Universidad Nacional Autónoma de México

Facultad de Ciencias

Fundamentos de Bases de Datos
Práctica 08. JDBC

Equipo : eSosQLones

Estrada Garcia Luis Gerardo	319013832
Jiménez Hernández Allan	420003478
Mancera Quiroz Javier Alejandro	319274831
Mora Hernández Dulce Julieta	319236448
Peña Nuñez Axel Yael	318279754

Actividades

- Deberan realizar un reporte en formato pdf, donde definan cuales fueron las queries que tuvieron que utilizar para la realización de CRUD para cada una de sus tablas y explicar lo que realiza la query, ademas de las sentencias que usa, este documento lo llamaran Practica08.pdf.
- Utilizaran POSTMAN para probar su aplicación REST, para cada una de las consultas que realizaron. Las evidencias las deberan incluir en el reporte.

1. Reporte Completo de la Estructura CRUD para la Entidad Insumo en Spring Boot

- **Modelo: Insumo.java**

  Función:

  Define la estructura de datos de Insumo con propiedades que reflejan los campos de la tabla de Insumo en la base de datos.

  Roles:

  Encapsula la información del insumo con propiedades privadas.
  Ofrece constructores para la creación de instancias con datos predefinidos o sin ellos.
  Provee métodos getters y setters para manipular sus atributos.
  Implementa el método toString para obtener una representación de texto del estado actual del objeto.

- **Interfaz de Repositorio: InsumoRepository.java**

  Función:

  Establece un contrato para las operaciones CRUD en la base de datos para Insumo.

  Roles:

  Define métodos para la obtención (findAll), inserción (insertInsumo), actualización (updateInsumo y executeUpdateInsumo), y eliminación (deleteInsumo) de registros de insumos.

- **Implementación del Repositorio: InsumoRepositoryImp.java**

  Función:

  Implementa InsumoRepository usando NamedParameterJdbcTemplate para ejecutar operaciones de base de datos.

  Roles:

  Implementa lógica para consultar (findAll), insertar (insertInsumo), actualizar (updateInsumo y executeUpdateInsumo), y eliminar (deleteInsumo) insumos.
  Utiliza InsumoRowMapper para mapear filas de resultados SQL a objetos Insumo.

- **Mapper: InsumoRowMapper.java**

  Función:

  Mapea las filas de ResultSet a objetos Insumo.

  Roles:

  Implementa RowMapper para convertir filas SQL en instancias de Insumo con un mapeo de columna a atributo.

- **Interfaz de Servicio: InsumoService.java**

  Función:

  Declara la lógica de negocio abstracta para las operaciones con Insumo.

  Roles:

  Define los métodos que serán usados para operaciones CRUD en el nivel de servicio.

- **Implementación del Servicio: InsumoServiceImp.java**

  Función:

Provee la implementación de la interfaz InsumoService.

Roles:

Encapsula la lógica de negocio y directamente coordina las llamadas al repositorio, ofreciendo una capa adicional entre el controlador y el acceso a datos.

- **Controlador: InsumoController.java**

  Función:

  Controlador de Spring Boot que gestiona las peticiones HTTP relacionadas con Insumo.

  Roles:

  Define los endpoints para operaciones CRUD y delega las tareas al servicio InsumoService.
  Utiliza anotaciones como @GetMapping, @PostMapping, @PutMapping, y @DeleteMapping para mapear las rutas de la API a las operaciones de servicio correspondientes.

- **Conclusión**

  Este conjunto de clases e interfaces refleja una arquitectura típica de una aplicación Spring Boot construida alrededor del patrón CRUD. El flujo de datos sigue un camino claro desde el controlador hasta la capa de servicio, y luego al repositorio, con la respuesta que finalmente regresa al cliente. El uso de abstracciones como interfaces y la implementación de patrones de diseño facilita el mantenimiento y la escalabilidad del código.

2. Reporte Completo de la Estructura CRUD para la Entidad Animal en Spring Boot

- **Modelo: Animal.java**

  Función:

Representa una entidad que corresponde a una tabla en la base de datos con campos como id, nombre, especie, etc.

Roles:

1. Encapsula la información del insumo con propiedades privadas.

2. Ofrece constructores para la creación de instancias con datos predefinidos o sin ellos.

3. Provee métodos getters y setters para manipular sus atributos.

4. Implementa el método toString para obtener una representación de texto del estado actual del objeto.

– **Interfaz de Repositorio: AnimalRepository.java**

Función:

Define las operaciones CRUD abstractas para acceder y manipular los datos de animales en la base de datos.

Roles:

1. findAll() para obtener una lista de todos los animales.

2. insertAnimal(Animal animal) para añadir un nuevo animal.

3. updateAnimal(Animal animal) para modificar un animal existente.

4. deleteAnimal(Animal animal) para eliminar un animal.

– **Implementación del Repositorio: AnimalRepositoryImp.java**

Función:

Implementa Implementa la interfaz AnimalRepository utilizando NamedParameterJdbcTemplate para la ejecución de consultas SQL.

Roles:

1. findAll() realiza una consulta SQL para recuperar todos los animales.

2. insertAnimal(Animal animal) inserta un nuevo animal en la base de datos.

3. updateAnimal(Animal animal) actualiza la información de un animal existente.

4. deleteAnimal(Animal animal) elimina un animal de la base de datos.

– **Mapper: AnimalRowMapper.java**

Función:

Convierte las filas del ResultSet de SQL en objetos Java Animal.

Roles:

mapRow(ResultSet rs, int rowNum) mapea los datos de la base a la entidad Animal.

– **Interfaz de Servicio: AnimalService.java**

Función:

Declara los métodos de servicio que operan sobre la entidad Animal.

Roles:

1. findAll() para recuperar animales.

2. insertAnimal(Animal animal) para la creación de animales.

3. updateAnimal(Animal animal) para actualizar información de animales.

4. deleteAnimal(Animal animal) para borrar animales de la base.

– **Implementación del Servicio: AnimalBiomaServiceImp.java**

Función:

Implementa la interfaz AnimalService, proporcionando la lógica de negocio y conectividad entre el controlador REST y la capa de acceso a datos.

Roles:

1. findAll() utiliza animalRepositorio para obtener todos los animales.

2. insertAnimal(Animal animal) pasa el objeto Animal a animalRepositorio para su inserción.

3. updateAnimal(Animal animal) envía el objeto Animal a animalRepositorio para su actualización.

4. deleteAnimal(Animal animal) solicita a animalRepositorio que elimine un animal específico.

– **Controlador: AnimalController.java**

Función:

Maneja las solicitudes HTTP entrantes relacionadas con los animales y delega las operaciones CRUD a AnimalBiomaServiceImp.

Roles:

1. @GetMapping para obtener la lista de animales.

2. @PostMapping para crear un nuevo animal.

3. @PutMapping para actualizar un animal existente.

4. @DeleteMapping para eliminar un animal por su ID.

– **Conclusión**

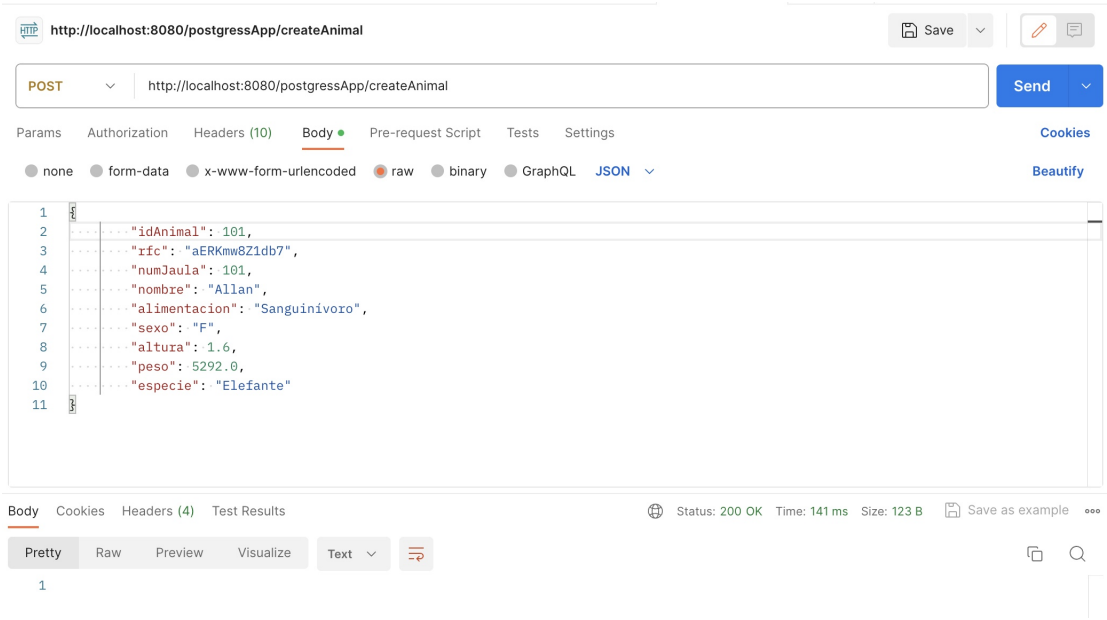
El conjunto de clases proporcionadas demuestra una arquitectura típica para una aplicación CRUD en Spring Boot. El modelo Animal define la entidad principal, mientras que el repositorio y su implementación manejan las operaciones de base de datos. El servicio encapsula la lógica del negocio, y el controlador expone la funcionalidad a través de una API REST. Cada capa tiene responsabilidades claramente definidas, facilitando el mantenimiento y la expansión del código.

Queries:

1. **Animal:**

   – POST:

   Crea un nuevo animal llamado "Allan" y se inserta a la tabla Animal con el id = 101, utiliza la sentencia POST para crearlo e insertarlo a la tabla.

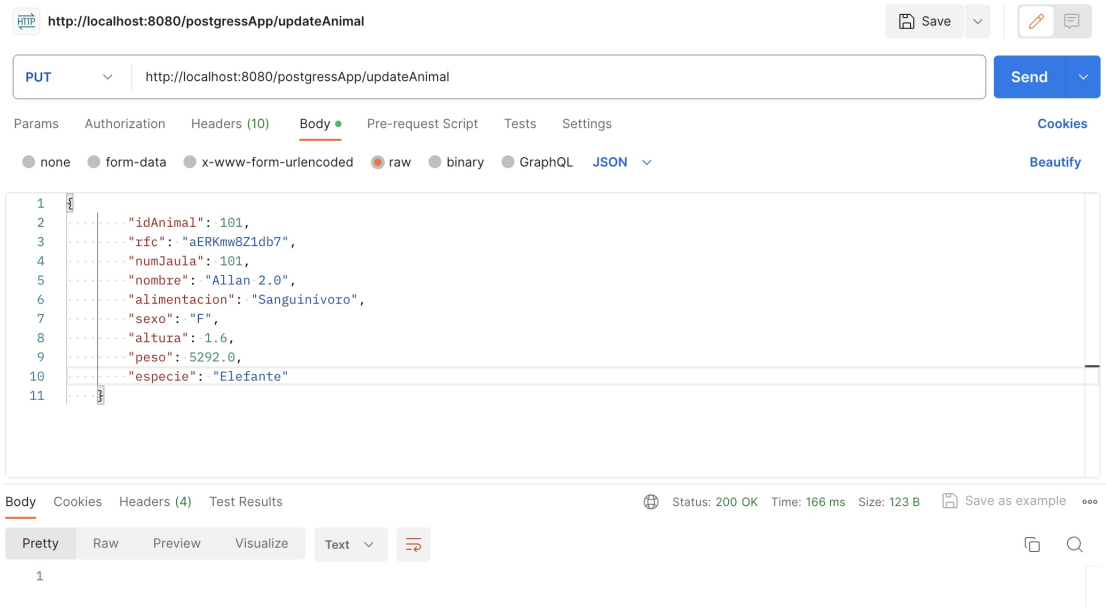


Evidencia:

98	98	v8c6Rp261FzH	98	Janice	Herbívoro	F
99	99	gG28QtJ20m9Y	99	Elliot	Folívoro	M
100	100	XxlA2P2u8pkx	100	Giovanni	Insectívoro	M
101	101	aERKmw8Z1db7	101	Allan	Sanguinívoro	F

– PUT:

Se actualiza el nombre del animal creado antes a "Allan 2.0" y se actualiza en la tabla Animal, utiliza la sentencia PUT para actualizarlo en la tabla, con la información a actualizar.

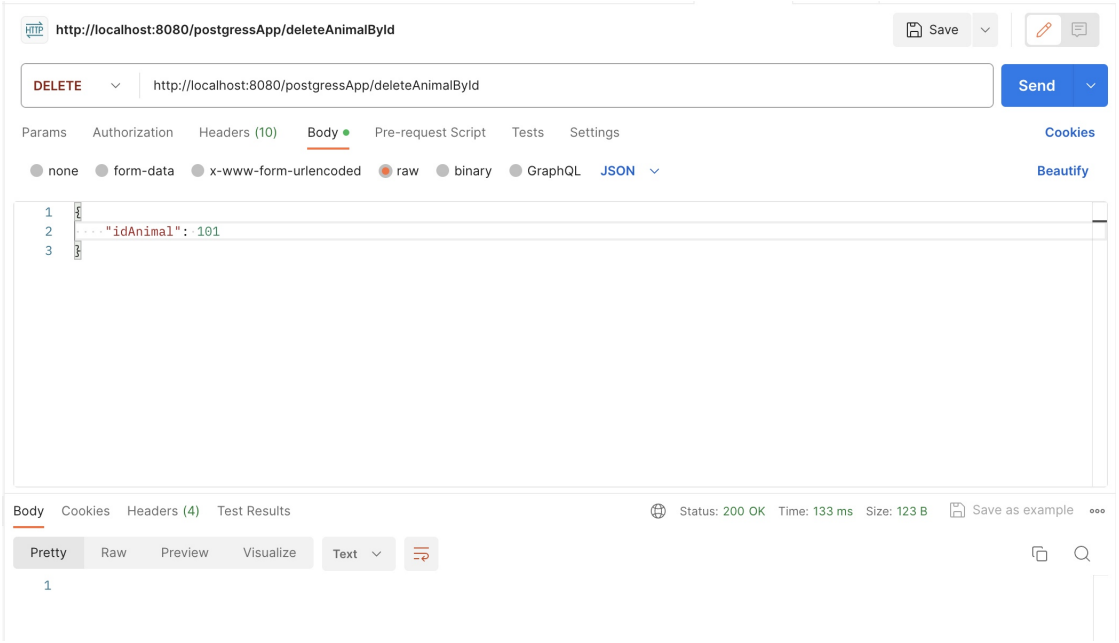


Evidencia:

99	gG28QtJ20m9Y	99	Elliot	Folívoro	M
100	XxIA2P2u8pkx	100	Giovanni	Insectívoro	M
101	aERKmw8Z1db7	101	Allan 2.0	Sanguinívoro	F

– DELETE:

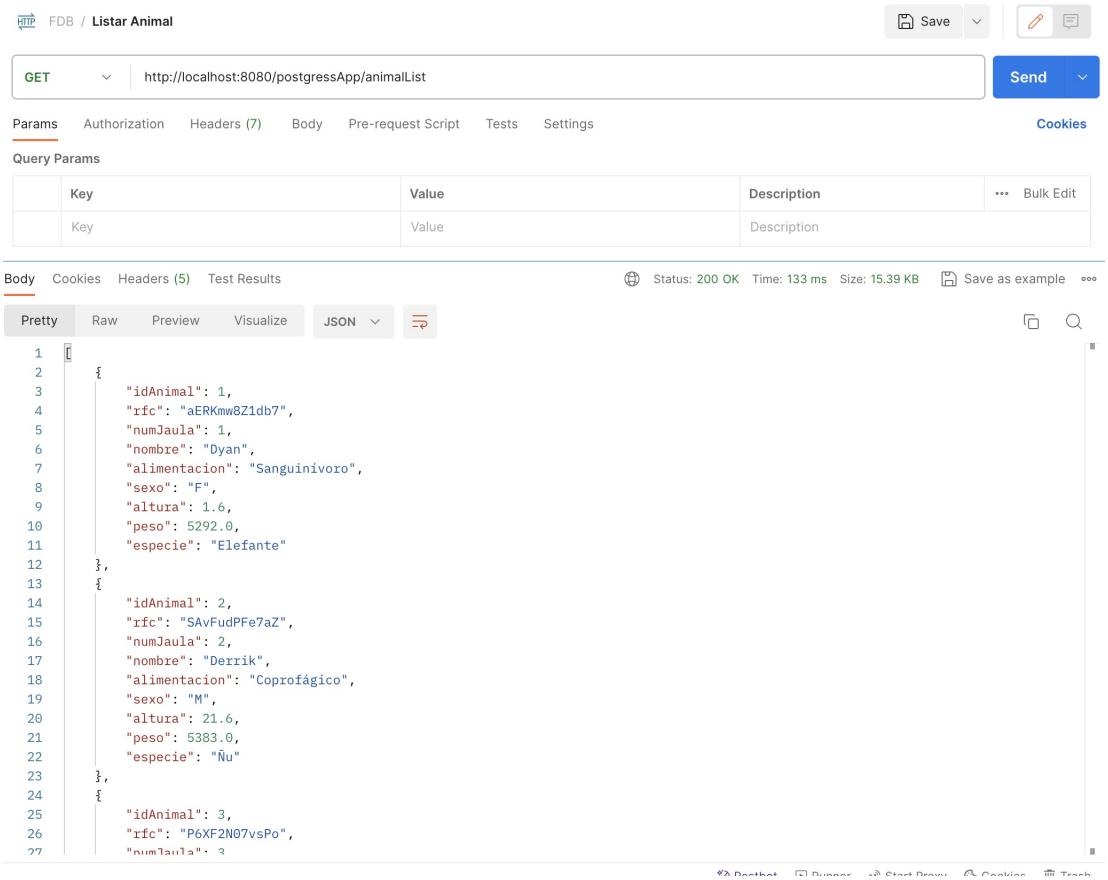
Se elimina el animal creado con POST, al indicar únicamente el id = 101 del animal a eliminar, se usa la sentencia DELETE para eliminar una tupla de la tabla.



Evidencia:

97	97	ww4nK7ny9YwU	97	Linus	Frugívoro	M
98	98	v8c6Rp261FzH	98	Janice	Herbívoro	F
99	99	gG28QtJ20m9Y	99	Elliot	Folívoro	M
100	100	XxIA2P2u8pkx	100	Giovanni	Insectívoro	M

– GET:
Devuelve la información de toda la tabla, se usa la sentencia GET para esto.



2. **Insumo:**

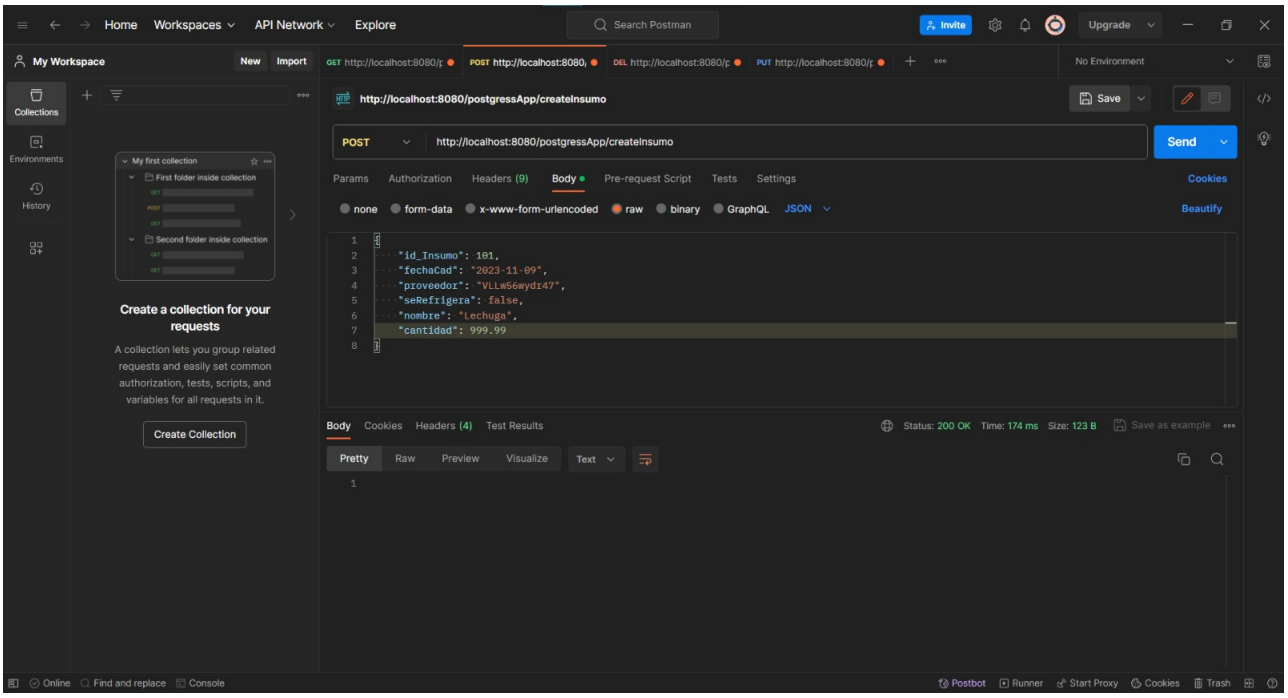
   – POST:

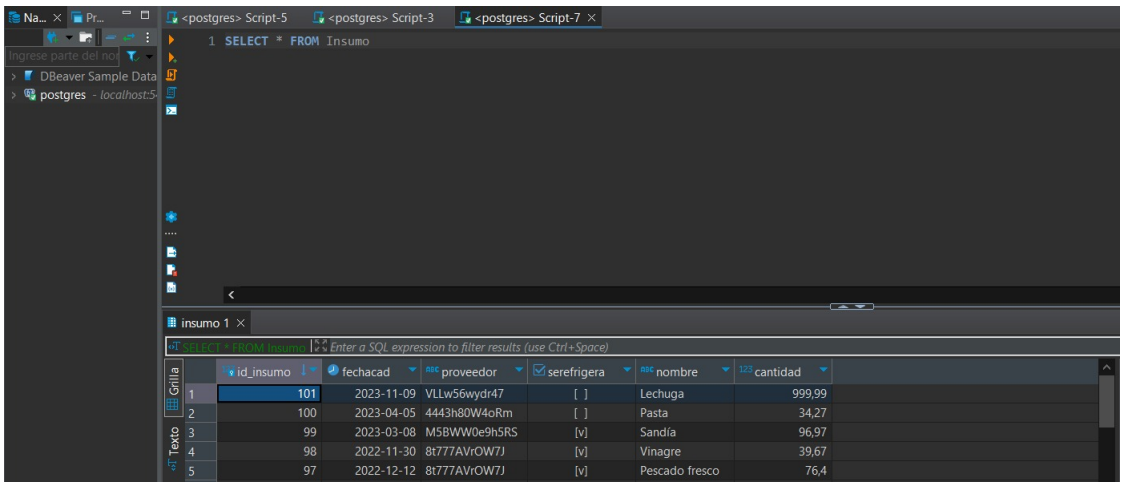
   Crea un nuevo insumo llamado ”Lechuga” y se inserta a la tabla Insumo con el id = 101, utiliza la sentencia POST para crearlo e insertarlo a la tabla.
   Evidencia:

   – PUT:

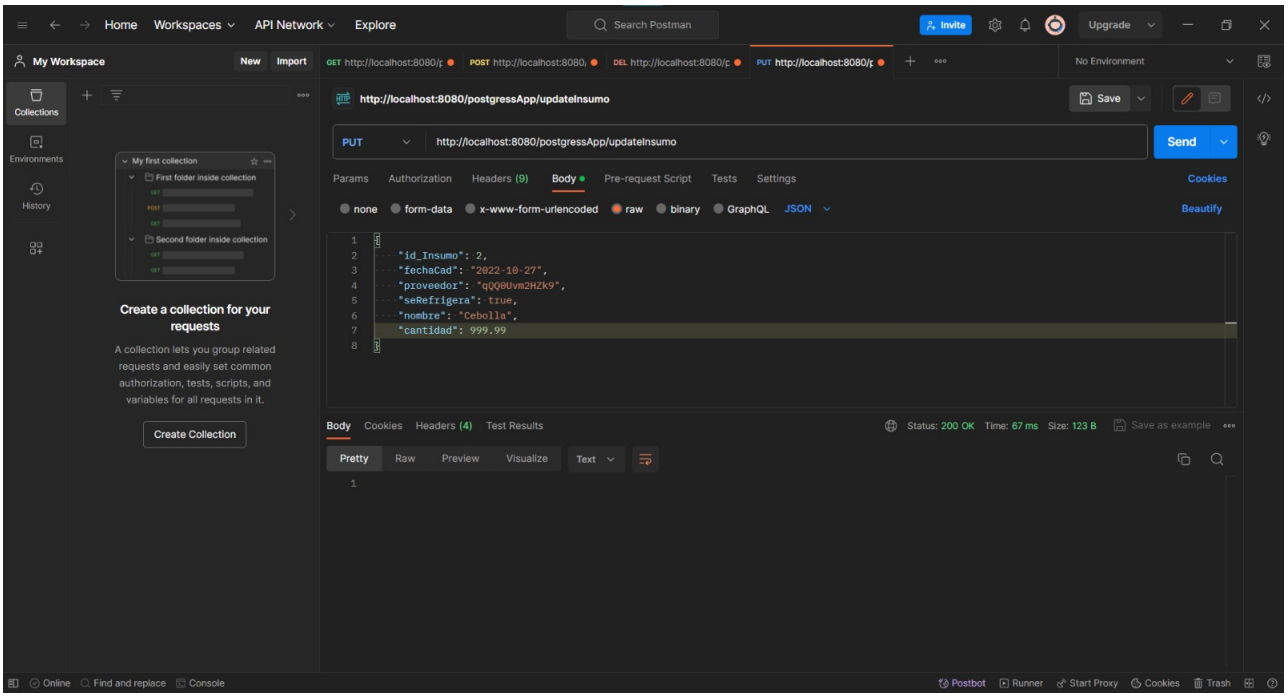
   Se actualiza la cantidad del insumo con id = 2 a 999.99 y se actualiza en la tabla Insumo, utiliza la sentencia PUT para actualizarlo en la tabla, con la información a actualizar.
   Evidencia:

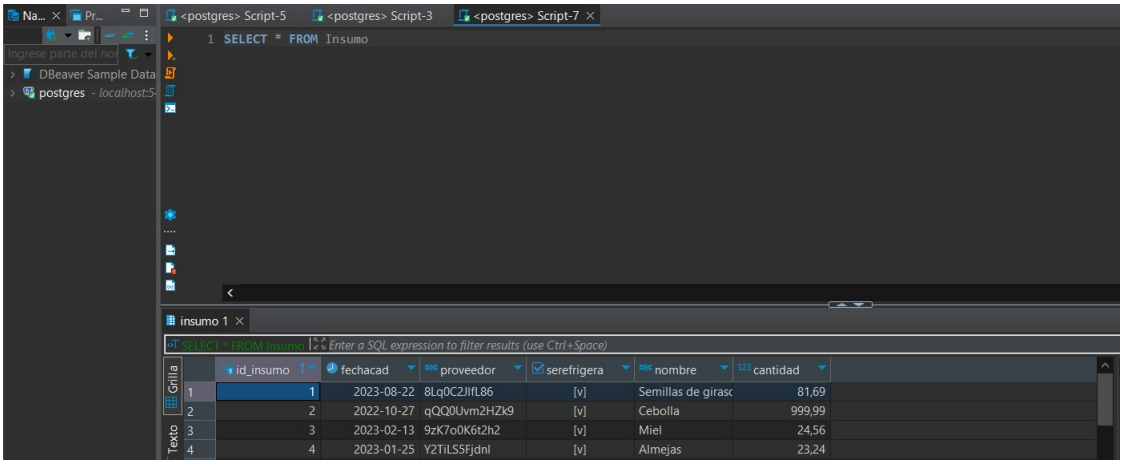
   – DELETE:



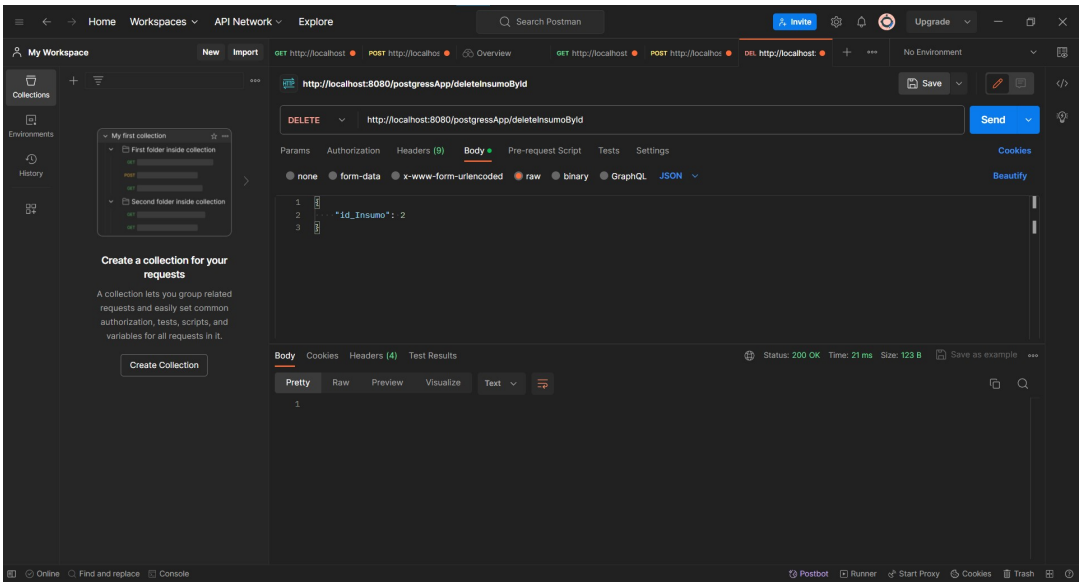


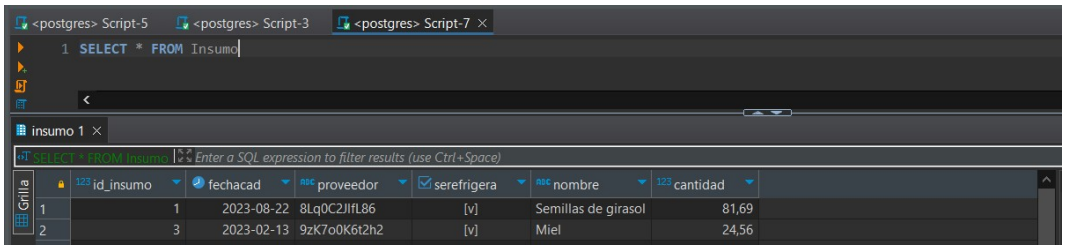
Se elimina el insumo actualizado antes con el id = 2, al indicar únicamente el id = 2 del insumo a eliminar, se usa la sentencia DELETE para eliminar una tupla de la tabla.
Evidencia:

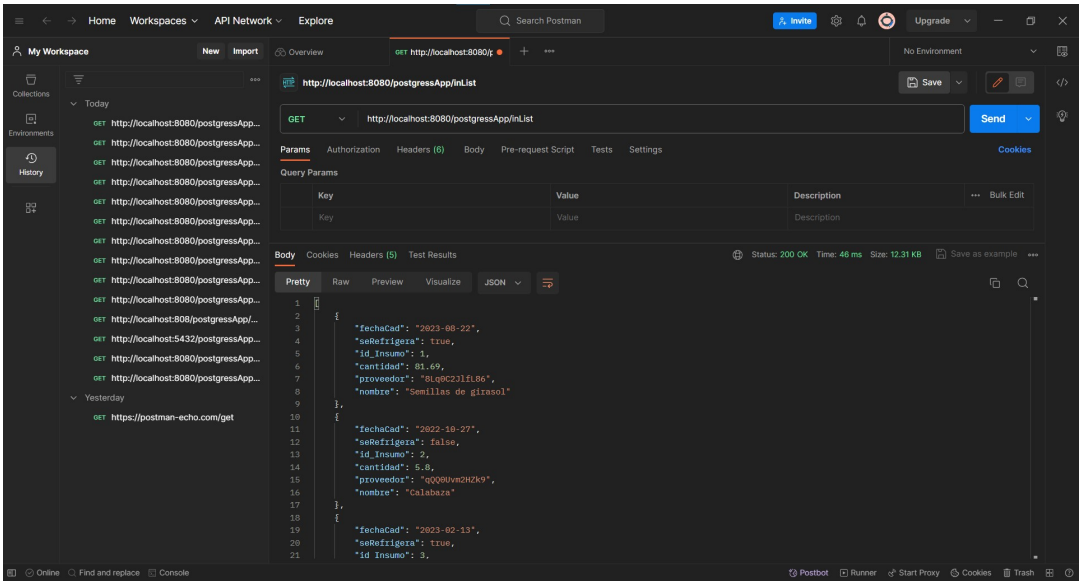




– GET:
Devuelve la información de toda la tabla, se usa la sentencia GET para esto.


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
No Environment
Collections
Environments
History
http://localhost:8080/postgressApp/deleteInsumoById
Save
DELETE
http://localhost:8080/postgressApp/deleteInsumoById
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
"id_Insumo": 2
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (4)
Test Results
Status: 200 OK
Time: 21 ms
Size: 123 B
Pretty
Raw
Preview
Visualize
Text
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash


<postgres> Script-5
<postgres> Script-3
<postgres> Script-7
SELECT * FROM Insumo
insumo 1
SELECT * FROM Insumo
Enter a SQL expression to filter results (use Ctrl+Space)
Grilla
id_insumo
fechacad
proveedor
serefrigera
nombre
cantidad
1
2023-08-22
8Lq0C2JlfL86
[v]
Semillas de girasol
81,69
3
2023-02-13
9zK7o0K6t2h2
[v]
Miel
24,56


Home
Workspaces
API Network
Explore
Search Postman
Invite
Upgrade
My Workspace
New
Import
No Environment
Collections
Environments
History
Today
Yesterday
http://localhost:8080/postgressApp/inList
Save
GET
http://localhost:8080/postgressApp/inList
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 46 ms
Size: 12.31 KB
Pretty
Raw
Preview
Visualize
JSON
"fechaCad": "2023-08-22",
"seRefrigera": true,
"id_Insumo": 1,
"cantidad": 81.69,
"proveedor": "8Lq0C2JlfL86",
"nombre": "Semillas de girasol"
"fechaCad": "2022-10-27",
"seRefrigera": false,
"id_Insumo": 2,
"cantidad": 5.8,
"proveedor": "qQO0Uvm2HZk9",
"nombre": "Calabaza"
"fechaCad": "2023-02-13",
"seRefrigera": true,
"id_Insumo": 3,
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash