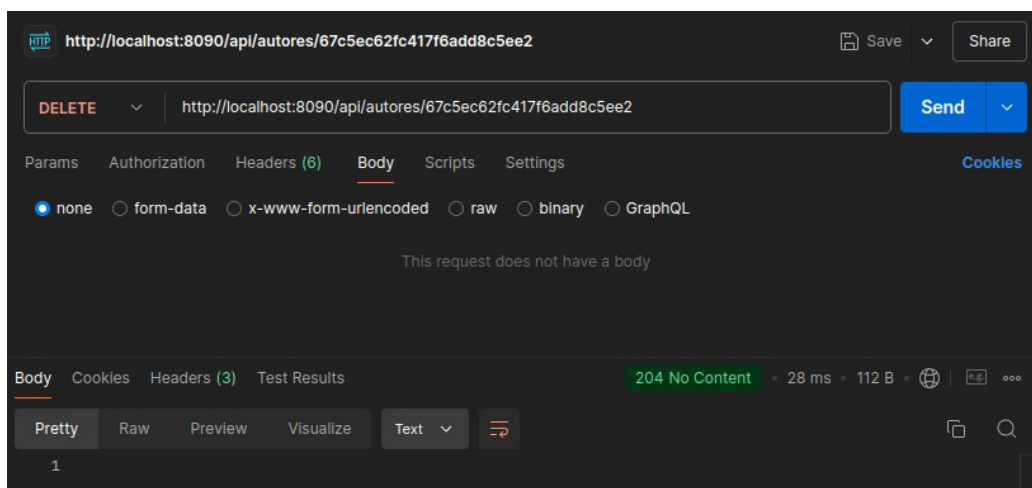


Petición PUT para un elemento de Autores.

Petición DELETE

Se ha realizado una petición DELETE vacía.

Podemos ver que se ha borrado, pues devuelve una respuesta con el código 204 NO CONTENT.



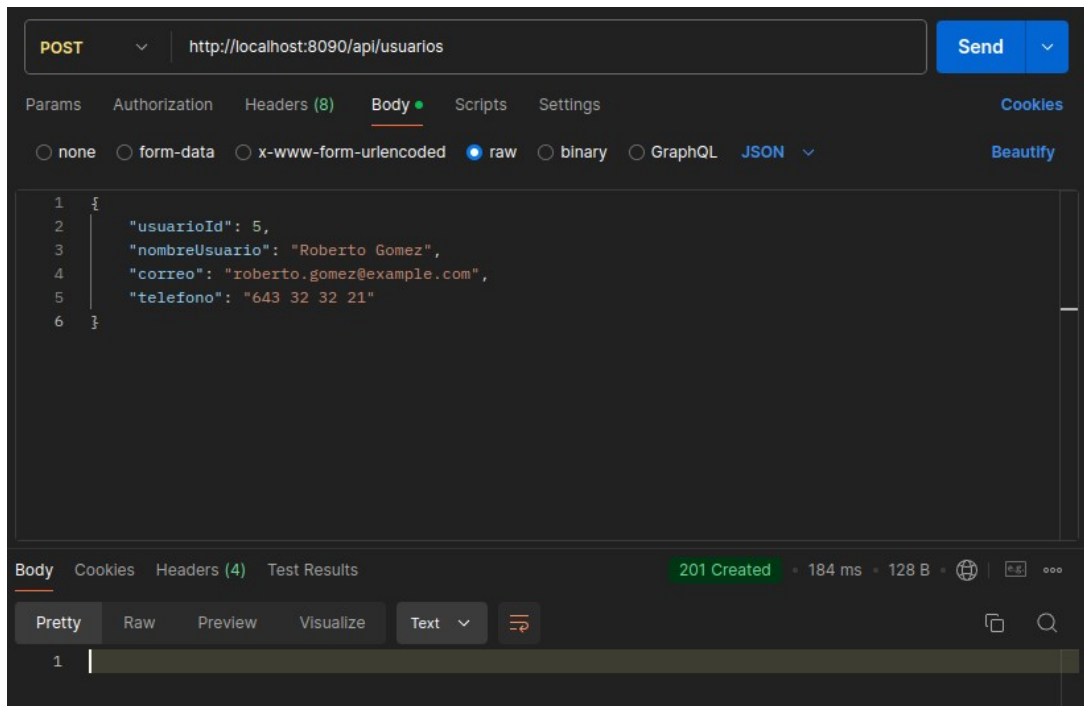
Petición DELETE para un elemento de Autores.

Usuarios

Petición POST

Se ha realizado una petición POST con un JSON con los datos del usuario.

Podemos ver que se ha creado, pues devuelve una respuesta con el código 201 CREATED.

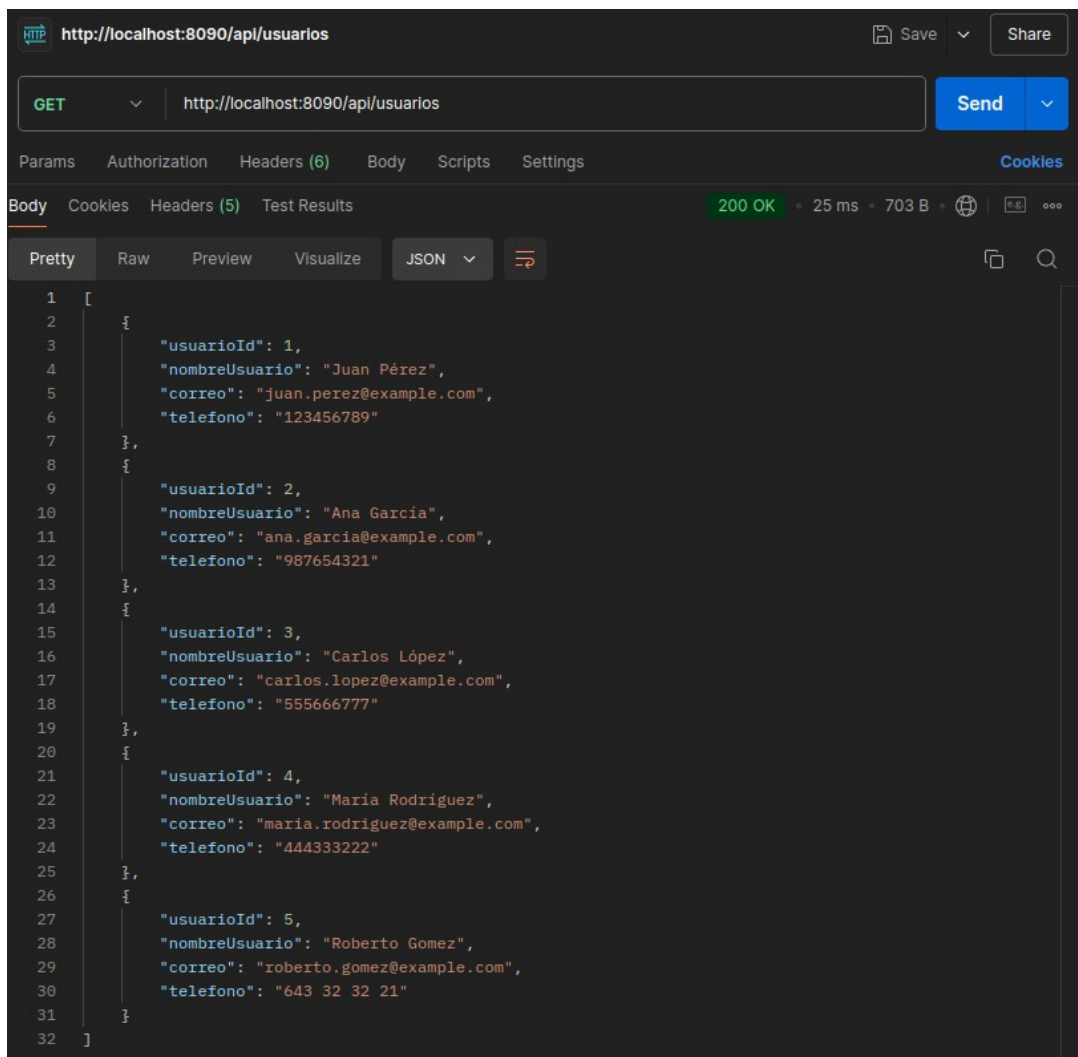


Petición POST para Usuarios.

Petición GET

Se ha realizado una petición GET vacía.

Podemos ver que devuelve la lista de usuarios, con código 200 OK.



```
1 [
2   {
3     "usuarioId": 1,
4     "nombreUsuario": "Juan Pérez",
5     "correo": "juan.perez@example.com",
6     "telefono": "123456789"
7   },
8   {
9     "usuarioId": 2,
10    "nombreUsuario": "Ana Garcia",
11    "correo": "ana.garcia@example.com",
12    "telefono": "987654321"
13  },
14  {
15    "usuarioId": 3,
16    "nombreUsuario": "Carlos López",
17    "correo": "carlos.lopez@example.com",
18    "telefono": "555666777"
19  },
20  {
21    "usuarioId": 4,
22    "nombreUsuario": "Maria Rodriguez",
23    "correo": "maria.rodriguez@example.com",
24    "telefono": "444333222"
25  },
26  {
27    "usuarioId": 5,
28    "nombreUsuario": "Roberto Gomez",
29    "correo": "roberto.gomez@example.com",
30    "telefono": "643 32 32 21"
31  }
32 ]
```

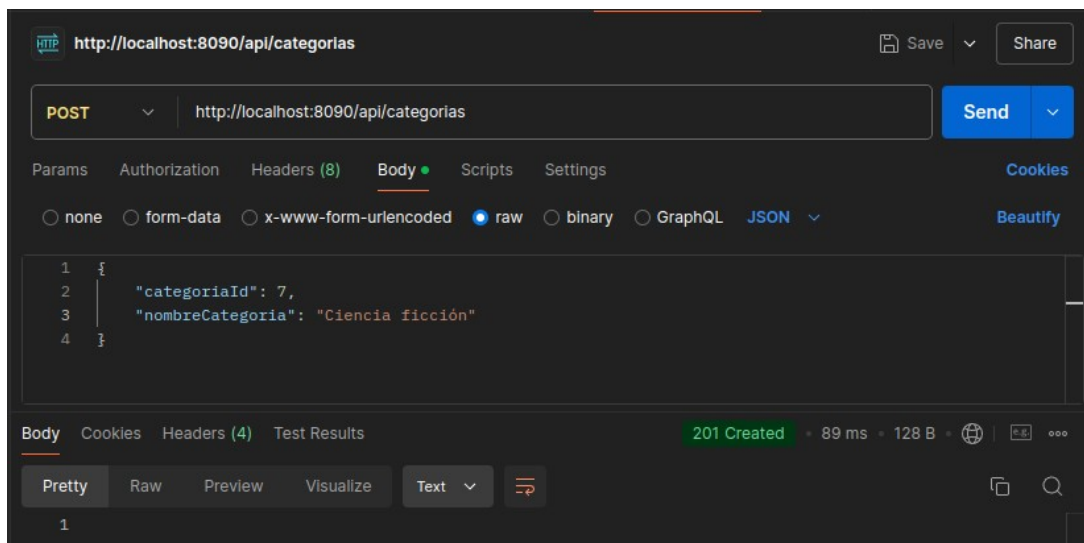
Petición GET para Usuarios.

Categorías

Petición POST

Se ha realizado una petición POST con un JSON con los datos de la categoría.

Podemos ver que se ha creado, pues devuelve una respuesta con el código 201 CREATED.

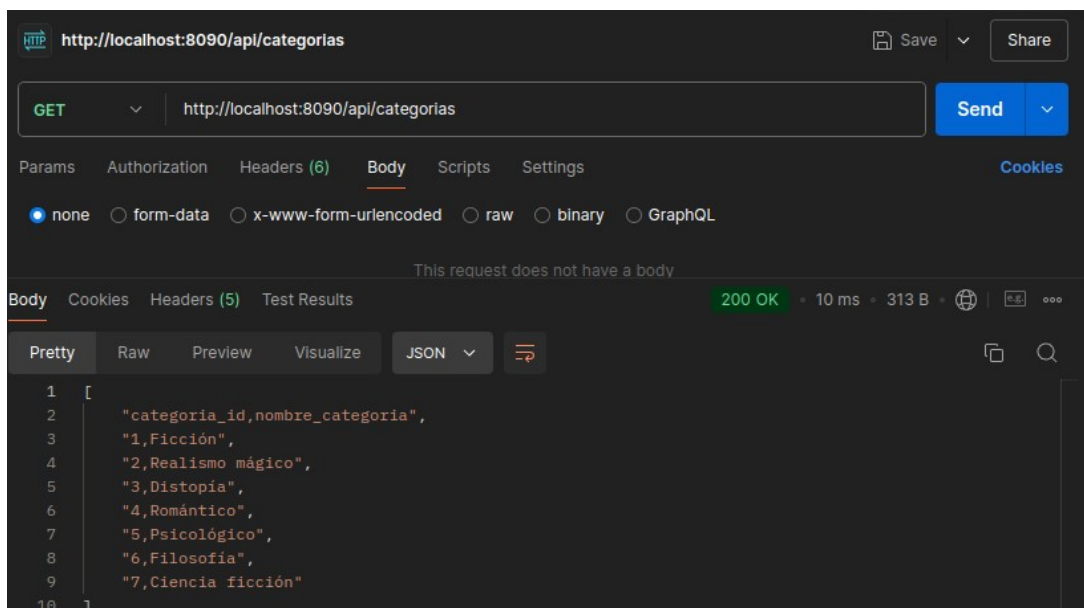


Petición POST para Categorías.

Petición GET

Se ha realizado una petición GET vacía.

Podemos ver que devuelve la lista de categorías, con código 200 OK.



Petición GET para Categorías.