****

**Transformación digital de Perfulandia SPA**

**Integrantes:**

**Rocío Trujillo**

**Gabriel Olguin**

**Matías Olivera**

**Índice**

[Parte 1: Estructura del proyecto 5](#_heading=h.z8c4dvnvrpg7)

[Parte 2: Base de datos 7](#_heading=h.rx9lhjw4knc7)

[Parte 3: Implementación de los servicios 8](#_heading=h.e95wqshyauop)

[Parte 4: Git - GitHub 15](#_heading=h.walhedz2mrc0)

# 

# **Parte 1: Estructura del proyecto**

El proyecto está construido con el framework Spring Boot, y se gestiona con Maven, que es una herramienta que permite automatizar la construcción del proyecto, gestionar librerías y ejecutar tareas de forma eficiente.

**Dependencias:**

* **Spring Boot Starter Web:** Para crear servicios REST.
* **Spring Boot Starter Data JPA:** Para conectarse y trabajar con la base de datos.
* **MySQL Driver:** Para conectar el proyecto con MySQL.
* **Spring Boot DevTools:** Para facilitar el desarrollo con recarga automática.

**Componentes implementados:**

El proyecto se organiza en diferentes paquetes para mantener una estructura más comprensible. Los principales componentes son:

* **Controladores (Controllers)**: Manejan las peticiones REST que llegan desde el cliente.
* **Servicios (Services)**: Contienen la lógica del negocio (Interfaces e Implementaciones)
* **Entidades (Entities)**: Representan las tablas de la base de datos; Productos, Proveedores y Usuarios
* **Repositorios (Repositories)**: Implementamos Interfaces que manejan operaciones con la base de datos.

## 

## 

## 

## 

## **Operaciones CRUD**

### **Usuario**

* **Descripción**: Representa a los usuarios registrados en el sistema de Perfulandia SPA.
* **Atributos**: id, nombre, correo, contraseña.
* **CRUD implementado en**:
  + **UsuarioController:** rutas REST para crear, actualizar, consultar y eliminar usuarios
  + **UsuarioService + UsuarioServiceImpl**: contienen la lógica de negocio asociada a la gestión de usuarios.
  + **UsuarioRepository:** interfaz JPA que permite acceder a los datos en la base de datos.

### **Producto**

* **Descripción**: Define los productos disponibles en Perfulandia SPA para su venta o gestión.
* **Atributos**: id, nombre, precio, stock.
* **CRUD implementado en**:  
  + **ProductoController:** expone las rutas para operaciones CRUD de productos.
  + **ProductoService + ProductoServiceImpl:** manejan la lógica para registrar, modificar, listar y eliminar productos.
  + **ProductoRepository:** interfaz JPA para interactuar con la base de datos de productos.

**Proveedor**

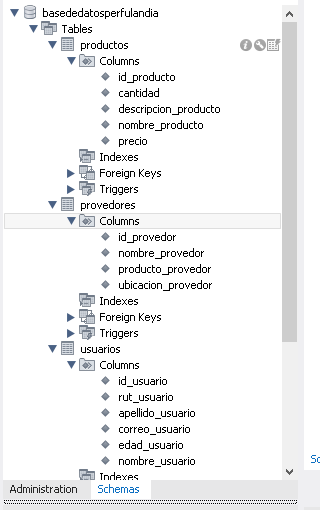
* **Descripción**: Representa a los proveedores asociados con Perfulandia SPA.
* **Atributos**: id, nombre, contacto, dirección.
* **CRUD implementado en**:  
  + **ProveedorController:** rutas REST para registrar, consultar, actualizar y eliminar proveedores.
  + **ProveedorService + ProveedorServiceImpl:** contienen la lógica de negocio relacionada a proveedores.
  + **ProveedorRepository:** interfaz JPA que gestiona el acceso a los datos de proveedores.

# 

# **Parte 2: Base de datos**

Se utilizó MySQL como motor de base de datos. Cada microservicio tiene asociada al menos una entidad, lo que se refleja en la estructura de la base de datos:

**Estructura de tablas:**

* **Productos:** id, cantidad, descripcion, nombre, precio
* **Proveedores:** id, nombre, producto, ubicacion
* **Usuarios:** id, nombre, apellido, edad, rut, correp

# 

# 

# 

# 

# 

# 

# 

# 

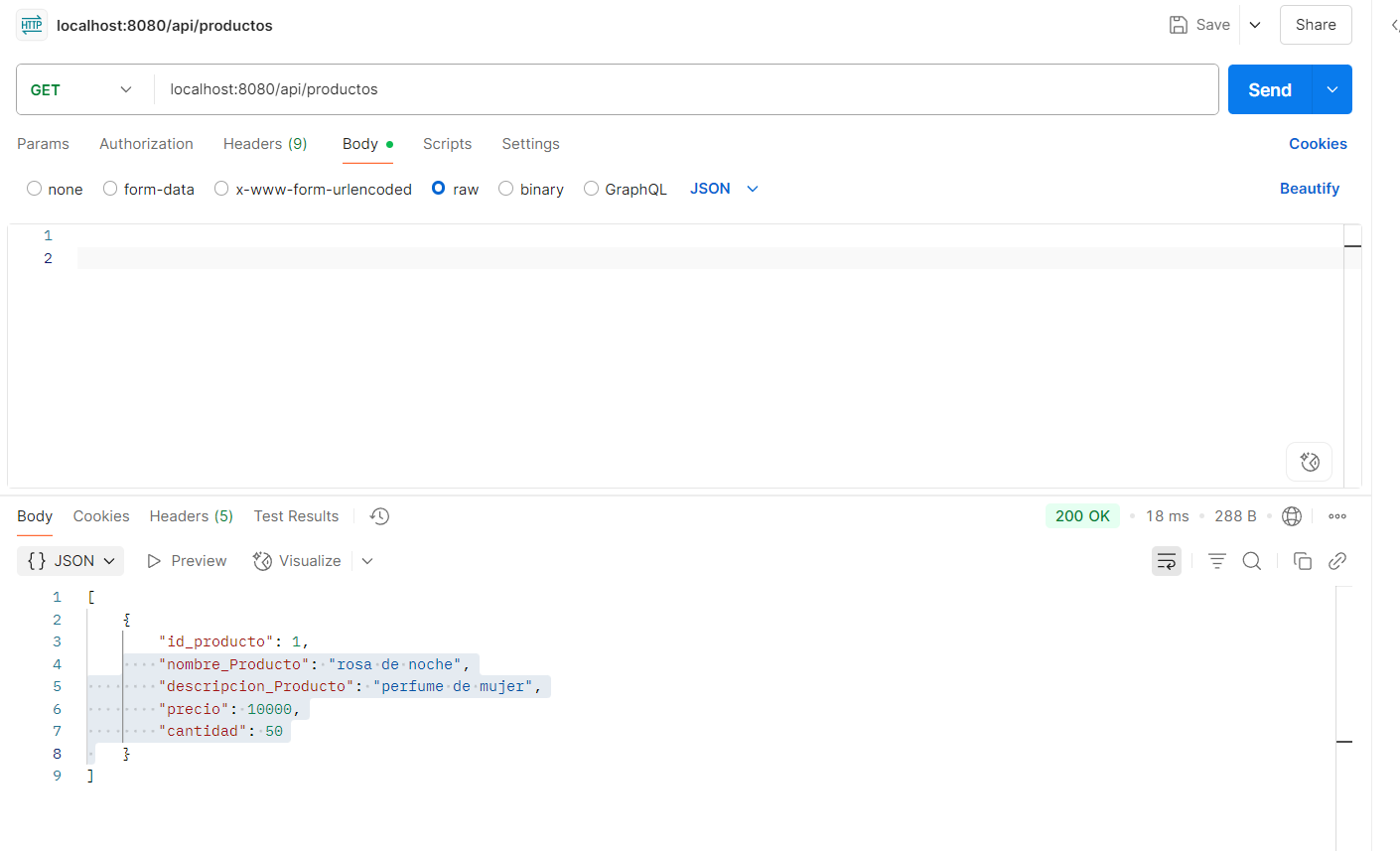
# 

# **Parte 3: Implementación de los servicios**

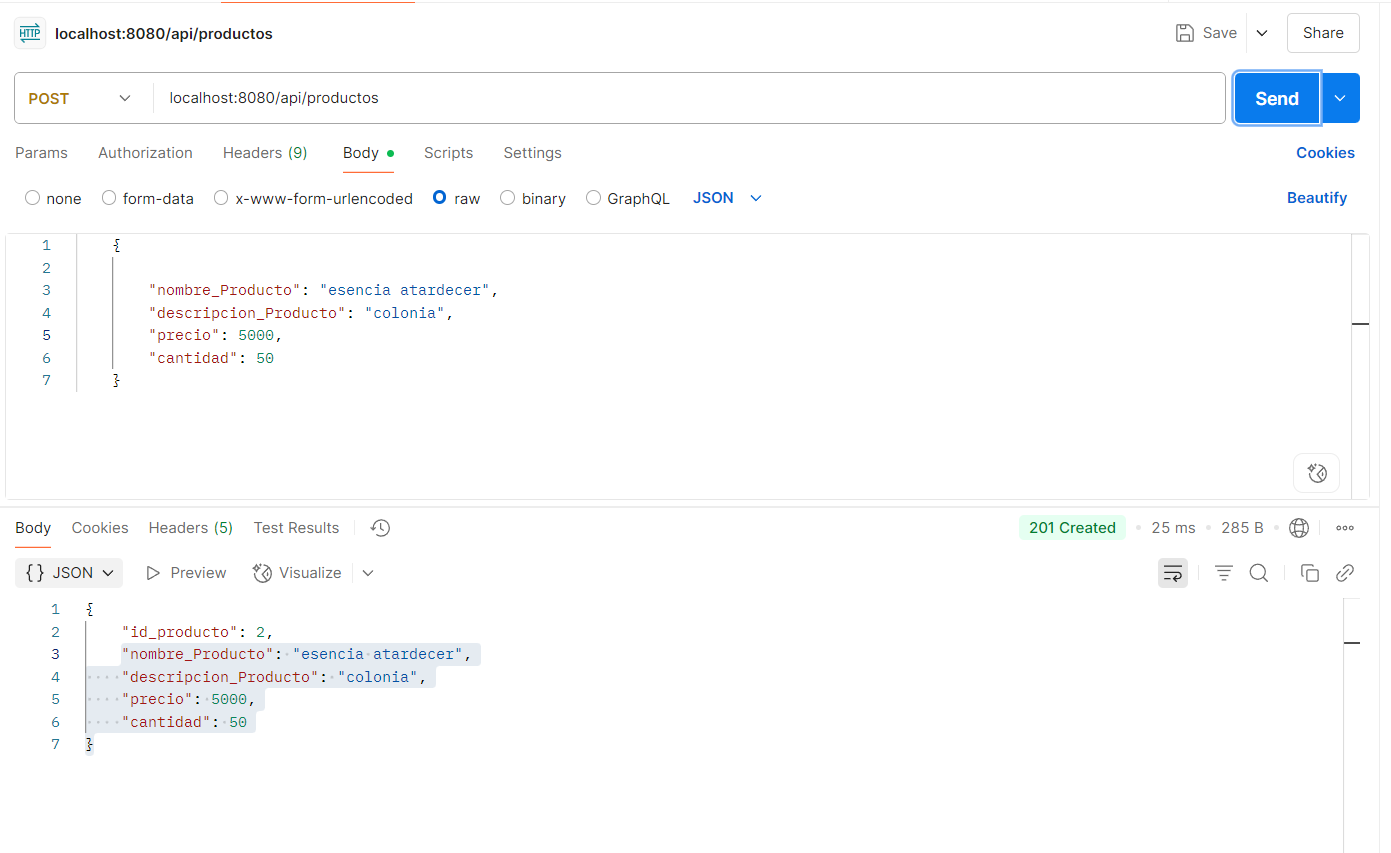
Los servicios se desarrollaron como APIs RESTful que permiten realizar operaciones CRUD. A continuación, se describen algunos ejemplos:

**Servicio de productos (/productos)**

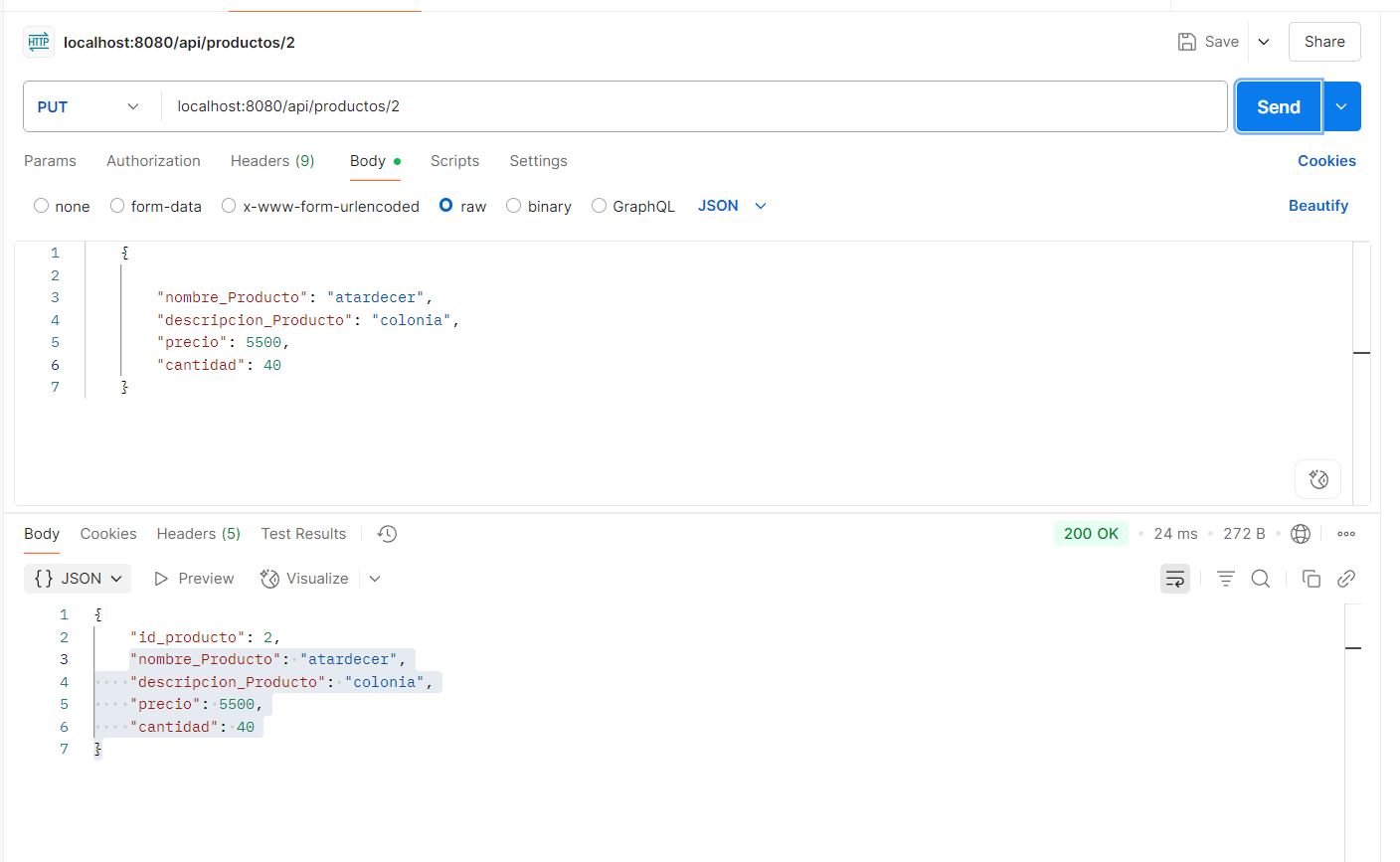
* **GET:** Listar todos los productos



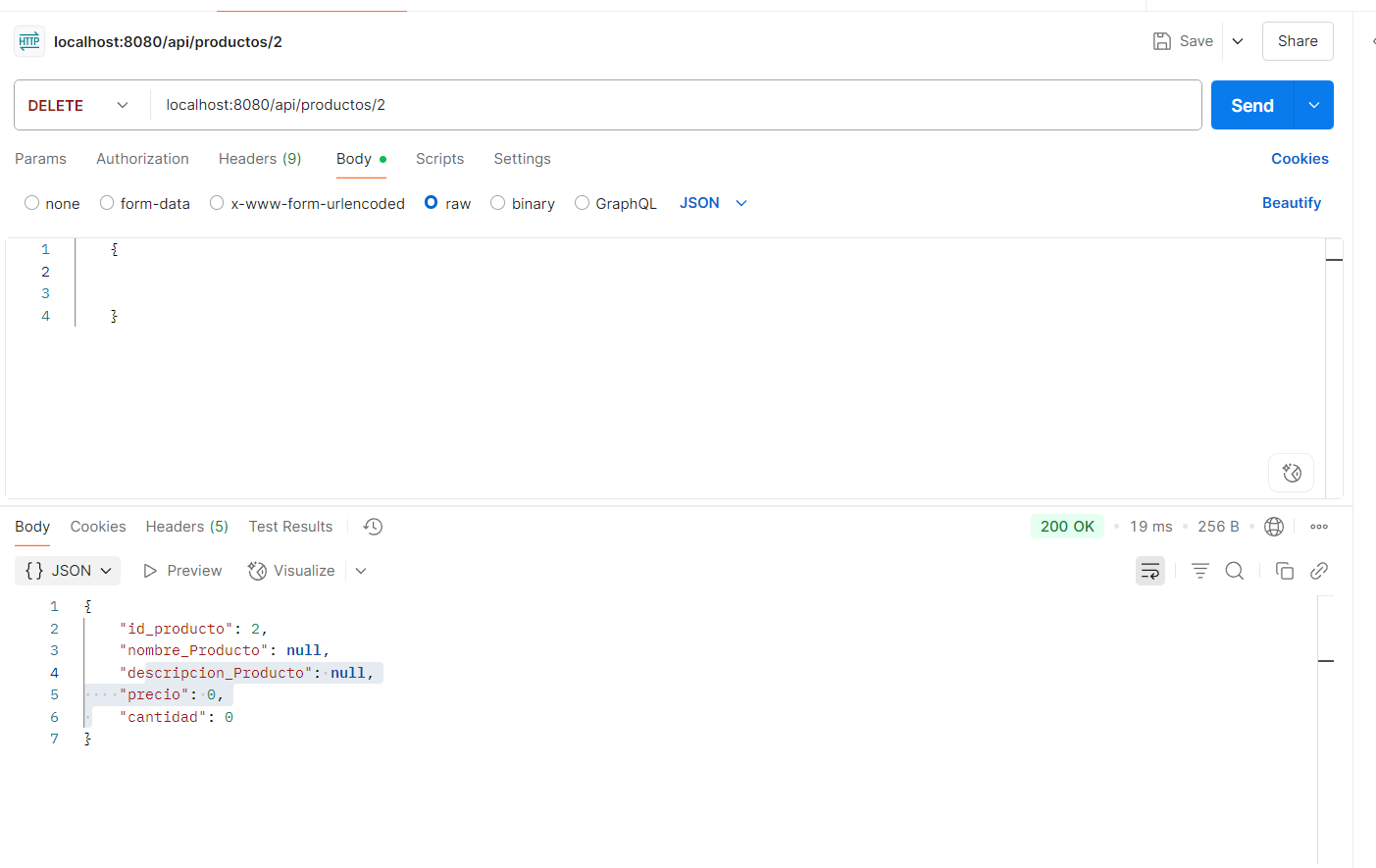
* **POST:**: Registrar un nuevo producto



* **PUT:** Actualizar información de un producto

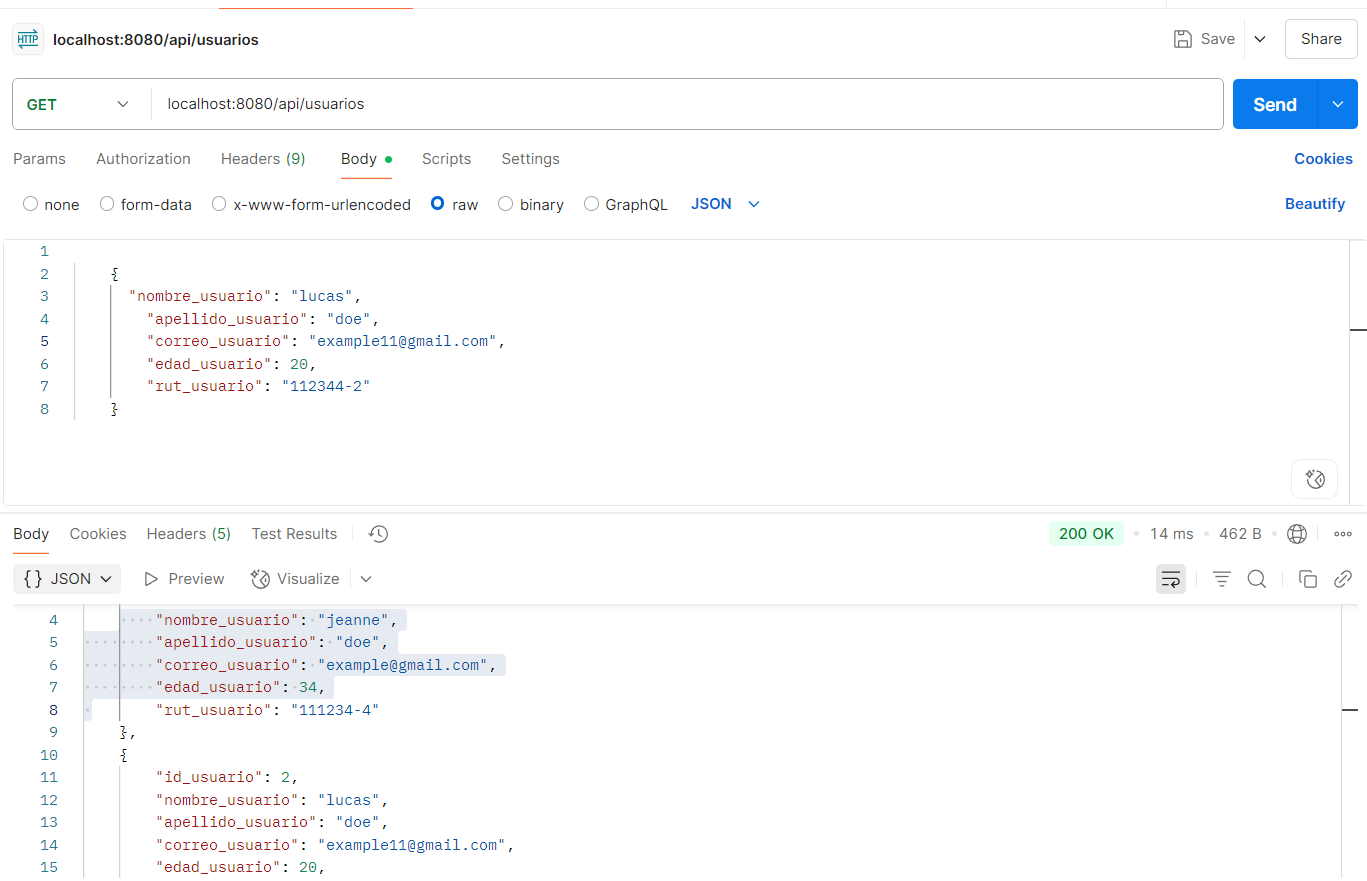


* **DELETE:** Eliminar un producto

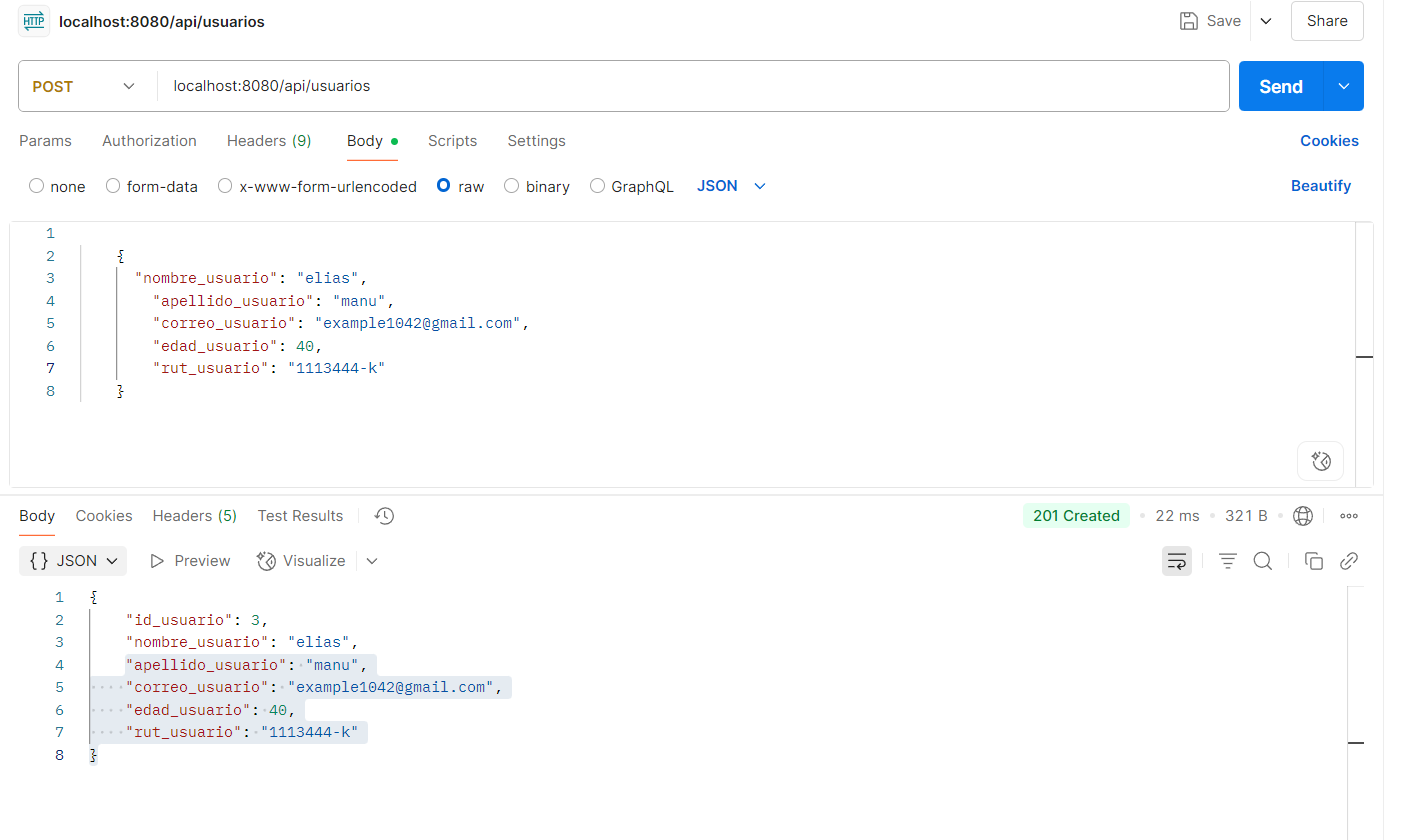


**Servicio de usuarios (/usuarios)**

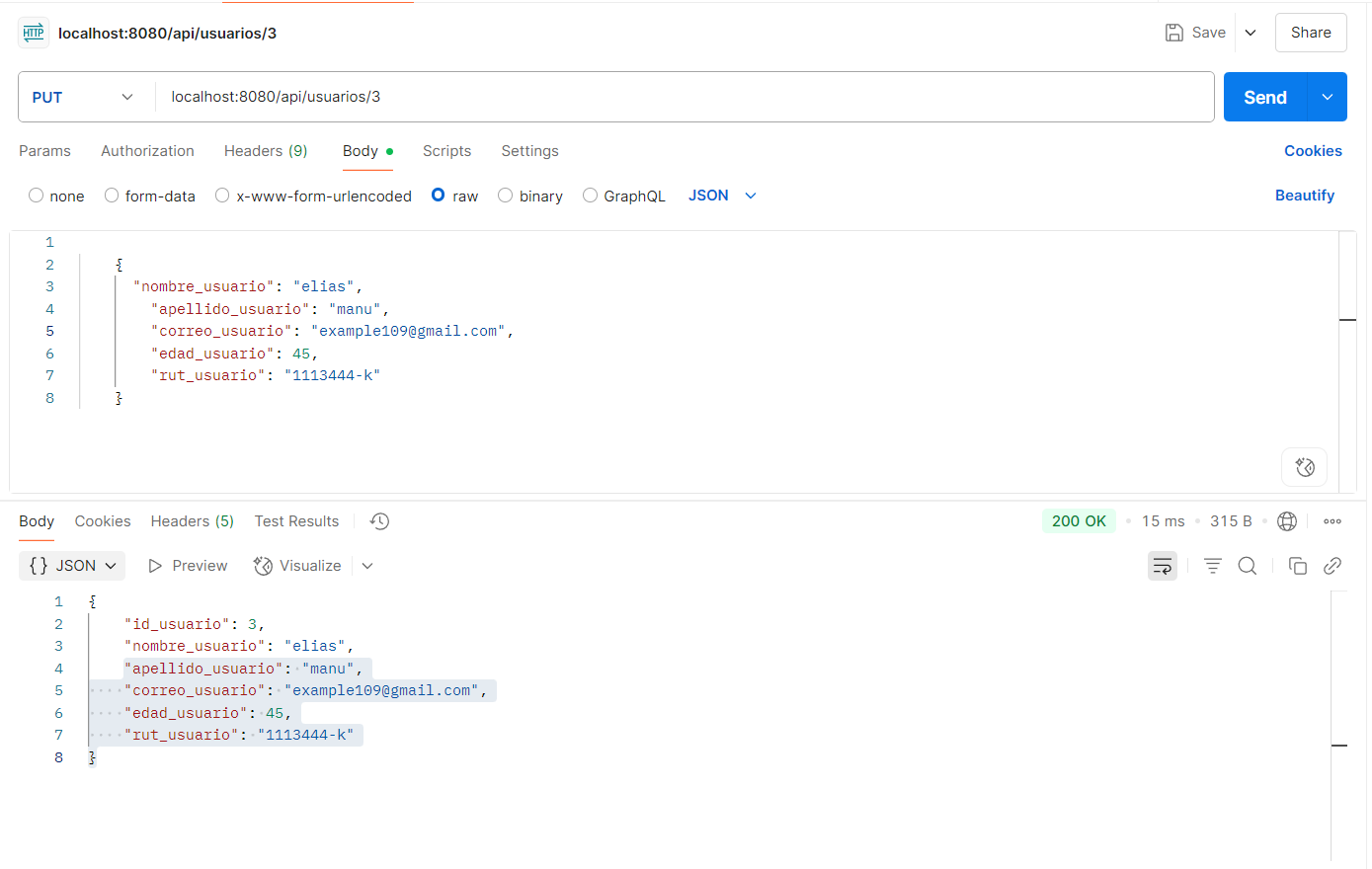
* **GET:** Listar todos los usuarios



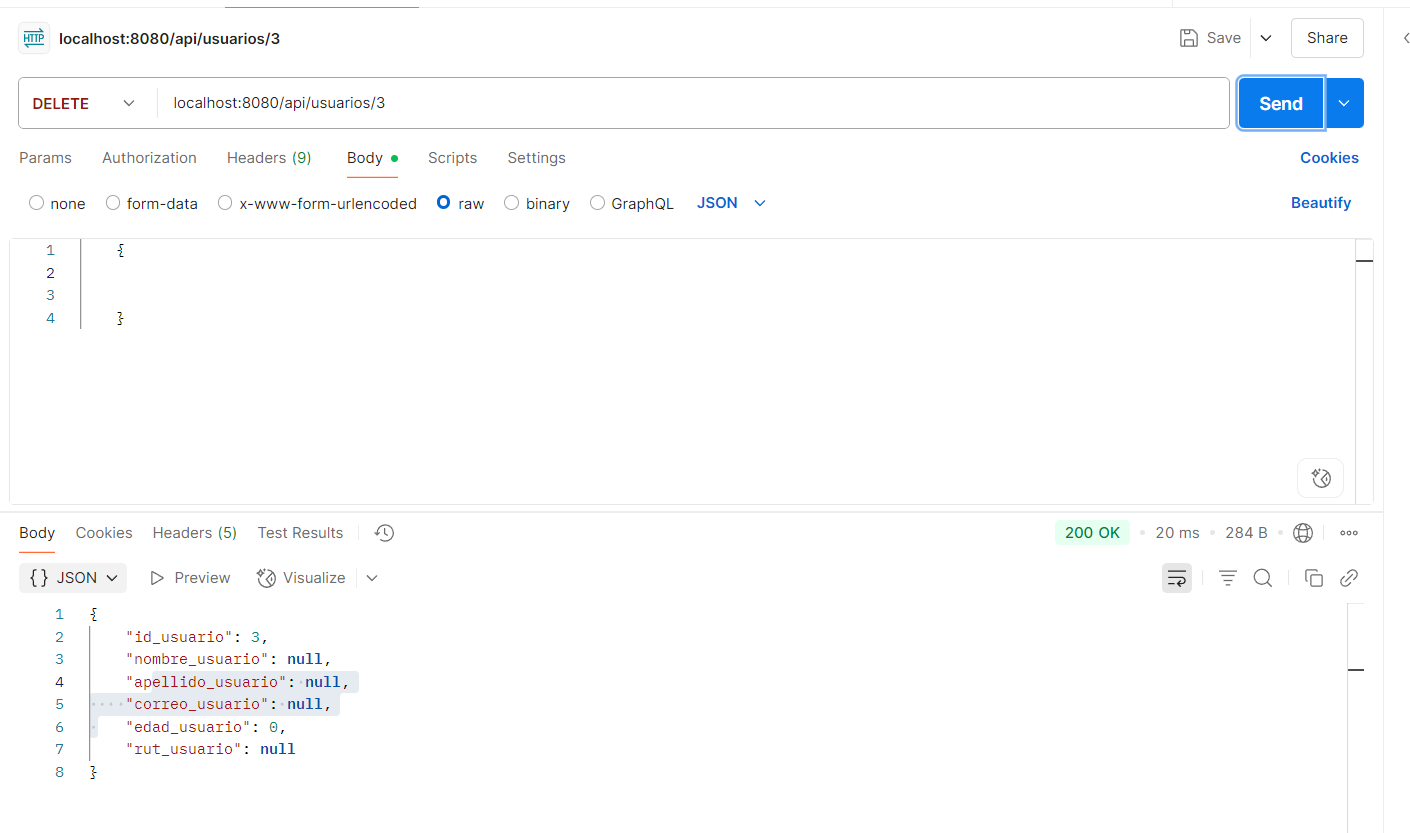
* **POST:** Registrar un nuevo usuario



* **PUT:** Actualizar información de un usuario

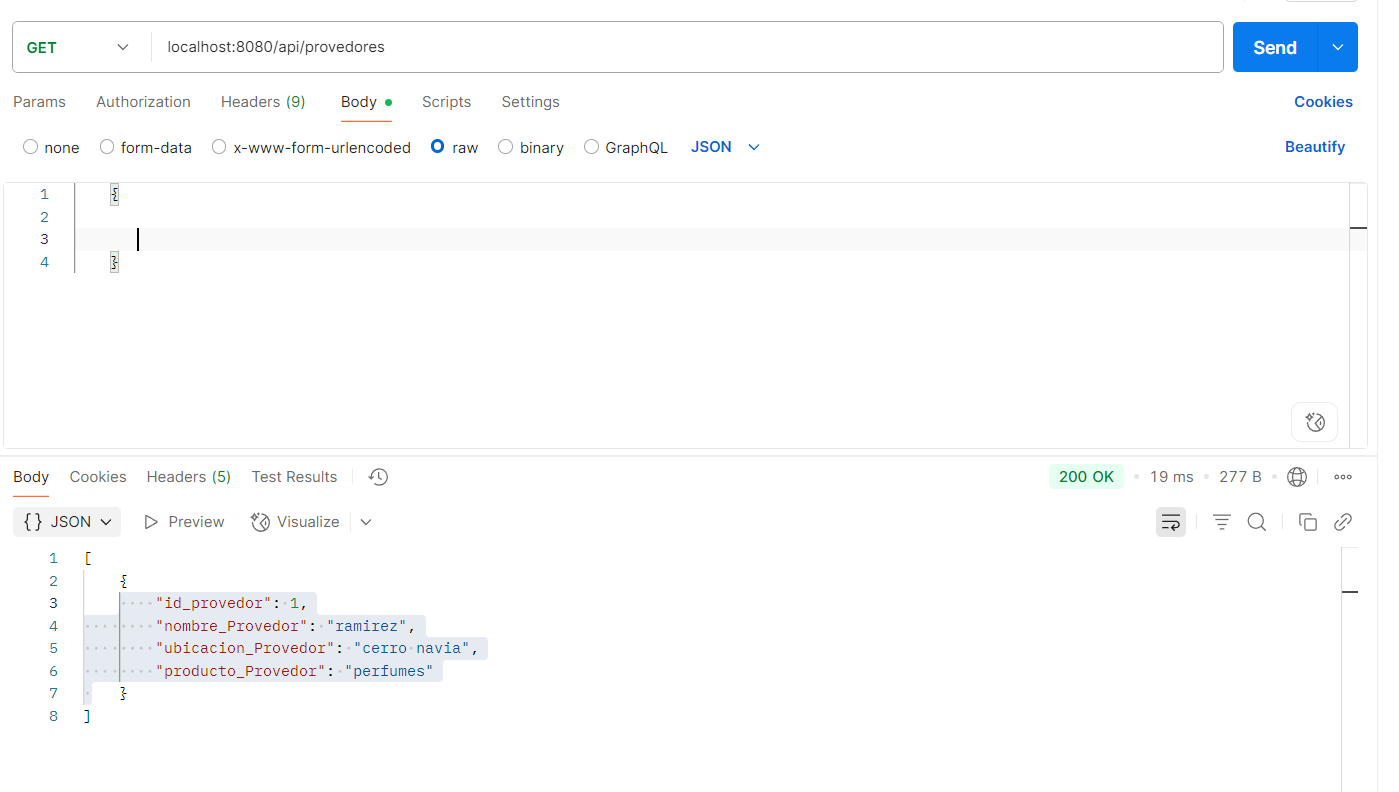


* **DELETE:** Eliminar usuario

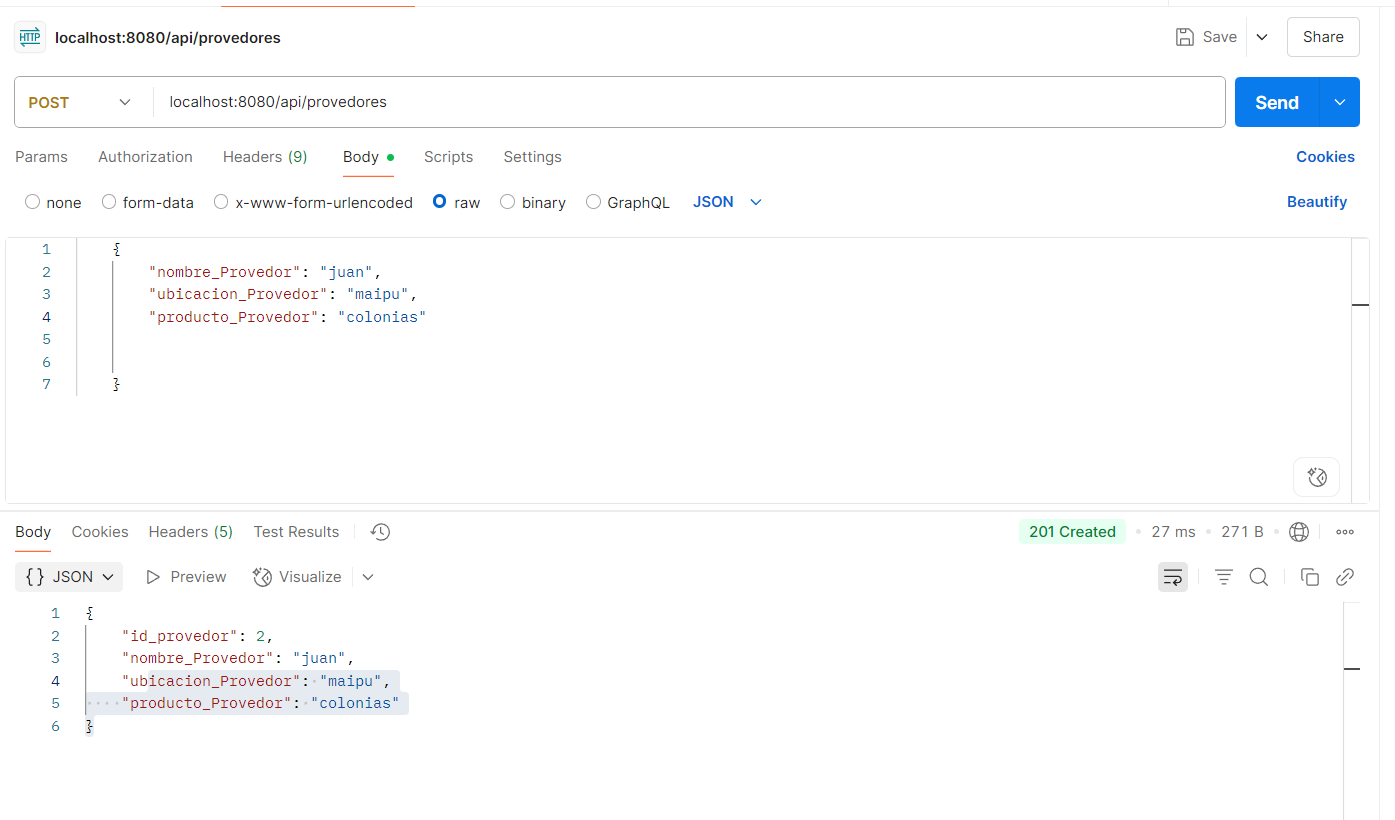


**Servicio de proveedores (/proveedores)**

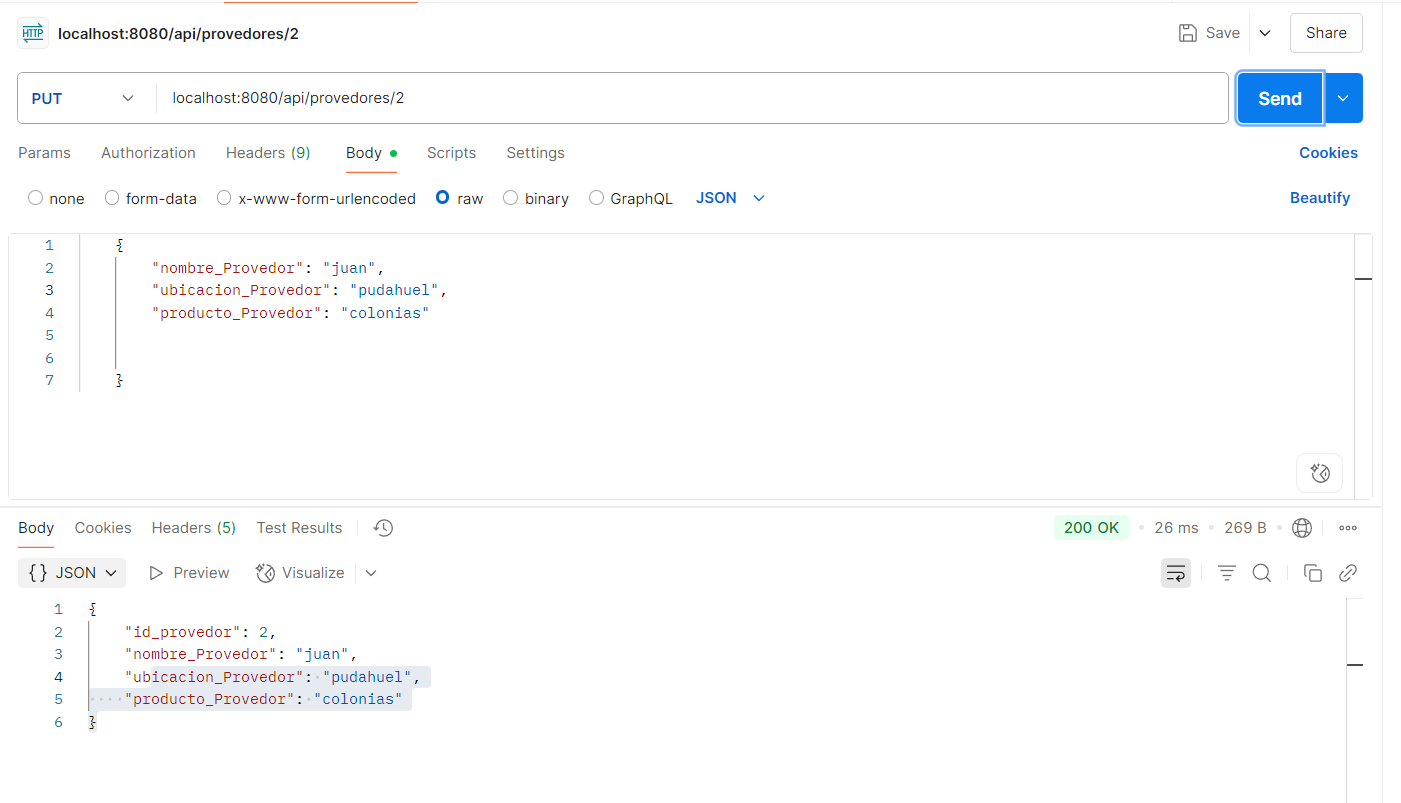
* **GET:** Listar todos los proveedores



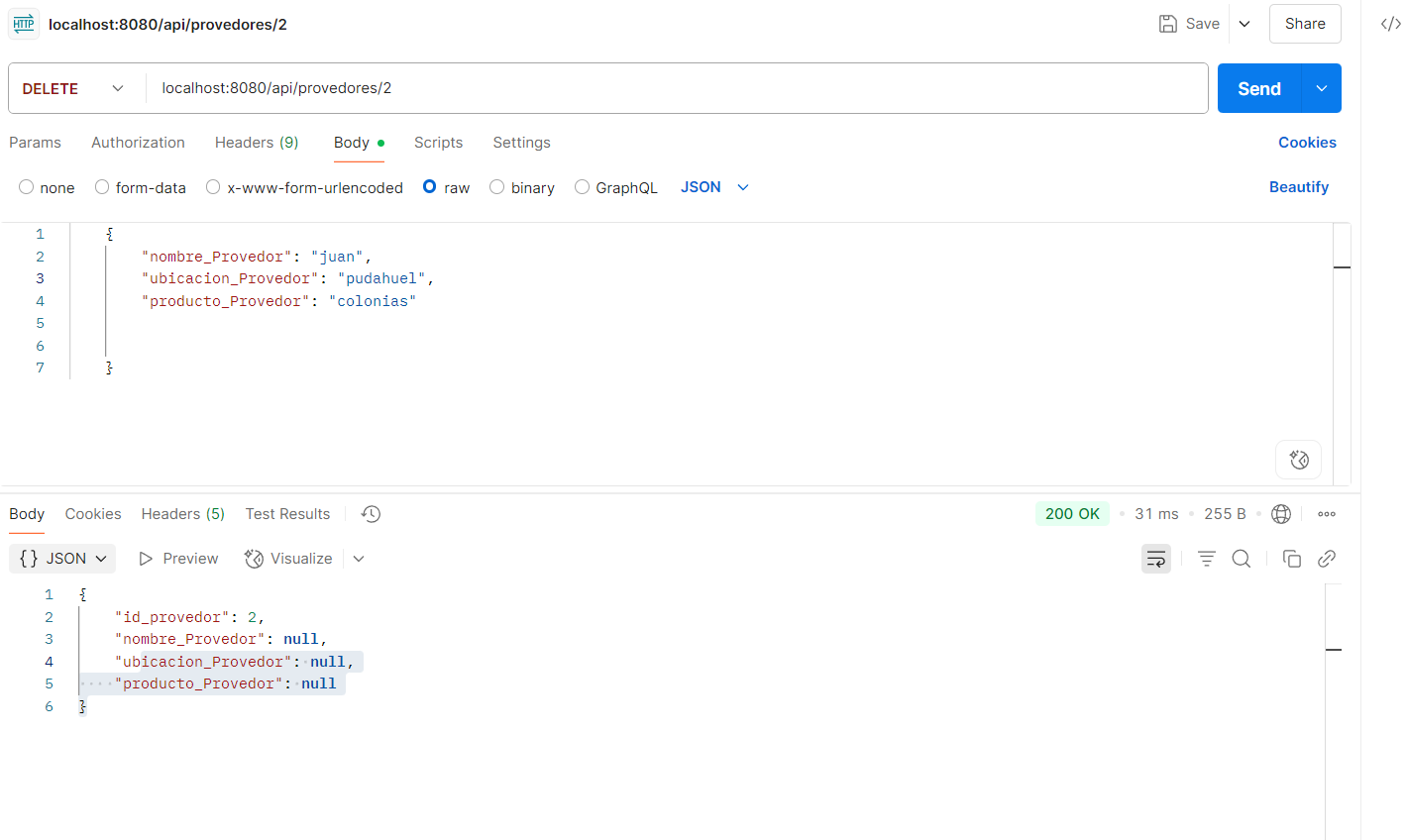
* **POST:** Registrar un nuevo proveedor



* **PUT:** Actualizar información de un proveedor



* **DELETE:** Eliminar un proveedor



# 

# 

# 

# 

# 

# 

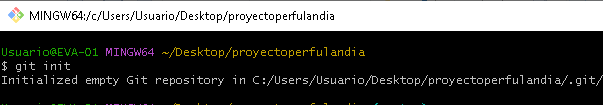
# 

# 

# **Parte 4: Git - GitHub**

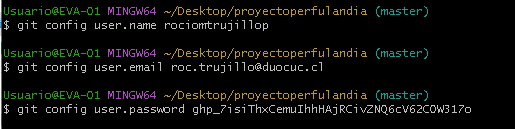
**git init**

inicializamos el nuevo repositorio en la carpeta del proyecto para versionar los archivos localmente

****

**git config**

Este comando se usa para configurar opciones de Git, como el nombre de usuario y el correo electrónico que se asocian con los commits

****

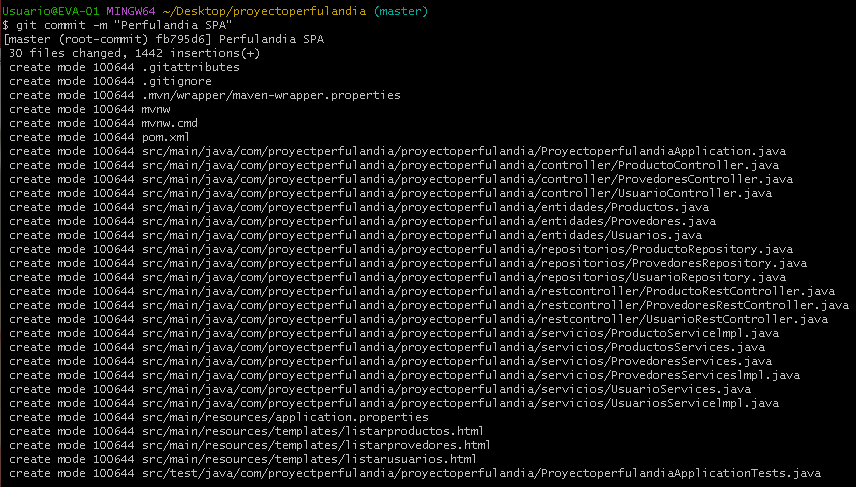
**git add**

Este comando prepara los archivos para ser confirmados

****

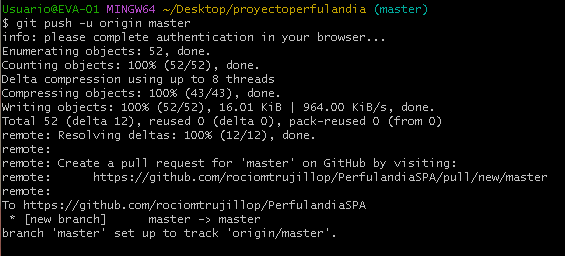
**git commit**

Guarda los cambios preparados en el repositorio local

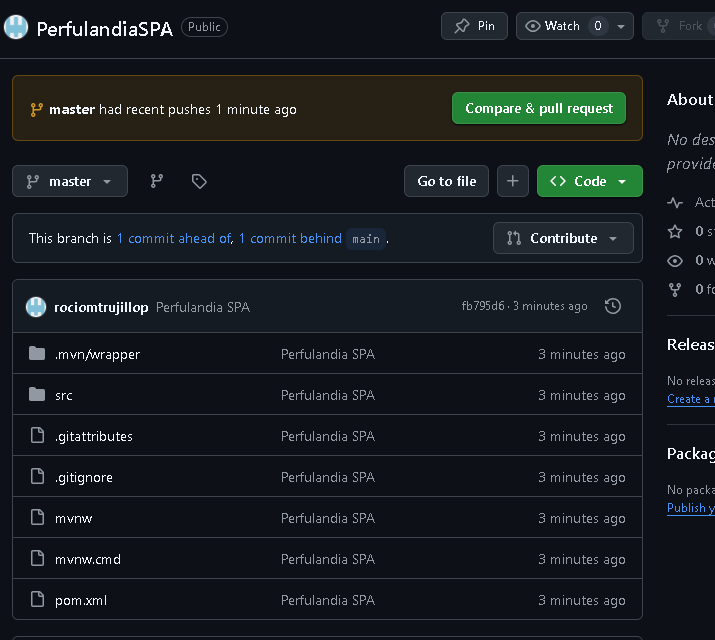


**git push**

Este comando envía los commits localmente al repositorio remoto

****

**Resultado:**

****

# 

# 