

Transformación Digital de EcoMarket SPA

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

introducción

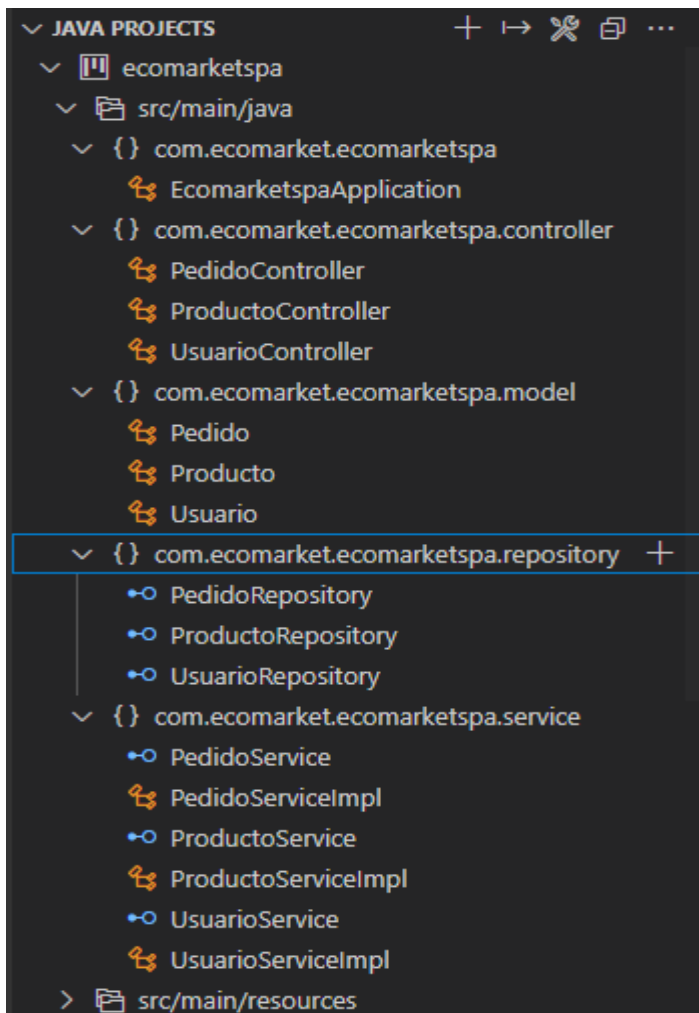
EcoMarketSPA enfrenta desafíos de escalabilidad debido a su sistema monolítico. Este proyecto propone una solución basada en microservicios para mejorar el rendimiento y la eficiencia operativa. Se desarrollaron tres servicios REST: Usuarios, Inventario y Pedidos, utilizando Spring Boot, Maven, MySQL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

Dependencias Maven

```
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
23 <scm>
24 <connection/>
25 <developerConnection/>
26 <tag/>
27 <url/>
28 </scm>
29 <properties>
30 <java.version>21</java.version>
31 </properties>
32 <dependencies>
33
34 <dependency>
35 <groupId>org.springframework.boot</groupId>
36 <artifactId>spring-boot-starter-actuator</artifactId>
37 </dependency>
38
39 <dependency>
40 <groupId>org.springframework.boot</groupId>
41 <artifactId>spring-boot-starter-data-jpa</artifactId>
42 </dependency>
43 <dependency>
44 <groupId>org.springframework.boot</groupId>
45 <artifactId>spring-boot-starter-thymeleaf</artifactId>
46 </dependency>
47 <dependency>
48 <groupId>org.springframework.boot</groupId>
49 <artifactId>spring-boot-starter-web</artifactId>
50 </dependency>
51
52 <dependency>
53 <groupId>org.springframework.boot</groupId>
54 <artifactId>spring-boot-devtools</artifactId>
55 <scope>runtime</scope>
56 <optional>true</optional>
57 </dependency>
58 <dependency>
59 <groupId>com.mysql</groupId>
60 <artifactId>mysql-connector-j</artifactId>
61 <scope>runtime</scope>
62 </dependency>
63 <dependency>
64 <groupId>org.springframework.boot</groupId>
65 <artifactId>spring-boot-starter-test</artifactId>
66 <scope>test</scope>
67 <exclusions>
68 <exclusion>
69 <groupId>org.mockito</groupId>
```

La estructura de paquetes usada fue la siguiente:

- Controller: donde van todas las clases de controlador de la clase y la implementación del controlador REST juntos.
- Model: Posee todas las clases base de cada servicio con sus variables y métodos para construir, get, setter y toString
- Repository: Incluye todas las clases de repository de cada servicio para implementar con JPArepository
- Service: Posee todas las clases de servicio y servicio de implementación de cada servicio



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).

Diagrama de Arquitectura

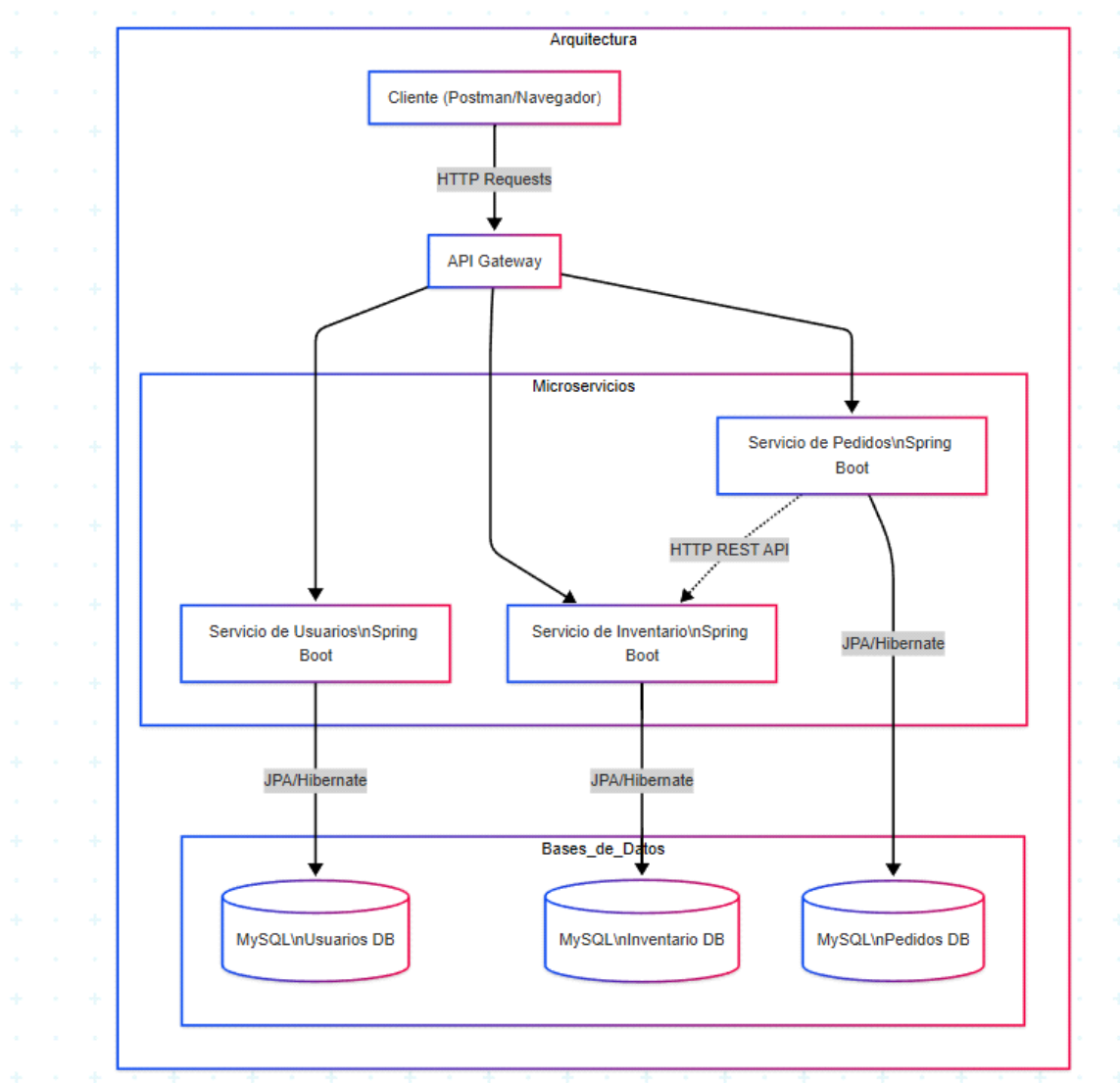
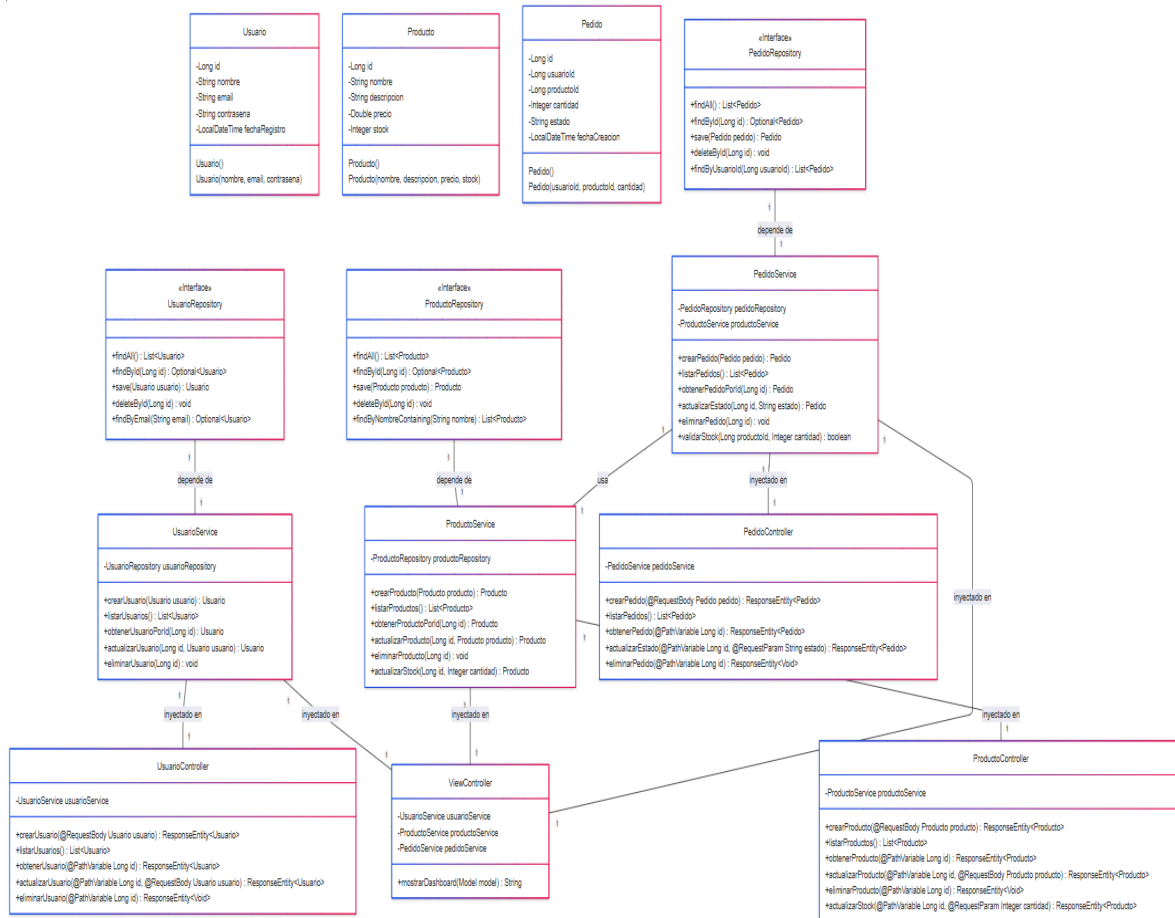
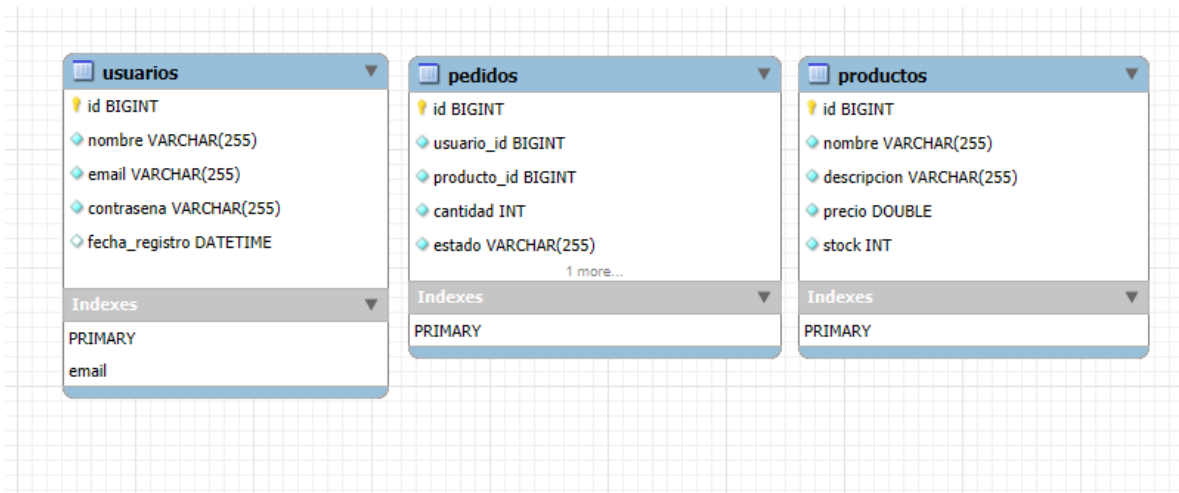


Diagrama de Clases



El motor de la base de datos utilizado fue MySQL WORKBENCH 8.0 CE

Modelo SCHEME de MySQL de la base de datos de Usuarios, Productos y Pedidos



Capturas de Postman Para Api de Usuarios

- Get de todos los usuarios en la base de datos

The screenshot shows the Postman interface with a GET request to `http://localhost:8080/api/usuarios`. The response is a 200 OK status with a JSON body containing three user records.

Query Params

Key	Value	Description
Key	Value	Description

Body

```
1 [
2   {
3     "id": 2,
4     "nombre": "María González",
5     "email": "maria@ecomarket.com",
6     "contrasena": "",
7     "fechaRegistro": null
8   },
9   {
10    "id": 3,
11    "nombre": "Luis Valdes",
12    "email": "luisvaldes@ecomarket.com",
13    "contrasena": "1234",
14    "fechaRegistro": "2025-06-06T14:38:27.493673"
15  },
16  {
17    "id": 4,
```

Postman interface elements visible: Overview, GET http://localhost:8080/api/usuarios, Save, Share, Params, Authorization, Headers (8), Body, Scripts, Settings, Cookies, Body, Cookies, Headers (5), Test Results, 200 OK, 602 ms, 812 B, JSON, Preview, Visualize, Postbot, Runner, Start Proxy, Cookies, Vault, Trash.

- 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`
- Query Params Table:**

Key	Value	Description
Key	Value	Description
- Status:** `200 OK` (49 ms, 267 B)
- Body (JSON):**

```
{
  "id": 2,
  "nombre": "Maria González",
  "email": "maria@ecomarket.com",
  "contrasena": "",
  "fechaRegistro": null
}
```

- Post de usuario creado con éxito

The screenshot displays a REST client interface for a POST request to `http://localhost:8080/api/usuarios`. The request body is a JSON object with the following fields:

```
1 {
2   "nombre": "Nombre nuevo POST",
3   "email": "pruebaPOST@ecomarket.com",
4   "contrasena": "contrasenapost"
5 }
```

The response status is **201 Created**, with a response time of 205 ms and a body size of 317 B. The response body is shown in JSON format:

```
1 {
2   "id": 7,
3   "nombre": "Nombre nuevo POST",
4   "email": "pruebaPOST@ecomarket.com",
5   "contrasena": "contrasenapost",
6   "fechaRegistro": "2025-06-07T13:49:50.9773406"
7 }
```

- Put de Usuario actualizado por ID

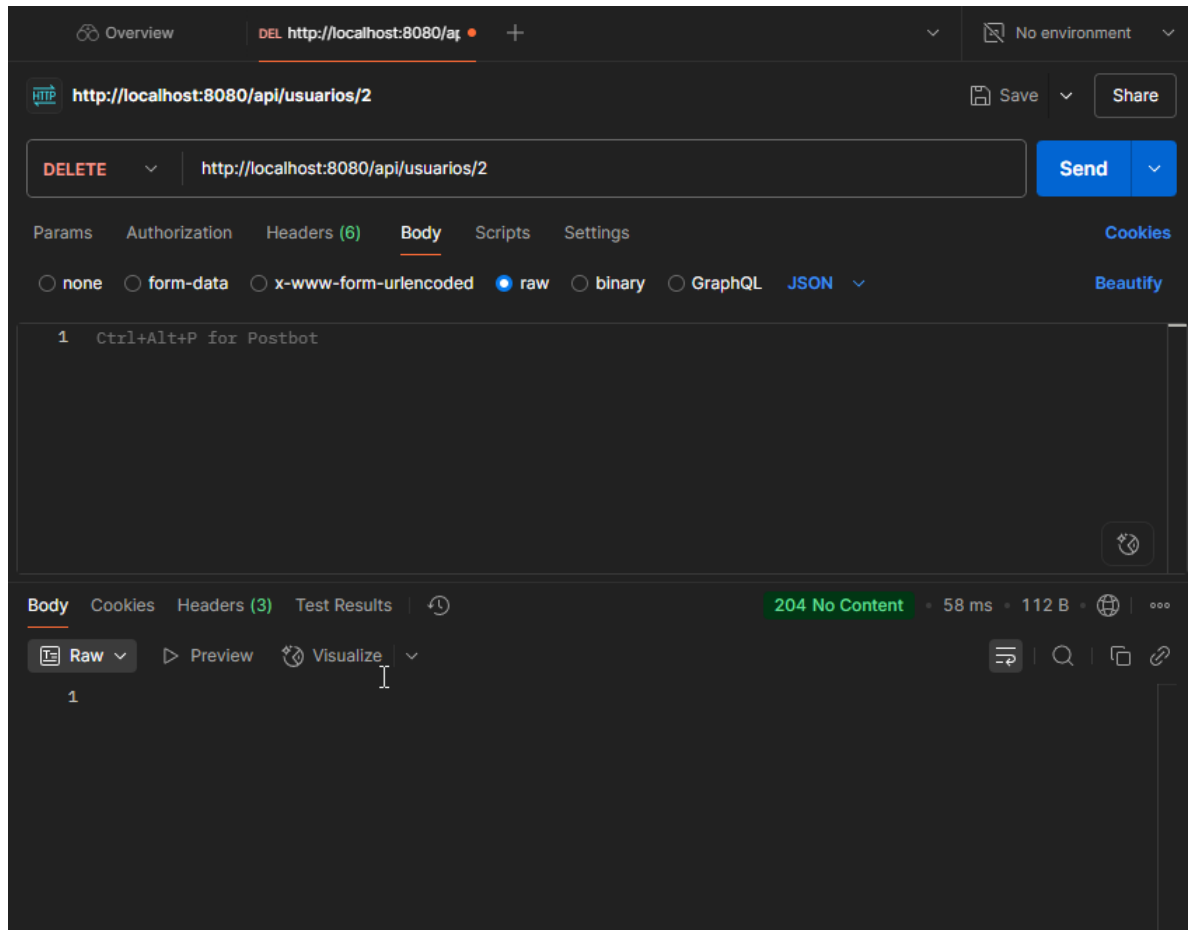
The screenshot shows a REST client interface with a PUT request to `http://localhost:8080/api/usuarios/2`. The request body is a JSON object with the following fields:

```
1 {  
2   "nombre": "Nombre Actualizado por put",  
3   "email": "pruebaput@ecomarket.com",  
4   "contrasena": "contrasenaput"  
5 }
```

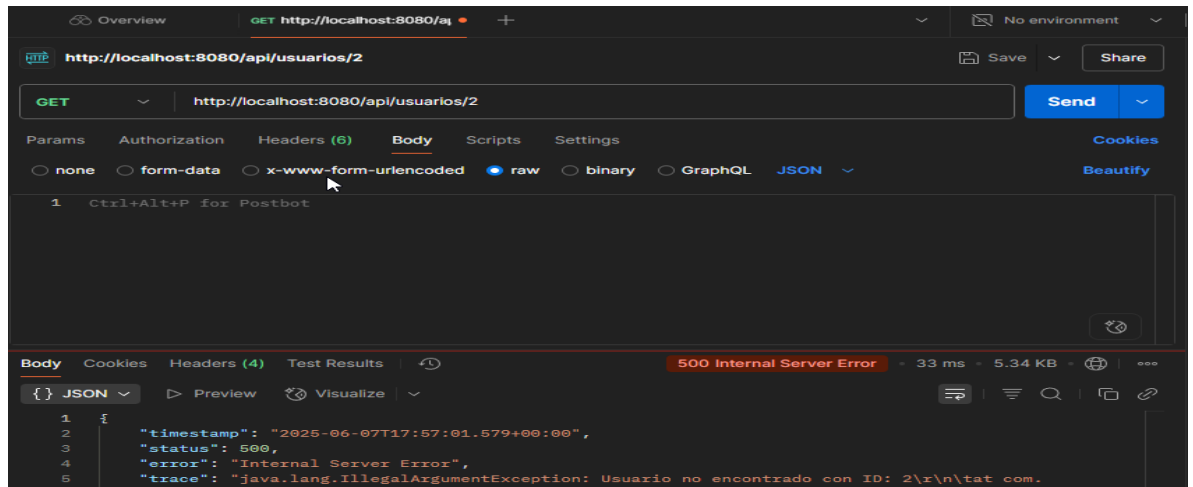
The response is a 200 OK status with a response time of 38 ms and a body size of 294 B. The response body is a JSON object with the following fields:

```
1 {  
2   "id": 2,  
3   "nombre": "Nombre Actualizado por put",  
4   "email": "pruebaput@ecomarket.com",  
5   "contrasena": "contrasenaput",  
6   "fechaRegistro": null  
7 }
```

- Delete Usuario Eliminado por ID

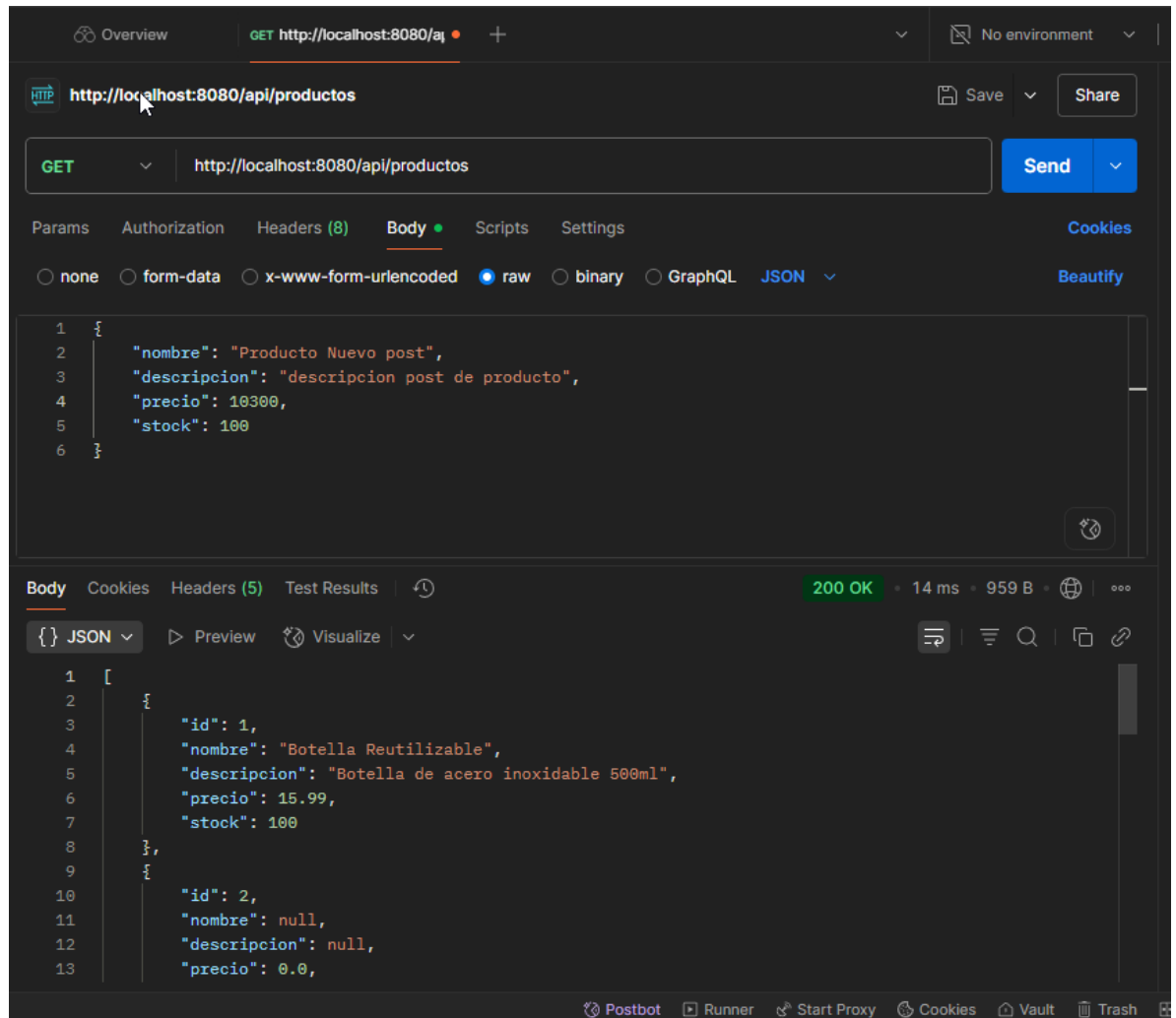


Prueba de que el usuario con ID “2” dejo de existir y no se encuentra al buscarlo.



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/` + No environment

HTTP `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:

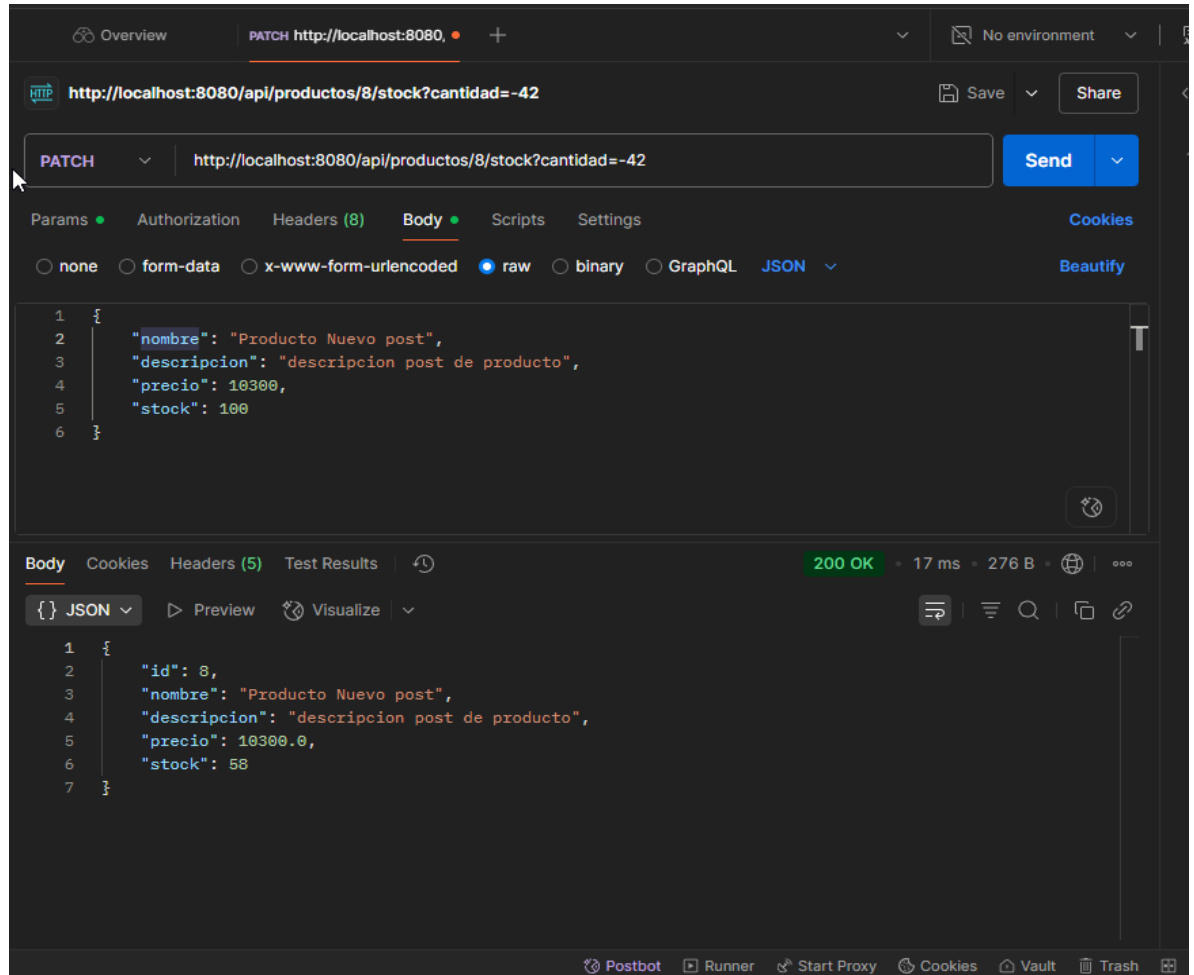
```
{  "nombre": "Producto Nuevo post",  "descripcion": "descripcion post de producto",  "precio": 10300,  "stock": 100}
```

. 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:

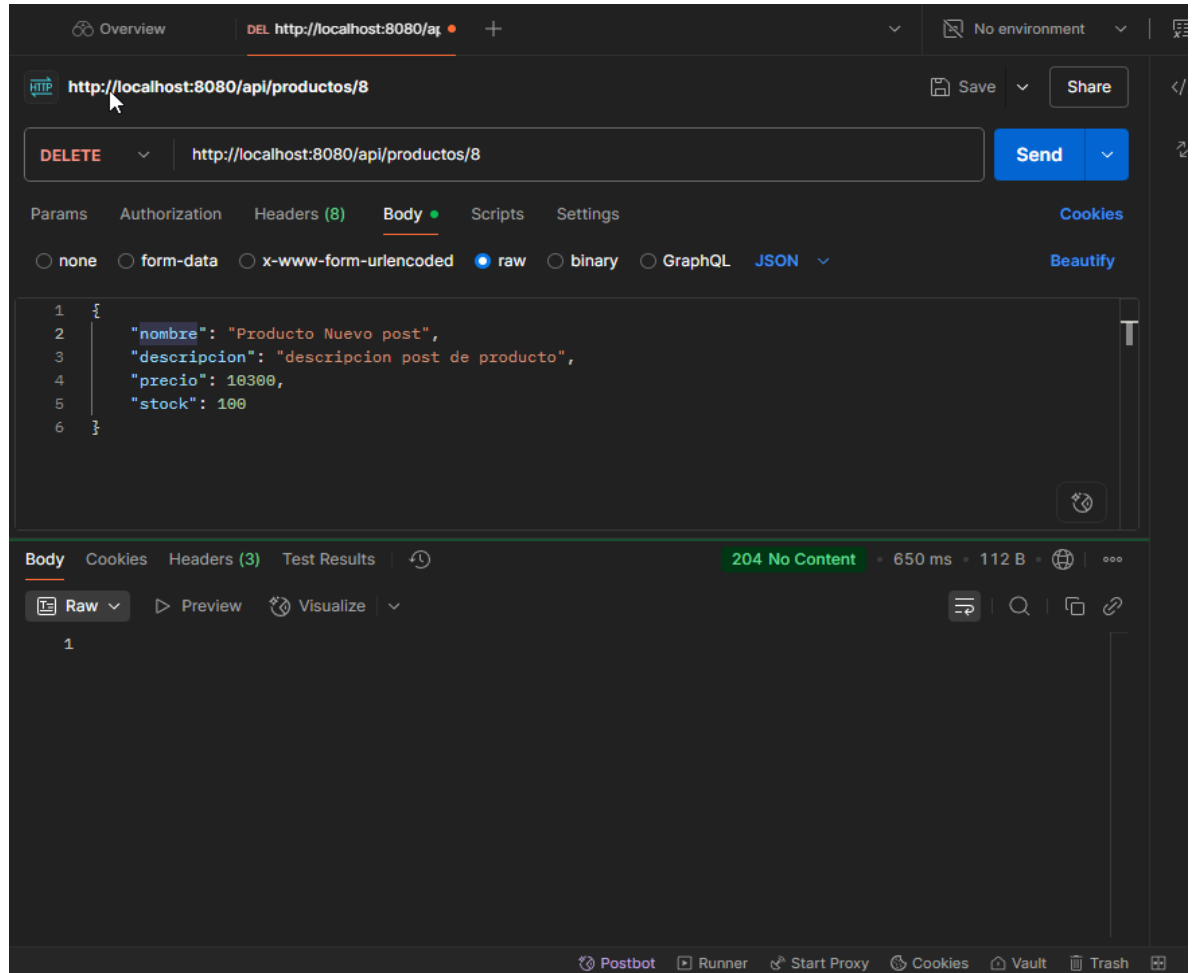
```
{  "id": 5,  "nombre": "Baaaaa",  "descripcion": "Botella de acero inoxidable 500ml",  "precio": 15.99,  "stock": 100}
```

. The interface includes tabs for Params, Authorization, Headers (8), Body, Scripts, and Settings. The response status bar shows `200 OK`, `12 ms`, and `267 B`.

- Patch petición para actualizar el Stock de un producto



- Petición Delete 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:** 200 OK, 531 ms, 285 B
- Response Body (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 }
```

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

Capturas de Postman Para Api de Pedidos.

- Peticion Get de todos los Pedidos

The screenshot shows the Postman interface for a GET request to `http://localhost:8080/api/pedidos`. The request is successful, returning a `200 OK` status with a response time of 19 ms and a body size of 1.45 KB. The response body is displayed in JSON format, showing an array of two order objects.

Query Params

Key	Value	Description
Key	Value	Description

Body

```
[
  {
    "id": 1,
    "usuarioId": 2,
    "productoId": 1,
    "cantidad": 2,
    "estado": "PENDIENTE",
    "fechaCreacion": "2025-06-17T17:10:00"
  },
  {
    "id": 2,
    "usuarioId": 2,
    "productoId": 3,
    "cantidad": 5,
    "estado": "ENVIADO",
    "fechaCreacion": "2025-06-17T17:10:00"
  }
]
```

- Peticion Get Para buscar pedido por ID

The screenshot shows a REST client interface with a GET request to `http://localhost:8080/api/pedidos/3`. The response is a 200 OK status with a JSON body containing order details.

Request:

- Method: GET
- URL: `http://localhost:8080/api/pedidos/3`

Response:

- Status: 200 OK
- Time: 14 ms
- Size: 274 B

JSON Body:

```
1 {
2   "id": 3,
3   "usuarioId": 3,
4   "productoId": 2,
5   "cantidad": 3,
6   "estado": "COMPLETADO",
7   "fechaCreacion": "2025-06-17T17:10:08"
8 }
```

Table: Query Params

Key	Value	Description
Key	Value	Description

- Peticion Post para crear un nuevo Pedido

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

- URL:** `http://localhost:8080/api/pedidos`
- Method:** `POST`
- Body (raw):**

```
1 {  
2   "usuarioId": 2,  
3   "productoId": 3,  
4   "cantidad": 4  
5 }
```
- Response Status:** `201 Created` (13 ms, 287 B)
- Response Body (JSON):**

```
1 {  
2   "id": 13,  
3   "usuarioId": 2,  
4   "productoId": 3,  
5   "cantidad": 4,  
6   "estado": "PENDIENTE",  
7   "fechaCreacion": "2025-06-20T13:04:21.2819972"  
8 }
```

The interface includes tabs for Params, Authorization, Headers (8), Body, Scripts, and Settings. The Body tab is active, showing the raw JSON input and the JSON response. The response status bar indicates a successful creation of the resource.

- Peticion Patch para actualizar estado del pedido

Antes


The screenshot shows a REST client interface with a PATCH request to `http://localhost:8080/api/pedidos/1/estado?estado=COMPLETADO`. The request is configured with the following parameters:

Key	Value	Description
estado	COMPLETADO	

The response is a 200 OK status with a 26 ms response time and 1.56 KB of data. The response body is a JSON array of two order objects:

```
[
  {
    "id": 1,
    "usuarioId": 2,
    "productoId": 1,
    "cantidad": 2,
    "estado": "PENDIENTE",
    "fechaCreacion": "2025-06-17T17:10:08"
  },
  {
    "id": 2,
    "usuarioId": 2,
    "productoId": 3,
    "cantidad": 5,
    "estado": "ENVIADO",
    "fechaCreacion": "2025-06-17T17:10:08"
  }
]
```

después

 **http://localhost:8080/api/pedidos/1/estado?estado=COMPLETADO**

Save

Share

PATCH

http://localhost:8080/api/pedidos/1/estado?estado=COMPLETADO

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	estado	COMPLETADO			
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

200 OK

38 ms

274 B

...

{ } JSON

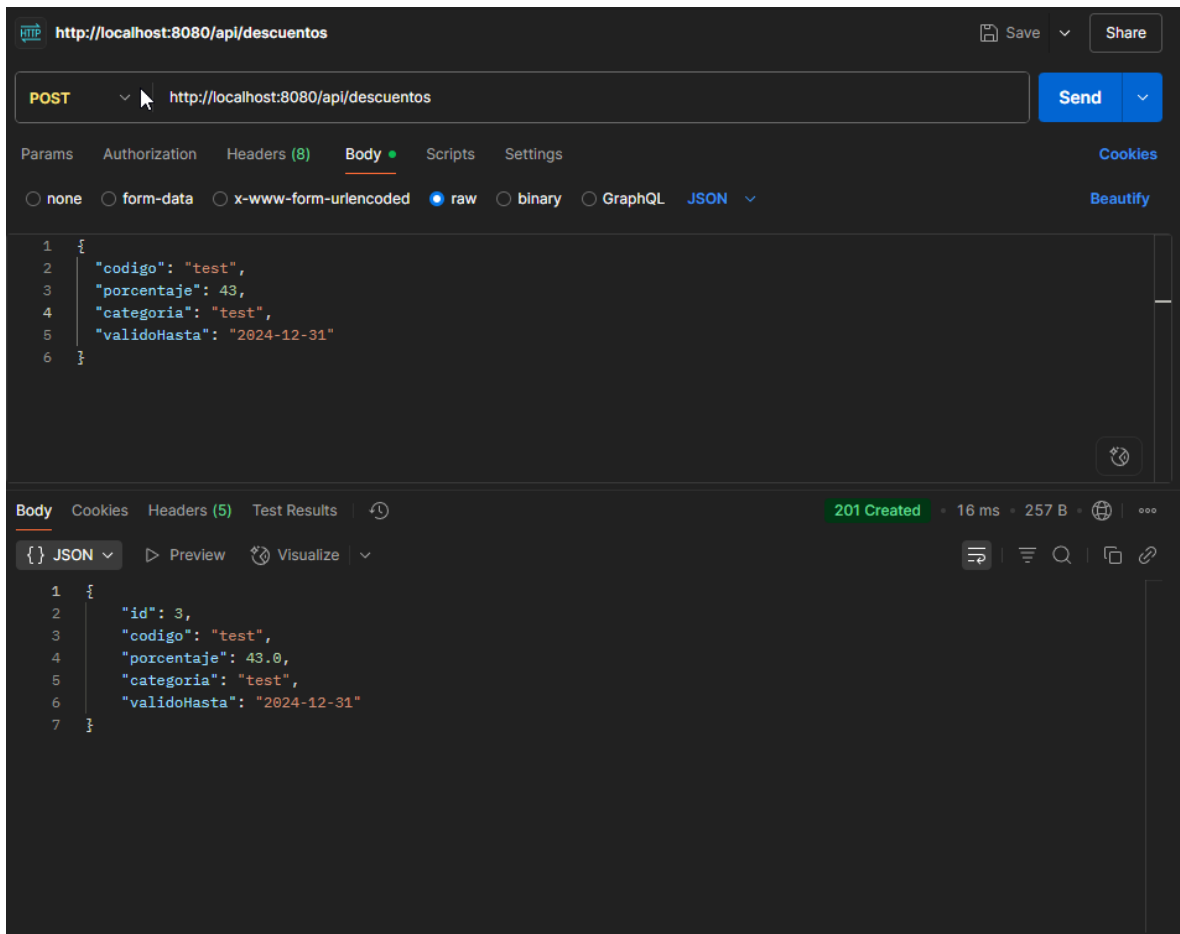
Preview

Visualize

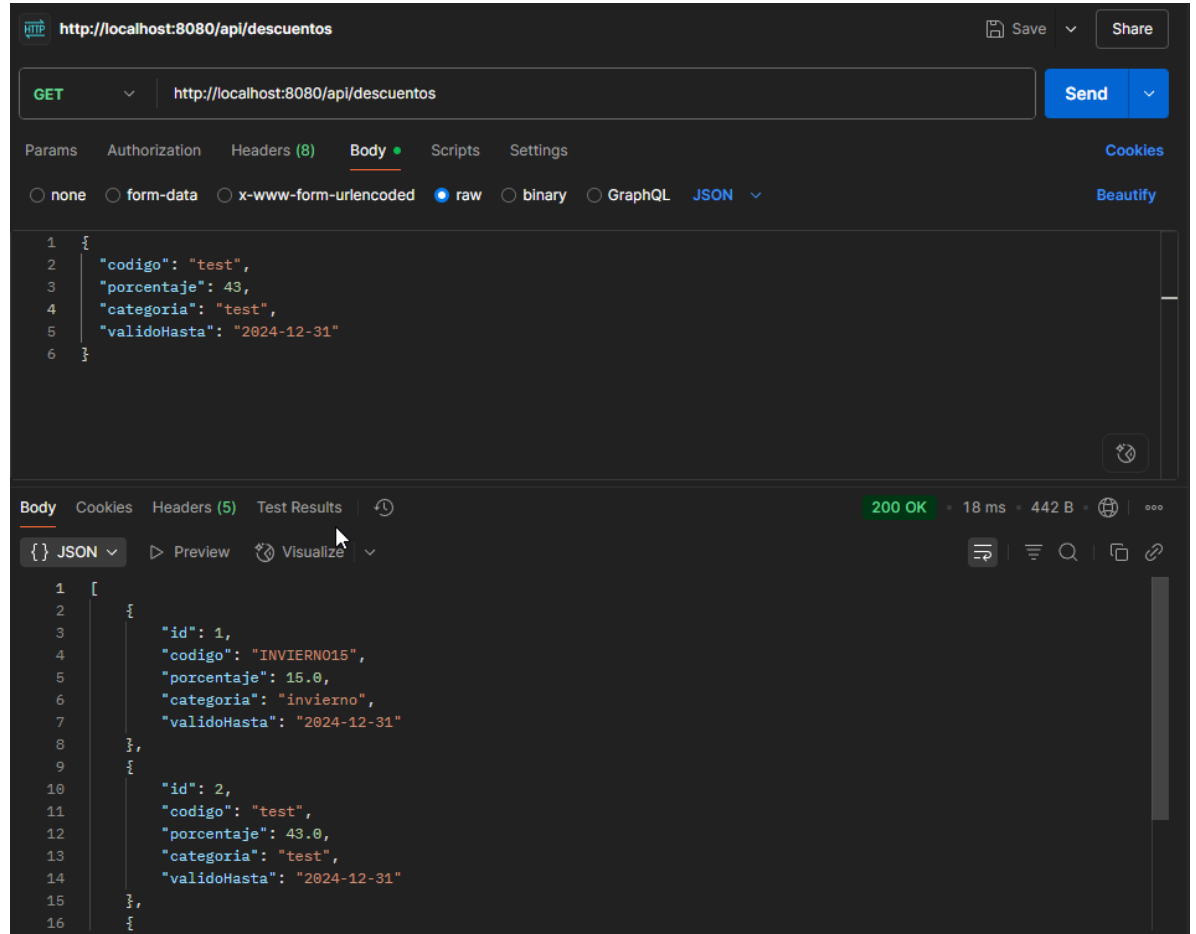
```
1 {
2   "id": 1,
3   "usuarioId": 2,
4   "productoId": 1,
5   "cantidad": 2,
6   "estado": "COMPLETADO",
7   "fechaCreacion": "2025-06-17T17:10:08"
8 }
```


Capturas de Postman Para Api de Descuentos.

- Peticion Post para crear descuento



- Peticion Get para obtener lista de todos los descuentos existentes



```
C:\WINDOWS\system32\cmd. x + v
Reinitialized existing Git repository in C:/Users/Victor/Desktop/subida/Exp3/.git/
[main 3fa96b8] Subida de Exp3
Already up to date.
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 1.29 MiB | 1.13 MiB/s, done.
Total 9 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/VictorG238/Exp3_Baeza_Paredes.git
1c86fbb..3fa96b8 main -> main
branch 'main' set up to track 'origin/main'.
Presione una tecla para continuar . . . |
```

```
MINGW64 ~/Desktop/subida
$ git add Exp2

Victor@DESKTOP-ORFG561 MINGW64 ~/Desktop/subida (main)
$ git commit -m "Experiencia 3 Testing"
[main 1e68e2c] Experiencia 3 Testing
1 file changed, 1 insertion(+), 1 deletion(-)

Victor@DESKTOP-ORFG561 MINGW64 ~/Desktop/subida (main)
$ git push origin main
To https://github.com/VictorG238/Exp3_Baeza_Paredes.git
! [rejected] main -> main (non-fast-forward)
error: failed to push some refs to 'https://github.com/VictorG238/Exp3_Baeza_Paredes.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. If you want to integrate the remote changes,
hint: use 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

Victor@DESKTOP-ORFG561 MINGW64 ~/Desktop/subida (main)
$ AC

Victor@DESKTOP-ORFG561 MINGW64 ~/Desktop/subida (main)
$ git push -u origin main
To https://github.com/VictorG238/Exp3_Baeza_Paredes.git
! [rejected] main -> main (fetch first)
error: failed to push some refs to 'https://github.com/VictorG238/Exp3_Baeza_Paredes.git'
hint: Updates were rejected because the remote contains work that you do not
hint: have locally. This is usually caused by another repository pushing to
hint: the same ref. If you want to integrate the remote changes, use
hint: 'git pull' before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

Victor@DESKTOP-ORFG561 MINGW64 ~/Desktop/subida (main)
$
```