Documentación de la API - ServidorAPI_20222297

Alumno: Cesar Fabricio Tirado Vera
Código: 20222297

Proyecto: Servidor API para gestionar inventario de productos (Northwind - tabla “products”).

Base de datos:
- URL: “jdbc:mysql://lewisrp.dev:3306/Northwind”
- Usuario: “root”
- Contraseña: “root”

Contenido: descripción de los endpoints expuestos por el servicio REST.

Endpoints

1) Listar productos
- Método: GET
- URL: “/product"
- Descripción: Devuelve la lista completa de productos.

Entrada:
- Ninguna

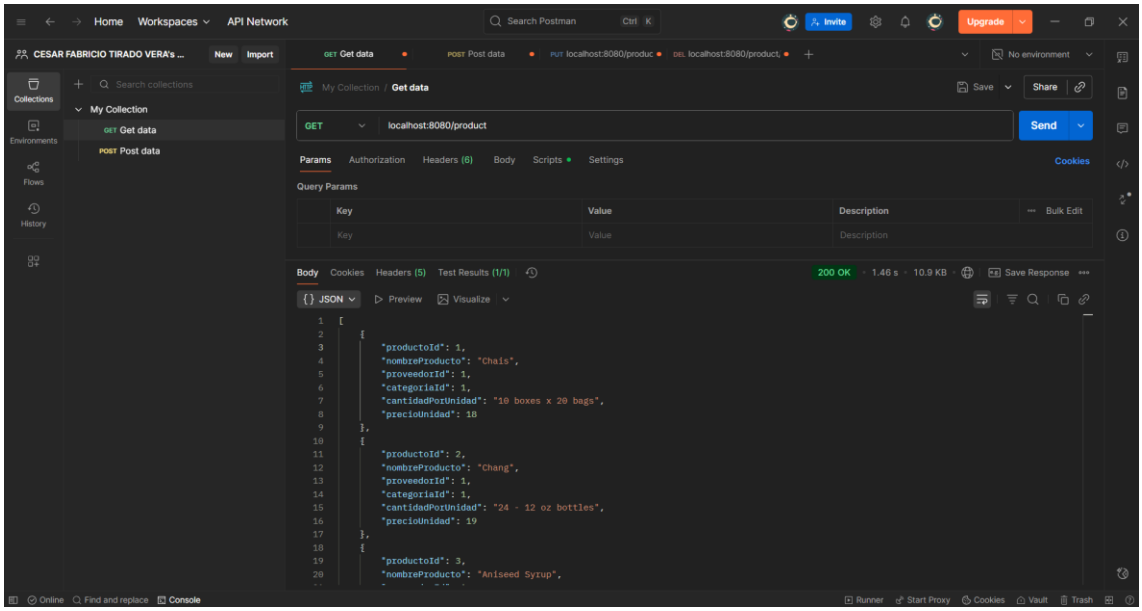
Salida (200 OK):
- Array JSON con objetos “Producto”.
- Ejemplo (considerando los dos primeros productos):


```
 {
   "productoId": 1,
   "nombreProducto": "Chai",
   "proveedorId": 1,
   "categoriaId": 1,
   "cantidadPorUnidad": "10 boxes x 20 bags",
   "precioUnidad": 18.00,
 },
 {
   "productoId": 2,
   "nombreProducto": "Chang",
   "proveedorId": 1,
   "categoriaId": 1,
   "cantidadPorUnidad": "24 - 12 oz bottles",
   "precioUnidad": 19.00,
 }
```

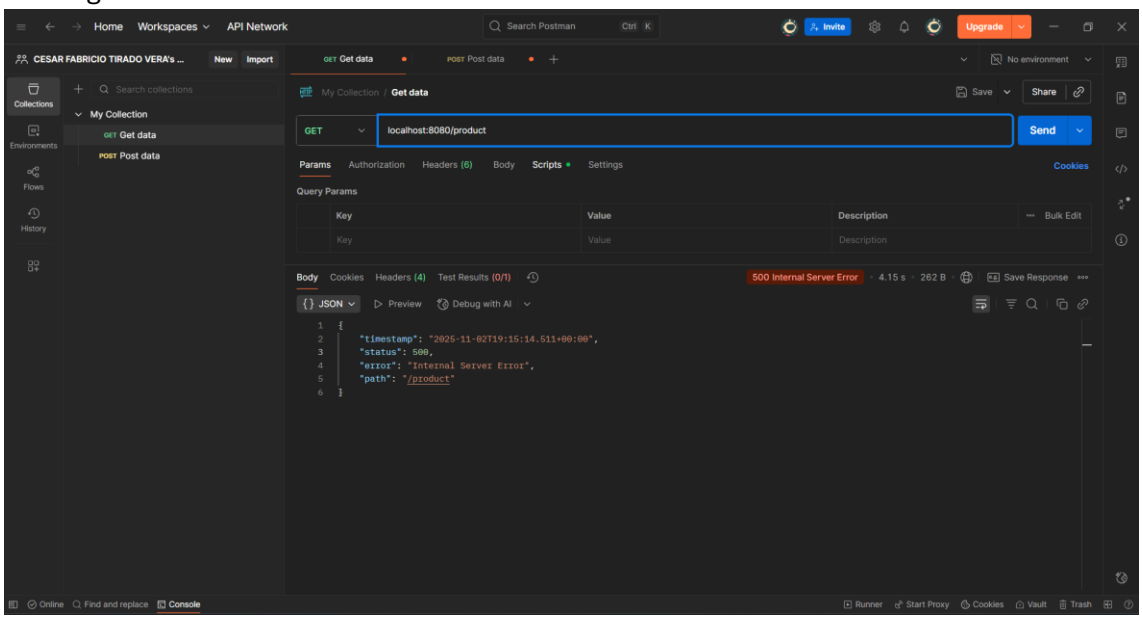

Códigos HTTP posibles:
- 200 OK: listado devuelto correctamente.
- 500 Internal Server Error: error en el servidor (BD, etc.).

Imágenes desde Postman:

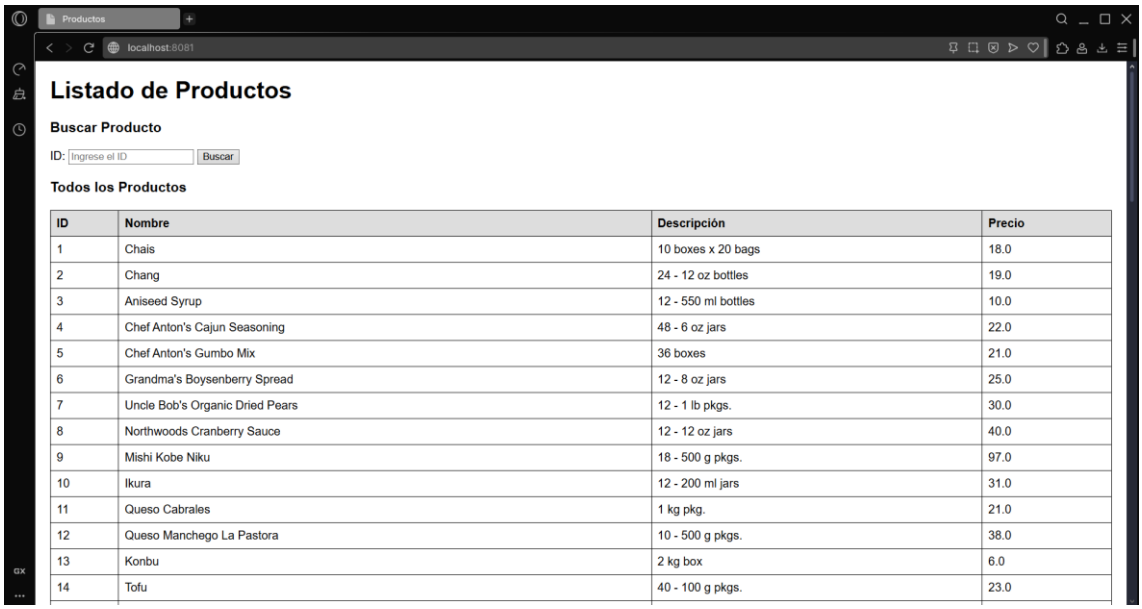
- Código HTTP 200 OK



- Código HTTP 500 Internal Server Error



Vista desde el Cliente Web



ID	Nombre	Descripción	Precio
1	Chais	10 boxes x 20 bags	18.0
2	Chang	24 - 12 oz bottles	19.0
3	Aniseed Syrup	12 - 550 ml bottles	10.0
4	Chef Anton's Cajun Seasoning	48 - 6 oz jars	22.0
5	Chef Anton's Gumbo Mix	36 boxes	21.0
6	Grandma's Boysenberry Spread	12 - 8 oz jars	25.0
7	Uncle Bob's Organic Dried Pears	12 - 1 lb pkgs.	30.0
8	Northwoods Cranberry Sauce	12 - 12 oz jars	40.0
9	Mishi Kobe Niku	18 - 500 g pkgs.	97.0
10	Ikura	12 - 200 ml jars	31.0
11	Queso Cabrales	1 kg pkg.	21.0
12	Queso Manchego La Pastora	10 - 500 g pkgs.	38.0
13	Konbu	2 kg box	6.0
14	Tofu	40 - 100 g pkgs.	23.0

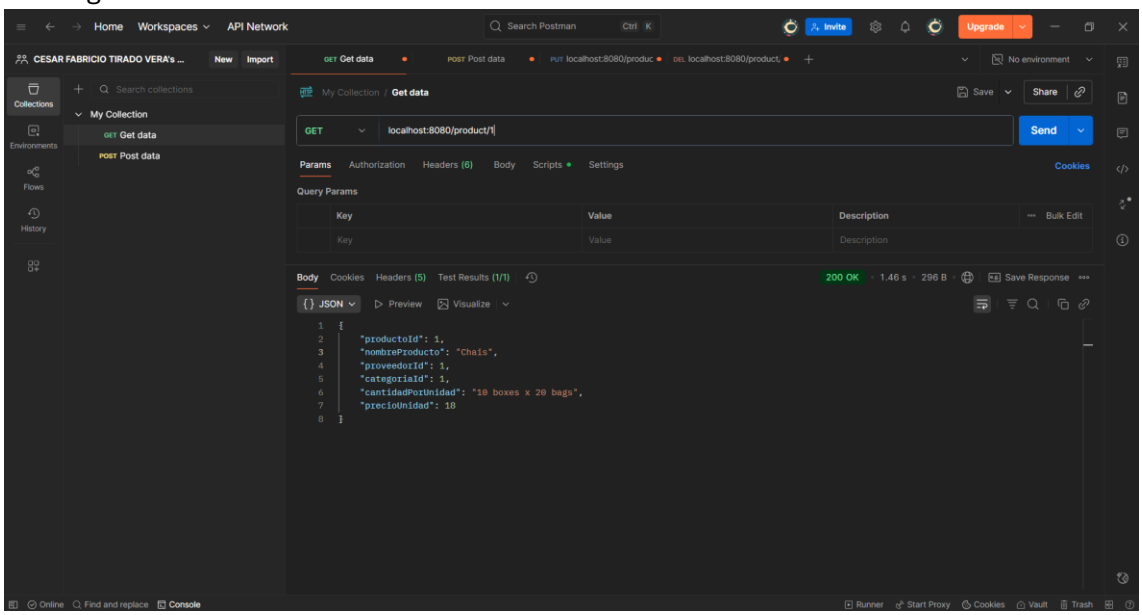
2) Obtener producto por ID
- Método: GET
- URL: “/product/{id}”
- Descripción: Devuelve el producto con el ID especificado.

Entrada:
 - Parámetro de ruta: “id” (entero)
Salida:
 - 200 OK: objeto JSON con los detalles del producto.
 - Ejemplo:
{
 "productoId": 1,
 "nombreProducto": "Chai",
 "proveedorId": 1,
 "categoriaId": 1,
 "cantidadPorUnidad": "10 boxes x 20 bags",
 "precioUnidad": 18.00,
}

- 400 Bad Request: si el “id” no es numérico válido o si el producto con ese ID no existe.
Respuesta JSON con mensaje:
{
"mensaje": "Producto con id X no encontrado"
}
o
{
"mensaje": "ID de producto inválido"
}

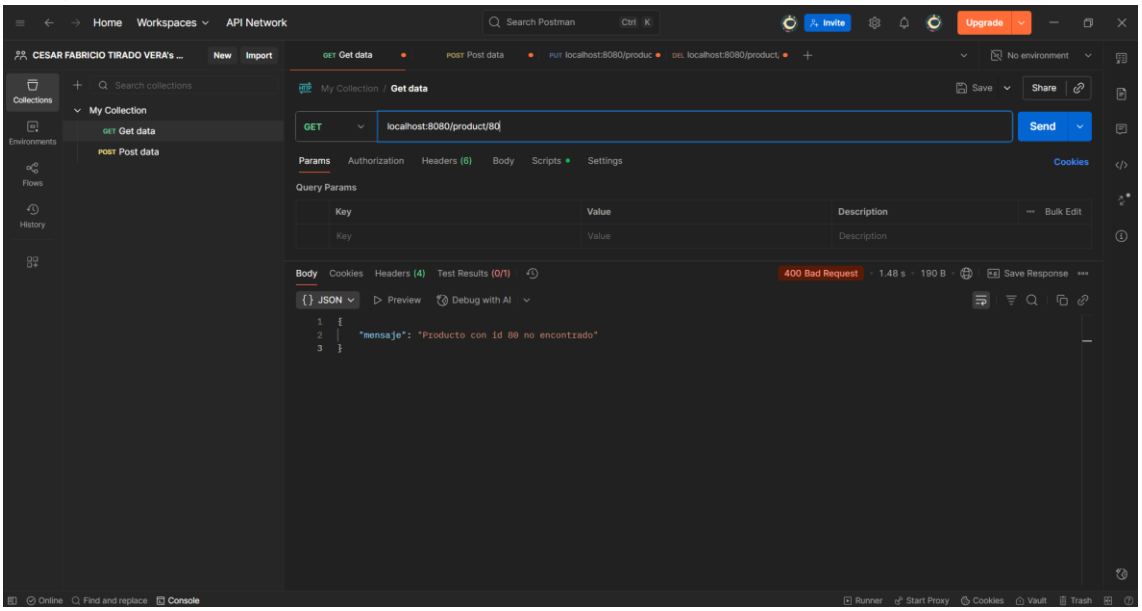
Imágenes desde Postman:

- Código HTTP 200 OK

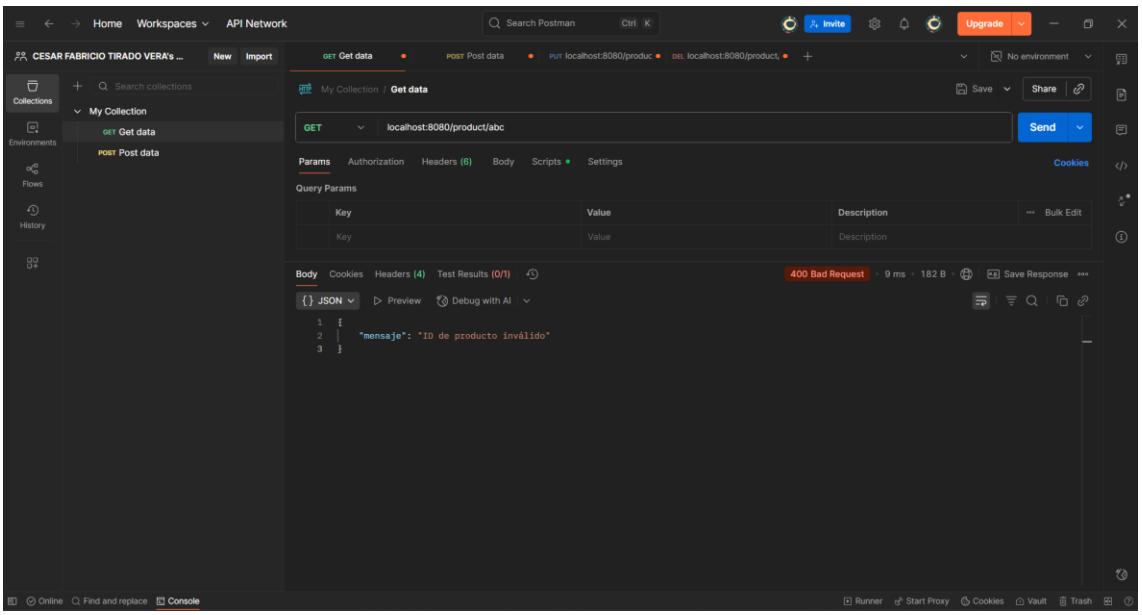


- Código HTTP 400 Bad Request:

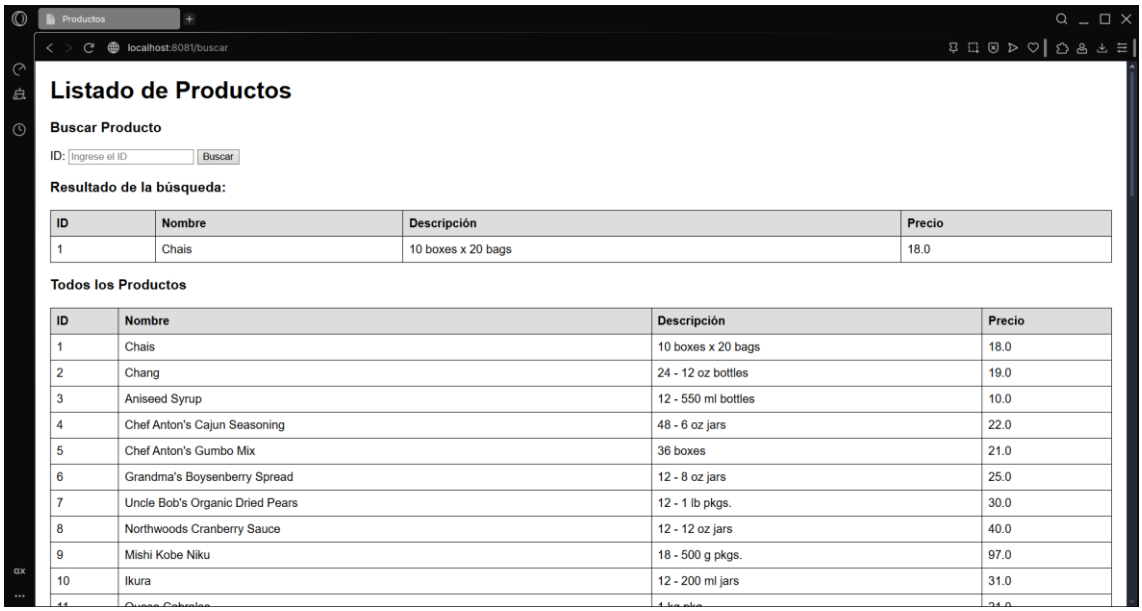
    - Si el producto con ese ID no existe



    - Si el ID no es numérico válido



Vista desde el Cliente Web



3) Crear un producto
- Método: POST
- URL: “/producto”
- Descripción: Crea un nuevo producto en la base de datos.

Entrada (body JSON): objeto “Producto” (los campos que no se envíen se dejarán null si la BD lo permite).
- Ejemplo de body:

```
{
  "productoId": 100,
  "nombreProducto": "Nuevo Producto",
  "proveedorId": 1,
  "categoriaId": 1,
  "cantidadPorUnidad": "10 cajas",
  "precioUnidad": 25.50,
}
```

Salida:
- 201 Created: JSON indicando estado y el producto creado.
  - Ejemplo:

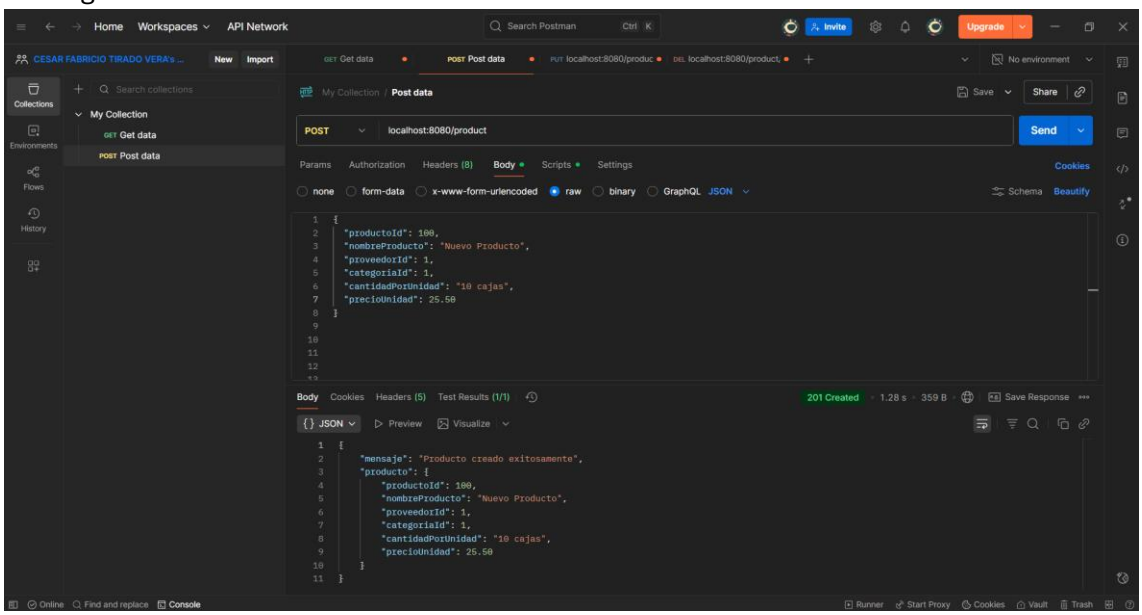
```
{
"mensaje": "Producto creado exitosamente",
  "producto": {
    "productoId": 100,
    "nombreProducto": "Nuevo Producto",
    "proveedorId": 1,
    "categoriaId": 1,
    "cantidadPorUnidad": "10 cajas",
    "precioUnidad": 25.50,
   }
}
```

- 400 Bad Request: en caso de error al crear (por ejemplo violation de constraints).
Ejemplo:

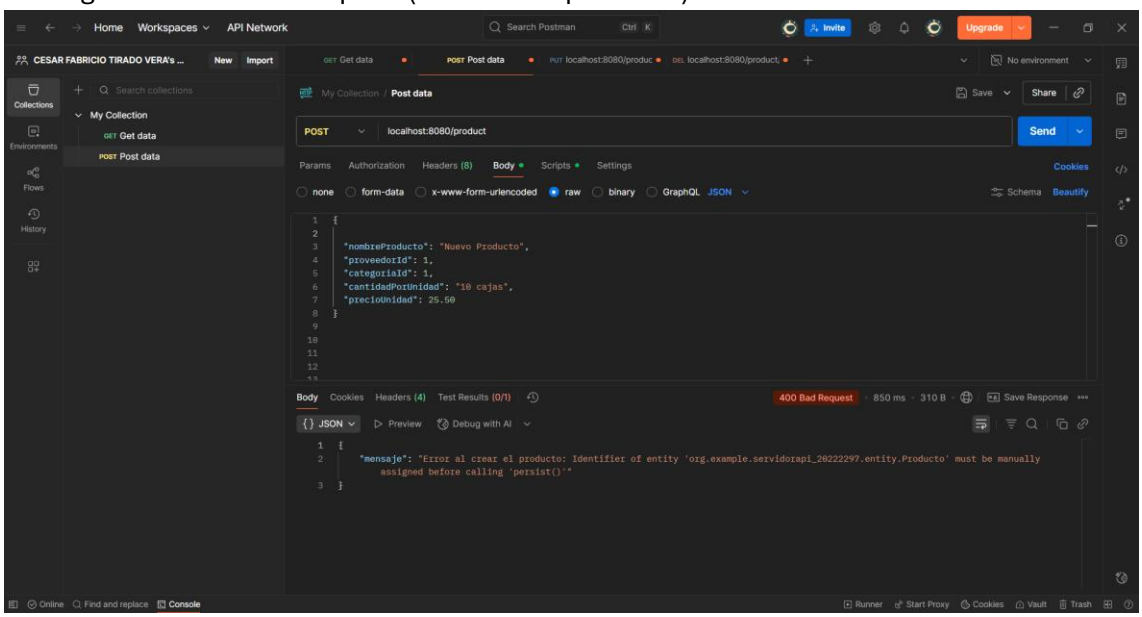
```
{
"mensaje": "Error al crear el producto: [detalle]"
}
```

Imágenes del Postman:

- Código HTTP 201 Created



- Código HTTP 400 Bad Request (Falta ID del producto)



4) Actualizar producto
- Método: PUT
- URL: “/producto”
- Descripción: Actualiza parcialmente un producto. En esta implementación el “productoId” se envía en el body y el servidor actualiza solo los campos que no son “null” en el objeto recibido.

Entrada (body JSON): objeto “Producto” con “productoId” obligatorio y los campos a actualizar (campos no enviados o “null” no se modifican).
- Ejemplo (actualizar precio y stock):

```
{
  "productoId": 1,
  "precioUnidad": 22.50,
  "cantidadPorUnidad": "15 cajas"
}
```


Salida:
- 200 OK: JSON indicando resultado:

```
{
"resultado": "ok"
}
```


- 400 Bad Request: si no se envía “productoId” o si el “productoId” no existe.
  - Ejemplo (sin ID):

```
{
 "resultado": "error", "mensaje": "Debe enviar un producto con ID"
 }
```

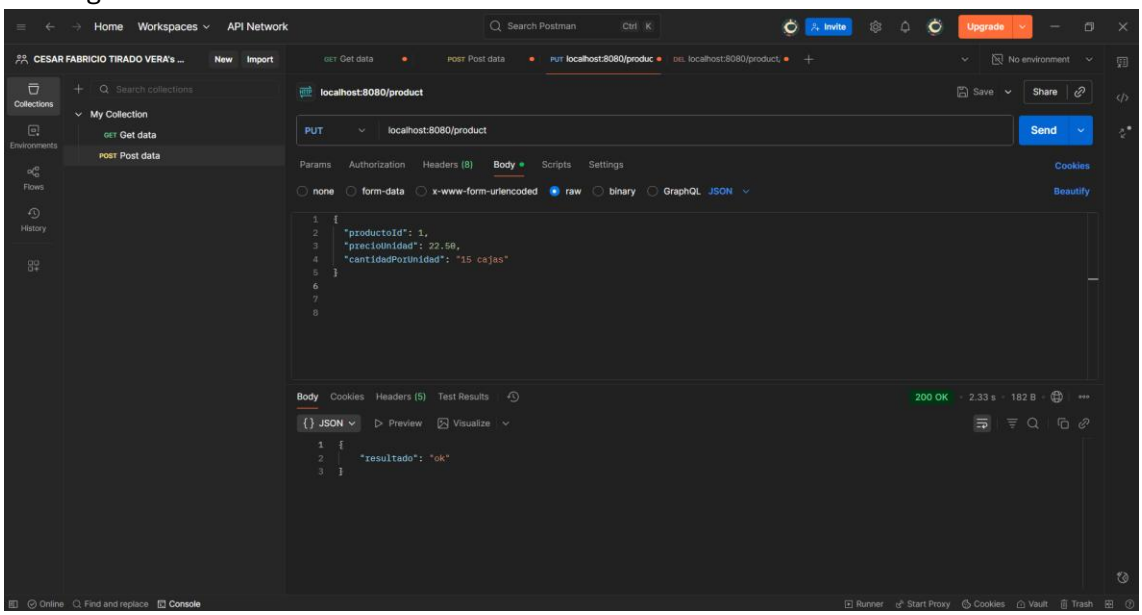

  - Ejemplo (no existe):

```
{
"resultado": "error", "mensaje": "El ID del producto enviado no existe"
}
```


Observaciones:
- Este endpoint realiza una actualización parcial comprobando cada campo con “!= null”.
- Cabe recalcar que “null” significa "no cambiar" en esta API.

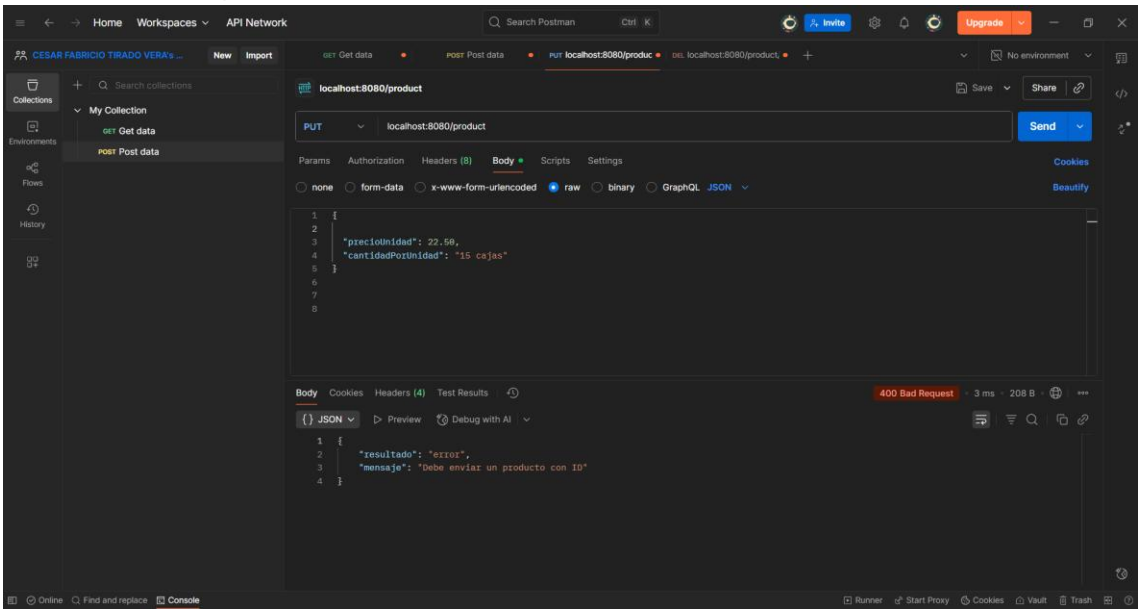
Imágenes del Postman:

- Código HTTP 200 OK

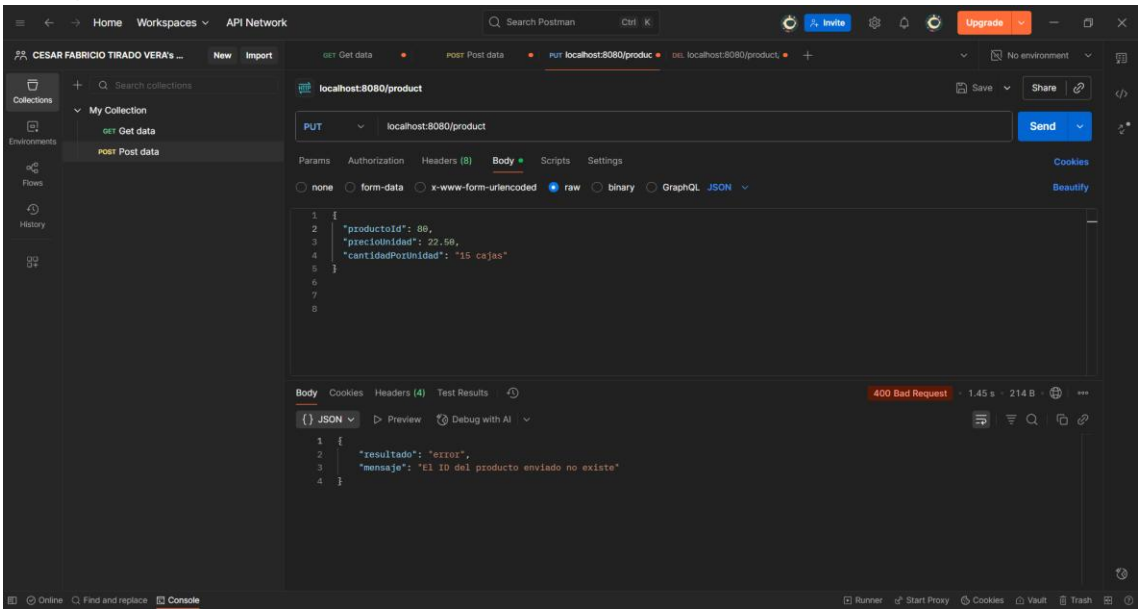


- Código HTTP 400 Bad Request

    - Si no se envía el “productoId”



- Si el “productoId” no existe



5) Eliminar producto
- Método: DELETE
- URL: “/product/{id}”
- Descripción: Elimina el producto por ID.

ADVERTENCIA IMPORTANTE: Este endpoint elimina en cascada:

1. Primero: Todos los registros de “OrderDetails” asociados al producto (pérdida permanente de historial de ventas)

2. Luego**:** El producto mismo

Esta operación es **irreversible** y elimina información histórica de pedidos.

Entrada:
- Parámetro de ruta: “id” (entero)

Salida:
- 200 OK: si la eliminación fue exitosa.
  - Ejemplo:
    - Cuando tiene registros de “OrderDetails”
      {
        "mensaje": "Producto eliminado exitosamente",
        "orderDetailsEliminados": 8
      }
    - Cuando no tiene registros de “OrderDetails”
      {
        "mensaje": "Producto eliminado exitosamente",
      }

El campo “orderDetailsEliminados” indica cuántos registros de pedidos fueron eliminados junto con el producto.

- 400 Bad Request: si el producto no existe.
  - Ejemplo:

{
"mensaje": "Producto con id X no encontrado"
}

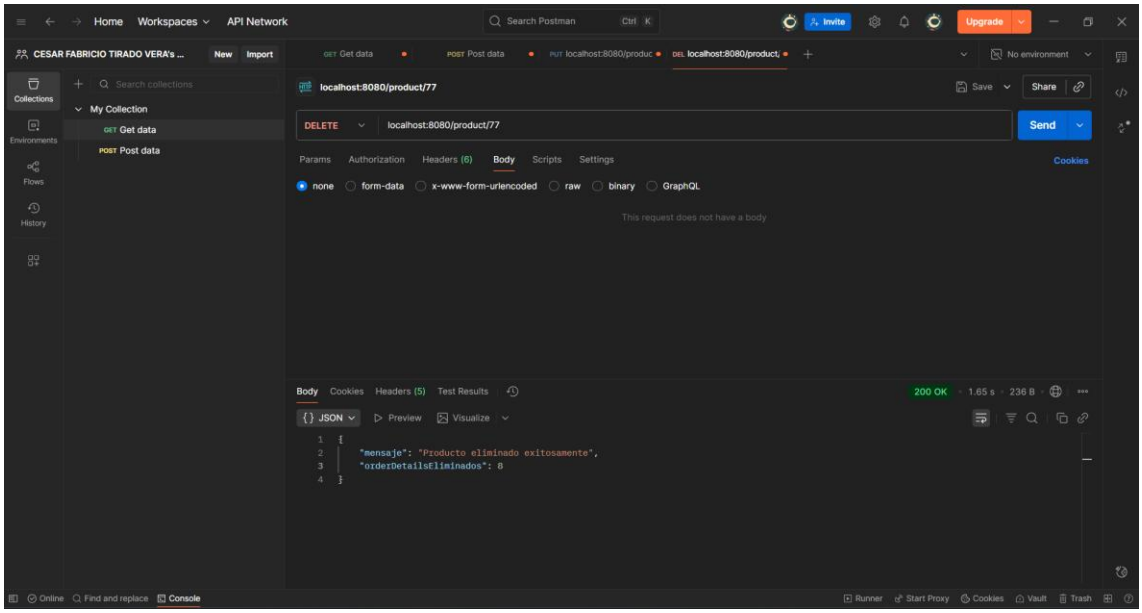
- 500 Internal Server Error: si ocurre un error durante la transacción.
  - Ejemplo:

{
"mensaje": "Error al eliminar el producto: [detalle]"
}

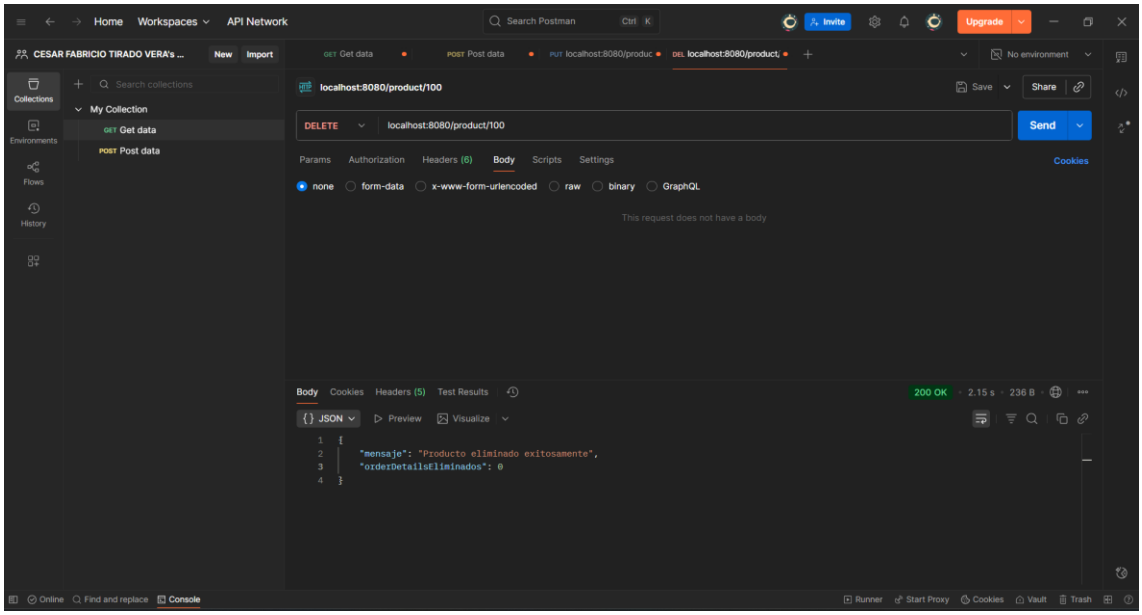
Imágenes del Postman:

- Código HTTP 200 OK

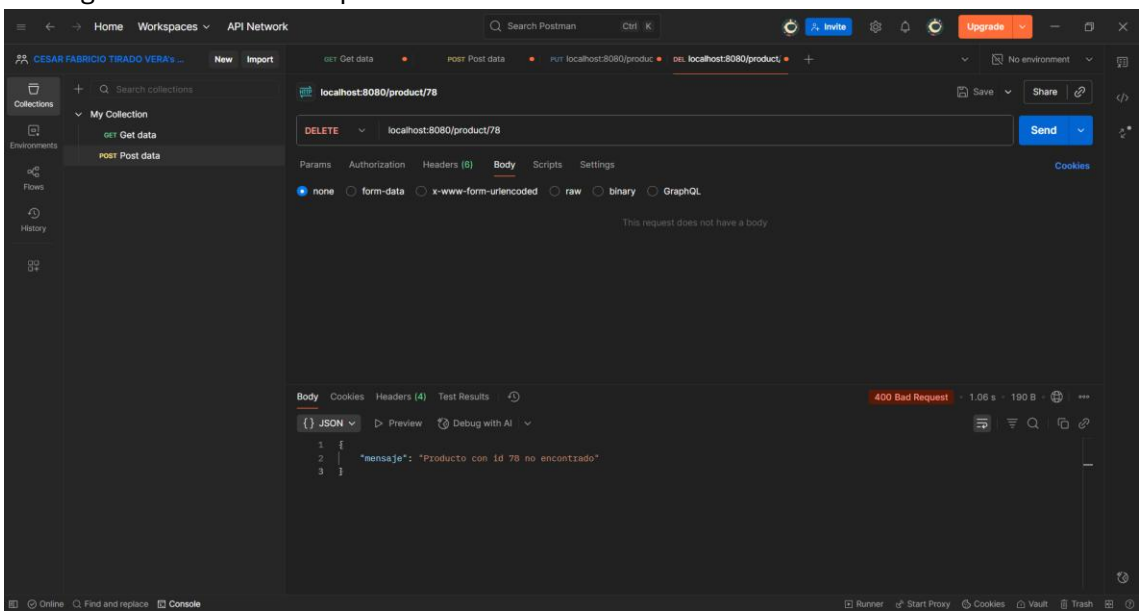
    - Cuando tiene registros de “OrderDetails”



    - Cuando no tiene registros de “OrderDetails”



- Código HTTP 400 Bad Request



Estructura del objeto Producto (campos manejados por la API)
Nota: La tabla “Products” en la base de datos tiene una estructura con solo 6 campos.

- “productoId” (Integer) — clave primaria, corresponde a columna “ProductID”

- “nombreProducto” (String) — corresponde a columna “ProductName”

- “proveedorId” (Integer) — corresponde a columna “SupplierID”

- “categoriaId” (Integer) — corresponde a columna “CategoryID”

- “cantidadPorUnidad” (String) — corresponde a columna “Unit”

- “precioUnidad” (Decimal / BigDecimal) — corresponde a columna “Price”

Errores comunes y mensajes
- “ID de producto inválido” : cuando un ID no numérico es pasado en la ruta.
- “Producto con id X no encontrado” : cuando no existe registro con ese ID.
- “Debe enviar un producto con ID” : cuando el body de actualización no incluye “productoId”.
- “Error al crear/actualizar/eliminar el producto: [detalle]” : error interno o constraint violado.

Consideraciones de eliminación en cascada

El endpoint DELETE implementa eliminación en cascada manual mediante transacción:

1. Elimina primero todos los registros de “OrderDetails” relacionados

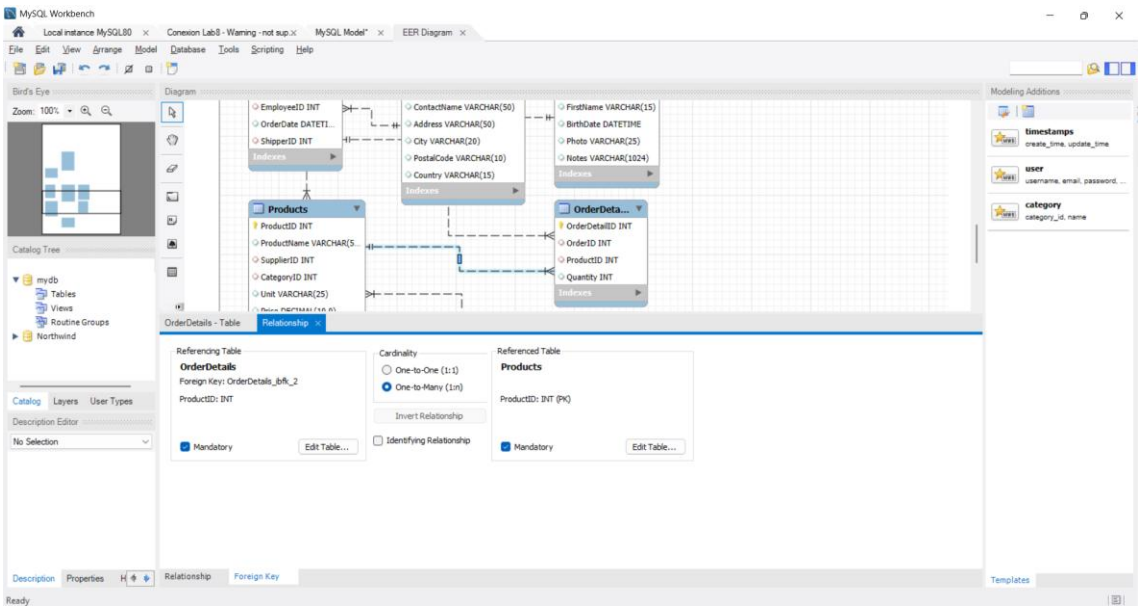
2. Luego elimina el producto

3. Si cualquier paso falla, la transacción completa se revierte

Impacto: Pérdida permanente de historial de ventas. Los registros de pedidos asociados al producto se eliminan sin posibilidad de recuperación.

Por qué se borra explícitamente “OrderDetails”

La razón por la que el controlador elimina primero los registros de la tabla “OrderDetails” es que la base de datos impone una restricción de clave foránea (FK) entre “OrderDetails” y “Products”. MySQL no permite borrar una fila padre (un “Product”) si existen filas hijas (“OrderDetails”) que la referencian; en ese caso la operación DELETE falla con un error de restricción.



En resumen:

- Si intentáramos ejecutar “DELETE FROM products WHERE ProductID = ?” mientras existan “OrderDetails” apuntando a ese producto, la base de datos rechazará la operación por integridad referencial.

- Para poder eliminar el producto desde la API sin cambiar la estructura de la BD , el controlador borra primero las filas hijas y luego la fila padre dentro de la misma transacción.