

Transformación Digital de EcoMarket SPA

Integrantes del Equipo: Daniel Paredes **Fecha de Entrega:** 26/05/2025

introducción

EcoMarketSPA enfrenta desafíos de escalabilidad debido a su sistema monolítico. Este proyecto propone una solución basada en microservicios para mejorar el rendimiento y la eficiencia operativa. Se desarrollaron tres servicios REST: Usuarios, Inventario y Pedidos, utilizando Spring Boot, Maven, MySQL y Postman para validaciones.

DIAGRAMA DE ARQUITECTURA DE SERVICIOS.

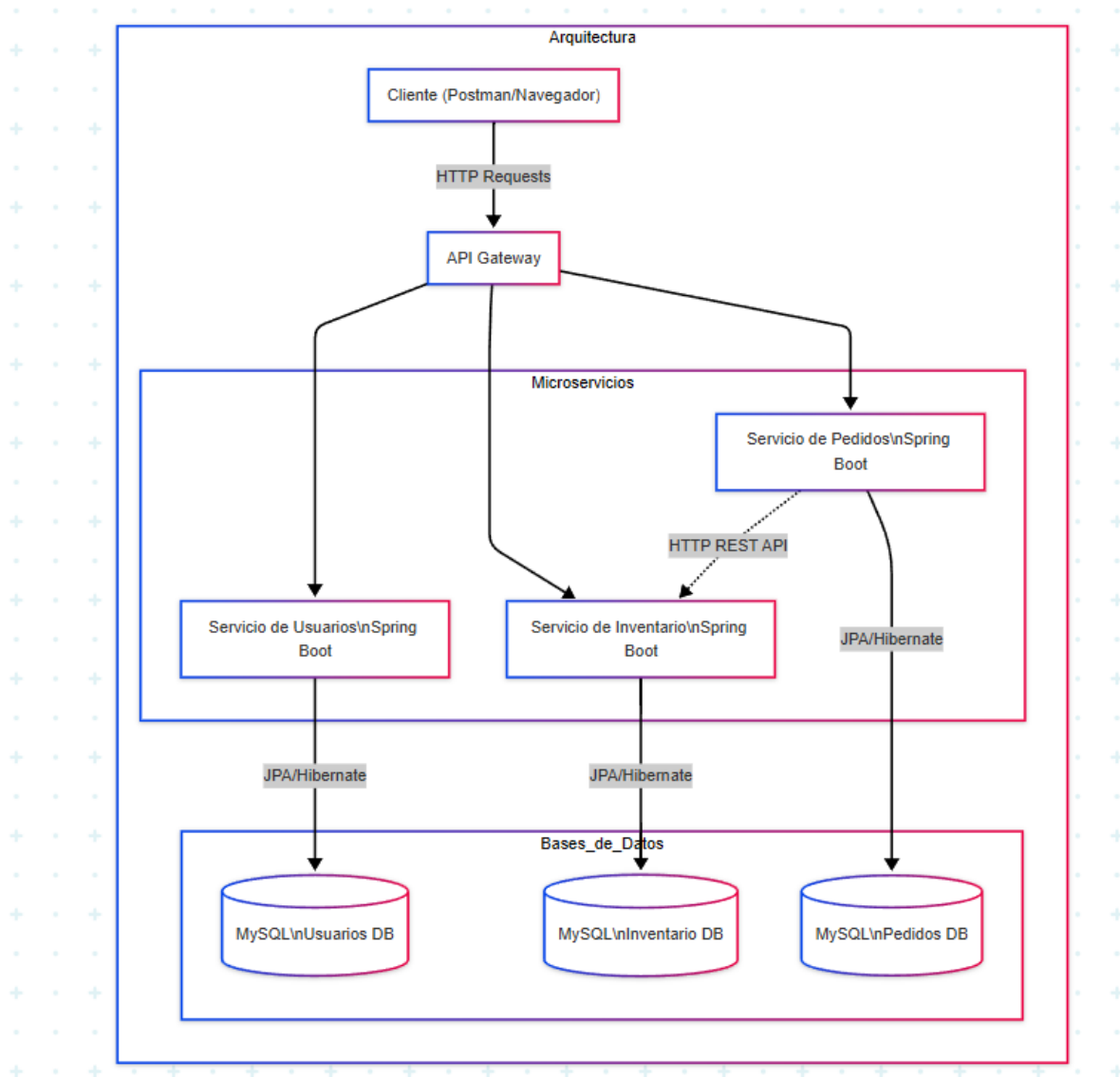
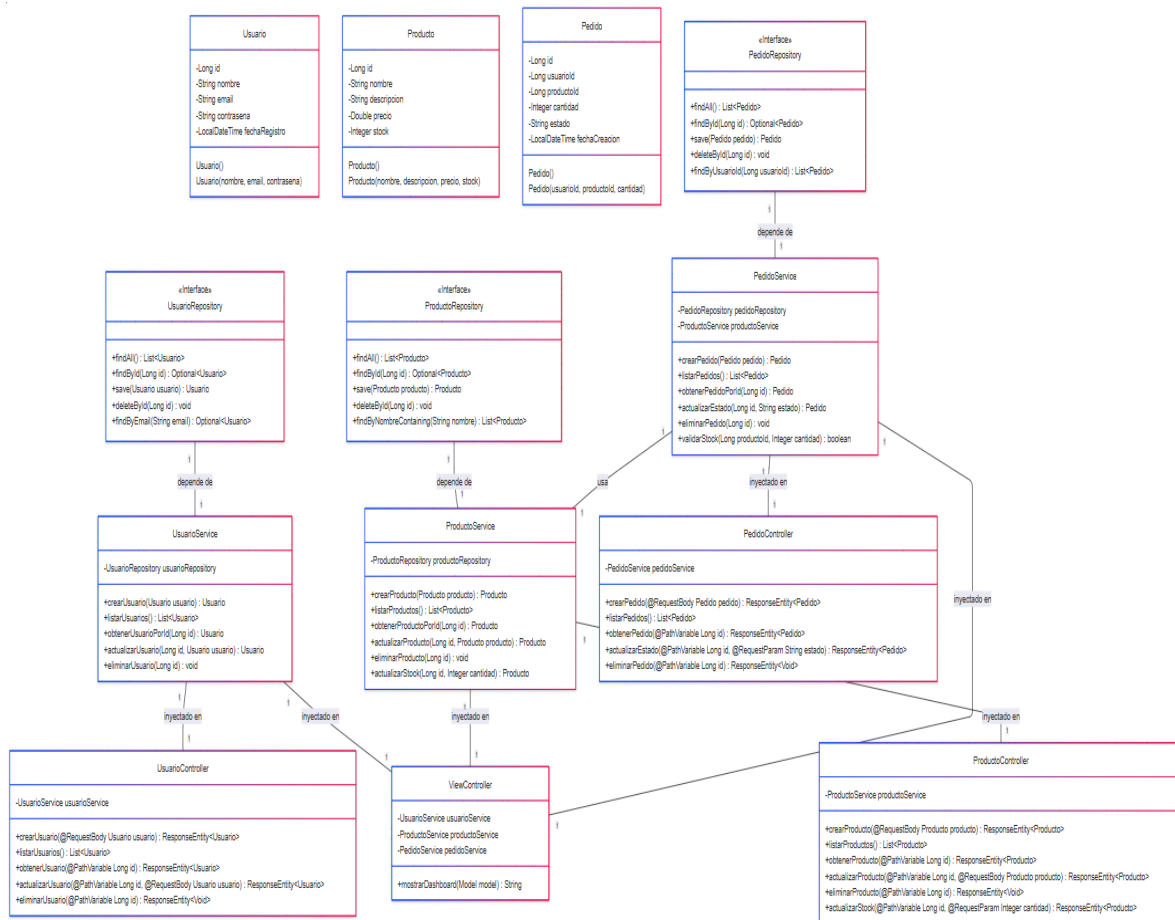


Diagrama de Clases



Capturas de Postman Para Api de Usuarios

- Get de todos los usuarios en la base de datos

The screenshot shows the Postman interface for a GET request to `http://localhost:8080/api/usuarios`. The request is successful, returning a 200 OK status with a response time of 602 ms and a body size of 812 B. The response body is displayed in JSON format, showing an array of three user objects.

Request:

- Method: GET
- URL: `http://localhost:8080/api/usuarios`

Response:

```
200 OK • 602 ms • 812 B
```

```
[{"id": 2, "nombre": "María González", "email": "maria@ecomarket.com", "contrasena": "", "fechaRegistro": null}, {"id": 3, "nombre": "Luis Valdes", "email": "luisvaldes@ecomarket.com", "contrasena": "1234", "fechaRegistro": "2025-06-06T14:38:27.493673"}, {"id": 4, "nombre": "Luis Valdes", "email": "luisvaldes@ecomarket.com", "contrasena": "1234", "fechaRegistro": "2025-06-06T14:38:27.493673"}]
```

- Get de Usuario por ID

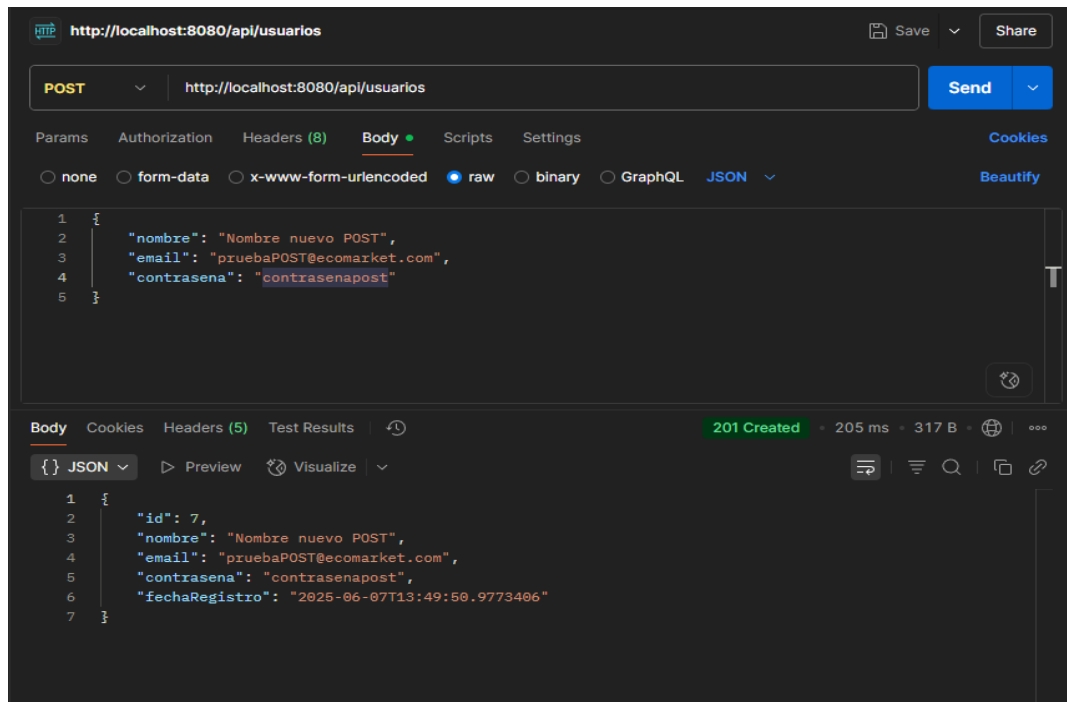
The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/api/usuarios/2`
- Method:** `GET`
- Response Status:** `200 OK` (49 ms, 267 B)
- Response Body (JSON):**

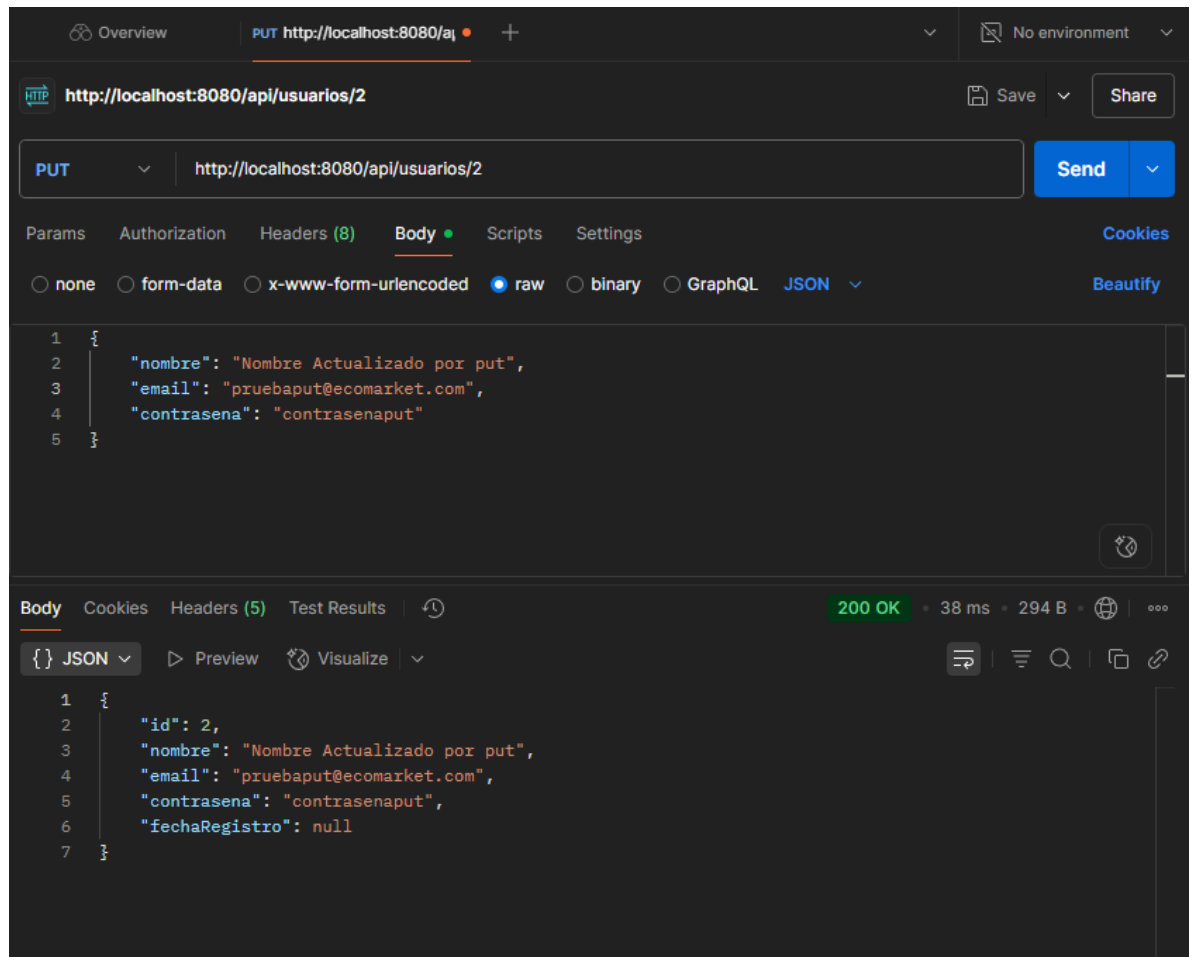
```
1 {
2   "id": 2,
3   "nombre": "María González",
4   "email": "maria@ecomarket.com",
5   "contrasena": "",
6   "fechaRegistro": null
7 }
```
- Query Params Table:**

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

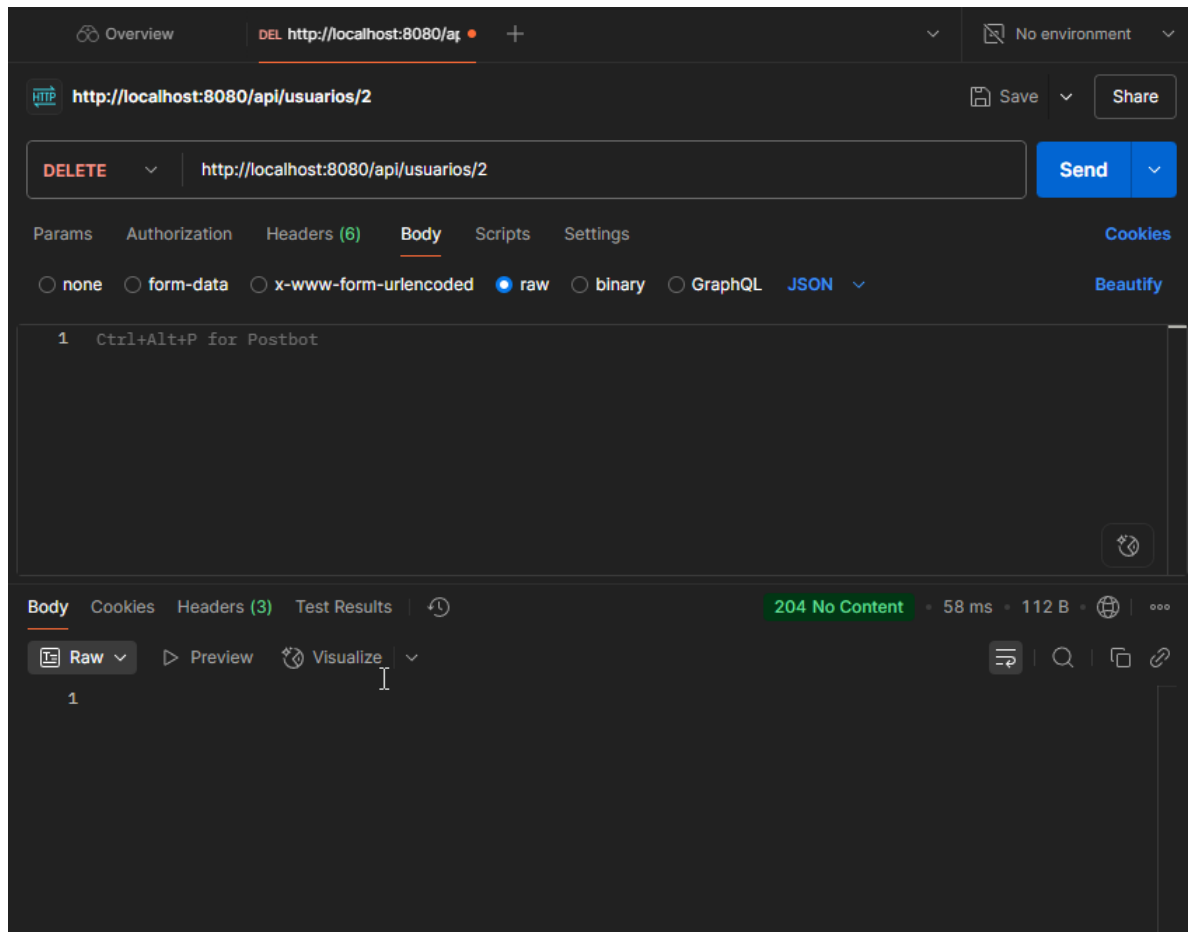
- Post de usuario creado con éxito



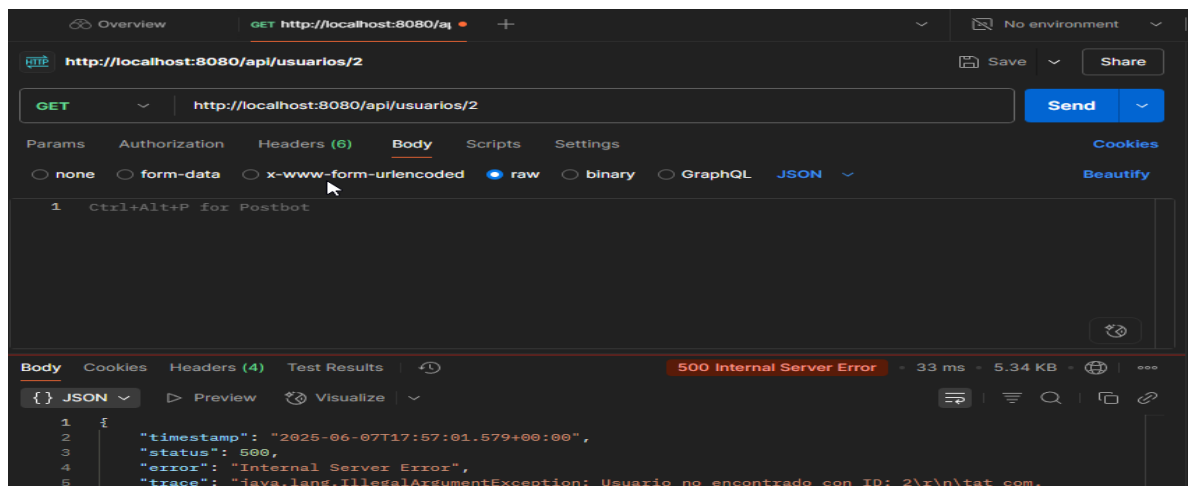
- Put de Usuario actualizado por ID



- Delete Usuario Eliminado por ID

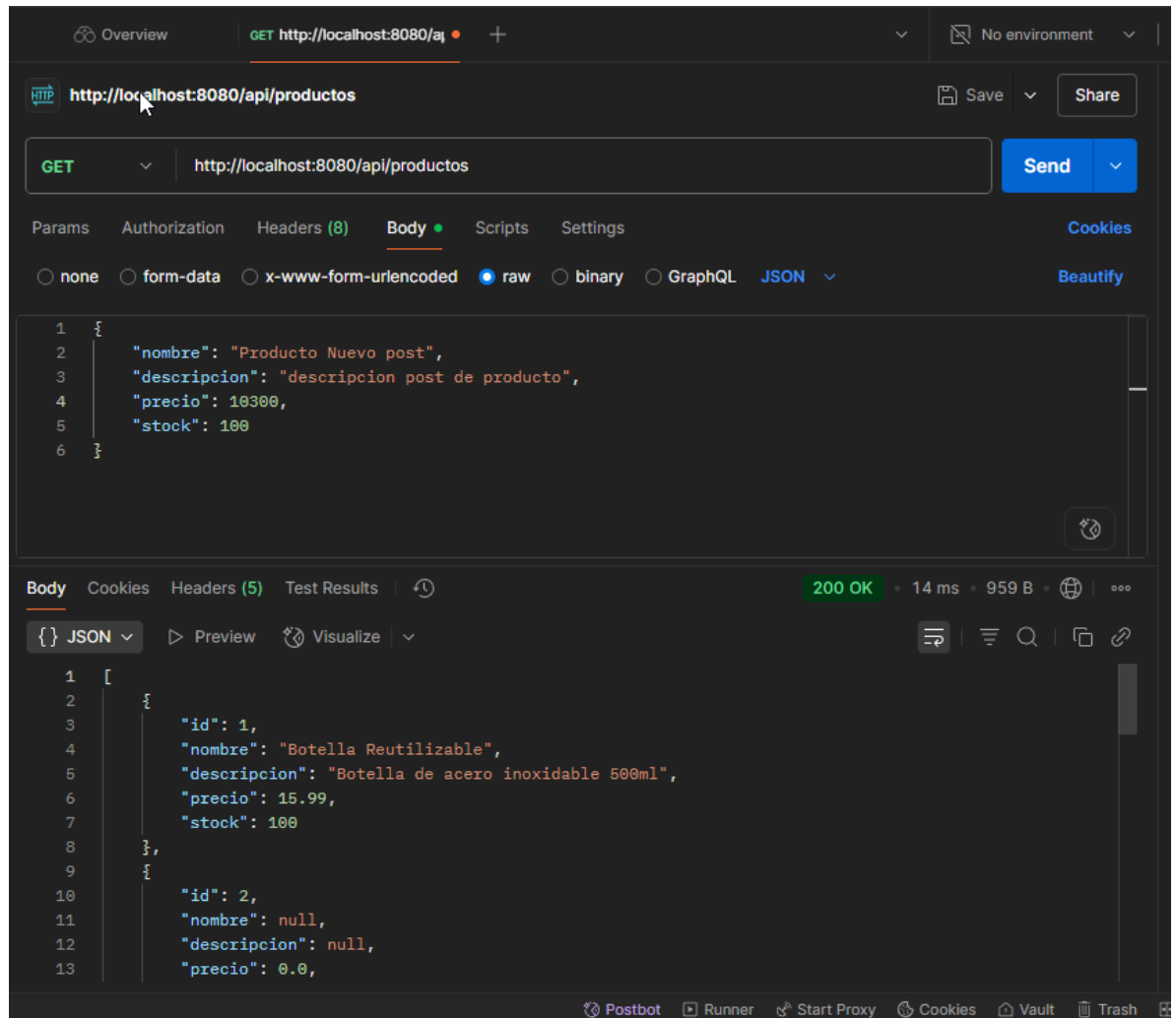


Prueba de que el usuario con ID “2” deajo de existir y no se encuentra al buscarlo.



Capturas de Postman Para Api de Productos.

- Get de todos los Productos en la BD



- Post Peticion para Crear nuevo producto

The screenshot displays a REST client interface with a dark theme. At the top, the URL bar shows `http://localhost:8080/api/productos` with a `POST` method selected. The `Body` tab is active, showing a JSON payload:

```
{  "nombre": "Producto Nuevo post",  "descripcion": "descripcion post de producto",  "precio": 10300,  "stock": 100}
```

. Below the request, the response is shown in the `Body` tab, indicating a `201 Created` status with a response time of 15 ms and a size of 282 B. The response JSON is:

```
{  "id": 8,  "nombre": "Producto Nuevo post",  "descripcion": "descripcion post de producto",  "precio": 10300.0,  "stock": 100}
```

Overview **POST** http://localhost:8080/api/productos

Save Share

POST http://localhost:8080/api/productos Send

Params Authorization Headers (8) **Body** Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

```
1 {
2   "nombre": "Producto Nuevo post",
3   "descripcion": "descripcion post de producto",
4   "precio": 10300,
5   "stock": 100
6 }
```

Body Cookies Headers (5) Test Results 201 Created • 15 ms • 282 B

{ } JSON Preview Visualize

```
1 {
2   "id": 8,
3   "nombre": "Producto Nuevo post",
4   "descripcion": "descripcion post de producto",
5   "precio": 10300.0,
6   "stock": 100
7 }
```

- Get Peticion para Buscar Producto por su ID

The screenshot displays a REST client interface with a dark theme. At the top, the 'Overview' tab is active, showing the URL `http://localhost:8080/api/productos/5` and the method `GET`. Below the URL bar, the 'Body' tab is selected, showing a JSON request body:

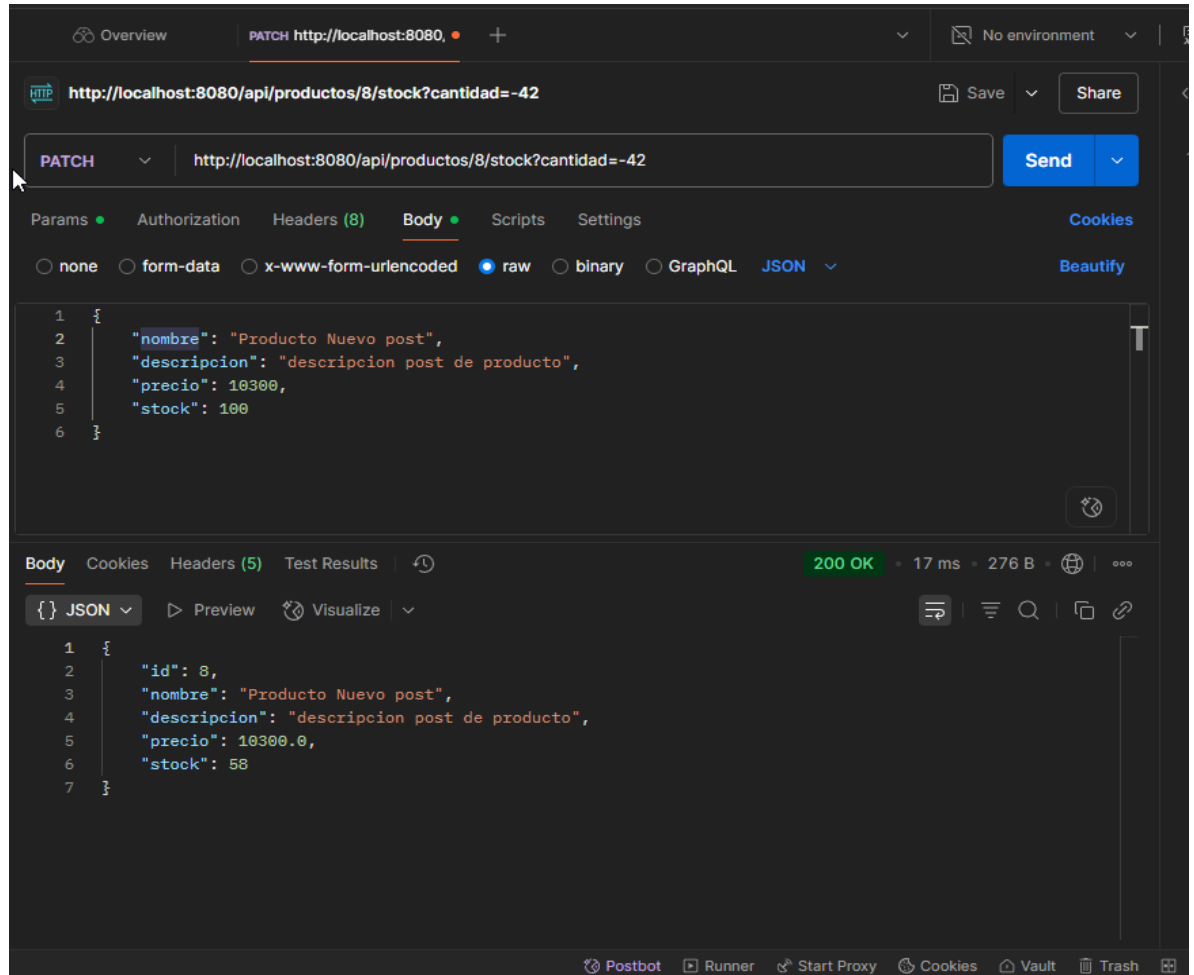
```
{  "nombre": "Producto Nuevo post",  "descripcion": "descripcion post de producto",  "precio": 10300,  "stock": 100}
```

. The 'Send' button is visible. Below the request body, the 'Body' tab of the response is selected, showing a `200 OK` status and a JSON response body:

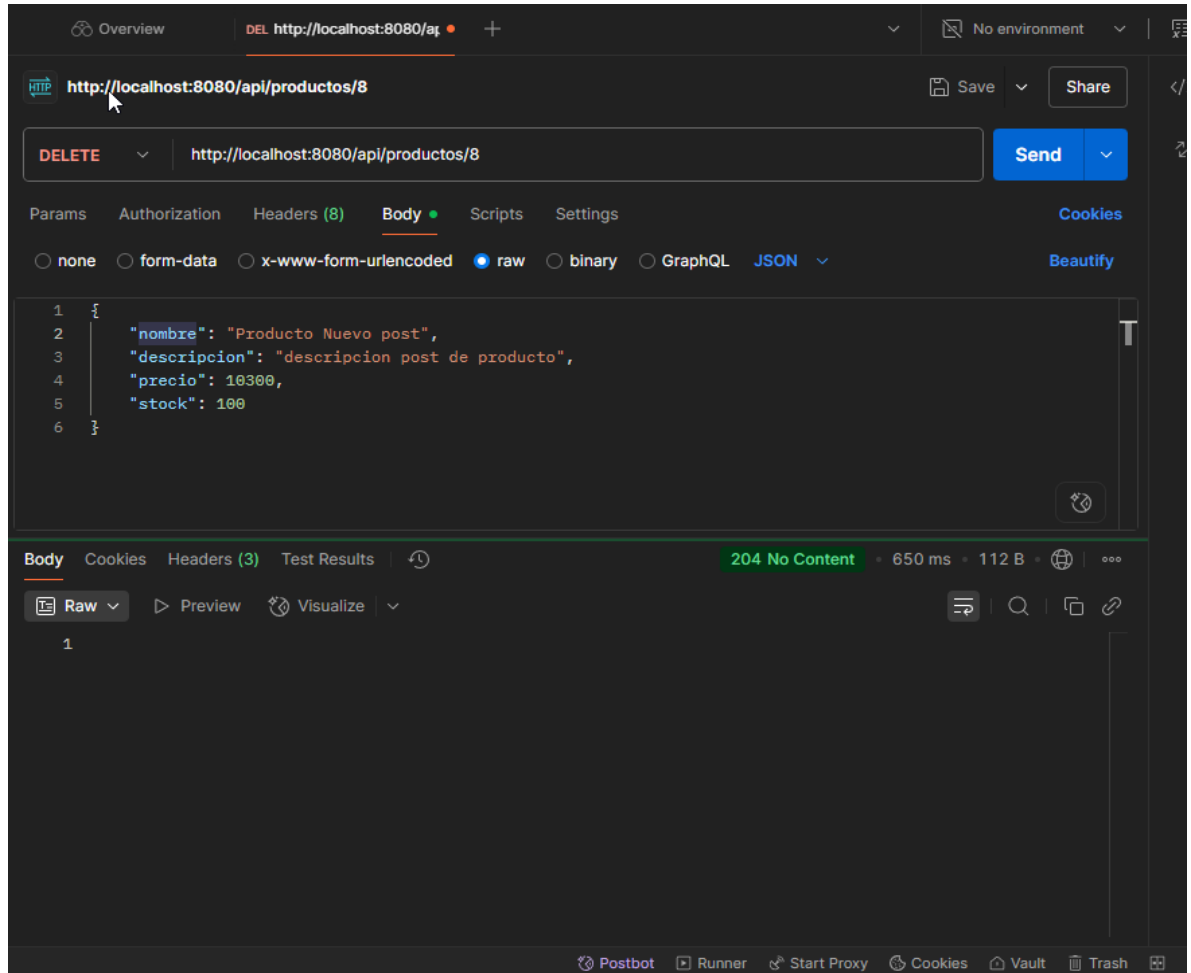
```
{  "id": 5,  "nombre": "Baaaaa",  "descripcion": "Botella de acero inoxidable 500ml",  "precio": 15.99,  "stock": 100}
```

. The interface includes various tabs like 'Params', 'Authorization', 'Headers (8)', 'Scripts', 'Settings', 'Cookies', and 'Test Results'.

- Patch petición para actualizar el Stock de un producto



- Delete petición para eliminar producto por ID



- PUT Petición para actualizar producto por ID

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/api/productos/7`
- Method:** PUT
- Body (raw):**

```
1 {
2   "nombre": "Producto actualizado con put",
3   "descripcion": "descripcion actualizada con put",
4   "precio": 5.99,
5   "stock": 33
6 }
```
- Response (JSON):**

```
1 {
2   "id": 7,
3   "nombre": "Producto actualizado con put",
4   "descripcion": "descripcion actualizada con put",
5   "precio": 5.99,
6   "stock": 33
7 }
```
- Status:** 200 OK
- Time:** 531 ms
- Size:** 285 B

The bottom status bar includes icons for Postbot, Runner, Start Proxy, Cookies, Vault, and Trash.

DEPENDENCIAS USADAS

SPRING DATA JPA: Para Facilitar la integración con bases de datos usando JPA y Hibernate como implementación por defecto y también permitir crear repositorios.

MySQL Connector: Usado para poder conectar el proyecto con la base de datos en MySQL WorkBench

ThymeLeaf: usado para la Vista

Spring Boot devTools: Son las herramientas de desarrollo usadas para la ejecución y reinicio del proyecto reiteradas veces

Componentes Implementados

Entidades: Usuario, Producto, Pedido (JPA)

Repositorios: Interfaces JpaRepository.

Controladores: Endpoints REST (@RestController).

Servicios: Lógica de negocio (@Service).

Base de Datos:

EL motor de base de datos usado fue MySQL Workbench para que interactue con el proyecto

