

Ximena López – 202312848

Santiago Pineda - 202023262

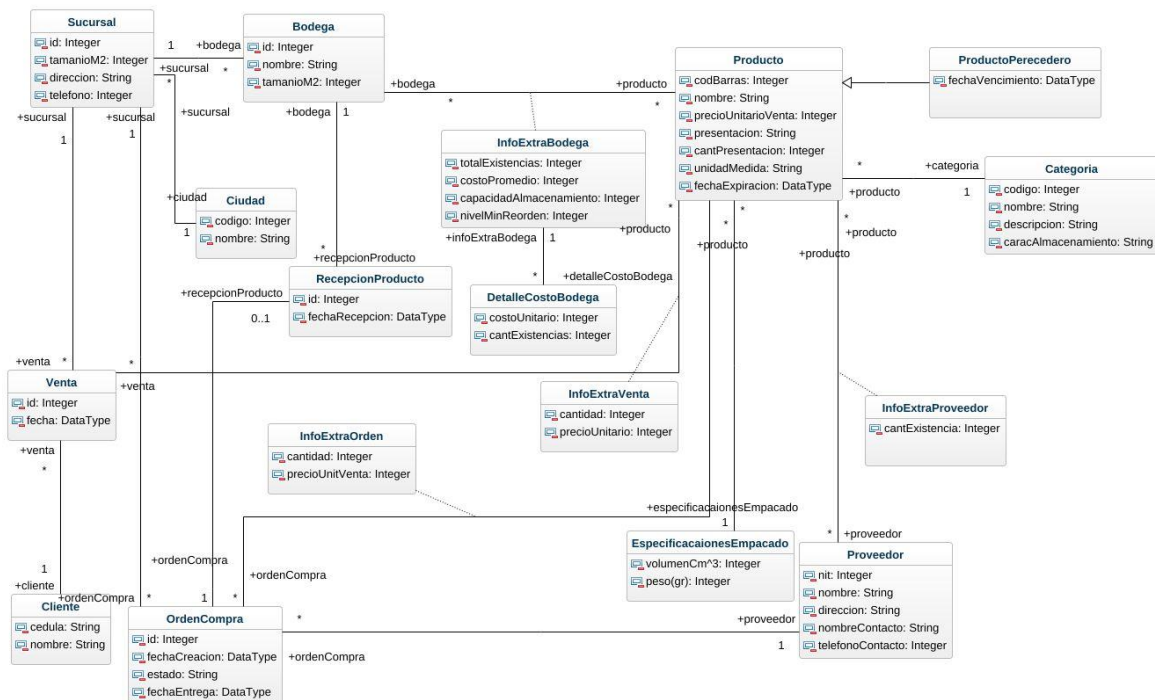
Sofia Losada - 202221008

Documentación Proyecto 1

Nombre de usuario Oracle:

ISIS2304A20202420

Diagrama UML actualizado:



Análisis UML:

- Descripción de los elementos clave:
 - Sucursal: Atributos: id, tamañoM2, dirección, teléfono.
 - Relaciones:
 - Bodega: Una sucursal puede tener múltiples bodegas (relación 1 a *), pero una bodega pertenece a una única sucursal (relación * a 1).
 - Ciudad: Una sucursal está ubicada en una ciudad (relación 1 a 1), pero una ciudad puede tener múltiples sucursales (relación * a 1).

- Bodega: Atributos: id, nombre, tamañoM2.
 - Relaciones:
 - Sucursal: Una bodega pertenece a una única sucursal (relación * a 1).
 - Producto: Una bodega puede almacenar múltiples productos (relación * a *), y cada producto puede estar presente en varias bodegas.
 - InfoExtraBodega: Relaciona una bodega con información adicional de productos específicos en dicha bodega (relación 1 a *).

- Producto: Atributos: codBarras, nombre, precioUnitarioVenta, presentación, cantidadPresentación, unidadMedida, fechaExpiración.
 - Relaciones:
 - InfoExtraBodega: Un producto puede estar almacenado en múltiples bodegas con diferentes informaciones extras, como existencias y costos (relación * a 1 con respecto a InfoExtraBodega).
 - InfoExtraProveedor: Un producto puede ser provisto por múltiples proveedores con diferentes cantidades de existencias (relación * a 1).
 - RecepcionProducto: Un producto puede estar relacionado con múltiples recepciones de productos (relación * a 1), pero una recepción está relacionada con un único producto (relación 0..1 a *).

- Producto Perecedero: Subclase de Producto, agregando el atributo fechaVencimiento.
 - Relaciones: Hereda todas las relaciones de Producto.

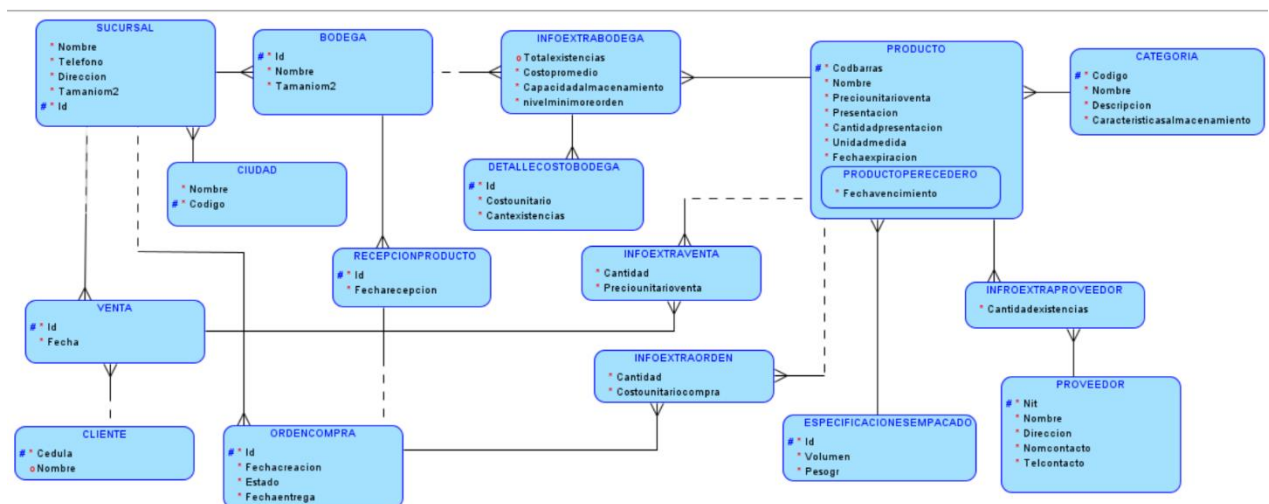
- Recepción de Productos: Atributos: id, fechaRecepción.
 - Relaciones:
 - Producto: Relación 0..1 a * (un producto puede tener múltiples recepciones, y una recepción puede no tener productos asociados o puede tener un solo producto).
 - DetalleCostoBodega: Múltiples productos recibidos pueden estar asociados a un único registro de costos (relación * a *).

- Cliente: Atributos: cédula, nombre.
 - Relaciones:
 - Venta: Un cliente puede realizar múltiples ventas (relación 1 a *), pero una venta está relacionada con un único cliente (relación * a 1).
 - OrdenCompra: Un cliente puede hacer múltiples órdenes de compra (relación 1 a *), pero una orden de compra está asociada a un solo cliente (relación * a 1).

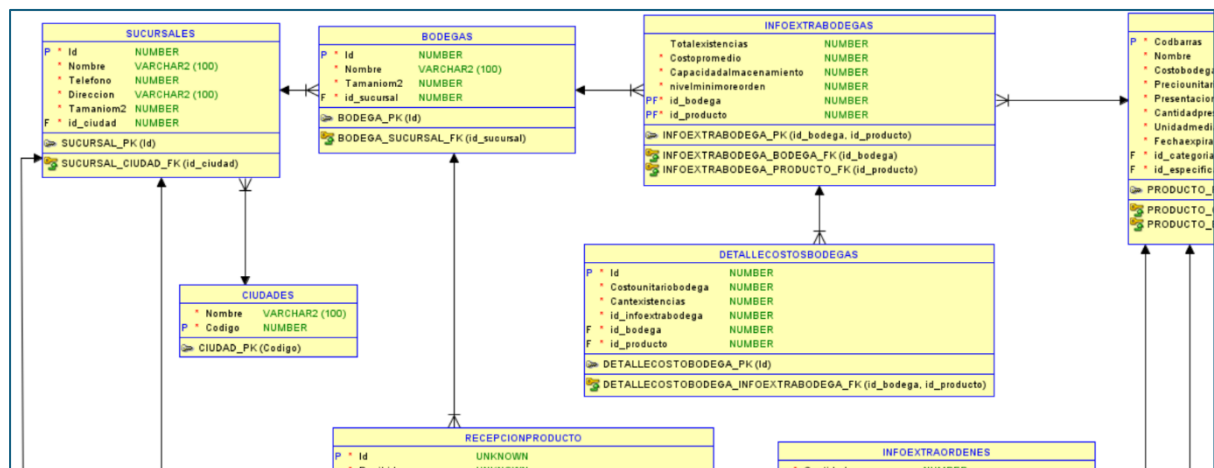
- Orden de Compra: Atributos: id, fechaCreación, estado, fechaEntrega.
 - Relaciones:
 - Cliente: Relación * a 1 (múltiples órdenes para un cliente).
 - InfoExtraOrden: Cada orden de compra puede tener múltiples detalles asociados (relación 1 a *).

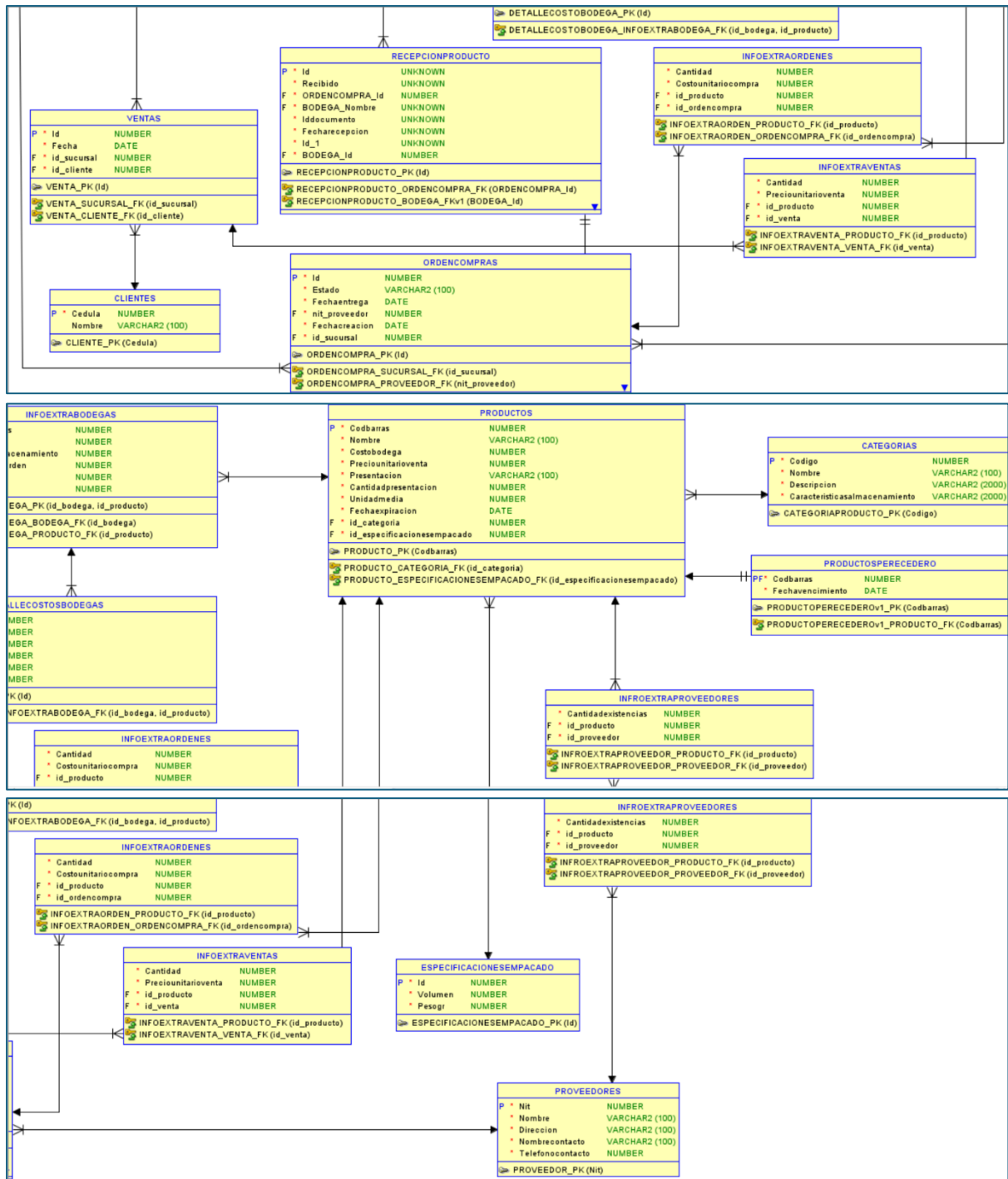
- Venta: Atributos: id, fecha.
 - Relaciones:
 - Cliente: Relación * a 1 (múltiples ventas pueden estar asociadas a un cliente).
 - InfoExtraVenta: Una venta puede tener múltiples detalles adicionales (relación 1 a *).
- InfoExtraOrden: Atributos: cantidad, precioUnitVenta.
 - Relaciones:OrdenCompra: Múltiples detalles de la orden pueden estar asociados a una única orden de compra (relación * a 1).
 - Producto: Cada detalle de la orden está relacionado con un único producto (relación * a 1).
- InfoExtraVenta: Atributos: cantidad, precioUnitario.
 - Relaciones:
 - Venta: Múltiples detalles de la venta pueden estar asociados a una venta (relación * a 1).
 - Producto: Cada detalle de venta se refiere a un único producto (relación * a 1).
- Proveedor: Atributos: nit, nombre, dirección, nombreContacto, teléfonoContacto.
 - Relaciones:
 - EspecificacionesEmpacado: Un proveedor puede tener múltiples especificaciones de empaque para sus productos (relación 1 a *).
 - InfoExtraProveedor: Un proveedor puede estar relacionado con múltiples productos que proporciona (relación 1 a *).
- EspecificacionesEmpacado: Atributos: volumenCm3, pesoGr.
 - Relaciones:
 - Producto: Un producto puede tener diferentes especificaciones de empaquetado (relación * a 1).
 - Proveedor: Un proveedor puede tener múltiples especificaciones de empaquetado para sus productos (relación 1 a *).
- InfoExtraProveedor: Atributos: cantExistencia.
 - Relaciones:
 - Producto: Múltiples relaciones de proveedores a productos, con información extra como las existencias (relación * a 1).
 - Proveedor: Múltiples productos pueden estar relacionados con un proveedor específico (relación * a 1).

En base al UML se actualizo el Modelo Entidad-Relación:



Con esto llegamos al Modelo Relacional siguiente (para ver mejor el modelo mirar el archivo ModeloRelacionalProyecto1.dmd):





Asimismo, por el cambio en el UML, se realizaron las debidas actualizaciones a las tablas relacionales. A las cuales, se le añadieron las nuevas clases como venta, y se corrigieron las relaciones intermedias. Para poder visualizarlas mejor se adjunta el archivo en formato Excel.

Sucursales					
id	nombre	direccion	telefono	tamanoM2	id_ciudad
SA, PK	NN	NN	NN	NN	FKciudades.codigo
Suc-01	sucursal norte	Carrera 4	3222345931	500	1001
Suc-01	sucursa sur	calle 55	3112018882	100	1001

Proveedores				
nit	nombre	direccion	nomContacto	telContacto
PK, UA	NN	NN	NN	NN
1302	Lay	calle 4#2-34	Juan	3112018888

Productos								
cod Barras	nombre	precio Unitario	presentacion	cantPresentacion	unidad Medida	fechaExpiracion	id_categoria	id_especificaciones
PK,SA	NN	NN	NN	NN	NN	NN,NC	NN,FKcategoriaProducto.id	NN,FKEspecificacionesEmpacado.id
Pro-01	Papas fritas les frites	9000	paquetón de 5 paquetes de 200 gr. cada uno	1000	gr	12/12/2027	123	123

Bodegas			
id	nombre	tamanoM2	id_sucursal
PK, SA	NN	NN	NN, FKsucursal.id
Bod-01	Bodeguita	100	Suc-01

Ciudades	
codigo	nombre
PK,UA	NN
1001	Bogotá
1001	

RecepcionProductos			
idDocumento	fechaRecepcion	id_OrdenCompra	id_bodega
PK,UA	NN,NC	FKordenCompra.id	Fkbodega.id
recepcion-01	17/02/2006	Orden-01	Bod-01

OrdenCompras					
id	estado	fecha_entrega	id_sucursal	fechaCreacion	id_proveedor
PK,SA	NN	NN	FKsucursal.id, NN	NN	FKproveedores.id, NN
Orden-01	Entregada	12/02/2020	Suc-01	12/01/2992	1302

Orden-03	Entregada		Suc-70		2029
----------	-----------	--	--------	--	------

Categorías			
codigo	nombre	descripción	caracAlmacenamiento
PK,UA	NN	NN	NN
124	No-perecedero	No vencimiento	No frio

Cliente	
cedula	nombre
PK,UA	NN
1001	Juliana

Ventas			
id	fecha	id_sucursal	id_cliente
PK,SA	NN	Fksucursal.id,NN	NN,Fkclientes.id
124	17/02/2002	SUC11	1001

Esta clase se añadió, debido a que no la teníamos contemplada y estábamos dejando de lado información necesaria al relacionarla.

ProductosPerecederos									
cod Barras	nombre	precioUnitario	presentacion	cantPresentacion	unidadMedida	fechaExpiracion	id_categoria	id_especificaciones	fechaVencimiento
PK,SA	NN	NN	NN	NN	NN	NN,NC	NN,FKcategoriaProducto.id	NN,FKEspecificacionesEmpacado.id	NN
Pro-01	Papas fritas les frites	9000	paquetón de 5 paquetes de 200 gr. cada uno	1000	gr	12/12/2027	123	123	12/02/2002

InfoExtraVentas			
id_venta	id_producto	cantidad	precioUnitario
PK,Fkventas.id	PK,Fkproductos.id	NN	NN
VENTA100	PRO 12	12	1000

tabla se añadió para poder tener datos que se necesitan de la relación que ya estaba.

InfoExtraOrdenes

id_ordenCompra	id_producto	cantidad	costoUnitario
PK, Fkordencompras.id	PK, FK productos.id	NN	NN
ORDEN100	PRO 12	12	1000

Esta tabla se añadió para poder tener datos que se necesitan de la relación que ya estaba.

InfoExtraProveedores		
id_proveedor	id_producto	cantidadExistencias
PK, Fkproveedore.id	PK, Fkproductos.id	NN
VENTA100	PRO 12	12

Esta tabla se añadió para poder tener datos que se necesitan de la relación que ya estaba.

InfoExtraBodegas					
id_bodega	id_producto	totalExistencias	costoPromedio	capacidadAlmacenamiento	nivelMinimoReorden
PK, Fjbodegas.id	PK, Fkproductos.id	NN	NN	NN	NN
VENTA100	PRO 12	12	1000	1000	1000

Esta tabla se añadió para poder tener datos que se necesitan de la relación que ya estaba.

DetalleCostosBodegas					
id	costoUnitarioBodega	cantidadExistencias	id_bodega	id_InfoExtra	id_producto
PK	NN	NN	Fkbodegas.id	PKinfoExtraBodegas.pk	Fkproductos.codbarras
VENTA100	PRO 12	12	1000	1000	1000

EspecificacionesEmpacado		
id	Volumen	peso
PK, SA	NN	NN
es-01	1001	5

La clase especificaciones empacado se añadió debido a que sin ella se podía violar la 1FN poner varias especificaciones de empacado en un mismo atributo de la tabla Productos.

Normalización:

- Primera forma normal:

El modelo que se planteó para el diseño sigue la primera forma normal (1FN) debido a que cada celda de las tablas contiene valores atómicos (individuales), y no se encuentran grupos repetitivos o valores múltiples como listas o arreglos dentro de estas. Cabe resaltar que, en el caso de la tabla “EspecificacionesEmpacado”, fue creada para mantener esta forma normal ya que como había más de una característica a tomar en cuenta se decidió crear una tabla aparte y dejar el id de esta tabla como llave foránea en Productos. Así en ambos casos quedarían valores individuales en cada atributo de las tablas, y hay forma de saber las especificaciones de empaque del producto, mientras se mantiene el 1FN. De igual forma, las demás tablas del modelo fueron creadas de manera que ninguna celda tenga valores no atómicos y el modelo quede dentro del 1FN.

- Segunda forma normal:

Siguiendo con el proceso de normalización, en el caso de la segunda forma normal (2FN), aparte de que se cumpla la 1FN, es necesario que todos los atributos de la tabla dependan completamente de las llaves primarias y no solo de una parte. Así que, para el modelo se logró llegar al cumplimiento de estos criterios. De tal manera que, por ejemplo, en uno de los casos donde hay una llave primaria compuesta como lo es para InfoExtraProveedores:

InfoExtraProveedores		
id_proveedor	id_producto	cantidadExistencias
PK,Fkproveedore.id	PK,Fkproductos.id	NN
VENTA100	PRO 12	12

donde están las llaves id_proveedor y id_producto. Así que, para el atributo es necesario conocer ambas llaves ya que para “cantidadExistencias”, sin el id_producto no se puede saber a qué producto se le necesita saber la cantidad de existencias que son y sin el id del proveedor no se va a saber a qué proveedor le corresponde esta cantidad. De manera similar, se puede observar este caso con las demás tablas que tienen llaves compuestas, ya que en general estas representan relaciones entre dos tablas. Por lo que siempre será necesario conocer todos los datos para saber específicamente a que atributos de la relación están asignados los valores. Por lo tanto, el modelo si cumple con la 2FN.

- Tercera forma normal:

En el caso de la tercera forma normal (3FN), es necesario que se cumpla la 2FN y que no haya dependencias transitivas entre los atributos no clave (atributo que no hace parte de la llave primaria) y la PK. Es decir que ningún atributo que no esté en la llave primaria debe depender de otro que tampoco lo esté. En general, las tablas del modelo propuesto si cumple con la 3FN y un buen ejemplo para mostrarlo es con la tabla de “Productos”:

Productos								
cod Barras	nombre	precio Unitario	presentacion	cantPresentacion	unidadMedida	fechaExpiracion	id_categoria	id_especificaciones
PK,SA	NN	NN	NN	NN	NN	NN,NC	NN,FKcategoriaProducto.id	NN,FKEspecificacionesEmpacado.id
Pro-01	Papas fritas les frites	9000	paquetón de 5 paquetes de 200 gr. cada uno	1000	gr	12/12/2027	123	123

Ya que en esta están la mayor cantidad de atributos y todos solo dependen de la llave primaria que sería el código de barras del producto. Con esto ya se puede saber todo lo necesario para esta tabla, como la “unidadMedida” en la que se sabe como se mide el producto, o tambien su presentación. Y esto con cada uno de los atributos, dependiendo únicamente del Código de barras (clave primaria), sin depender entre sí, lo que demuestra que la tabla está en 3FN.

Esto igualmente se cumple para las demás tablas del modelo, ya sea porque tienen una PK que permite obtener cualquier atributo y/o no existen dependencias transitivas entre las llaves no clave.

- Forma Normal de Boyce-Codd:

Para la forma normal de Boyce-Codd (BCNF), se necesita que el modelo este en 3FN y que, además todas las llaves determinantes (atributos que determinan otros atributos en una dependencia funcional) deben ser atributos clave (primos) o triviales con la llave. Lo que significa que es una versión más restrictiva que el 3FN, así que por lo general las tablas del modelo que tienen una única llave primaria (solo un atributo) y ya están en 3FN, pertenecen también a BCNF. Por lo que, quedaría verificar las tablas que tienen llaves primarias compuestas.

InfoExtraBodegas					
id_bodega	id_producto	totalExistencias	costoPromedio	capacidadAlmacenamiento	nivelMinimoReorden
PK,Fjbodegas.id	PK,Fkproductos.id	NN	NN	NN	NN
VENTA100	PRO 12	12	1000	1000	1000

En primer lugar, está “InfoExtraBodegas”, que como se había establecido anteriormente, para esta tabla que es una relación intermedia, los atributos que no hacen parte de la llave primaria dependen de esta, y véase que como ambos son necesarios para saber los atributos, realmente no tienen un elemento en común individualmente. O sea, de

id_producto no es posible saber costo promedio o nivelMinimoReorden, es necesario saber el id_bodega, y viceversa. Esto es aplicable para cualquier atributo dentro de la tabla. De esta manera la tabla si entrará en BCNF.

Este mismo caso se repite para las tablas ya mencionadas en los otros casos de normalización. Hay otros casos en donde las llaves primarias tienen acceso a un único atributo como en el caso de “InfoExtraProveedores” y entraría en la misma circunstancia que la tabla anterior, dado que en cada caso ambas claves son necesarias para determinar los atributos no clave, no existen dependencias funcionales que violen BCNF. Así que, el modelo finalmente entraría en BCNF.

Arquitectura de la aplicación:

La implementación se desarrolla bajo la arquitectura de modelo vista controlador. De esta manera, es posible gestionar múltiples entidades y aspectos del negocio.

La separación en capas permite que cada parte del sistema tenga una responsabilidad bien definida, facilitando el mantenimiento y la escalabilidad del sistema. Las principales capas son: Capa de Lógica de Negocio, Capa de Persistencia: Mapea las clases del diagrama UML a tablas en la base de datos y maneja la persistencia de los datos. Capa de Base de Datos: Almacena los datos de forma estructurada y relacional. De esta manera, las entidades y relaciones previamente descritas se contienen en estas capas y permiten un flujo de procesos más eficiente en la ejecución de operaciones.

Traza de la ejecución del plan de pruebas

- **RF1:**

Acá estamos creando una ciudad con el nombre de “Barranquilla”, y el código de la ciudad se genera automáticamente por la secuencia creada en el script SQL.

HTTP New Collection / Crear una ciudad

POST http://localhost:8080/ciudades/new/save

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

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

```

1 {
2   "nombre": "Baranquilla"
3 }

```

Body Cookies Headers (5) Test Results 201 Created 58 ms 195 B Save as example

Pretty Raw Preview Visualize Text

1 Ciudad creada exitosamente

Y así queda añadida en la tabla Ciudades:

	CODIGO	NOMBRE
1	1	Baranquilla

- RF2:**

Ahora creamos una nueva sucursal la cual esta relacionada con la ciudad anteriormente insertada, por lo cual se crean los atributos que tendrá esta sucursal, y además se pone la ciudad existente con la que se relaciona.

HTTP New Collection / Crear una sucursal

POST http://localhost:8080/sucursales/new/save

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

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

```

2 "nombre": "sucursalprueba",
3 "tamanoM2": 100,
4 "direccion": "Carrea 1 #1-2",
5 "telefono": 3841234,
6 "id_ciudad": {
7   "codigo": 1,
8   "nombre": "Baranquilla"
9 }

```

Body Cookies Headers (5) Test Results 201 Created 68 ms 197 B

Pretty Raw Preview Visualize Text

1 Sucursal creada exitosamente

Y así es como queda ya en la tabla de Sucursales en la base de datos:

	ID	NOMBRE	TAMANIOM2	DIRECCION	TELEFONO	ID_CIUADAD
1	23	sucursalprueba	100	Carrea 1 #1-2	3841234	1

- RF3:**

Ahora bien, para esta parte se necesita crear una nueva bodega que igualmente tiene una relación. Esta vez con una sucursal, entonces al crear sus atributos dentro de la tabla, también se pone la sucursal a la cual estaría relacionada.

NEW COLLECTION / Crear una bodega

POST http://localhost:8080/BODEGAS/new/save

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "nombre": "Bodega Prueba",
3   "tamaniom2": 25,
4   "id_sucursal": {
5     "id": 23,
6     "nombre": "sucursalprueba",
7     "tamaniom2": 100,
8     "direccion": "Carrea 1 #1-2",
9     "telefono": 3841234,
10    "id_ciudad": {
11      "codigo": 1,
12      "nombre": "Barranquilla"
13    }
14  }
15 }

```

201 Created • 978 ms • 196 B

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

1 Bodega creada exitosamente

En la tabla se ve la bodega creada con nombre “Bodega Prueba” de esta manera:

	ID	NOMBRE	TAMANIOM2	ID_SUCURSAL
1	1000	Bodegal	20	1
2	1	Bodega Prueba	25	23

Y para la segunda parte, se tiene que eliminar una bodega. Así que se eliminara la bodega previamente creada, la cual tiene como id, 1.

NEW COLLECTION / Borrar una bodega

GET http://localhost:8080/BODEGAS/1/delete

Params Authorization Headers (6) Body Scripts Settings

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

200 OK • 256 ms • 193 B

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

1 Bodega eliminada exitosamente

Y ahora si miramos en la tabla de la base de datos en BODEGAS, veremos que la bodega ya no estará ahí:

	ID	NOMBRE	TAMANIOM2	ID_SUCURSAL
1	1000	Bodegal	20	1

- RF4:

Acá se necesita crear y actualizar un proveedor, en el ejemplo, vamos a crear uno de nombre “empresa2”. En donde se confirma que no hubo errores al crearse con el mensaje adjunto

The screenshot shows a REST client interface for a POST request to `http://localhost:8080/proveedores/new/save`. The request body is a JSON object: `{ "nit": "1011", "nombre": "empresa2", "direccion": "23", "nombreContacto": "Lina", "telefonoContacto": "3112" }`. The response status is `200 OK` with a message: `Proveedor guardado exitosamente`. Below the response, a table displays the data:

	NIT	NOMBRE	DIRECCION	NOMBRECONTACTO	TELEFONOCONTACTO
1	1011	empresa2	23	Lina	3112

Además, es necesario poder actualizarlo, en este caso, se le hace un cambio a el nombre, el cual va a ser “empresa5”. Esta actualización se confirma con el mensaje adjunto.

The screenshot shows a REST client interface for a POST request to `http://localhost:8080/proveedores/1011/edit/save`. The request body is a JSON object: `{ "nit": "1011", "nombre": "empresa5", "direccion": "23", "nombreContacto": "Lina", "telefonoContacto": "3112" }`. The response status is `200 OK` with a message: `Proveedor actualizado exitosamente`. Below the response, a table displays the data:

	NIT	NOMBRE	DIRECCION	NOMBRECONTACTO	TELEFONOCONTACTO
1	1011	empresa5	23	Lina	3112

RF5:

En este requerimiento hay que crear una nueva Categoría. Como la categoría no contiene llaves foráneas, simplemente se ponen sus atributos propios de la tabla y se crea.

HTTP New Collection / Crear una categoria Save Share

POST ▼ `http://localhost:8080/Categorias/new/save` Send ▼

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

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

```

1 {
2   "nombre": "Enlatados",
3   "descripcion": "Preservados en recipientes herméticos ",
4   "caracteristicasAlmacenamiento": "Almacenarse en un lugar fresco y seco"
5 }

```

Body Cookies Headers (5) Test Results 201 Created 778 ms 199 B 🌐 🔍 ⋮

Pretty Raw Preview Visualize Text ▼ 🔍

1 Categoria creada exitosadamente

Ya en la tabla de categorías se guarda y quedaría así:

	CODIGO	NOMBRE	DESCRIPCION	CARACTERISTICASALMACENAMIENTO
1	3	categoriatest	descripcion producto	varias cosas
2	43	Enlatados	Preservados en recipientes herméticos	Almacenarse en un lugar fresco y seco

Ahora, también se requiere consultar la categoría por su nombre o código para obtener su información. Así que, primero tenemos la consulta de la categoría con código 1:

HTTP New Collection / Consultar categoria por codigo Save Share

GET ▼ `http://localhost:8080/Categorias/consulta?codigo=1` Send ▼

Params Authorization Headers (6) **Body** Scripts Settings Cookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	⋮	Bulk Edit
<input checked="" type="checkbox"/>	codigo	1			
	Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK 375 ms 285 B 🌐 🔍 ⋮

Pretty Raw Preview Visualize **JSON** ▼ 🔍

```

1 {
2   "codigo": 1,
3   "nombre": "categoriatest",
4   "descripcion": "descripcion producto",
5   "caracteristicasAlmacenamiento": "varias cosas"
6 }

```

También tenemos la consulta de la Categoría con nombre “Enlatados”:

HTTP New Collection / Consultar categoria por nombre

GET http://localhost:8080/Categorias/consulta?nombre=Enlatados

Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

Key	Value	Description
nombre	Enlatados	

Body Cookies Headers (5) Test Results 200 OK • 378 ms • 326 B

Pretty Raw Preview Visualize JSON

```

1 {
2   "codigo": 43,
3   "nombre": "Enlatados",
4   "descripcion": "Preservados en recipientes herméticos ",
5   "caracteristicasAlmacenamiento": "Almacenarse en un lugar fresco y seco"
6 }

```

- **RF6: Crear, leer y actualizar un producto:**

Para crear un producto es necesario tener creada la categoria y la especificación de empaquetado, en este caso los dos ya son pre-creados.

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "nombre": "Jabon",
3   "precioUnitarioVenta": 2,
4   "presentacion": "empaquetado",
5   "cantidadPresentacion": 5,
6   "unidadMedia": 2,
7   "fechaExpiracion": "01-OCT-2023",
8   "id_especificacionesEmpacado": {
9     "id": 24,
10    "volumen": 9,
11    "peso": 900,
12    "id_categoria": {
13      "codigo": 45,
14      "nombre": "categorias",
15      "descripcion": "test",
16      "caracteristicasAlmacenamiento": "Cuidado"
17    }
18  }
19 }

```

Body Cookies Headers (5) Test Results 200 OK • 151 ms • 194 B

Pretty Raw Preview Visualize Text

1 Producto guardado exitosamente

Ahí, se logra evidenciar que el producto fue correctamente creado, sin embargo, se confirma con una consulta en sql:

	CODBARRAS	NOMBRE	PRECIOUNITARIOVENTA	PRESENTACION	CANTIDADPRESENTACION	UNIDADMEDIA	FECHAEXPIRACION	ID_EXPECIFICACIONEEMPACADO	ID_CATEGORIA
1	1020	Jabon	2	empaquetado	5	2	30/09/23	24	45

Para la actualización del producto, utilizamos el creado anteriormente, sin embargo, le hacemos cambios precioUnitarioVenta y la presentacion:

HTTP Producto / Actualizar un producto

POST http://localhost:8080/productos/1020/edit/save

Params Authorization Headers (9) Body Scripts Settings

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

```

1  {
2    "codBarras":1020,
3    "nombre":"Jabon",
4    "precioUnitarioVenta":4,
5    "presentacion":"libre",
6    "cantidadPresentacio": 5,
7    "unidadMedia":2,
8    "fechaExpiracion":"01-OCT-2023",
9    "id_expecificacionesEmpacado": {
10     "id":24,
11     "volumen":9,
12     "peso":900},
13    "id_categoria": {
14     "codigo":45

```

Body Cookies Headers (5) Test Results 200 OK 94

Pretty Raw Preview Visualize Text

1 Producto actualizado exitosamente

Confirmación en SQL:

	CODBARRAS	NOMBRE	PRECIOUNITARIOVENTA	PRESENTACION	CANTIDADPRESENTACIO	UNIDADMEDIA	FECHAEXPIRACION	ID_EXPECIFICACIONEEMPACADO	ID_CATEGORIA
1	1020	Jabon	4	libre	5	2	30/09/23	24	45

Ahora para la lectura se debe presentar la información del producto y de la categoría.

Esta se puede de dos maneras:

Consulta por codBarras:

GET http://localhost:8080/productos/consulta/codBarras?codBarras=1020 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/>	codBarras	1020		
<input type="checkbox"/>	Key	Value	Description	

Body Cookies Headers (5) Test Results 200 OK 389 ms 262 B Save Response

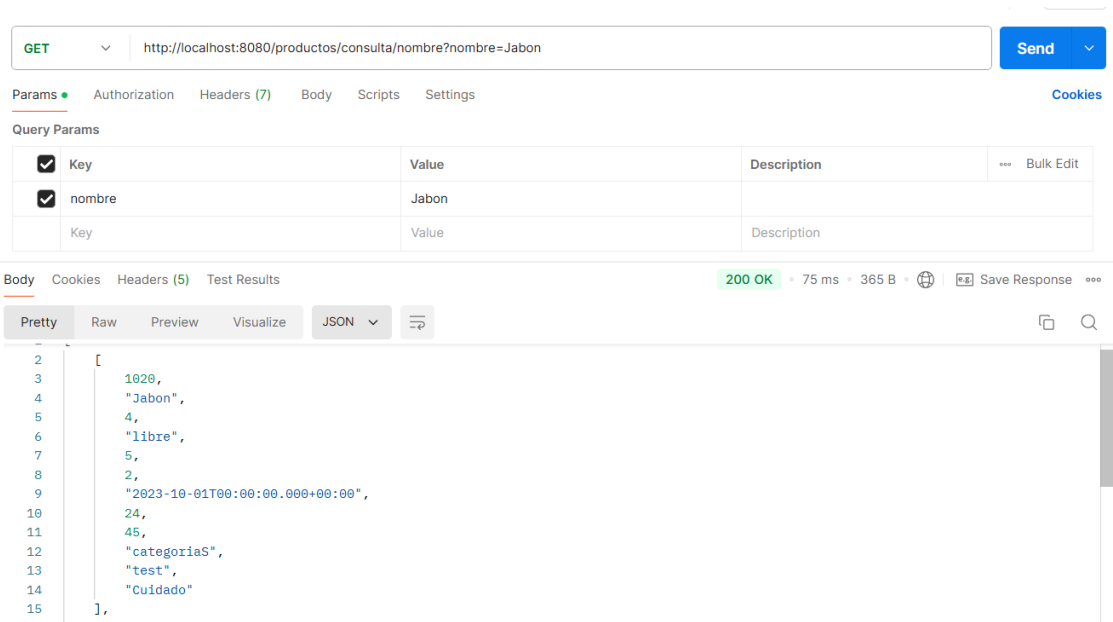
Pretty Raw Preview Visualize JSON

```

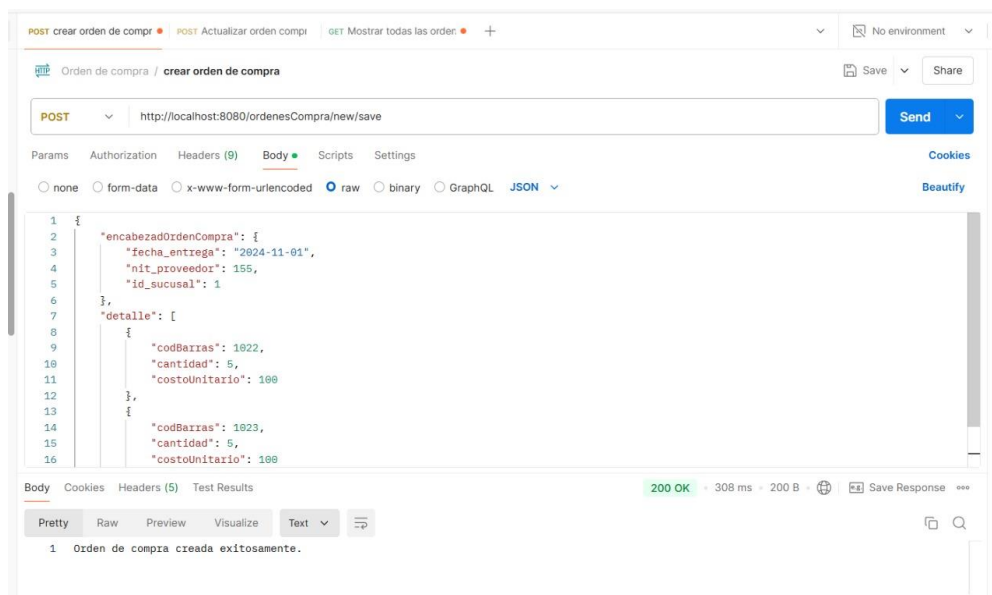
1  {
2    "codBarras":1020,
3    "nombre":"Jabon",
4    "precioUnitarioVenta":4,
5    "presentacion":"libre",
6    "cantidadPresentacio": 5,
7    "unidadMedia":2,
8    "fechaExpiracion":"2023-10-01T00:00:00.000+00:00",
9    "id_expecificacionesEmpacado": {
10     "id":24,
11     "volumen":9,
12     "peso":900},
13    "id_categoria": {
14     "codigo":45,
15     "categorias": "test",
16     "cuidado": "Cuidado"

```

Consulta por nombre:



- **RF7:** Crear una orden de compra para una sucursal, considerando la información de la orden de compra



Para la corrección del requerimiento se implementaron dos clases auxiliares en donde se establecen los atributos de las respuestas esperadas. OrdenCompraEspec y OrdenCompraHelper, son invocadas en el controller de la función para crear una orden de compra, con el fin de obtener todos los elementos del encabezado en la tabla OrdeCompra y agregar el detalle de los productos que componen la orden de compra.

- **RF8:** Actualizar una orden de compra pasando su estado de vigente a anulada:

Para este iniciamos con esta orden ya creada:

ID	FECHACREACION	ESTADO	FECHAENTREGA	ID_SUCURSAL	NIT_PROVEEDOR	
1	10001	10/09/02	vigente	10/10/02	1	20900

POST

http://localhost:8080/ordencompras/10001/edit/save

Params

Authorization

Headers (9)

Body

Scripts

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

"id": 10001

Body

Cookies

Headers (5)

Test Results

200 OK

96 ms

204 B

S

Pretty

Raw

Preview

Visualize

Text

1

Orden de compra actualizada exitosamente

Y finalmente el resultado en la consulta queda así:

ID	FECHACREACION	ESTADO	FECHAENTREGA	ID_SUCURSAL	NIT_PROVEEDOR	
1	10001	10/09/02	anulada	10/10/02	1	20900

- RF9: Mostrar todas las órdenes de compra:

[

13,

"2022-09-22T05:00:00.000+00:00",

"anulada",

"2002-09-10T05:00:00.000+00:00",

1,

20900

],

[

10001,

"2002-09-10T05:00:00.000+00:00",

"anulada",

"2002-10-10T05:00:00.000+00:00",

1,

20900

]

]

En la consulta SQL se confirma:

ID	FECHACREACION	ESTADO	FECHAENTREGA	ID_SUCURSAL	NIT_PROVEEDOR
1	13 22/09/22	anulada	10/09/02	1	20900
2	10001 10/09/02	anulada	10/10/02	1	20900

RFC1:

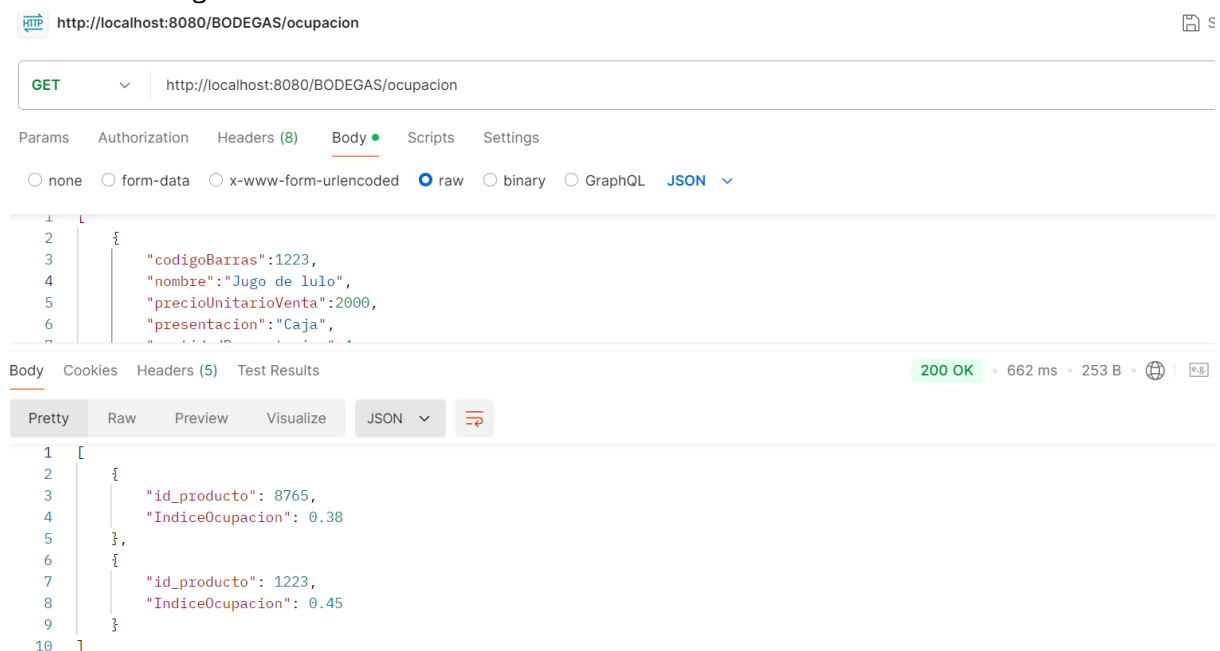
En este caso se quiere conocer el porcentaje de ocupación de un producto en una bodega por lo que la salida presenta el porcentaje de ocupación: volumen ocupado / capacidad de la bodega o estante para los productos indicados. Así que para la lista de productos siguiente

```

1  {
2    {
3      "codigoBarras":1223,
4      "nombre":"Jugo de lulo",
5      "precioUnitarioVenta":2000,
6      "presentacion":"Caja",
7      "cantidadPresentacion":4,
8      "unidadMedia":2,
9      "Date":"10-OCT-2002"
10   },
11   {
12     "codigoBarras":8765,
13     "nombre":"Kola Roman",
14     "precioUnitarioVenta":2500,
15     "presentacion":"Botella",
16     "cantidadPresentacion":1,
17     "unidadMedia":2,
18     "Date":"10-SEP-2002"
19   }

```

Se obtuvo el siguiente resultado:



HTTP http://localhost:8080/BODEGAS/ocupacion

GET http://localhost:8080/BODEGAS/ocupacion

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1  [
2    {
3      "id_producto": 8765,
4      "IndiceOcupacion": 0.38
5    },
6    {
7      "id_producto": 1223,
8      "IndiceOcupacion": 0.45
9    }
10 ]

```

Body Cookies Headers (5) Test Results 200 OK • 662 ms • 253 B

Pretty Raw Preview Visualize JSON

Mostrando que el índice de ocupación para el producto “Kola Roman” es igual a 0.38 o el 38%. Igualmente, para el producto “Jugo de lulo” se obtuvo un índice de ocupación de 0.45 o el 45%. Estos productos se ven referenciados en la tabla INFOEXTRABODEGAS de la siguiente manera:

	ID_BODEGA	ID_PRODUCTO	TOTALEXISTENCIAS	COSTOPROMEDIO	CAPACIDADALMACENAMIENTO	NIVELMINIMOREORDEN
1	1000	8765	76	2200	200	2
2	1000	1223	54	2000	120	10
3	1000	12226	54	2000	120	10

RFC2: Mostrar todos los productos que cumplen con una cierta característica

Acá el usuario puede elegir bajo cuales características basar el filtrado:

Solo una:

GET http://localhost:8080/productos/consulta/avanzadas?id_categoria=45 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Key	Value	Description
<input type="checkbox"/> precio_ma	3	
<input type="checkbox"/> id_sucursal	1	
<input checked="" type="checkbox"/> id_categoria	45	

200 OK • 173 ms • 1.16 KB Save Response

Pretty Raw Preview Visualize JSON

```

46     "presentacion": "empaquetado",
47     "cantidadPresentacio": 5,
48     "unidadMedia": 2,
49     "fechaExpiracion": "01-oct-2023",
50     "id_expecificacionesEmpacado": {
51       "id": 24,
52       "volumen": 9,
53       "peso": 900
54     },
55     "id_categoria": {
56       "codigo": 45,
57       "nombre": "categoriaS",
58       "descripcion": "test",
59       "caracteristicasAlmacenamiento": "Cuidado"

```

Postbot Ctrl Shift \

Postbot Runner Capture requests Auto-select agent Cookies Vault Trash

Todas: que en este caso por no cumplir con el de id_sucursal nos da el siguiente resultado:

GET http://localhost:8080/productos/consulta/avanzadas?fechaMenor=01-OCT-2023&fechaMayor=01-OCT-2023&precio_in=1&precio_ma=5... Send

Params Authorization Headers (7) Body Scripts Settings Cookies

<input checked="" type="checkbox"/> fechaMenor	01-OCT-2023	
<input checked="" type="checkbox"/> fechaMayor	01-OCT-2023	
<input checked="" type="checkbox"/> precio_in	1	
<input checked="" type="checkbox"/> precio_ma	5	
<input checked="" type="checkbox"/> id_sucursal	1	
<input checked="" type="checkbox"/> id_categoria	45	

404 Not Found • 136 ms • 229 B Save Response

Pretty Raw Preview Visualize Text

```

1 No se encontraron productos que cumplan con los criterios.

```

RFC 3:

En este caso se tienen que reportar la lista de productos disponibles en una bodega y para cada uno su cantidad actual, la cantidad mínima que se requiere en inventario para esa bodega y su costo promedio. En este caso se va a hacer la consulta para la bodega con código 1000:

New Collection / New Request

GET http://localhost:8080/BODEGAS/inventario Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

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

```
1 {
2   "id":1000,
3   "nombre": "Bodega1",
4   "tamanio2":20
5 }
```

Body Cookies Headers (5) Test Results 200 OK • 411 ms • 669 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "cantidad_ACTUAL": 2,
4     "cantidad_MINIMA": 10,
5     "costo_PROMEDIO": 4,
6     "id_PRODUCTO": 1020
7   },
```

Y se mostraran los productos de esa bodega, junto con los datos que pide el requerimiento:

New Collection / New Request

GET http://localhost:8080/BODEGAS/inventario Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Body Cookies Headers (5) Test Results 200 OK • 411 ms • 669 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "cantidad_ACTUAL": 2,
4     "cantidad_MINIMA": 10,
5     "costo_PROMEDIO": 4,
6     "id_PRODUCTO": 1020
7   },
8   {
9     "cantidad_ACTUAL": 2,
10    "cantidad_MINIMA": 10,
11    "costo_PROMEDIO": 4,
12    "id_PRODUCTO": 1021
13  },
14  {
15    "cantidad_ACTUAL": 300,
16    "cantidad_MINIMA": 200,
17    "costo_PROMEDIO": 500,
18    "id_PRODUCTO": 1223
19  }
20 ]
```

New Collection / New Request

GET http://localhost:8080/BODEGAS/inventario Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Body Cookies Headers (5) Test Results 200 OK • 411 ms • 669 B Save Response

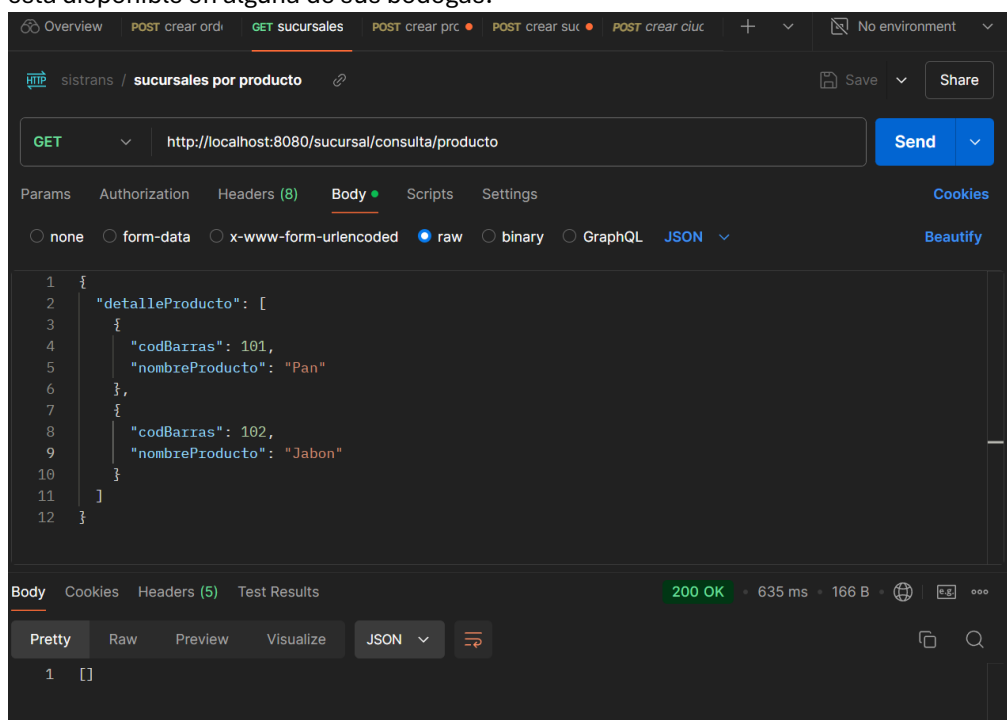
Pretty Raw Preview Visualize JSON

```
20 [
21   {
22     "cantidad_ACTUAL": 76,
23     "cantidad_MINIMA": 2,
24     "costo_PROMEDIO": 2200,
25     "id_PRODUCTO": 8765
26   },
27   {
28     "cantidad_ACTUAL": 54,
29     "cantidad_MINIMA": 10,
30     "costo_PROMEDIO": 2000,
31     "id_PRODUCTO": 12226
32   },
33   {
34     "cantidad_ACTUAL": 54,
35     "cantidad_MINIMA": 10,
36     "costo_PROMEDIO": 2000,
37     "id_PRODUCTO": 12227
38   }
39 ]
```

Como se puede observar se mostró la información requerida para 6 productos, los cuales son los que están asociados a la bodega requerida en la base de datos y se puede observar en la tabla de InfoExtraBodegas como se ve a continuación:

	ID_BODEGA	ID_PRODUCTO	TOTALEXISTENCIAS	COSTOPROMEDIO	CAPACIDADALMACENAMIENTO	NIVELMINIMOREORDEN
1	1000	8765	76	2200	200	2
2	1000	1223	300	500	1000	200
3	1000	1021	2	4	500	10
4	1000	1020	2	4	500	10
5	1000	12226	54	2000	120	10
6	1000	12227	54	2000	200	10

RCF 4: Mostrar las sucursales en las que hay disponibilidad de un producto. Dado el identificador o el nombre de un producto, se debe mostrar una lista de todas las sucursales en las cuales dicho producto está disponible en alguna de sus bodegas.



Para la corrección de este requerimiento se implementaron las clases SucursalProductoEspecc y SucursalProductoHelper, las cuales son invocadas en el controller para obtener los valores que se deben mostrar en la tabla

RFC 5: Mostrar todos los productos que requieren una orden de compra con la información explícita de bodega, sucursal y proveedor.

En este caso solo 2 tienen la cantidad de existencias menor a la cantidad mínima de reorden

GET ▼ http://localhost:8080/productos/consulta/necesitaordenCompra

Params Authorization Headers (7) Body Scripts Settings

Query Params

Key	Value
-----	-------

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ▼ ≡

```

1  [
2    [
3      1020,
4      "Jabon",
5      "Bodega1",
6      "Sucursal",
7      null,
8      2
9    ],
10   [
11     1021,
12     "Jabon",
13     "Bodega1",
14     "Sucursal",
15     null,
16     2
17   ]
18 ]

```

Confirmación con consulta en SQL:

Todas las Filas Recuperadas: 6 en 0,022 segundos						
	ID_BODEGA	ID_PRODUCTO	TOTALEXISTENCIAS	COSTOPROMEDIO	CAPACIDADALMACENAMIENTO	NIVELMINIMOREORDEN
1	1000	8765	76	2200	200	2
2	1000	1223	300	500	1000	200
3	1000	1021	2	4	500	10
4	1000	1020	2	4	500	10
5	1000	12226	54	2000	120	10
6	1000	12227	54	2000	200	10

Nuevos Requerimientos:

RF10:

Se registró el ingreso de productos a la bodega, en este caso de prueba para la Orden de Compra con id: 10004 y la bodega con id: 1000 se actualizaron los datos correspondientes en la base de datos para mostrar que esta orden quedó entregada, a continuación, se muestra el encabezado, junto con el detalle correspondiente de la Orden de Compra luego de hacer la transacción completa:

HTTP New Collection / New Request Save Share

POST ▼ Send ▼

Params ● Authorization Headers (7) Body Scripts Settings Cookies

Query Params

<input type="checkbox"/>	Key	Value	Description	...	Bulk Edit
--------------------------	-----	-------	-------------	-----	-----------

Body Cookies Headers (5) Test Results 200 OK 1134 ms 390 B 🌐 📄 ⋮

Pretty Raw Preview Visualize JSON ▼ 🔍

```

1  [
2    {
3      "Sucursal": "Sucursal",
4      "Bodega": "1000",
5      "Proveedor": "empresa2",
6      "Fecha Ingreso": "2024-10-31 19:00:00.0"
7    },
8    {
9      "precioUnitario": "100",
10     "id_producto": "1022",
11     "cantidad": "5"
12   },
13   {
14     "precioUnitario": "100",
15     "id_producto": "1023",

```

Así que, sobre la base de datos se creó la recepción correspondiente a esta orden de compra:

	ID	FECHARECEPCION	ID_BODEGA	ID_ORDENCOMPRA
1	42	31-OCT-24	1000	10004
2	23	03-NOV-24	1000	2000

Y además se actualizaron los niveles de inventario para cada producto dentro de la bodega correspondiente, como se puede observar a continuación para los productos con id 1022 y 1023:

	ID_BODEGA	ID_PRODUCTO	TOTALEXISTENCIAS	COSTOPROMEDIO	CAPACIDADALMACENAMIENTO	NIVELMINIMOREORDEN
1	1000	1042	76	2200	200	2
2	1000	1043	300	500	1000	200
3	1000	1021	2	4	500	10
4	1000	1020	2	4	500	10
5	1000	1022	25	3660	200	5
6	1000	1023	25	3660	200	5

Por último, se cambió el estado de la Orden de compra a 'ENTREGADA', como se observa al final de la tabla:

	ID	FECHACREACION	ESTADO	FECHAENTREGA	ID_SUCURSAL	NIT_PROVEEDOR
12	27	03-NOV-24	VIGENTE	31-DEC-23	1	155
13	28	03-NOV-24	VIGENTE	31-DEC-24	1	155
14	13	22-SEP-22	anulada	10-SEP-02	1	20900
15	24	03-NOV-24	VIGENTE	31-DEC-24	1	155
16	25	03-NOV-24	VIGENTE	31-DEC-23	1	155
17	10006	03-NOV-24	VIGENTE	31-OCT-24	1	155
18	15	03-NOV-24	VIGENTE	31-OCT-24	1	155
19	16	03-NOV-24	VIGENTE	31-OCT-24	1	155
20	17	03-NOV-24	VIGENTE	31-OCT-24	1	155
21	18	03-NOV-24	VIGENTE	31-OCT-24	1	155
22	19	03-NOV-24	VIGENTE	31-OCT-24	1	155
23	22	03-NOV-24	VIGENTE	31-OCT-24	1	155
24	23	03-NOV-24	VIGENTE	31-DEC-24	1	155
25	10002	03-NOV-24	VIGENTE	31-OCT-24	1	155
26	10003	03-NOV-24	VIGENTE	31-OCT-24	1	155
27	10004	03-NOV-24	ENTREGADA	31-OCT-24	1	155

RFC 6:

Mostrar los documentos de recepción de productos para una sucursal y una bodega en específico con un aislamiento serializable:

Orden de compra / ne Save Share

GET http://localhost:8080/recepcionproductos/serializable/1/1000?id_sucursal=1&id_bodega=1000 Send

Params • Authorization Headers (7) Body Scripts Settings Cookies

Query Params

<input checked="" type="checkbox"/> Key	Value	Description	Bulk Edit
<input checked="" type="checkbox"/> id_sucursal	1		
<input checked="" type="checkbox"/> id_bodega	1000		

Body Cookies Headers (5) Test Results 200 OK 335 ms 451 B Save Response

Pretty Raw Preview Visualize JSON more actions

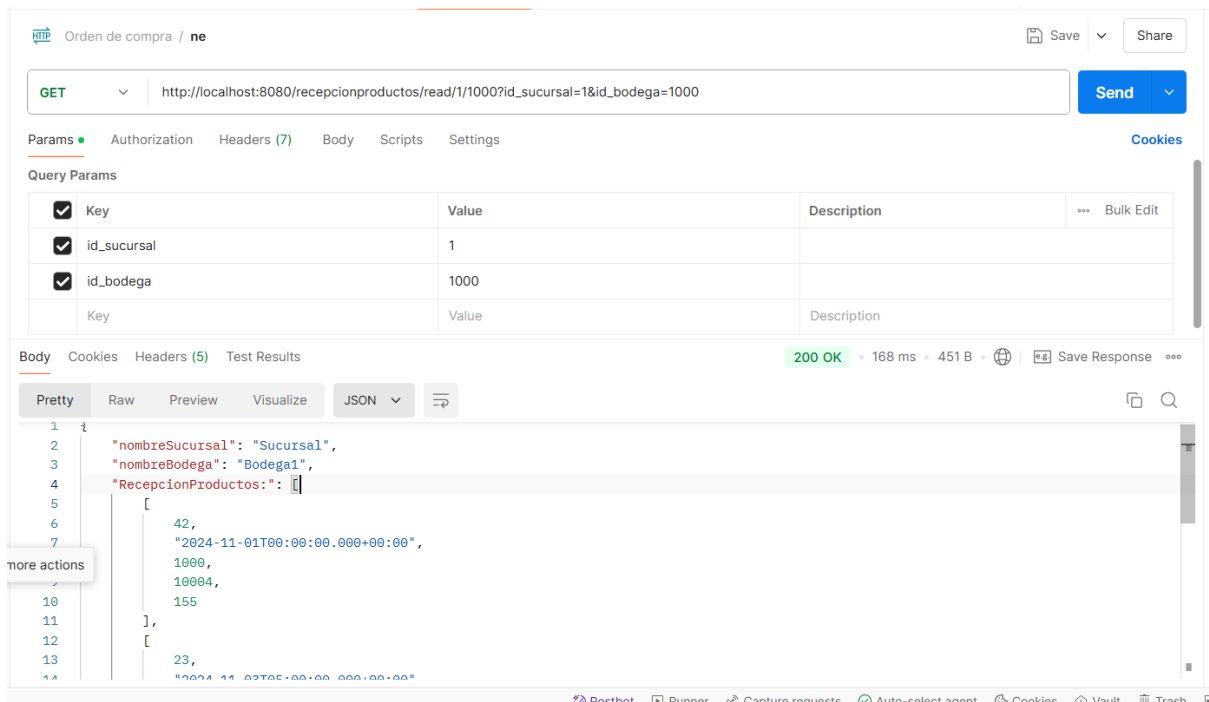
```

1 {
2   "nombreSucursal": "Sucursal",
3   "nombreBodega": "Bodega1",
4   "RecepcionProductos": [
5     [
6       42,
7       "2024-11-01T00:00:00.000+00:00",
8       1000,
9       10004,
10      155
11    ],
12    [
13      23,
14      "2024-11-03T05:00:00.000+00:00",
15      1000,

```

RFC7:

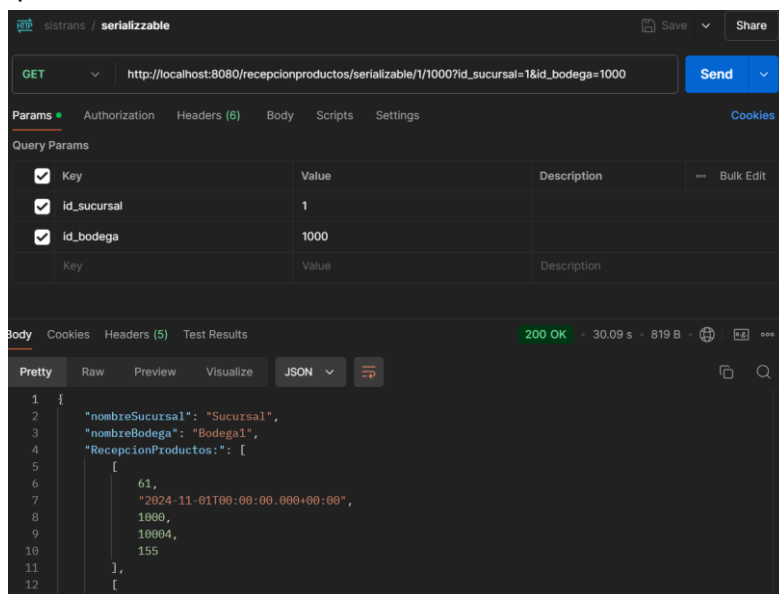
Mostrar los documentos de recepción de productos para una sucursal y una bodega en específico con un aislamiento read_committed:

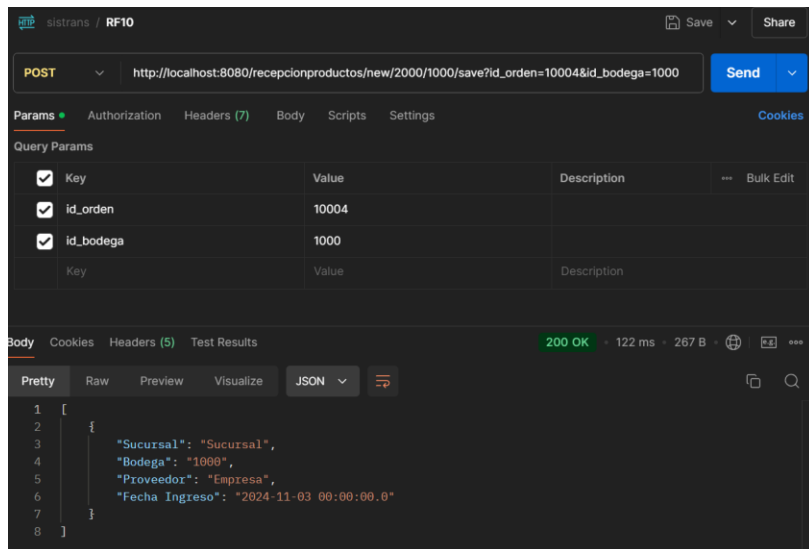


Escenario Concurrencia 1:

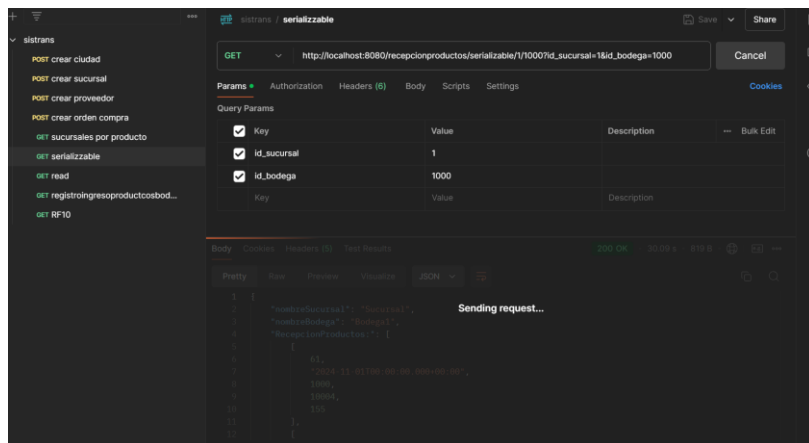
- Los pasos para la ejecución concurrente de RFC6 y RF10 a través de la línea de tiempo:

El escenario de prueba comienza dando los datos a Postman de ambos requerimientos



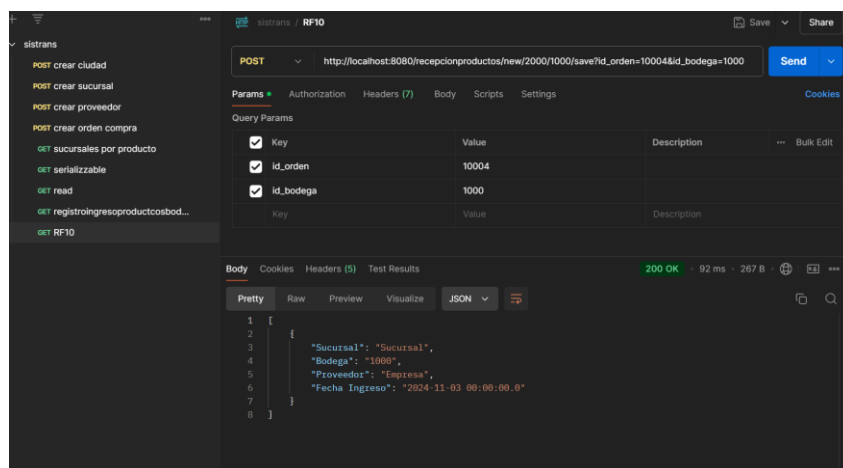


Después se inicia la ejecución del Serializable y aparece en postman lo siguiente:



Se demora 30 segundos en ejecutarse debido a los requerimientos de la implementación

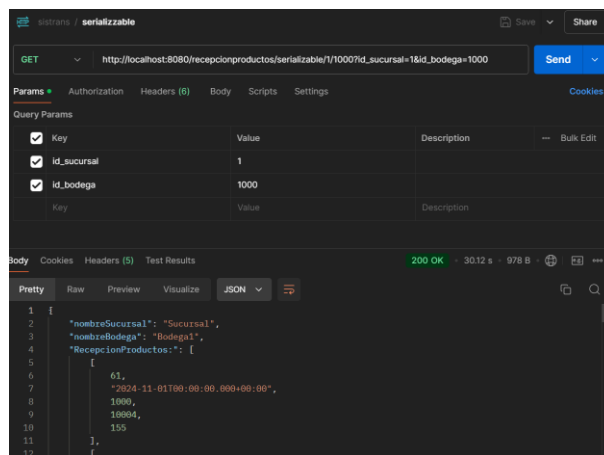
Antes de que los 30 segundos terminen, se inicia la ejecución de RF10, la cual arroja los siguientes resultados en el instante en que se ejecuta



- ¿Acaso el componente que implementa RF10 debió esperar a que terminara la ejecución de la consulta RFC6 para poder registrar el ingreso de productos?

En la ejecución del escenario de prueba Se implementó primero RFC6 Serializable y antes de que su ejecución terminara se ejecutó RF10. La segunda ejecución arrojó su resultado antes que la primera, por lo que no se vio afectada por los resultados de RFC6. De esta manera, se registraron los productos en bodega de manera correcta sin necesidad de que RFC6 terminara su ejecución.

- El resultado presentado por RFC6: presente el resultado de esta consulta. Diga si allí apareció el documento de Ingreso de producto realizado al ejecutar el componente que implementa RF10 de manera simultánea.

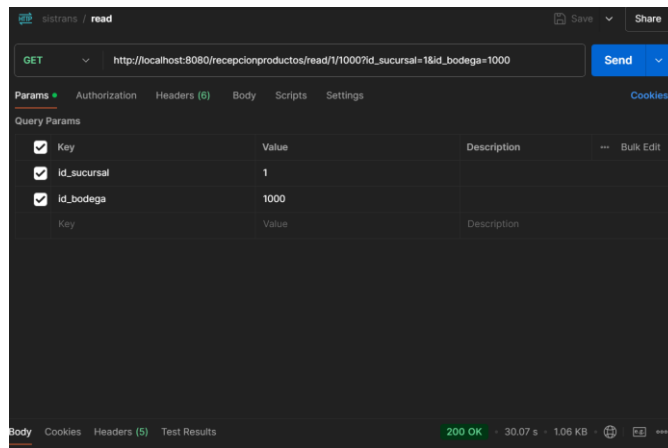


Al terminar su ejecución el componente RFC6 no incluyó los nuevos registros. Por lo que, es posible afirmar que al ejecutar de manera serializable el componente RFC6 y RF10, RFC6 termina su ejecución mucho después de RF10 y el ingreso de los nuevos productos no se hace de manera correcta.

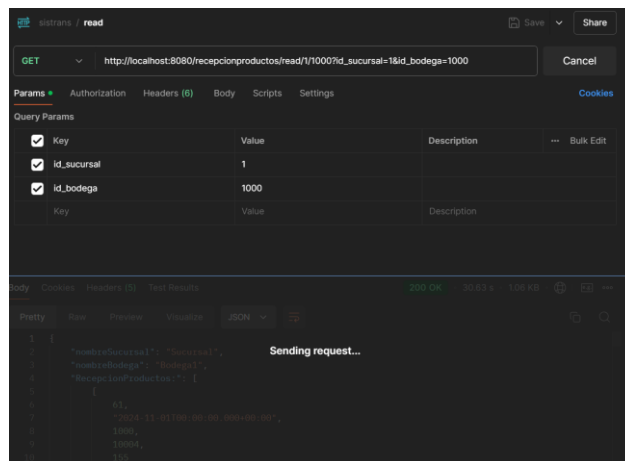
Escenario de Concurrency 2:

- Los pasos para la ejecución concurrente de RFC7 y RF10 a través de la línea de tiempo

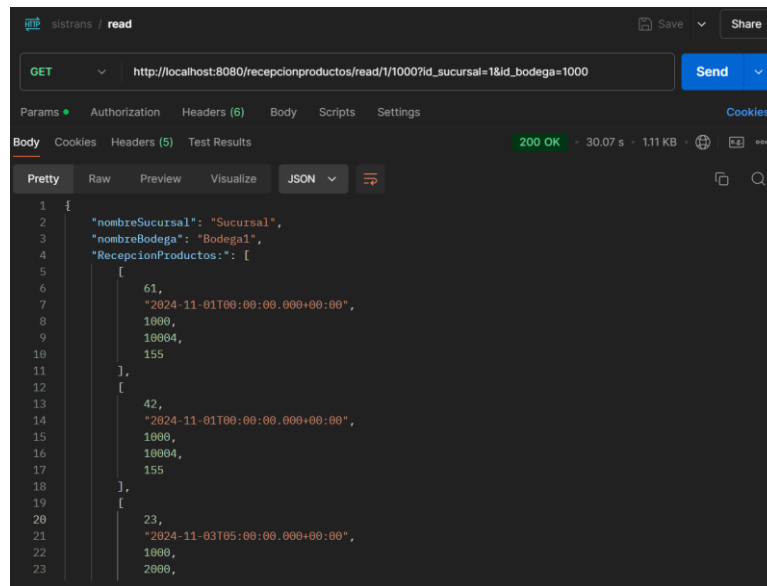
El escenario comienza generando el requerimiento en postman de RFC7 (read). Rf10 ya se generó con el escenario de prueba 1.



Después se inicia la ejecución de RFC7 (read), la cual dada las especificaciones de los requerimientos se demora 30 segundos en ejecutarse.



Antes de que los 30 segundos terminen se ejecuta RF10, cuya respuesta es inmediata. De igual manera, el requerimiento RFC7 termina antes su ejecución. Por lo que ambos procesos terminan al tiempo y arrojan los siguientes resultados.



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

- Method:** GET
- URL:** `http://localhost:8080/recepcionproductos/read/1/1000?id_sucursal=1&id_bodega=1000`
- Status:** 200 OK
- Response Body (JSON):**

```
{
  "nombreSucursal": "Sucursal",
  "nombreBodega": "Bodega1",
  "RecepcionProductos": [
    [
      61,
      "2024-11-01T00:00:00.000+00:00",
      1000,
      10004,
      155
    ],
    [
      42,
      "2024-11-01T00:00:00.000+00:00",
      1000,
      10004,
      155
    ],
    [
      23,
      "2024-11-03T05:00:00.000+00:00",
      1000,
      2000,
      2000
    ]
  ]
}
```

¿Acaso el componente que implementa RF10 debió esperar a que terminara la ejecución de la consulta RFC7 para poder registrar el ingreso de productos?

En este escenario la consulta RFC7 terminó antes de lo preestablecido, al tiempo con la respuesta de la consulta RF10, por lo que fue posible que se guardaran correctamente los nuevos documentos de ingreso.