202125304 -David Ortiz
202215891 - José Salgado
202215816 - Angelica Yeraldin Rodríguez

ENTREGA PROYECTO 1

SISTEMAS TRANSACCIONALES

2024-10

UNIVERSIDAD DE LOS ANDES

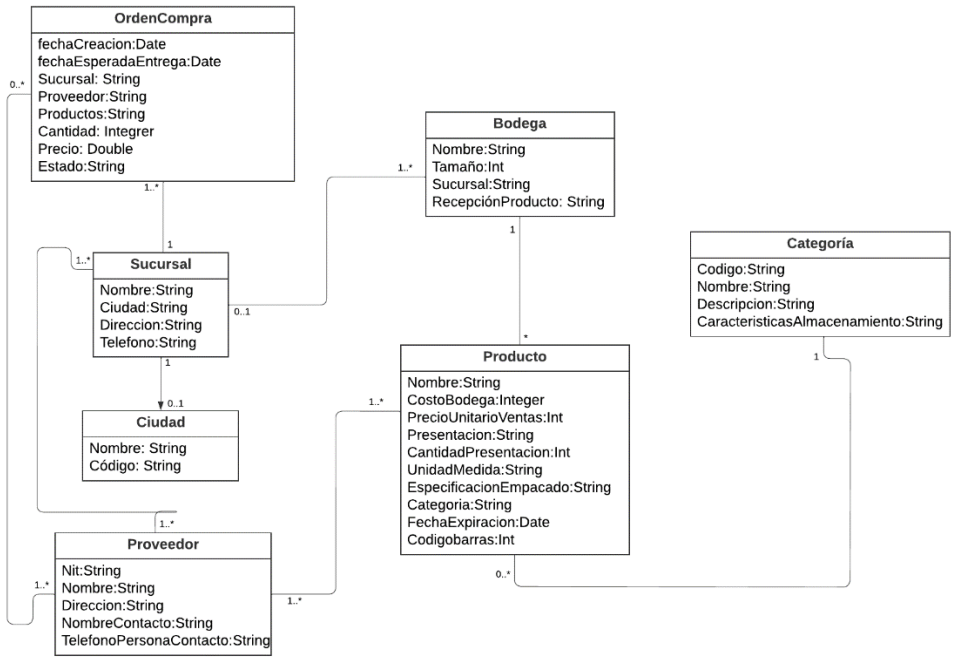
202125304 -David Ortiz
202215891 - José Salgado
202215816 - Angelica Yeraldin Rodríguez

Tabla de contenido

202125304 -David Ortiz
202215891 - José Salgado
202215816 - Angelica Yeraldin Rodríguez

INFORME- IMPLEMENTACIÓN SISTEMAS TRANSACCIONALES

Análisis y revisión del modelo de datos: modelos UML y relacional y su descripción, indicando las modificaciones hechas.



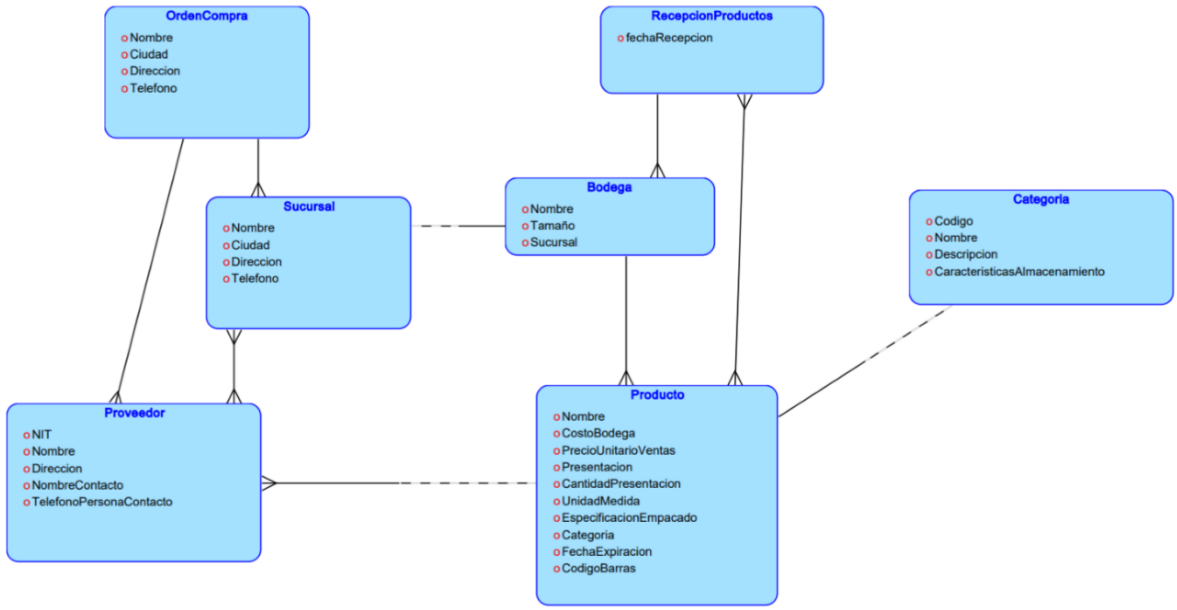
Modificaciones realizadas:

UML:

Las clases actualizadas en el modelo son Ciudad y OrdenDeCompra. Agregamos la clase Ciudad, como fue sugerido, y agregamos los atributos faltantes en OrdenDeCompra para su correcto funcionamiento. En cuanto a las relaciones, hicimos los siguientes cambios:

- Una **bodega** tiene de 0 a 1 **sucursal**, lo que significa que una bodega puede estar o no asociada a una sucursal.
- Un **proveedor** debe tener de 1 a muchos **productos**, asegurando que un proveedor siempre esté vinculado a al menos un producto.
- Un **producto** tiene 1 **categoría**, y una **categoría** tiene de 0 a muchos **productos**, lo que significa que cada producto pertenece a una categoría, pero una categoría puede tener múltiples productos.
- Un **proveedor** tiene de 0 a muchas **órdenes de compra**, lo que indica que un proveedor puede tener varias órdenes de compra o ninguna.
- Una **bodega** tiene varios **productos**, y un **producto** tiene una única **bodega**, lo que asegura que cada producto esté almacenado en una sola bodega, pero una bodega puede almacenar varios productos.

- Una **orden de compra** tiene una **sucursal**, y una **sucursal** tiene varias **órdenes de compra**, lo que significa que cada orden de compra está asociada a una sucursal, y una sucursal puede gestionar múltiples órdenes de compra.



RELACIONAL (Adjunto en el repositorio)

- La relación entre bodega y sucursal se arregló siendo así que Sucursal debe ser la que tenga la FK, NN de bodega.
- Se elimino toda tabla adicional que no fuera de muchos a muchos.
- La FK de id_producto en la recepción de productos se agregó bajo el comentario anteriormente dado.
- En producto se le agrego una FK de id_bodega para modelar la relación.
- Se cambio el NN de fecha de expiración por DATE.
- Se agrego la justificación sobre UML a relacional.
- Se modifico la normalización para especificar las unicas tablas que se tienen actualmente.

Comprobación de la forma normal de cada relación.

1. Sucursal

1FN (Primera Forma Normal)

- Todos los atributos contienen valores atómicos (indivisibles). Cada fila es única gracias a la clave primaria id_sucursal.
  Cumple 1FN.
  2FN (Segunda Forma Normal)
- La clave primaria es id_sucursal, y todos los demás atributos (nombre, ciudad, dirección, teléfono, id_bodega) dependen completamente de esta clave.
  Cumple 2FN.
  3FN (Tercera Forma Normal)
- No hay dependencias transitivas, ya que todos los atributos dependen directamente de la clave primaria id_sucursal. El id_bodega es una clave foránea que no crea dependencias transitivas.
  Cumple 3FN.

2. Bodega

1FN

- Todos los atributos tienen valores indivisibles, y cada fila es única por la clave primaria id_bodega.
  Cumple 1FN.
  2FN
- La clave primaria es id_bodega, y los atributos nombre y tamaño_m2 dependen completamente de esta clave.
  Cumple 2FN.
  3FN
- No existen dependencias transitivas, ya que todos los atributos dependen solo de id_bodega.
  Cumple 3FN.

3. Producto

1FN

- Todos los campos contienen valores atómicos, como nombre, costo_bodega, precio_unitario, etc. La clave primaria es id_producto, lo que asegura la unicidad de cada fila.
  Cumple 1FN.
  2FN

- La clave primaria es id_producto, y todos los atributos dependen completamente de esta clave. Las FKs id_bodega e id_categoria solo indican relaciones y no afectan la dependencia de los atributos de la clave principal.
  Cumple 2FN.
  3FN
- No existen dependencias transitivas entre los atributos. Aunque id_bodega y id_categoria son claves foráneas, estas no crean dependencias no deseadas entre los atributos.
  Cumple 3FN.

4. Categoría

1FN

- Todos los atributos (nombre, descripción, características_almacenamiento) son atómicos y cada fila es única gracias a la clave primaria id_categoria.
  Cumple 1FN.
  2FN
- Todos los atributos dependen completamente de id_categoria, que es la clave primaria.
  Cumple 2FN.
  3FN
- No hay dependencias transitivas. Cada atributo depende solo de id_categoria.
  Cumple 3FN.

5. Proveedor

1FN

- Todos los atributos (NIT, nombre, dirección, nombre_contacto, teléfono_contacto) son atómicos, y la clave primaria id_proveedor asegura la unicidad de las filas.
  Cumple 1FN.
  2FN
- Cada atributo depende completamente de la clave primaria id_proveedor.
  Cumple 2FN.
  3FN
- No hay dependencias transitivas en esta relación.
  Cumple 3FN.

6. Orden de Compra

1FN

- Todos los atributos (fecha_emisión, estado, id_sucursal, id_proveedor) son indivisibles, y la clave primaria id_orden_compra asegura que cada fila es única.
  Cumple 1FN.
  2FN
- Cada atributo depende completamente de la clave primaria id_orden_compra. Las claves foráneas id_sucursal e id_proveedor solo establecen relaciones, no afectan la dependencia de los atributos.
  Cumple 2FN.
  3FN
- No hay dependencias transitivas, ya que todos los atributos dependen directamente de la clave primaria.
  Cumple 3FN.

7. Recepción de Productos

1FN

- Todos los atributos son atómicos, y la clave primaria id_recepcion asegura que cada fila es única.
  Cumple 1FN.
  2FN
- Todos los atributos dependen completamente de la clave primaria id_recepcion. La FK id_bodega solo establece una relación y no afecta la dependencia.
  Cumple 2FN.
  3FN
- No hay dependencias transitivas en esta relación.
  Cumple 3FN.

8. Ciudad

1FN

- Todos los atributos son indivisibles, y la clave primaria id_ciudad garantiza la unicidad de las filas.

Cumple 1FN.
2FN

- Todos los atributos dependen completamente de id_ciudad.
  Cumple 2FN.
  3FN
- No hay dependencias transitivas en esta relación.
  Cumple 3FN.

9. Recepción de Productos - Producto

1FN

- Esta tabla tiene una clave compuesta por id_recepcion y id_producto.
  Ambos campos son atómicos y no hay valores repetidos o no atómicos.
  Cumple 1FN.
  2FN
- Los atributos dependen completamente de la clave compuesta (id_recepcion e id_producto).
  Cumple 2FN.
  3FN
- No hay dependencias transitivas, ya que todos los atributos dependen de la clave compuesta.
  Cumple 3FN.

Diseño de la base de datos: archivos .sql con los scripts utilizados para la creación del a BD.

También, se adjunta junto con este informe en la carpeta docs, los scripts utilizados para la creación de la base de datos.

```
CREATE TABLE CIUDAD (
  ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  NOMBRE VARCHAR2(100) NOT NULL,
  CODIGO VARCHAR2(10) NOT NULL
);

CREATE TABLE BODEGA (
  ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  NOMBRE VARCHAR2(100) NOT NULL,
  TAMANO NUMBER NOT NULL
);
```

202125304 -David Ortiz
202215891 - José Salgado
202215816 - Angelica Yeraldin Rodríguez


```
CREATE TABLE CATEGORIA_PRODUCTO (
  ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  NOMBRE VARCHAR2(100) NOT NULL,
  DESCRIPCION VARCHAR2(255),
  CARACTERISTICAS_ALMACENAMIENTO VARCHAR2(255)
);


CREATE TABLE PRODUCTO (
  ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  NOMBRE VARCHAR2(100) NOT NULL,
  COSTO_BODEGA NUMBER NOT NULL,
  PRECIO_UNITARIO NUMBER NOT NULL,
  PRESENTACION VARCHAR2(50) NOT NULL,
  CANTIDAD_PRESENTACION NUMBER NOT NULL,
  UNIDAD_MEDIDA VARCHAR2(50) NOT NULL,
  VOLUMEN_EMPAQUE NUMBER,
  PESO_EMPAQUE NUMBER,
  FECHA_EXPIRACION DATE,
  CODIGO_BARRAS VARCHAR2(50) UNIQUE,
  CATEGORIA_ID NUMBER,
  CONSTRAINT FK_CATEGORIA FOREIGN KEY (CATEGORIA_ID) REFERENCES
CATEGORIA_PRODUCTO(ID)
);

CREATE TABLE PROVEEDOR (
  ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  NIT VARCHAR2(50) NOT NULL UNIQUE,
  NOMBRE VARCHAR2(100) NOT NULL,
  DIRECCION VARCHAR2(255),
  NOMBRE_CONTACTO VARCHAR2(100),
  TELEFONO_CONTACTO VARCHAR2(20)
);

CREATE TABLE SUCURSAL (
  ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  NOMBRE VARCHAR2(100) NOT NULL,
  CIUDAD_ID NUMBER,
  DIRECCION VARCHAR2(255),
  TELEFONO VARCHAR2(20),
  CONSTRAINT FK_CIUDAD FOREIGN KEY (CIUDAD_ID) REFERENCES CIUDAD(ID)
```

```
);

CREATE TABLE ORDEN_DE_COMPRA (
  ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
  FECHA_CREACION DATE NOT NULL,
  FECHA_ESPERADA_ENTREGA DATE NOT NULL,
  ESTADO VARCHAR2(50) NOT NULL,
  SUCURSAL_ID NUMBER,
  PROVEEDOR_NIT VARCHAR2(50),
  PRODUCTO_ID NUMBER,
  PRECIO_PRODUCTO NUMBER NOT NULL,
  CANTIDAD_PRODUCTO NUMBER NOT NULL,
  CONSTRAINT FK_SUCURSAL FOREIGN KEY (SUCURSAL_ID) REFERENCES SUCURSAL(ID),
  CONSTRAINT FK_PROVEEDOR FOREIGN KEY (PROVEEDOR_NIT) REFERENCES PROVEEDOR(NIT),
  CONSTRAINT FK_PRODUCTO FOREIGN KEY (PRODUCTO_ID) REFERENCES PRODUCTO(ID)
);
```

Documentación de las clases de Java Spring y de la arquitectura de la aplicación

Arquitectura MVC en Nuestra Aplicación

Hemos adoptado el patrón arquitectónico MVC (Modelo-Vista-Controlador) para estructurar nuestra aplicación. Este enfoque nos permite mantener una clara separación de responsabilidades entre las diferentes capas, facilitando el desarrollo, mantenimiento y escalabilidad del proyecto.

Capa de Modelo:

Esta capa encapsula la representación de los datos de nuestra aplicación. Cada entidad (usuario, producto, etc.) se modela como una clase Java, anotando sus propiedades con JPA para mapearlas directamente a las tablas de la base de datos. Por ejemplo, las anotaciones @Entity, @Table, @ManyToOne, y @OneToMany definen las relaciones entre estas entidades.

Capa de Repositorio:

Para interactuar con la base de datos, utilizamos Spring Data JPA. Esta capa define interfaces que extienden de JpaRepository, proporcionándonos métodos predefinidos para realizar operaciones CRUD (crear, leer, actualizar y eliminar) de forma sencilla.

Además, podemos personalizar estas consultas utilizando para realizar operaciones más complejas.

Capa de Servicio:

Aquí es donde reside la lógica de negocio de nuestra aplicación. Los servicios encapsulan las reglas de negocio y las operaciones que modifican el estado de la aplicación. Por ejemplo, un servicio de "Usuarios" podría encargarse de crear nuevos usuarios, actualizar sus datos, o eliminarlos. Estos servicios interactúan con la capa de repositorio para persistir los cambios en la base de datos.

Capa de Controlador:

La capa de controlador es el punto de entrada de las solicitudes. Los controladores, anotados con @RestController, reciben las peticiones y las delegan a los servicios correspondientes para procesarlas. Una vez que el servicio ha realizado las operaciones necesarias, el controlador devuelve una respuesta al cliente, típicamente en formato JSON.

En la aplicación contamos con entities, controllers, repositories y services de cada una de las clases, no se realizó Front End de la aplicación para facilitar su desarrollo, se usó la herramienta Postman para probar en repetidas ocasiones las tablas de la base de datos, a continuación, haremos una descripción detallada de cada una de las clases y sus atributos:

Clase Ciudad:

- **Atributos**:
  - id: Identificador único de la ciudad, generado automáticamente.
  - nombre: Nombre de la ciudad, no puede ser nulo.
  - codigo: Código que representa la ciudad, no puede ser nulo.

Clase Bodega

Atributos:

- id: Identificador único de la bodega, generado automáticamente.
- nombre: Nombre de la bodega, no puede ser nulo.
- tamano: Tamaño de la bodega en metros cuadrados, no puede ser nulo.

Clase CategoriaProducto

Atributos:

- id: Identificador único de la categoría del producto.
- nombre: Nombre de la categoría, no puede ser nulo.
- descripcion: Breve descripción de la categoría.
- caracteristicasAlmacenamiento: Características especiales de almacenamiento para productos de esta categoría.

Clase OrdenDeCompra

Atributos:

- id: Identificador único de la orden de compra.
- fechaCreacion: Fecha de creación de la orden.
- fechaEsperadaEntrega: Fecha en la que se espera la entrega de la orden.
- estado: Estado actual de la orden (e.g., en proceso, completado).
- sucursal: Relación con la sucursal donde se genera la orden.
- proveedor: Relación con el proveedor responsable de la orden.
- producto: Relación con el producto que se está adquiriendo.
- precioProducto: Precio unitario del producto en la orden.
- cantidadProducto: Cantidad del producto que se compra.

Clase Producto

Atributos:

- id: Identificador único del producto.
- nombre: Nombre del producto.
- costoBodega: Costo del producto en la bodega.
- precioUnitario: Precio unitario del producto.
- presentacion: Tipo de presentación del producto (e.g., botella, caja).
- cantidadPresentacion: Cantidad por presentación (e.g., 12 unidades por caja).
- unidadMedida: Unidad de medida del producto.
- volumenEmpaque: Volumen del empaque en litros o metros cúbicos.
- pesoEmpaque: Peso del empaque en kilogramos.
- fechaExpiracion: Fecha de expiración del producto, si aplica.
- codigoBarras: Código de barras único.
- categoria: Relación con la categoría del producto.

Clase Proveedor

Atributos:

- id: Identificador único del proveedor.
- nit: Número de identificación tributaria del proveedor, único.
- nombre: Nombre del proveedor.
- direccion: Dirección del proveedor.
- nombreContacto: Nombre del contacto en el proveedor.
- telefonoContacto: Teléfono del contacto en el proveedor.

Clase Sucursal

Atributos:

- id: Identificador único de la sucursal.
- nombre: Nombre de la sucursal.
- ciudad: Relación con la ciudad donde está ubicada la sucursal.
- direccion: Dirección física de la sucursal.
- telefono: Teléfono de contacto de la sucursal.

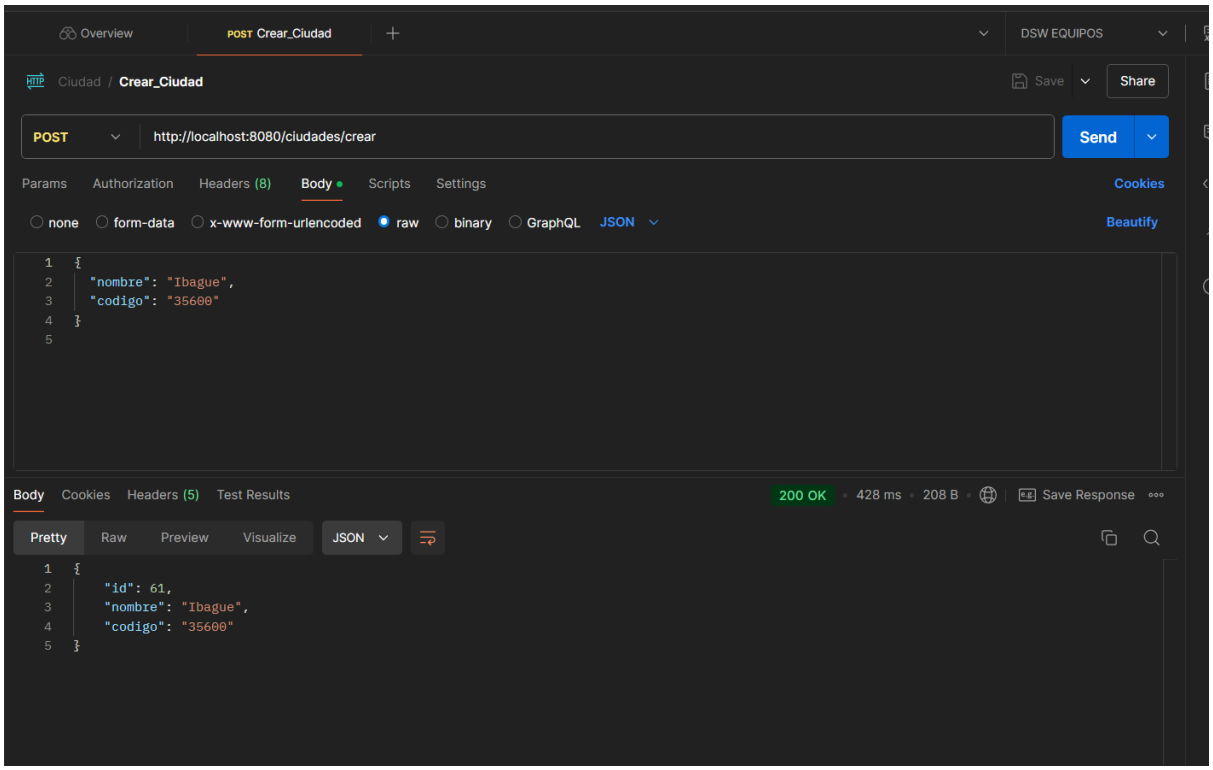
REQUERIMIENTOS FUNCIONALES

1. Ciudad

a) Crear una Ciudad (POST)

- **URL**: http://localhost:8080/ciudades
- **Método**: POST
- **Headers**:
  - Content-Type: application/json
- **Body (JSON)**:

```
{
 "codigo": "15420",
 "nombre": "Bogotá"
}
```



2. Sucursal

a) Crear una Sucursal (POST)

- **URL**: http://localhost:8080/sucursales
- **Método**: POST
- **Headers**:
  - Content-Type: application/json
- **Body (JSON)**:

{

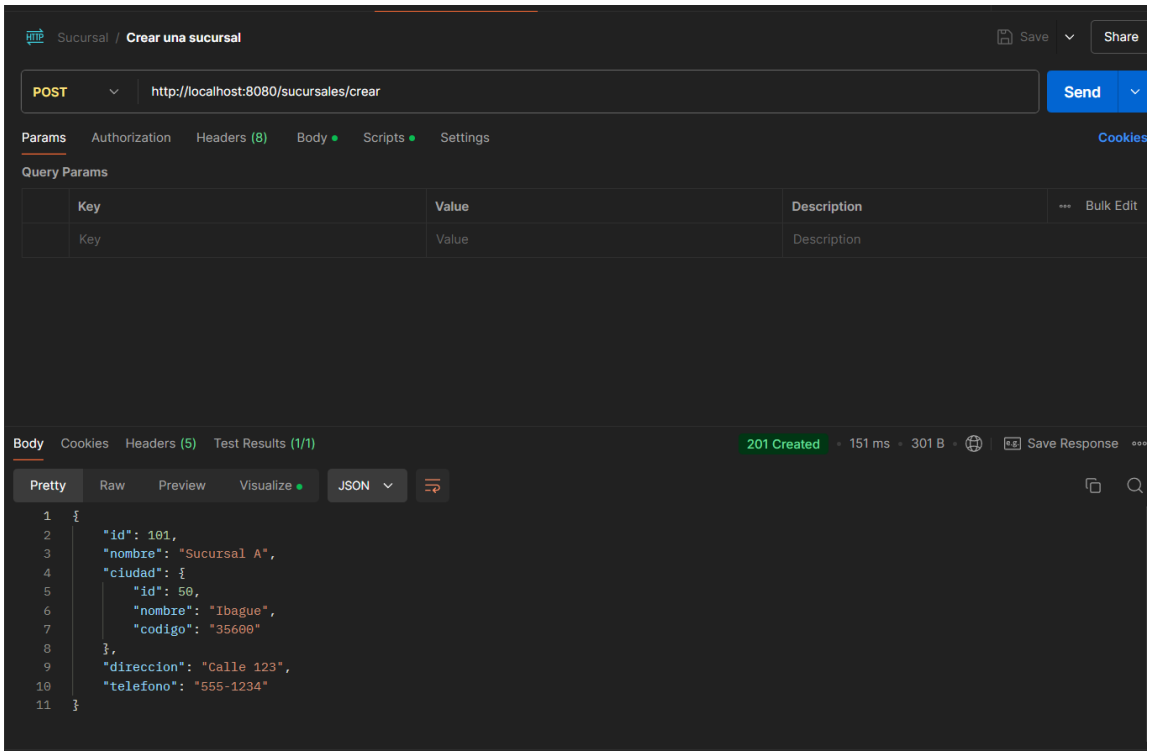
"nombre": "Sucursal Norte",

"direccion": "Calle 123",

"telefono": "1234567",

"ciudadId": 1

}



3. Bodega

a) Crear una Bodega (POST)

- **URL**: http://localhost:8080/bodegas
- **Método**: POST
- **Headers**:
  - Content-Type: application/json
- **Body (JSON)**:

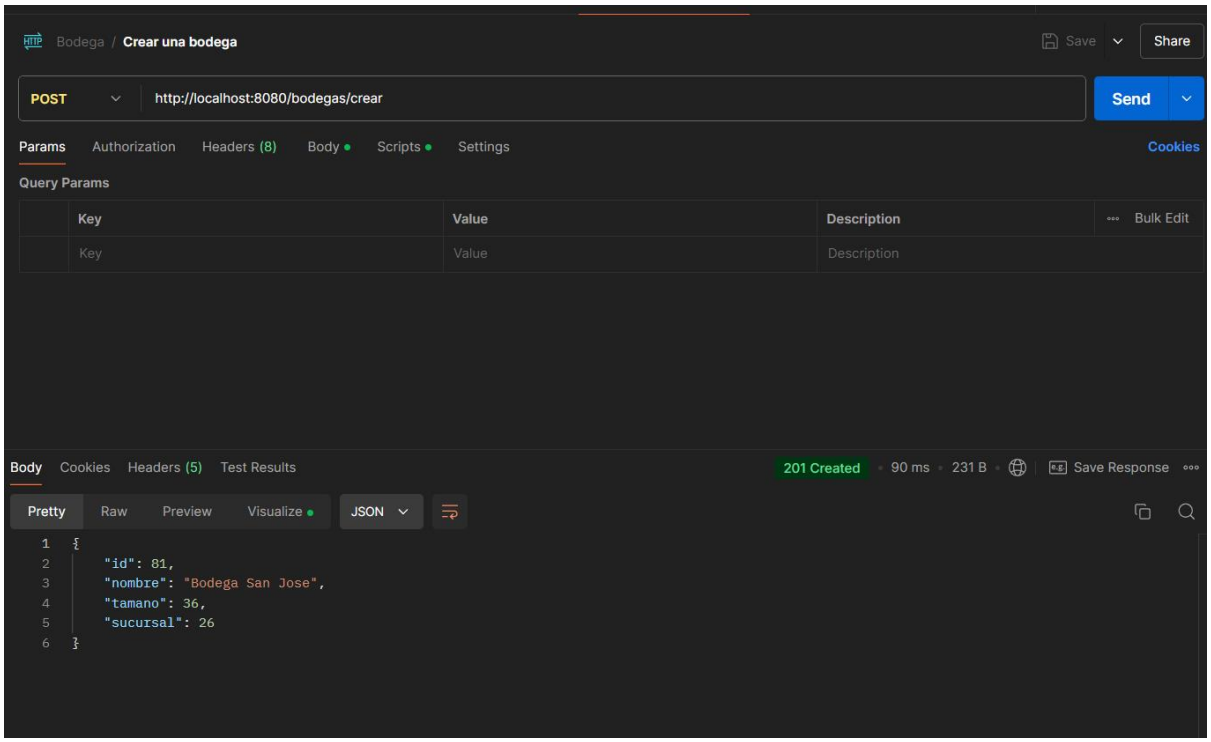
{

"nombre": "Bodega Principal",

"tamanio": 500,

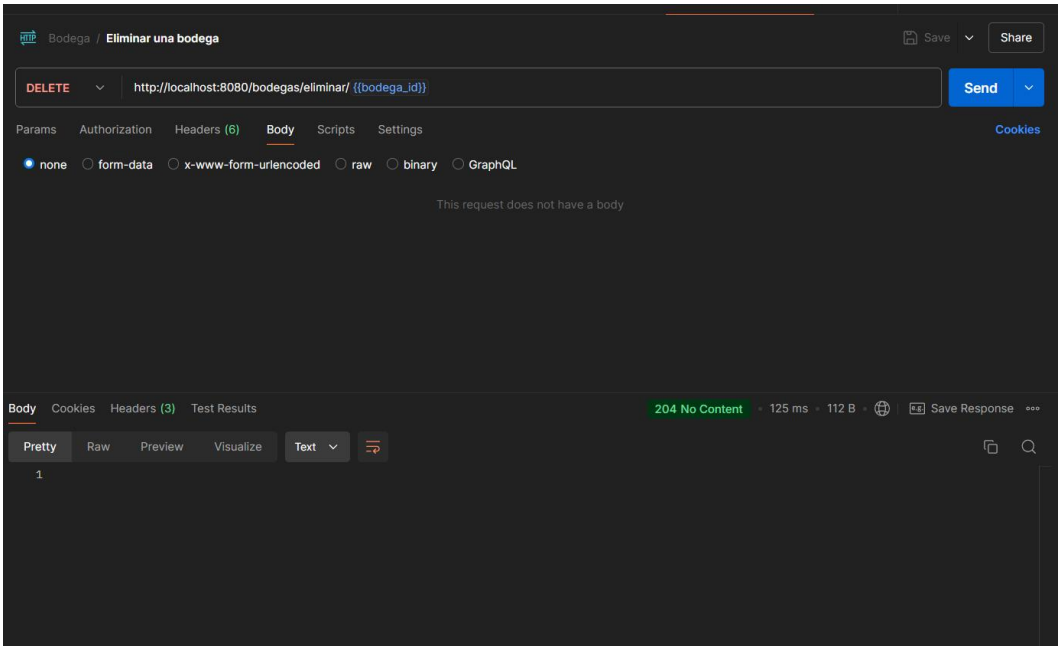
"sucursalId": 1

}



b) Borrar una Bodega (DELETE)

- **URL**: http://localhost:8080/bodegas/{id}
- **Método**: DELETE



4. Proveedor

a) Crear un Proveedor (POST)

- **URL**: http://localhost:8080/proveedores/crear

- **Método**: POST
- **Headers**:
  - Content-Type: application/json
- **Body (JSON)**:

{

"nombre": "Proveedor Ejemplo",

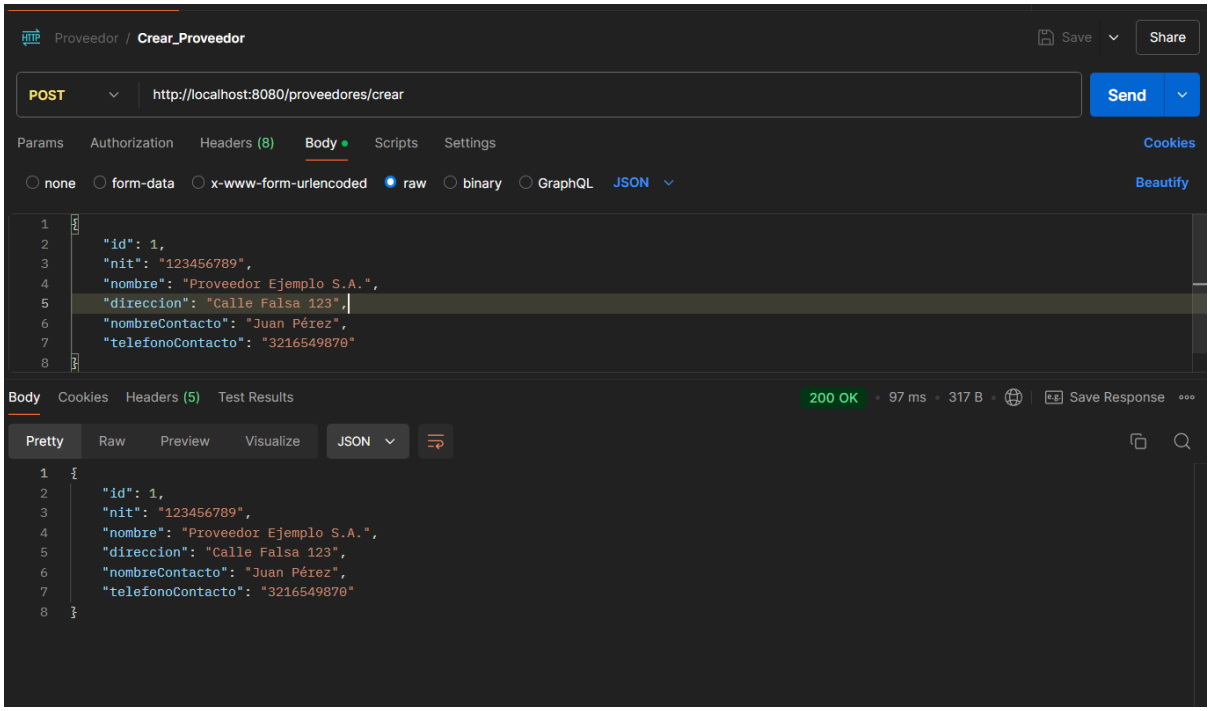
"telefonoContacto": "123456789",

"nombreContacto": "Juan Pérez",

"direccion": "Calle Falsa 123",

"nit": "1234567890"

}



b) Actualizar un Proveedor (PUT)

- **URL**: http://localhost:8080/proveedores/{id}
- **Método**: PUT
- **Headers**:
  - Content-Type: application/json
- **Body (JSON)**:

{

"nombre": "Proveedor Actualizado",

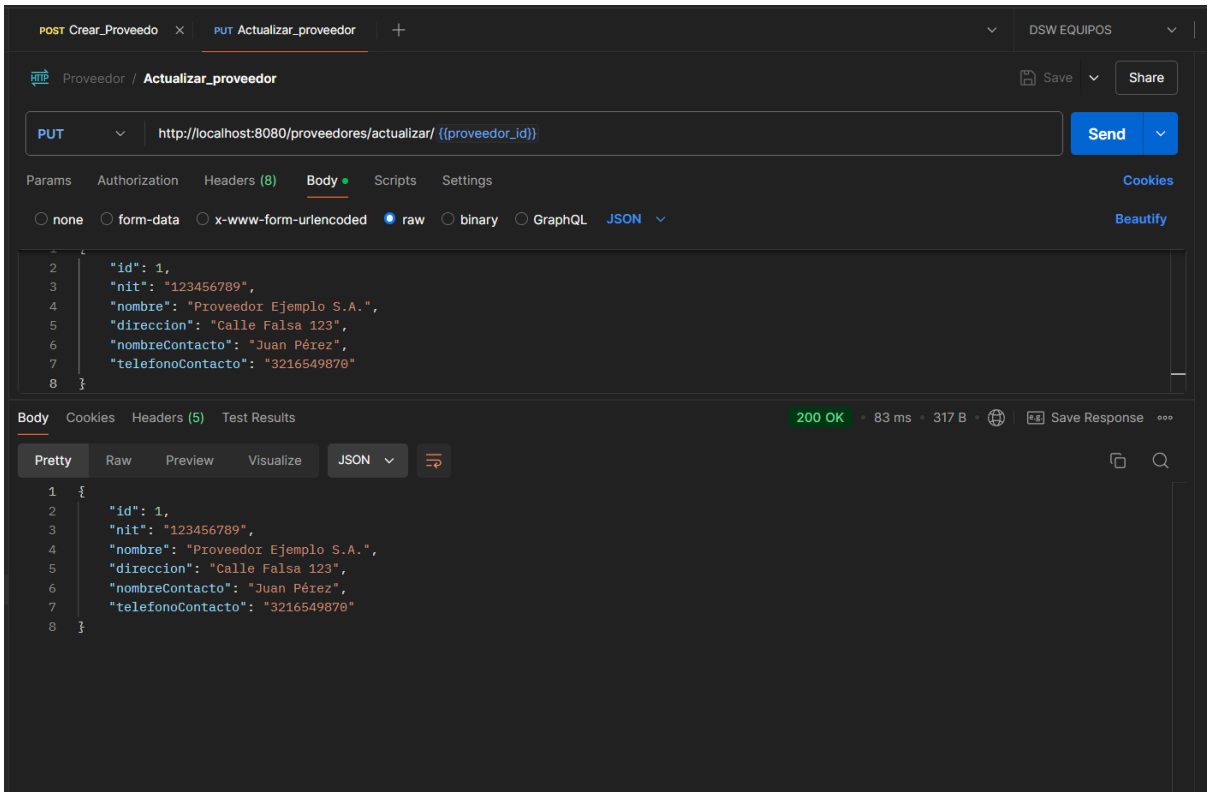
"telefonoContacto": "987654321",

"nombreContacto": "Carlos López",

"direccion": "Calle Verdadera 456",

"nit": "9876543210"

}



5. Categoría de Producto

a) Crear una Categoría de Producto (POST)

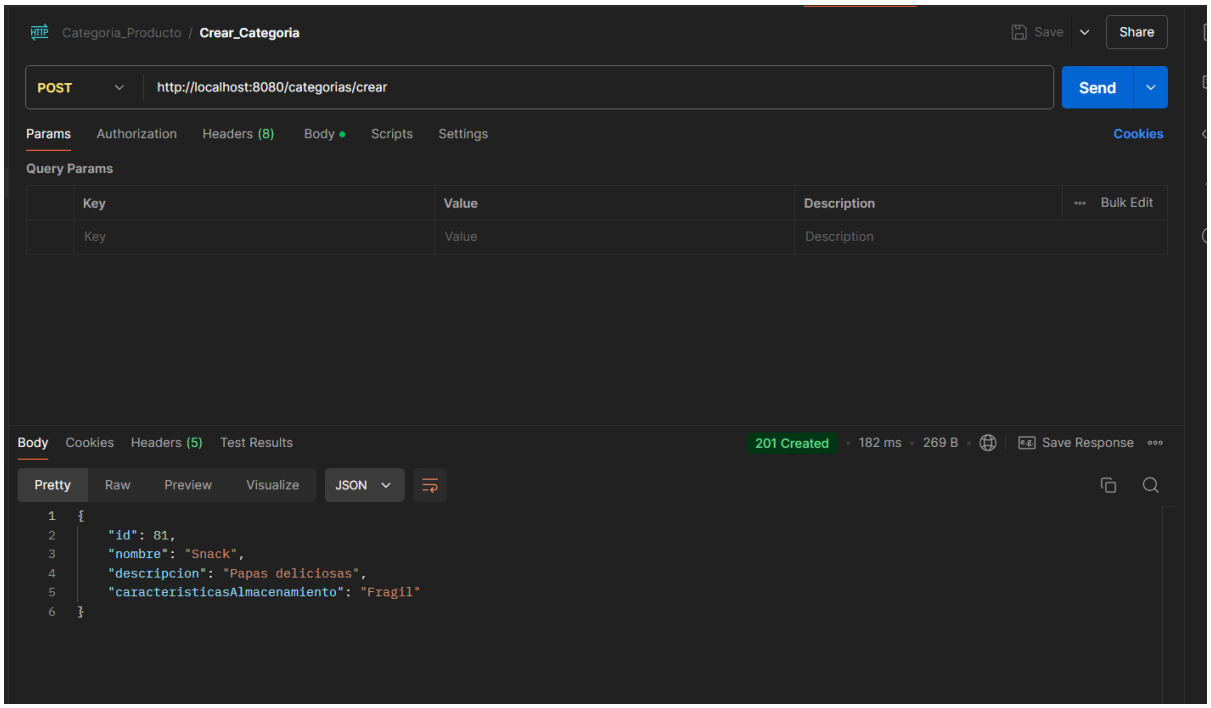
- **URL**: http://localhost:8080/categorias
- **Método**: POST
- **Headers**:
  - Content-Type: application/json
- **Body (JSON)**:

{

"nombre": "Perecedero",

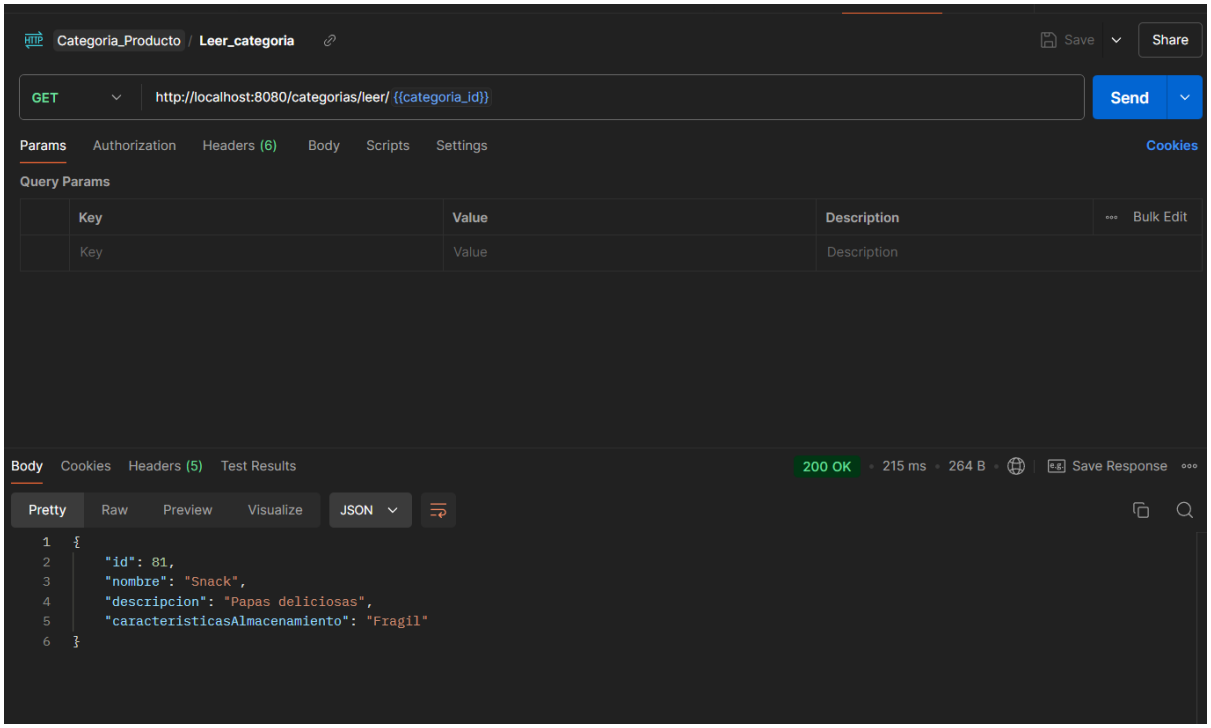
"descripcion": "Productos con fecha de vencimiento",

"caracteristicasAlmacenamiento": "Refrigeración"}



b) Leer una Categoría por ID (GET)

- **URL**: http://localhost:8080/categorias/{id}
- **Método**: GET



6. Producto

a) Crear un Producto (POST)

- **URL**: http://localhost:8080/productos
- **Método**: POST
- **Headers**:
  - Content-Type: application/json
- **Body (JSON)**:

{

"nombre": "Papas fritas Les frites",

"costoBodega": 4200,

"precioUnitario": 9000,

"presentacion": "paquetón de 5 paquetes de 200 gr",

"cantidadPresentacion": 1000,

"unidadMedida": "gr",

"volumenEmpaque": 150,

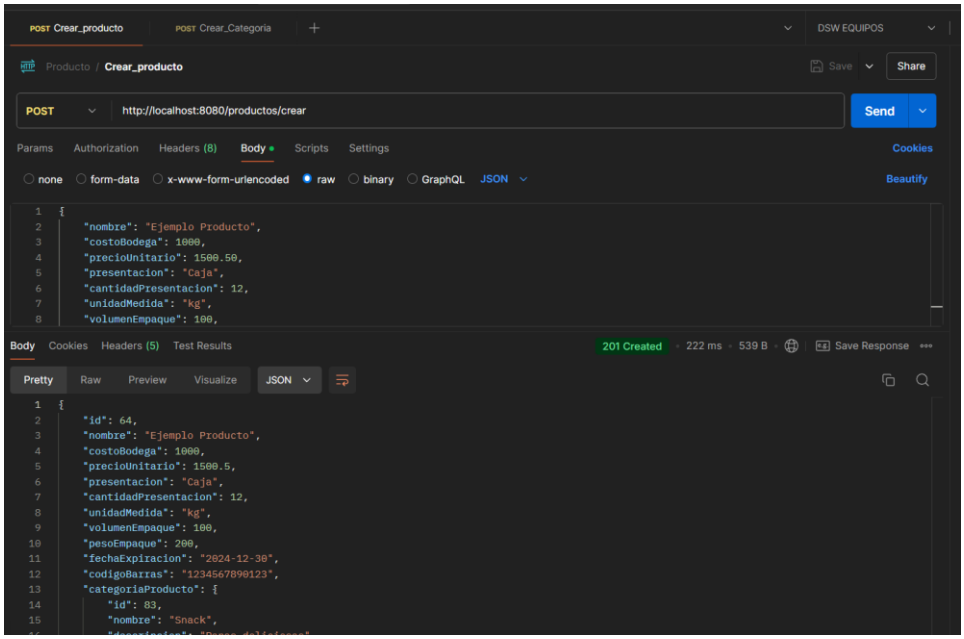
"pesoEmpaque": 15,

"fechaExpiracion": "2027-12-12",

"codigoBarras": "f0f0f0f0f0",

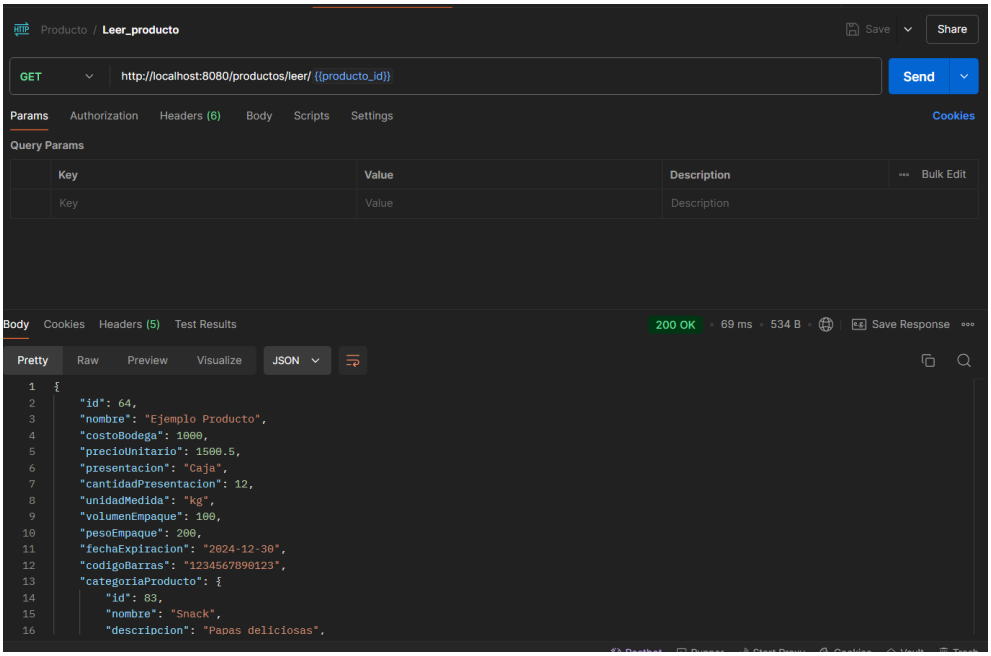
"categoriaId": 1

}



b) Leer un Producto (GET)

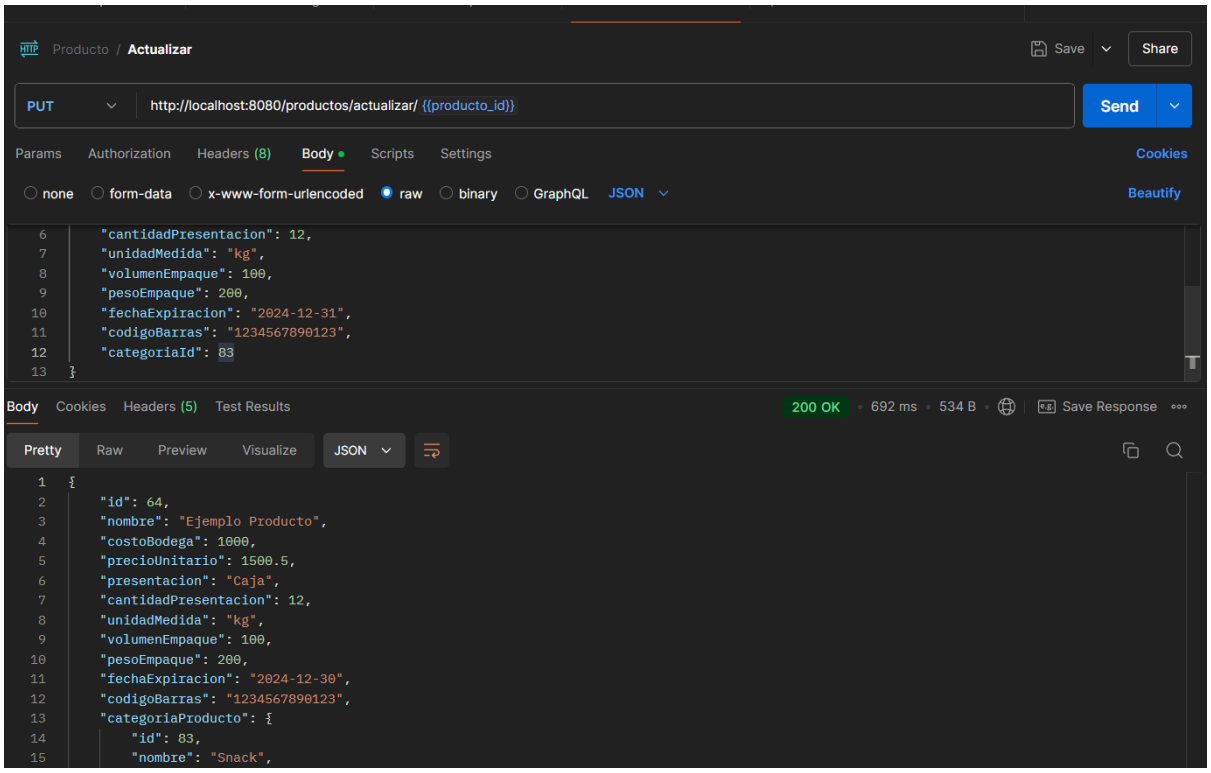
- **URL**: http://localhost:8080/productos/{id}
- **Método**: GET



c) Actualizar un Producto (PUT)

- **URL**: http://localhost:8080/productos/{id}
- **Método**: PUT
- **Headers**:
  - Content-Type: application/json
- **Body (JSON)**:

```
{
 "nombre": "Papas fritas Les frites actualizadas",
 "costoBodega": 4500,
 "precioUnitario": 9500,
 "presentacion": "paquetón de 5 paquetes de 250 gr",
 "cantidadPresentacion": 1250,
 "unidadMedida": "gr",
 "volumenEmpaque": 160,
 "pesoEmpaque": 20,
 "fechaExpiracion": "2028-12-12",
 "codigoBarras": "f0f0f0f0f0"
}
```



7,8,9. Orden de Compra

a) Crear una Orden de Compra (POST)

- **URL**: http://localhost:8080/ordenes-compra
- **Método**: POST
- **Headers**:
  - Content-Type: application/json
- **Body (JSON)**:

{

"fechaEsperadaEntrega": "2024-12-01",

"sucursalId": 1,

"proveedorId": 1,

"productos": [

{

"productoId": 1,

"cantidad": 10,

"precio": 9500

}]}

202125304 -David Ortiz
202215891 - José Salgado
202215816 - Angelica Yeraldin Rodríguez

b) Actualizar una Orden de Compra (Anular) (PUT)

- **URL**: http://localhost:8080/ordenes-compra/{id}/anular
- **Método**: PUT

c) Leer todas las Órdenes de Compra (GET)

- **URL**: http://localhost:8080/ordenes-compra
- **Método**: GET

RFC1 MOSTRAR EL ÍNDICE DE OCUPACIÓN DE CADA UNA DE BODEGAS DE UNA SUCURSAL:

Solicitudes sql para este requerimiento:

```
ALTER TABLE BODEGA ADD CAPACIDAD NUMBER NOT NULL;

SELECT
    B.NOMBRE AS nombre_bodega,
    SUM(P.VOLUMEN_EMPAQUE * P.CANTIDAD_PRESENTACION) AS volumen_ocupado,
    B.CAPACIDAD AS capacidad_total,
    CASE
        WHEN B.CAPACIDAD > 0 THEN ROUND((SUM(P.VOLUMEN_EMPAQUE *
P.CANTIDAD_PRESENTACION) / B.CAPACIDAD) * 100, 2)
        ELSE 0
    END AS indice_ocupacion
FROM BODEGA B
LEFT JOIN PRODUCTO P ON P.CATEGORIA_ID = B.ID
WHERE B.ID IN (SELECT SUCURSAL_ID FROM SUCURSAL WHERE ID = :idSucursal)
GROUP BY B.NOMBRE, B.CAPACIDAD
ORDER BY B.NOMBRE;
```

RFC2 MOSTRAR LOS PRODUCTOS QUE CUMPLEN CON CIERTA CARACTERÍSTICA:

```
SELECT
    P.ID AS producto_id,
    P.NOMBRE AS nombre_producto,
    P.COSTO_BODEGA AS costo_bodega,
    P.PRECIO_UNITARIO AS precio_unitario,
    P.PRESENTACION AS presentacion,
    P.CANTIDAD_PRESENTACION AS cantidad_presentacion,
    P.UNIDAD_MEDIDA AS unidad_medida,
    P.VOLUMEN_EMPAQUE AS volumen_empaque,
    P.PESO_EMPAQUE AS peso_empaque,
    P.FECHA_EXPIRACION AS fecha_expiracion,
    P.CODIGO_BARRAS AS codigo_barras,
    C.NOMBRE AS nombre_categoria
```

```
FROM PRODUCTO P
JOIN CATEGORIA_PRODUCTO C ON P.CATEGORIA_ID = C.ID
WHERE P.PRECIO_UNITARIO BETWEEN :precioMinimo AND :precioMaximo
    AND P.FECHA_EXPIRACION > :fechaVencimiento
    AND P.ID IN (SELECT P2.ID FROM SUCURSAL S JOIN BODEGA B ON S.ID = B.ID
JOIN PRODUCTO P2 ON B.ID = P2.ID
        WHERE S.ID = :idSucursal)
    AND C.ID = :idCategoria;
```

Para ejecutar esta consulta:

- Reemplazar los parámetros :precioMinimo, :precioMaximo, :fechaVencimiento, :idSucursal, y :idCategoria con los valores que se quiere realizar la consulta.

RFC3 INVENTARIO DE PRODUCTOS EN UNA BODEGA:

```
ALTER TABLE BODEGA ADD (CANTIDAD_MINIMA NUMBER, COSTO_PROMEDIO NUMBER);

CREATE TABLE INVENTARIO (
    ID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    BODEGA_ID NUMBER,
    PRODUCTO_ID NUMBER,
    CANTIDAD_ACTUAL NUMBER,
    CONSTRAINT FK_BODEGA FOREIGN KEY (BODEGA_ID) REFERENCES BODEGA(ID),
    CONSTRAINT FK_PRODUCTO FOREIGN KEY (PRODUCTO_ID) REFERENCES PRODUCTO(ID)
);

SELECT
    P.ID AS producto_id,
    P.NOMBRE AS nombre_producto,
    I.CANTIDAD_ACTUAL AS cantidad_actual,
    B.CANTIDAD_MINIMA AS cantidad_minima,
    B.COSTO_PROMEDIO AS costo_promedio
FROM INVENTARIO I JOIN PRODUCTO P ON I.PRODUCTO_ID = P.ID
JOIN BODEGA B ON I.BODEGA_ID = B.ID
JOIN SUCURSAL S ON B.ID = S.ID
WHERE S.ID = :sucursalId AND B.ID = :bodegaId;
```

Reemplazar los parámetros :sucursalId y :bodegaId con los valores correspondientes que se quieren usar en la consulta.

RFC4 MOSTRAR LAS SUCURSALES EN LAS QUE HAY DISPONIBILIDAD DE UN PRODUCTO:

```
SELECT s.*
FROM Sucursal s
JOIN Bodega b ON s.id = b.sucursal_id
JOIN Producto p ON b.id = p.bodega_id
WHERE p.id = :productoId
  AND p.cantidad_actual > 0;
```

RFC5 MOSTRAR TODOS LOS PRODUCTOS QUE REQUIEREN UNA ORDEN DE COMPRA:

```
SELECT P.ID AS producto_id,
    P.NOMBRE AS nombre_producto,
    B.NOMBRE AS bodega,
    S.NOMBRE AS sucursal,
    PR.NOMBRE AS proveedor,
SUM(O.CANTIDAD_PRODUCTO) AS cantidad_actual
FROM ORDEN_DE_COMPRA O
JOIN PRODUCTO P ON O.PRODUCTO_ID = P.ID
JOIN SUCURSAL S ON O.SUCURSAL_ID = S.ID
JOIN BODEGA B ON S.ID = B.ID
JOIN PROVEEDOR PR ON O.PROVEEDOR_NIT = PR.NIT
WHERE SUM(O.CANTIDAD_PRODUCTO) < (SELECT valor_minimo FROM BODEGA_CONFIG BC
WHERE BC.PRODUCTO_ID = P.ID AND BC.BODEGA_ID = B.ID)
GROUP BY P.ID, P.NOMBRE, B.NOMBRE, S.NOMBRE, PR.NOMBRE
HAVING SUM(O.CANTIDAD_PRODUCTO) < (SELECT valor_minimo FROM BODEGA_CONFIG BC
WHERE BC.PRODUCTO_ID = P.ID AND BC.BODEGA_ID = B.ID);
```

Balance del plan de pruebas (que se logró hacer y qué no).

Logramos cumplir con los requerimientos de tipo RF, creando la base de datos en SQL de acuerdo con el modelo UML y el diseño relacional indicados en la entrega anterior, integrando además las correcciones sugeridas. Finalizamos los requerimientos de tipo RFC mediante SQL, pero no fue posible implementarlos en código Java dentro del proyecto, ya que los cambios requeridos habrían comprometido el funcionamiento de las pruebas en **Postman**. Dado que estas pruebas eran más numerosas, decidimos priorizarlas, ya que además los RFC están resueltos en SQL.

Para los RFC, hicimos una suposición sobre los datos y la forma en que se crearían en **Postman** para verificar su implementación. A pesar de esto, las pruebas en **Postman** fueron cruciales para validar las operaciones en la base de datos, por lo que priorizarlas nos permitió asegurar un proceso de pruebas consistente en lugar de comprometer otras partes del proyecto.

202125304 -David Ortiz
202215891 - José Salgado
202215816 - Angelica Yeraldin Rodríguez

INFORME- IMPLEMENTACIÓN SISTEMAS TRANSACCIONALES

Lógica de RFC en Postman (ingresos y resultado esperados)

RFC1: Mostrar el índice de ocupación de cada una de las bodegas de una sucursal

a) Índice de Ocupación de Bodegas (GET)

- URL: http://localhost:8080/ocupacion-bodegas/{sucursalId}
- Método: GET
- Descripción: Este RFC muestra el índice de ocupación de todas las bodegas que pertenecen a una sucursal específica.
- Parámetros:
  - {sucursalId}: ID de la sucursal.

Ejemplo de URL:
http://localhost:8080/ocupacion-bodegas/1
Respuesta esperada (JSON):

```
[
 {
  "bodegaId": 1,
  "nombreBodega": "Bodega Principal",
  "capacidadTotal": 1000,
  "capacidadOcupada": 750,
  "indiceOcupacion": 0.75
 },
 {
  "bodegaId": 2,
  "nombreBodega": "Bodega Secundaria",
  "capacidadTotal": 500,
  "capacidadOcupada": 300,
  "indiceOcupacion": 0.6
 }
]
```

RFC2: Mostrar los productos que cumplen con cierta característica (precio, fecha de vencimiento, disponibilidad en sucursal o categoría)

a) Productos por Característica (GET)

- URL: http://localhost:8080/productos/caracteristicas
- Método: GET
- Descripción: Este RFC permite filtrar productos según características específicas como el precio, fecha de vencimiento, disponibilidad en sucursales, o categoría.
- Parámetros de consulta (Query Params):
  - precioMin (opcional): Precio mínimo del producto.
  - precioMax (opcional): Precio máximo del producto.
  - fechaVencimiento (opcional): Fecha de vencimiento máxima de los productos (formato: YYYY-MM-DD).
  - disponibilidadSucursalId (opcional): ID de la sucursal donde se desea verificar la disponibilidad.
  - categoriaId (opcional): ID de la categoría de los productos.

Ejemplo de URL:

http://localhost:8080/productos/caracteristicas?precioMin=1000&precioMax=5000&fechaVencimiento=2024-12-31&disponibilidadSucursalId=1&categoriaId=2

Respuesta esperada (JSON):


```
[
 {
  "productoId": 1,
  "nombre": "Galletas de Chocolate",
  "precioUnitario": 4500,
  "fechaExpiracion": "2024-11-30",
  "cantidadDisponible": 200,
  "sucursalId": 1,
  "categoriaId": 2
 },
 {
  "productoId": 2,
  "nombre": "Jugo de Naranja",
  "precioUnitario": 3000,
  "fechaExpiracion": "2024-09-15",
  "cantidadDisponible": 100,
  "sucursalId": 1,
  "categoriaId": 2
 }
]
```


RFC3: Inventario de productos en una bodega (detalles de cantidad actual, mínima requerida y costo promedio)

a) Inventario de una Bodega (GET)

- URL: http://localhost:8080/inventario/{sucursalId}/{bodegaId}
- Método: GET
- Descripción: Este RFC muestra el inventario de productos en una bodega específica de una sucursal, incluyendo detalles como la cantidad actual, cantidad mínima requerida y el costo promedio.
- Parámetros:
  - {sucursalId}: ID de la sucursal.
  - {bodegaId}: ID de la bodega.

Ejemplo de URL:

http://localhost:8080/inventario/1/2

Respuesta esperada (JSON):


```
[
 {
  "productoId": 1,
  "nombreProducto": "Leche Entera",
  "cantidadActual": 50,
  "cantidadMinimaRequerida": 30,
  "costoPromedio": 4000
 },
 {
  "productoId": 2,
  "nombreProducto": "Arroz",
```

"cantidadActual": 200,
"cantidadMinimaRequerida": 100,
"costoPromedio": 2500
}
]

RFC4: Mostrar las sucursales en las que hay disponibilidad de un producto

a) Disponibilidad de Producto en Sucursales (GET)

- URL: http://localhost:8080/productos/disponibilidad/{productoId}
- Método: GET
- Descripción: Este RFC muestra en qué sucursales está disponible un producto específico.
- Parámetros:
  - {productoId}: ID del producto.

Ejemplo de URL:
http://localhost:8080/productos/disponibilidad/1
Respuesta esperada (JSON):

[
{
"sucursalId": 1,
"nombreSucursal": "Sucursal Norte",
"direccion": "Calle 123",
"telefono": "1234567",
"cantidadDisponible": 50
},
{
"sucursalId": 2,
"nombreSucursal": "Sucursal Sur",
"direccion": "Avenida 456",
"telefono": "9876543",
"cantidadDisponible": 20
}
]

RFC5: Mostrar todos los productos que requieren una orden de compra (nombre del producto, bodega, sucursales y proveedores)

a) Productos que requieren Orden de Compra (GET)

- URL: http://localhost:8080/ordenes-compra/necesarios
- Método: GET
- Descripción: Este RFC muestra todos los productos que requieren una orden de compra debido a baja disponibilidad o agotamiento, detallando el nombre del producto, la bodega, las sucursales y los proveedores relacionados.

Ejemplo de URL:
http://localhost:8080/ordenes-compra/necesarios
Respuesta esperada (JSON):

[
{
"productoId": 1,
"nombreProducto": "Arroz",

"cantidadActual": 20,
"cantidadMinimaRequerida": 100,
"bodega": {
"bodegaId": 1,
"nombreBodega": "Bodega Principal"
},
"sucursales": [
{
"sucursalId": 1,
"nombreSucursal": "Sucursal Norte"
}],
"proveedores": [
{
"proveedorId": 1,
"nombreProveedor": "Proveedor Ejemplo"
}]},
{
"productoId": 2,
"nombreProducto": "Jugo de Naranja",
"cantidadActual": 10,
"cantidadMinimaRequerida": 50,
"bodega": {
"bodegaId": 2,
"nombreBodega": "Bodega Secundaria"
},
"sucursales": [
{
"sucursalId": 2,
"nombreSucursal": "Sucursal Sur"
}
],
"proveedores": [
{
"proveedorId": 2,
"nombreProveedor": "Proveedor ABC"
}] }]

Nombre del usuario en Oracle en donde se encuentran las tablas del modelo físico del proyecto

Credenciales de Angelica Rodriguez Gualtero:

Usuario: ISIS2304D22202420

Contraseña: bxdFKDMDjD