**Sistemas transaccionales**

**Proyecto grupo 6**

**Documentación con imágenes**

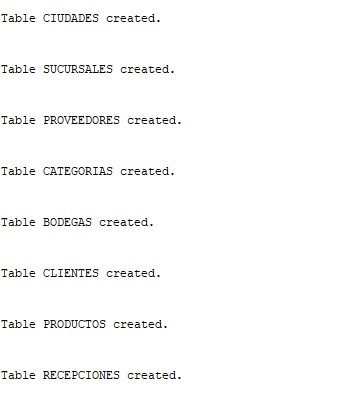
**Integrantes:**

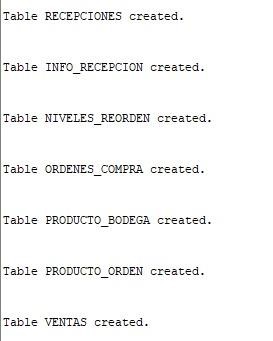
* María Lucia Benavides
* Ángela Jiménez
* Kevin Castillo

**Notas:**

Los cambios hechos para que funcionaran todas las pruebas de la entrega 1 eran más que todo de sintaxis y de conexión con bases de datos, además se crearon pruebas para los requerimientos de consulta de la entrega 1 y quedaron funcionando perfectamente.

* 1. **Creación de tablas:**

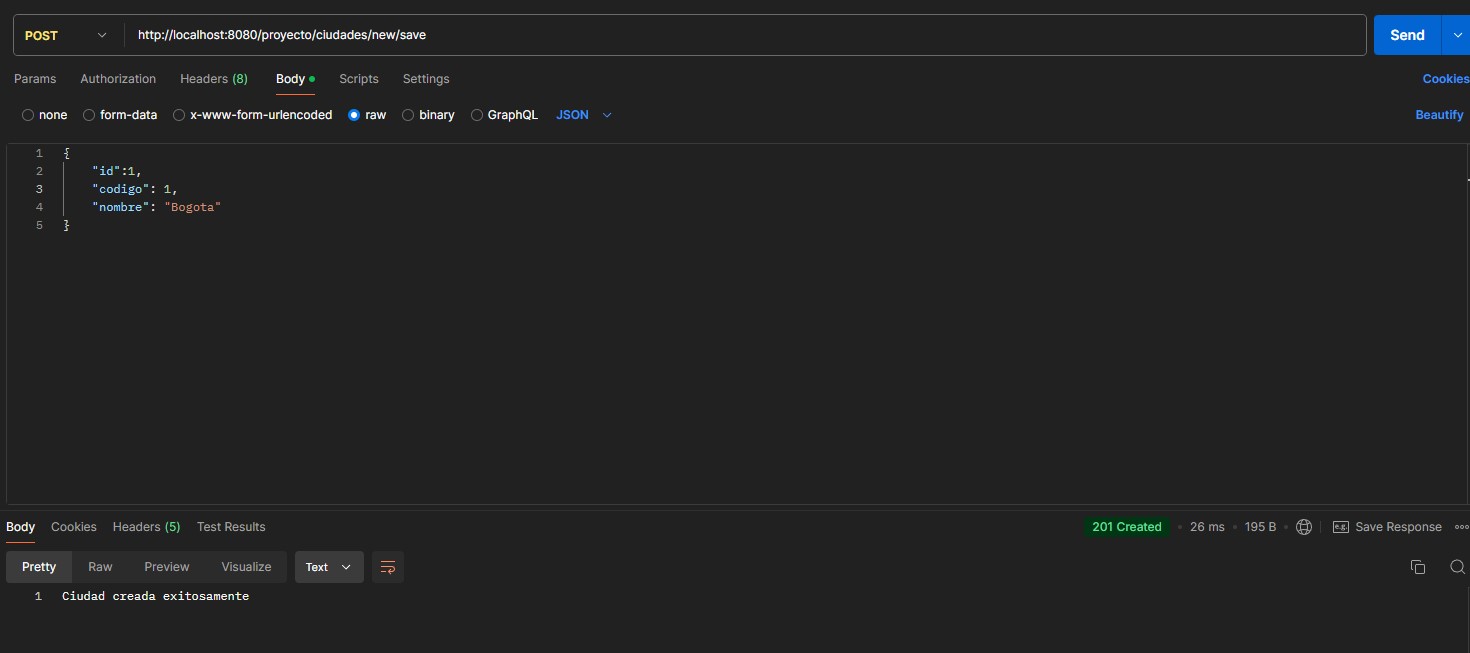




* 1. **Pruebas en PostMan de requerimientos:**

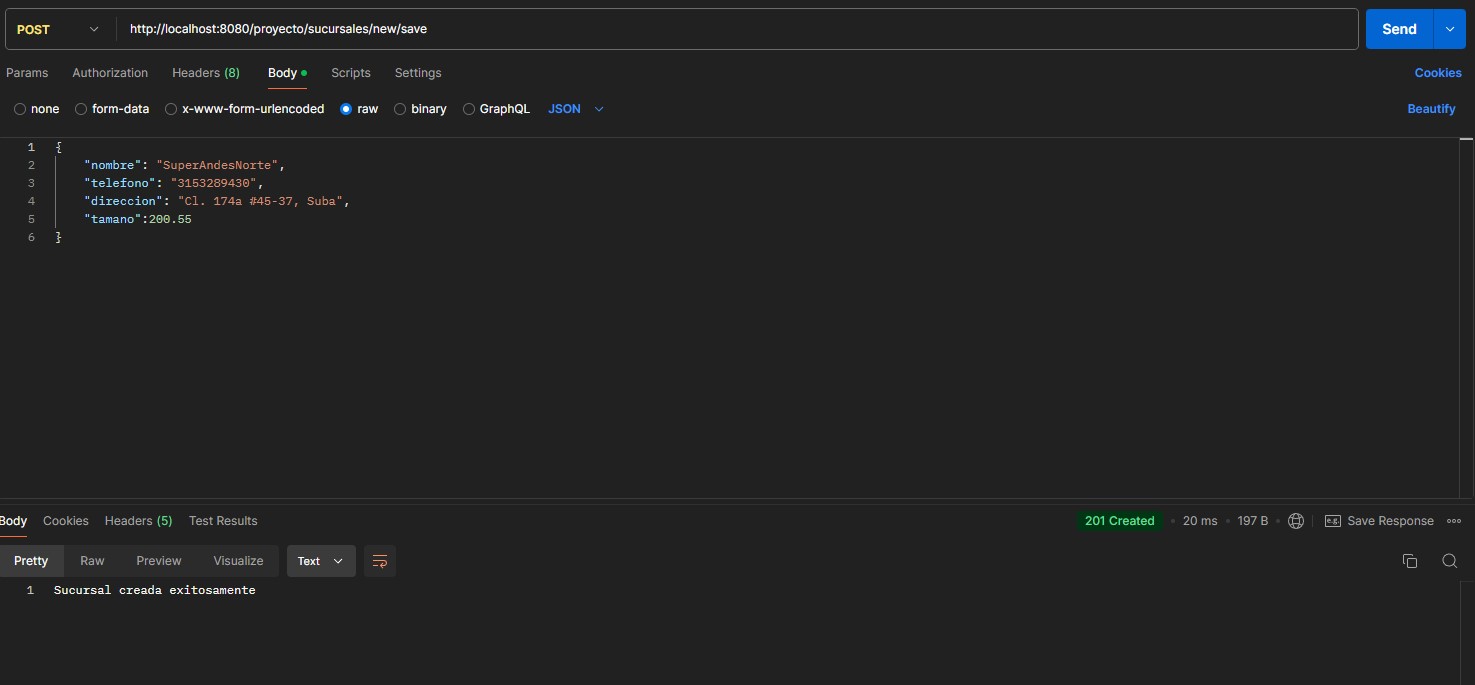
**Requerimientos funcionales:**

**-Requerimiento funcional 1: Crear una ciudad:**



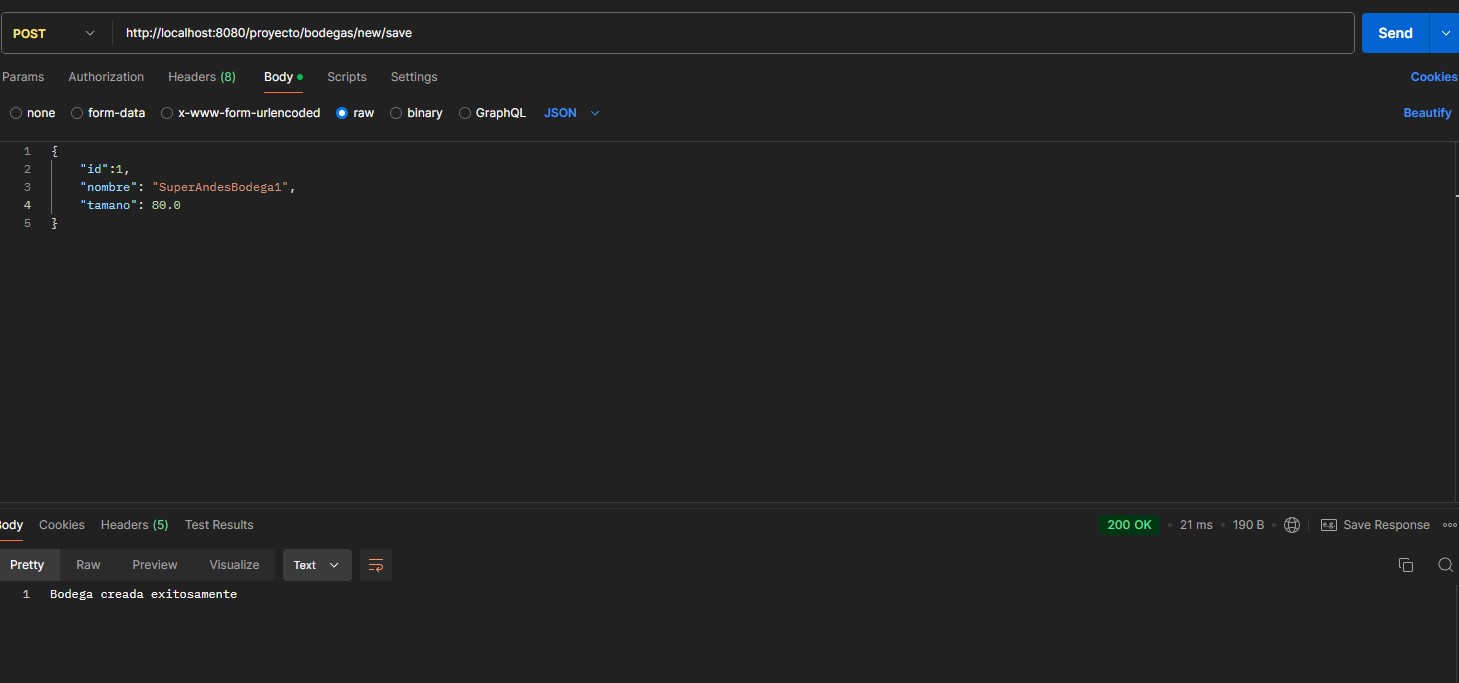
**-Requerimiento funcional 2:**

**Crear una sucursal:**

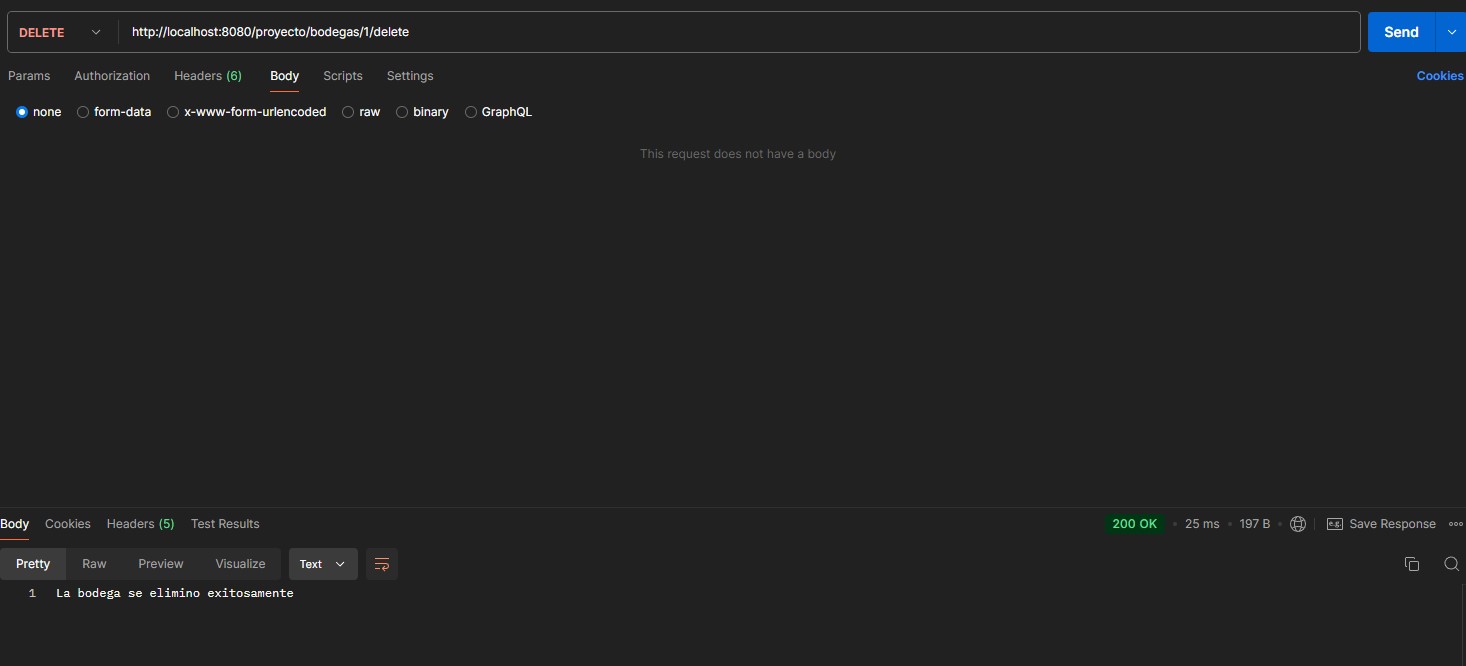


**-Requerimiento funcional 3:**

**Crear y borrar una bodega: Crear:**

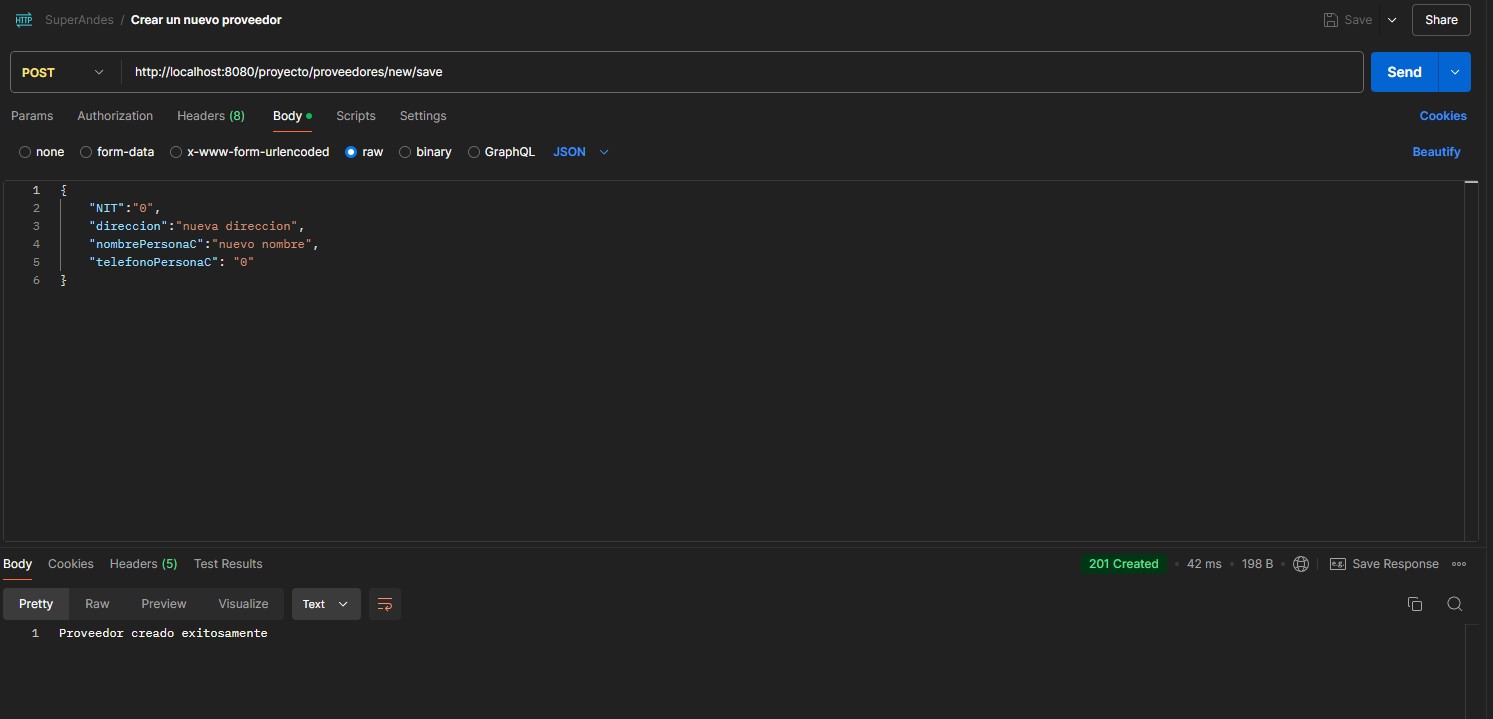


**Borrar:**

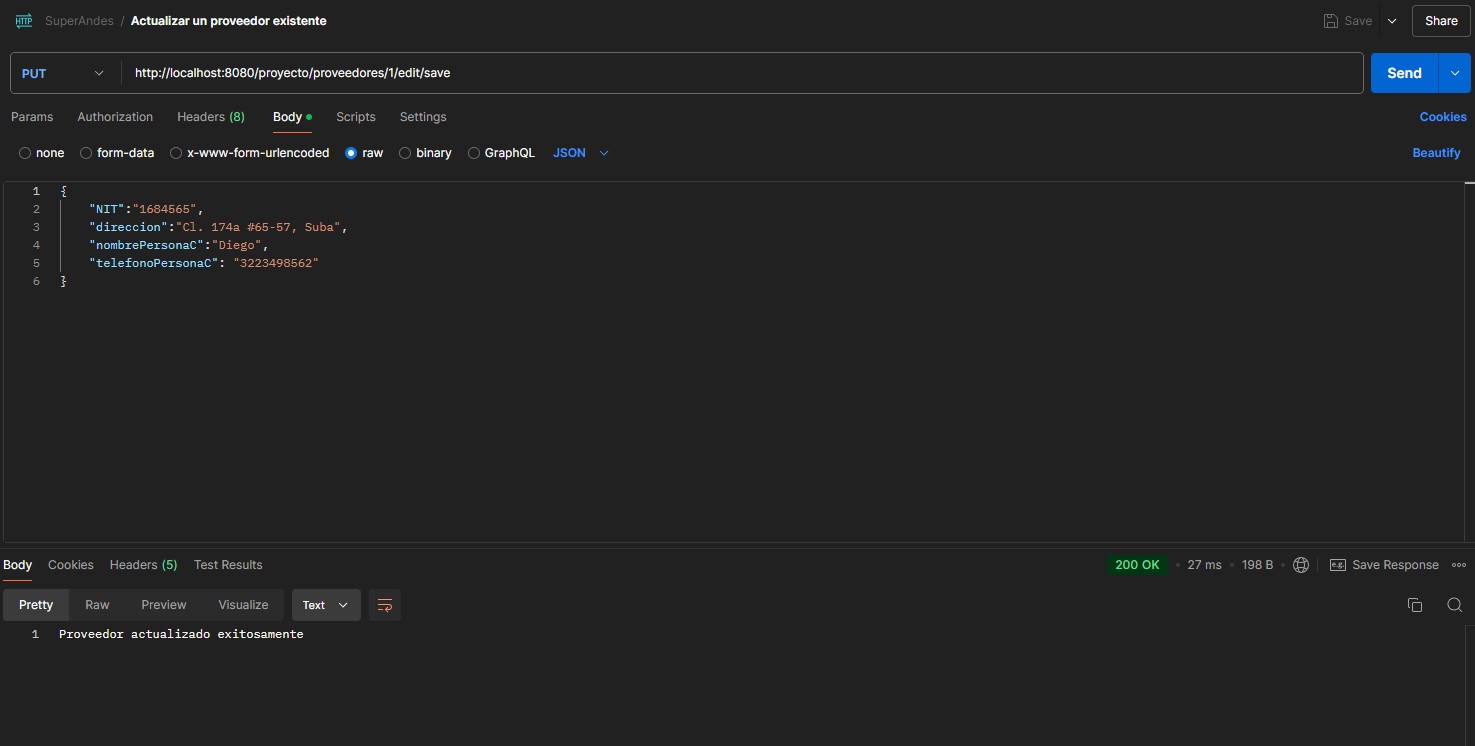


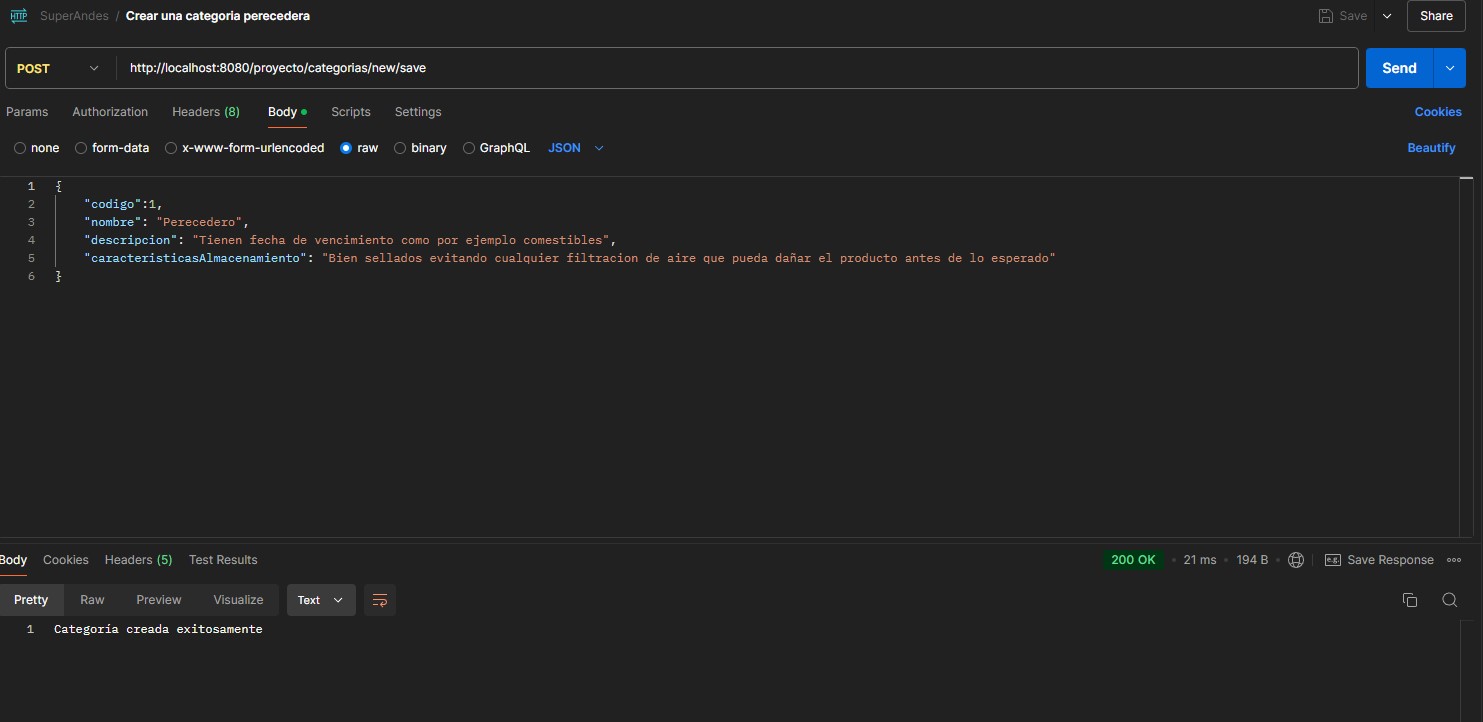
**-Requerimiento funcional 4:**

**Crear y Actualizar Proveedores: Crear:**



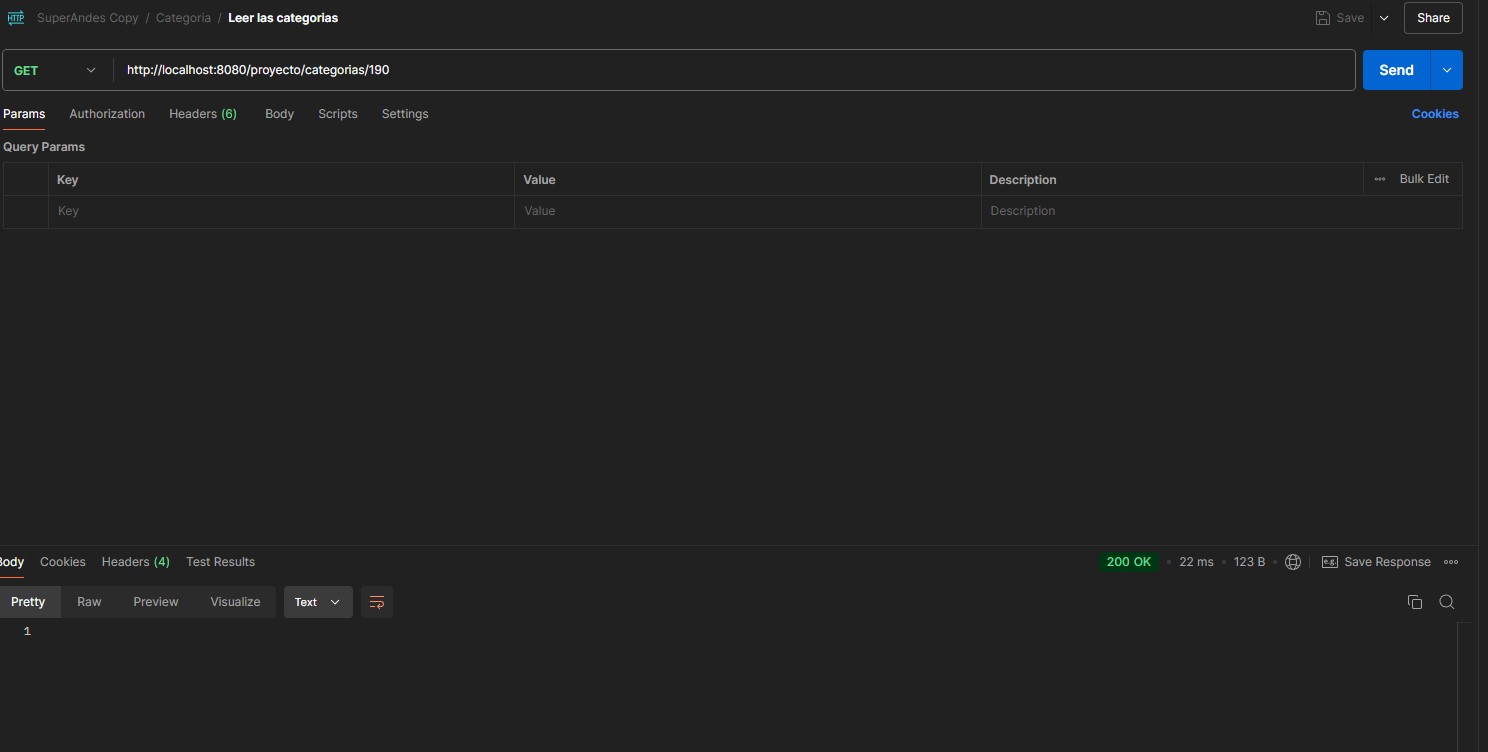
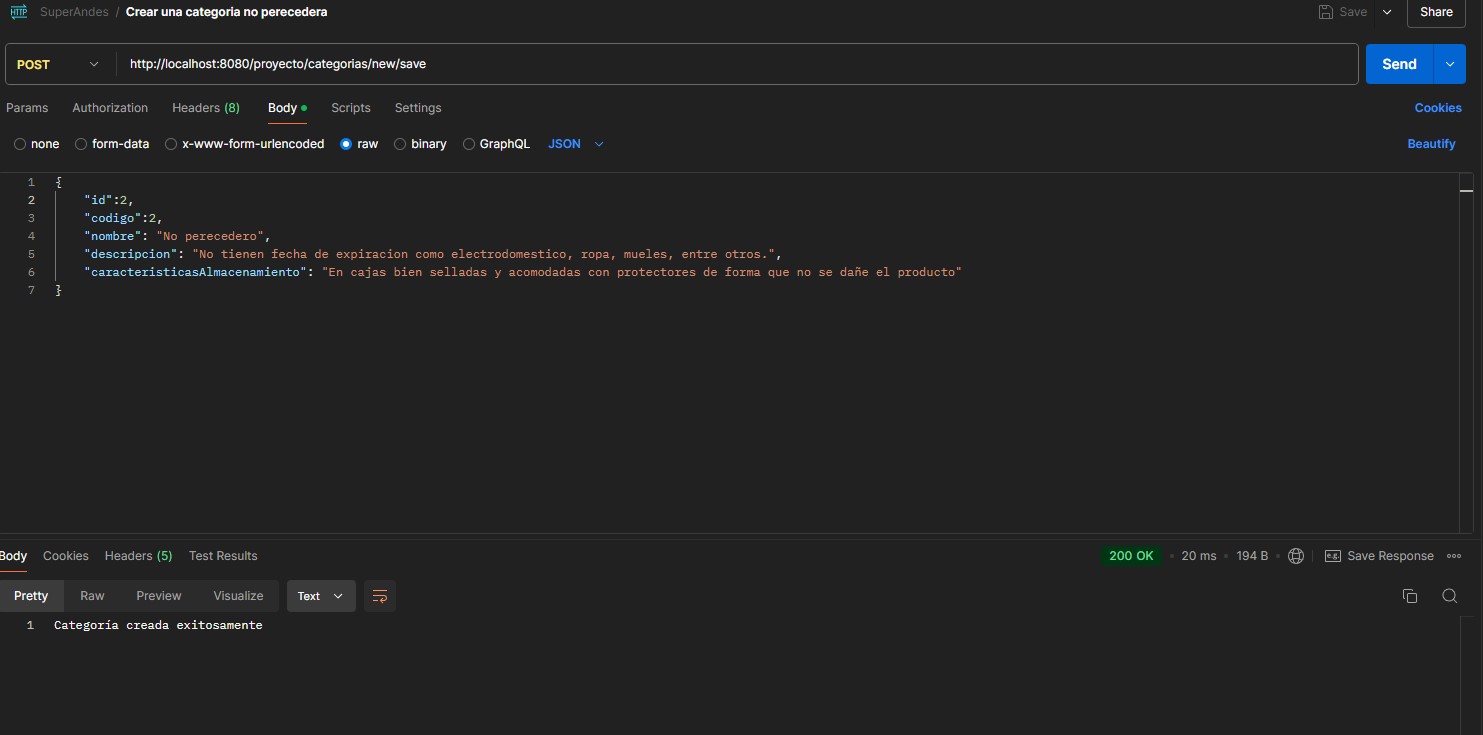
**Actualizar:**



**-Requerimiento funcional 5:**

**Crear y leer una categoría de producto:**

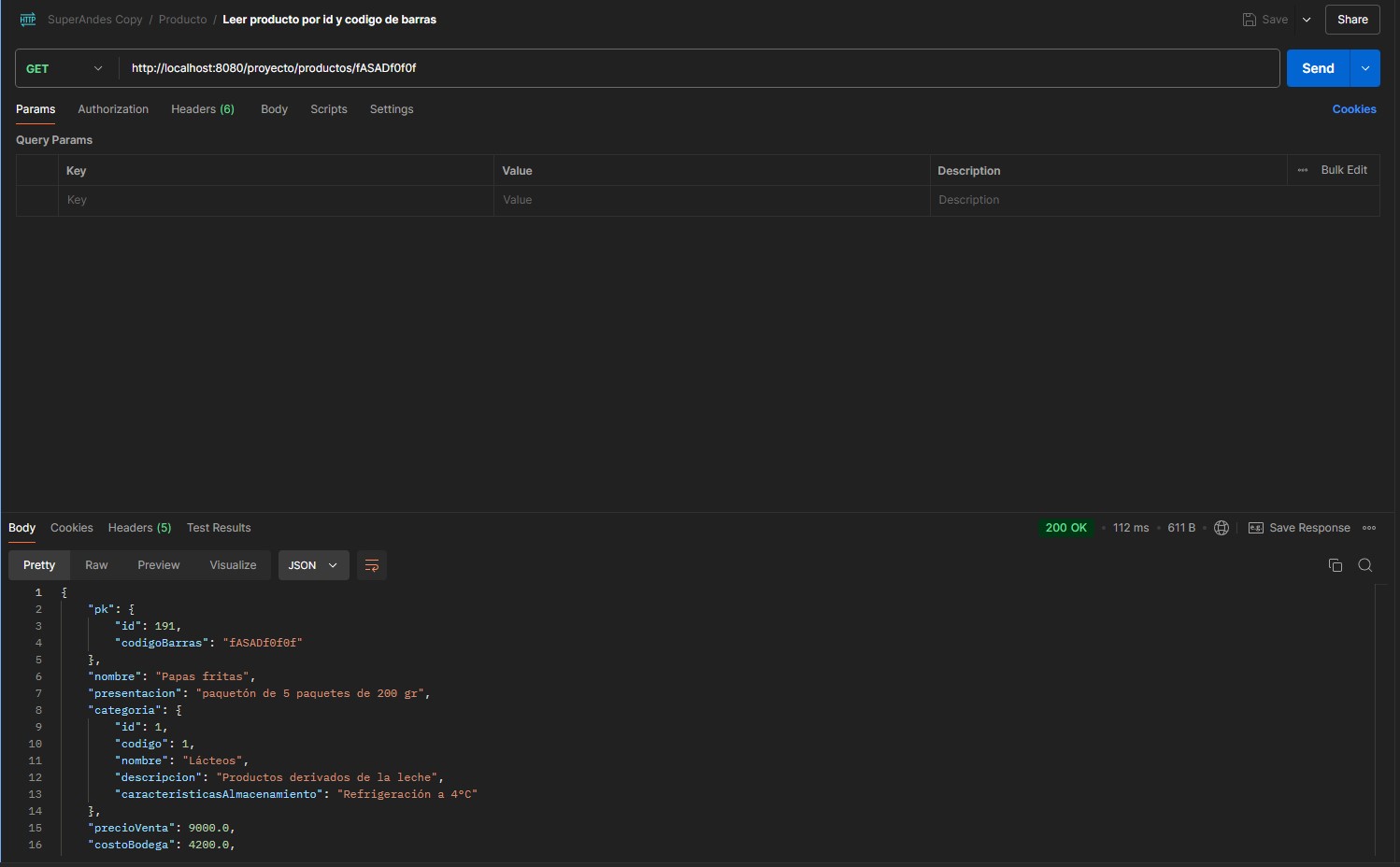
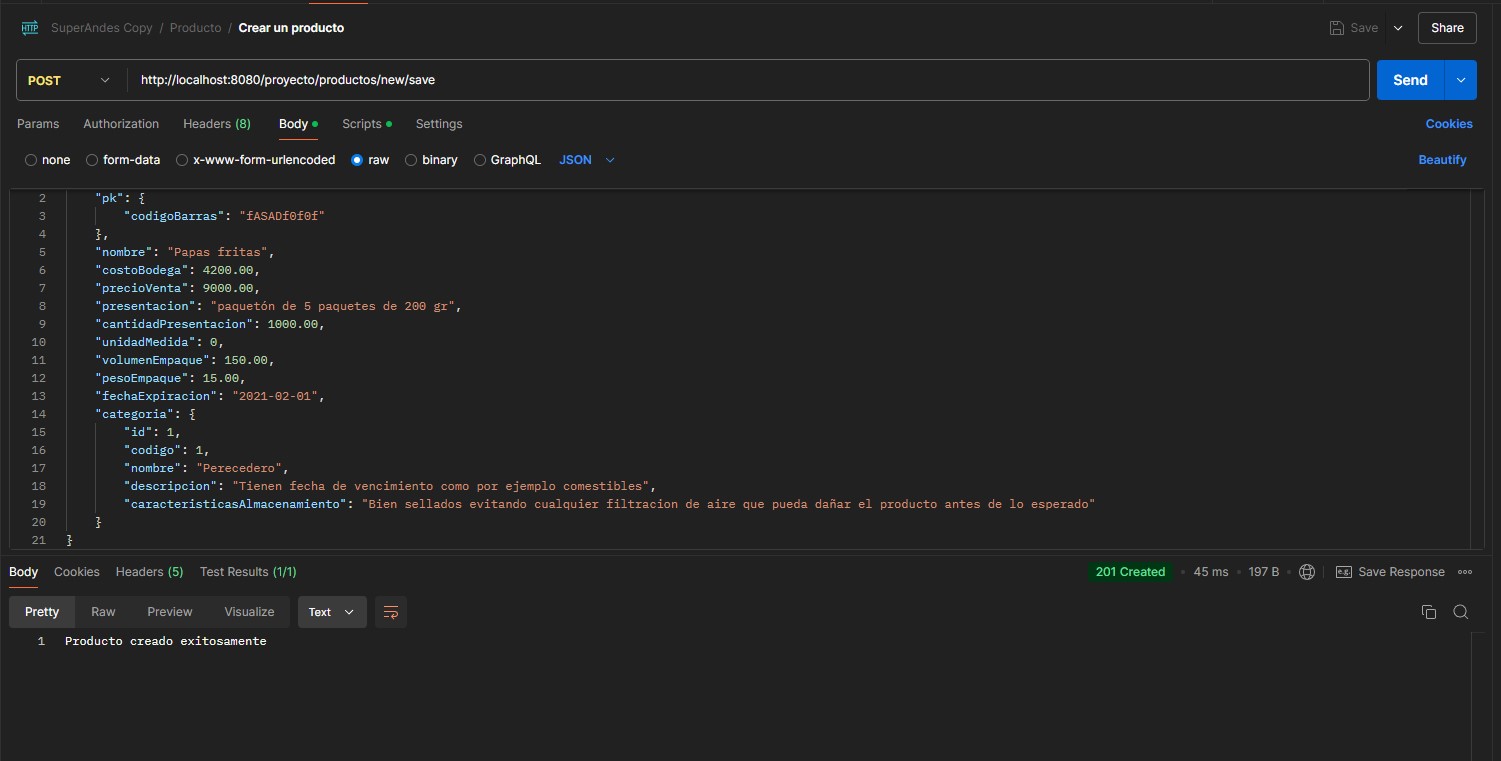
**Crear:**

**Leer:**

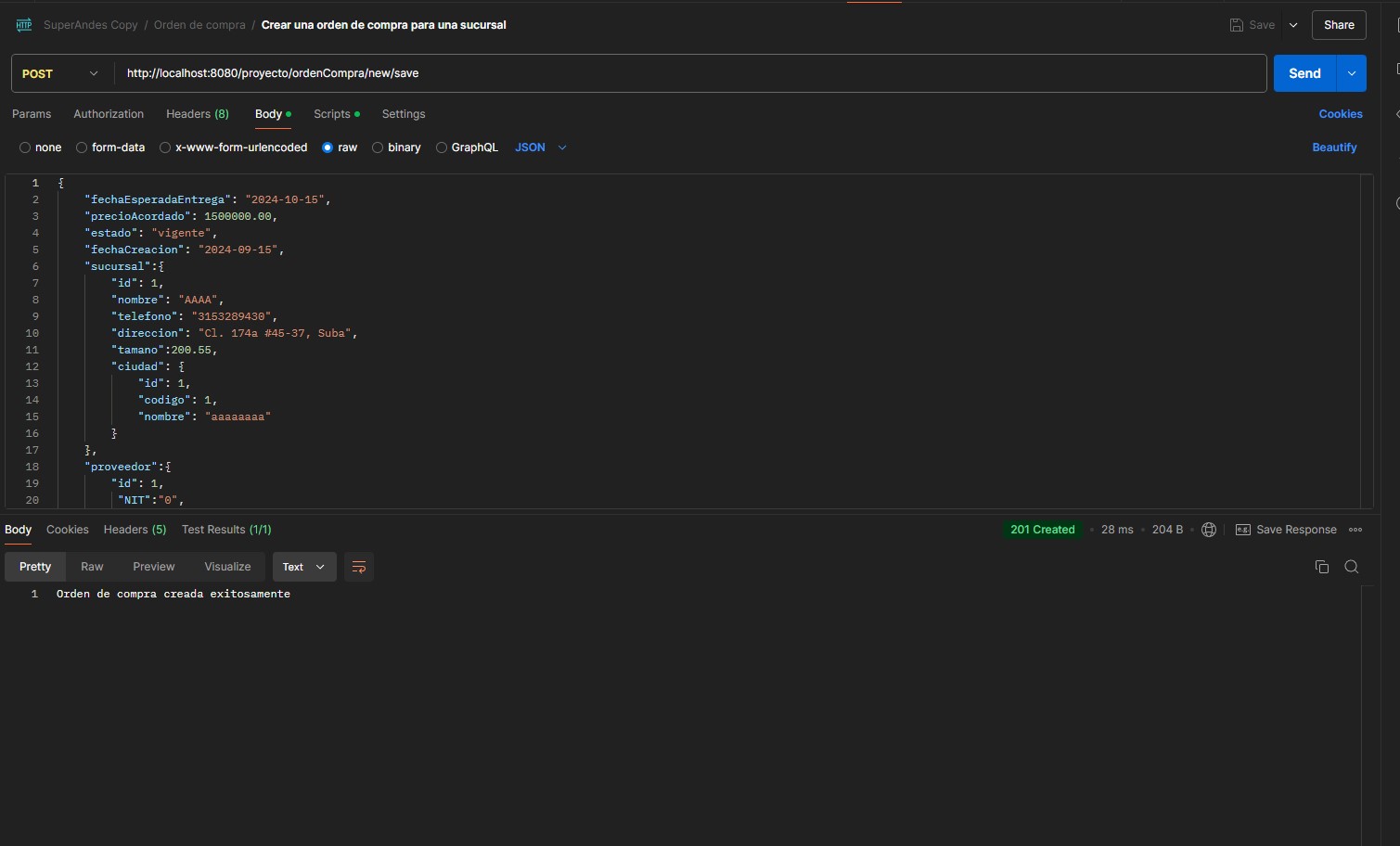
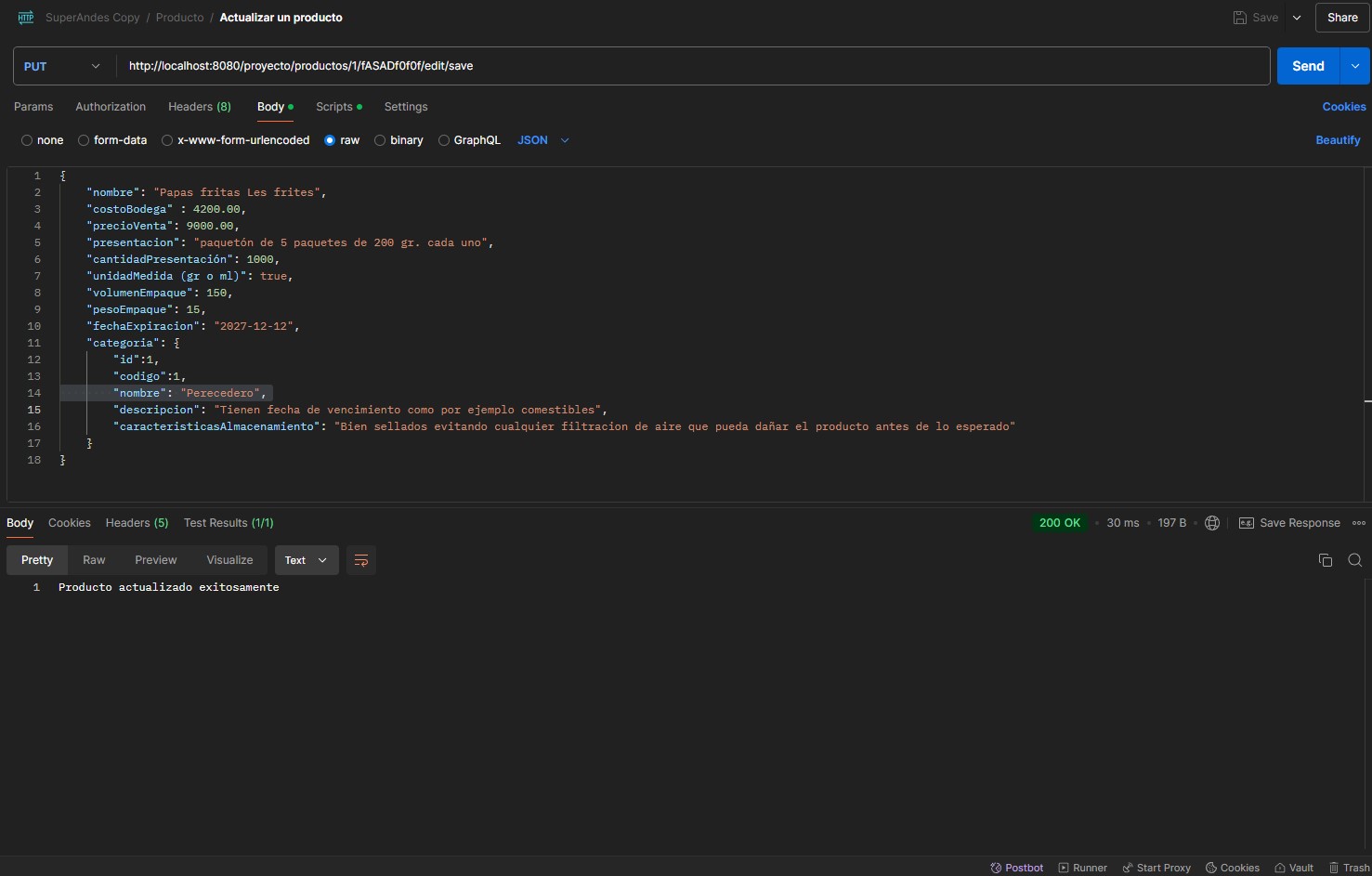
**-Requerimiento funcional 6:**

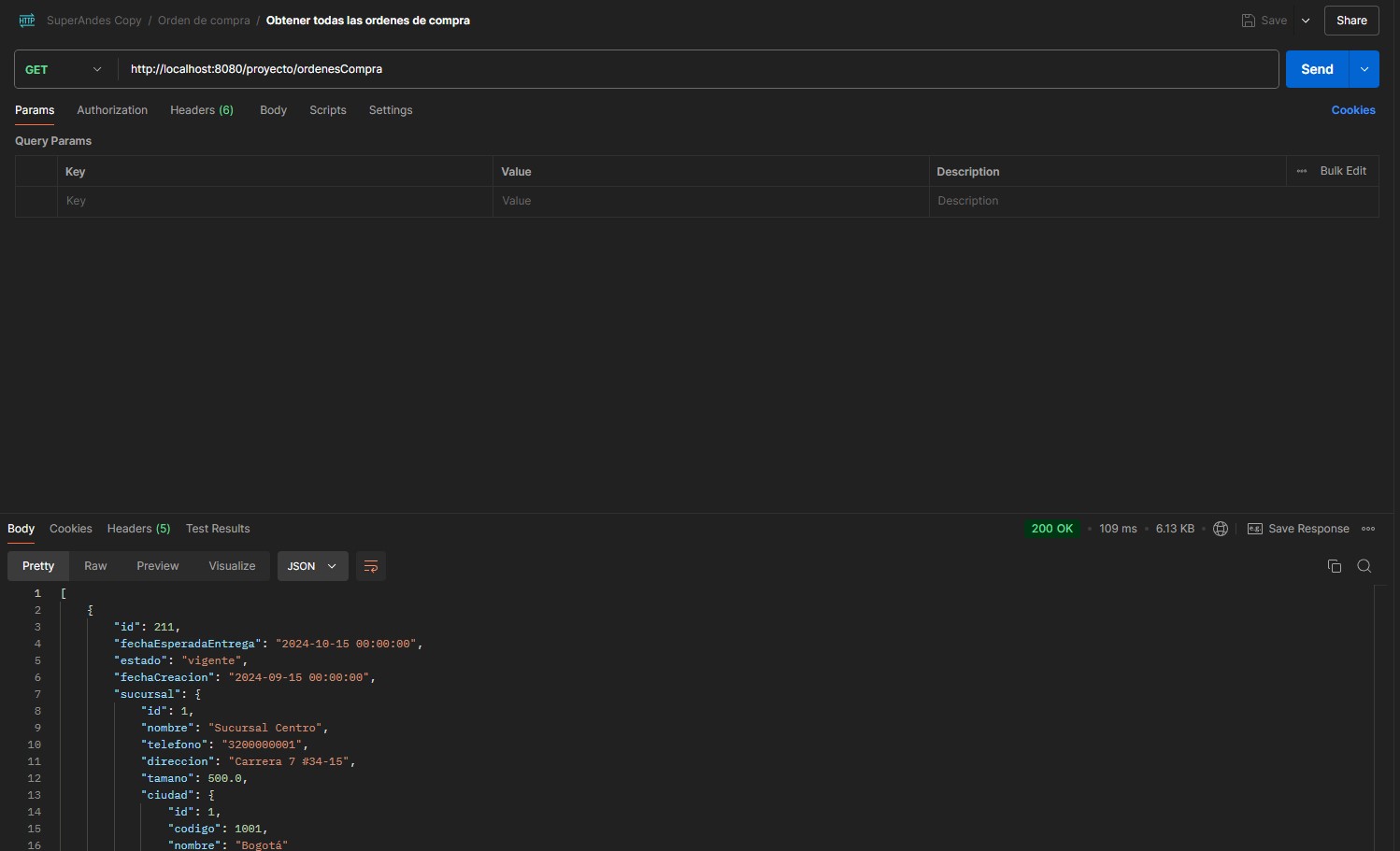
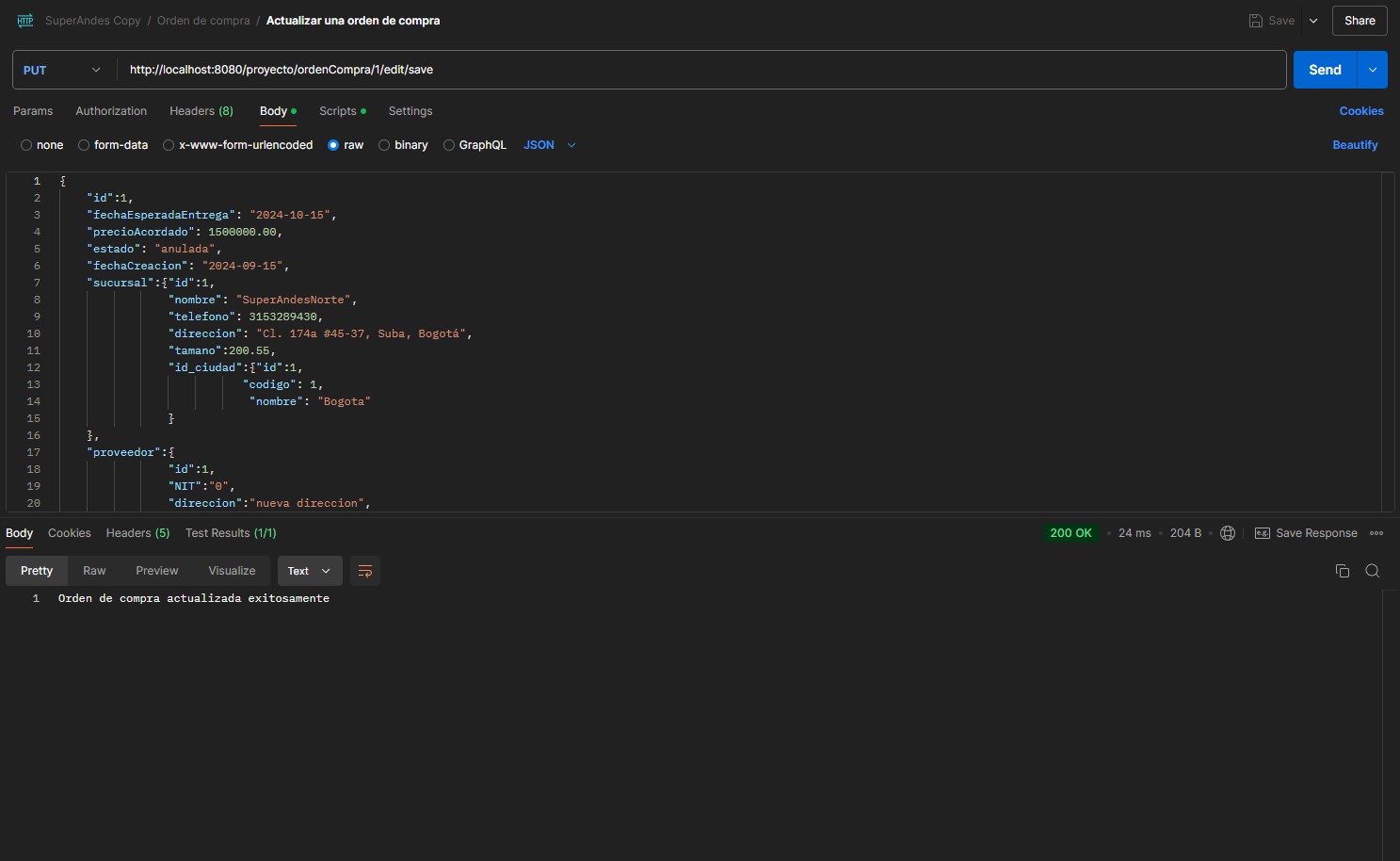
**Crear y leer y actualizar un producto:**

**Crear:**

**Leer:**

**Actualizar: -Requerimiento funcional 7:**

**Crear una orden de compra para una sucursal: -Requerimiento funcional 8:**

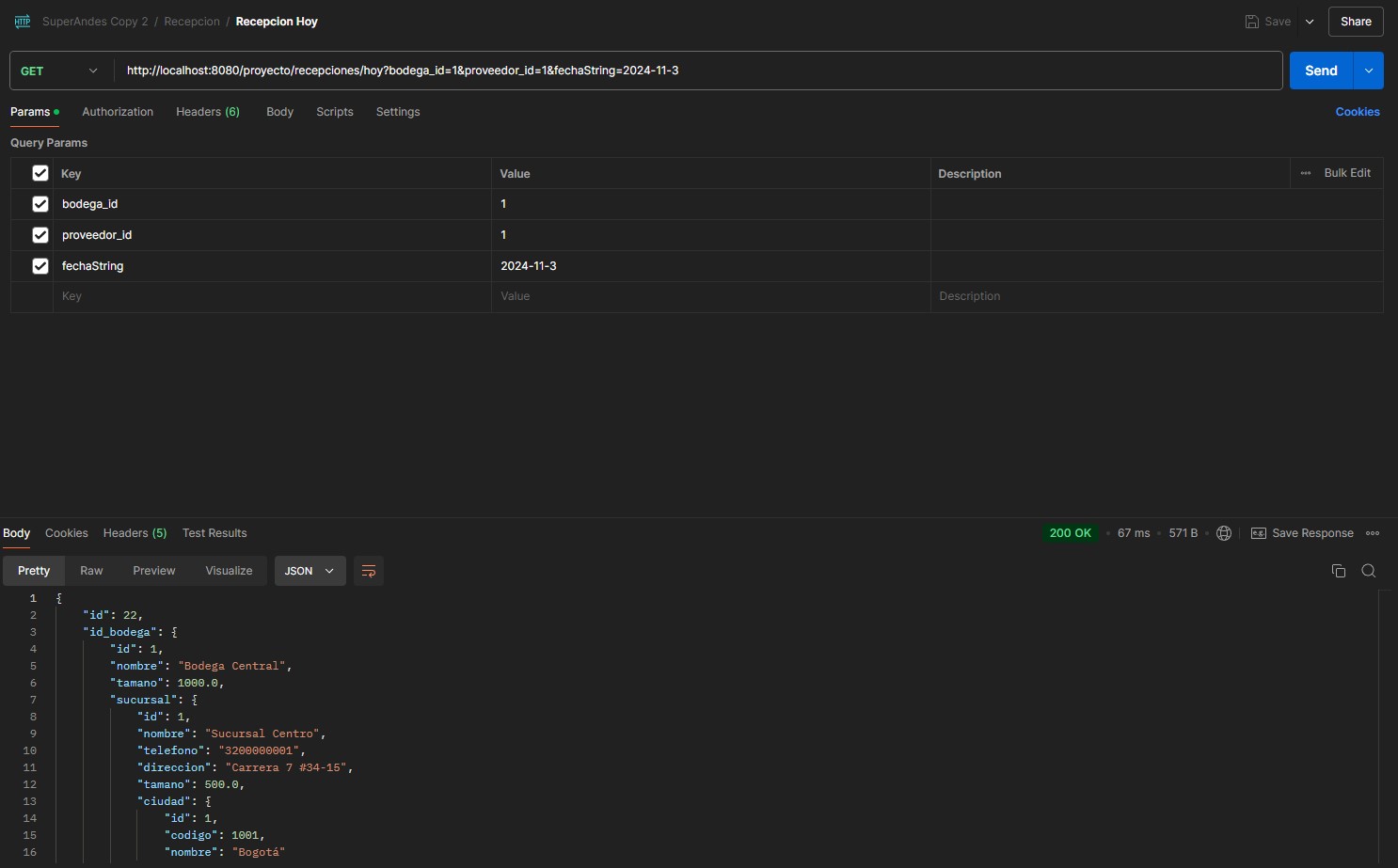
**Actualizar una orden de compra cambiando su estado a anulada:**

**-Requerimiento funcional 9:**

**Mostrar todas las órdenes de compra:**

**-Requerimiento funcional 10:**

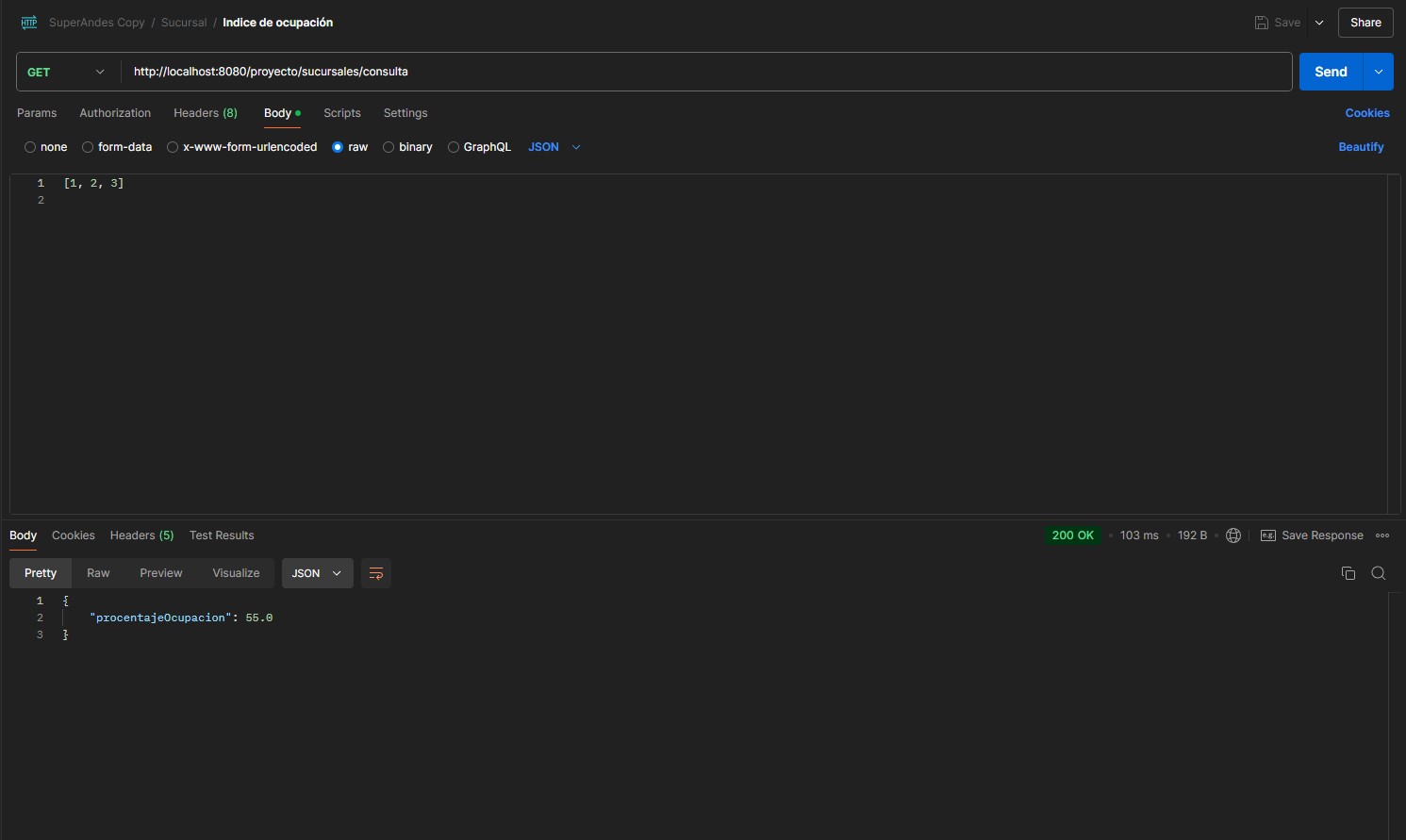
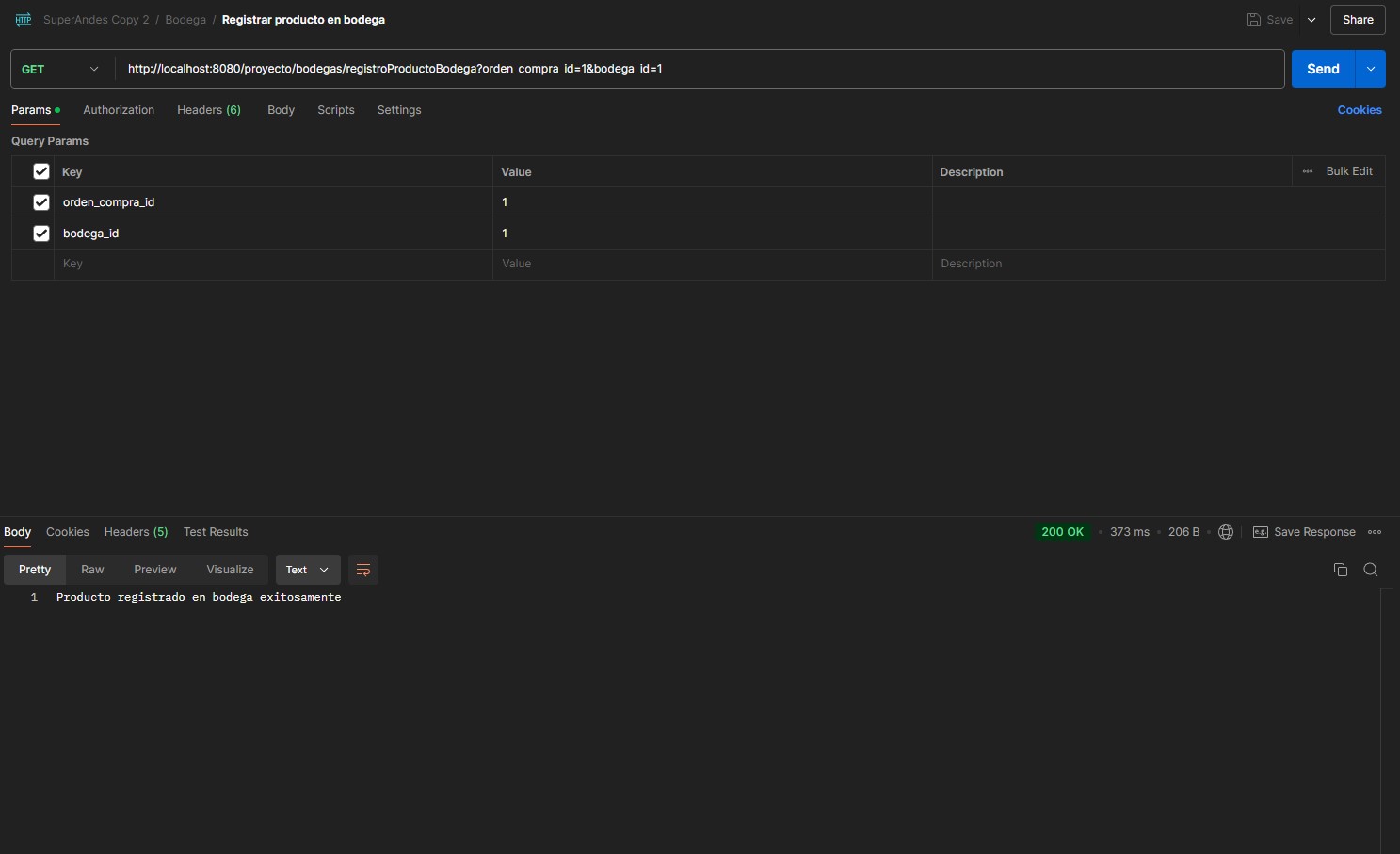
**Registrar ingreso de productos a bodega:**

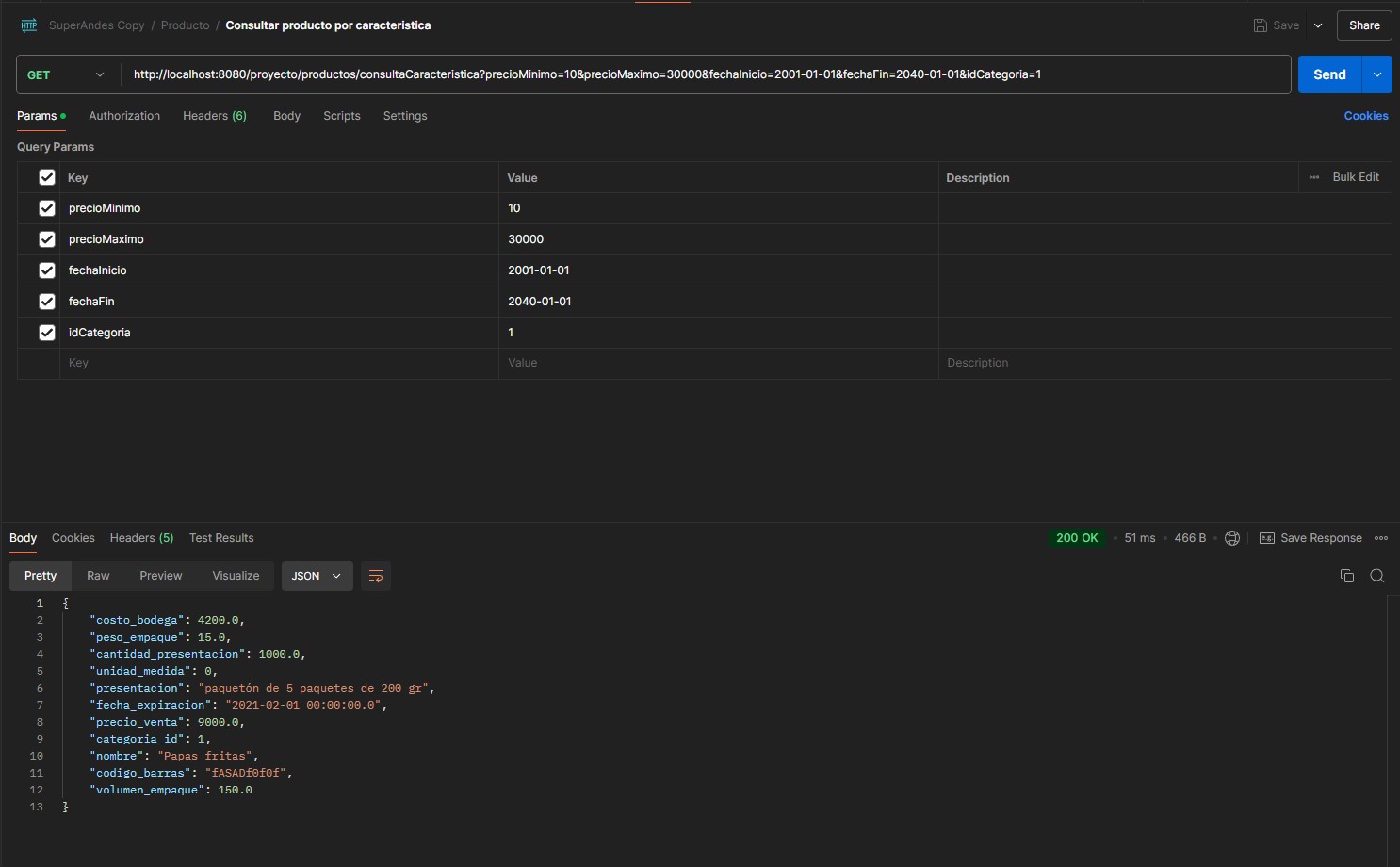
Para este requerimiento hicimos dos pruebas, una es recepción hoy la cual prueba una parte importante del requerimiento (principalmente para detectar errores y verificar la respuesta que da) y el otro es el requerimiento completo y se llama registrar productos en bodega. **Recepción hoy:**

**Registrar producto en bodega:**

**Requerimientos funcionales de consulta:**

**-Requerimiento funcional de consulta 1:**

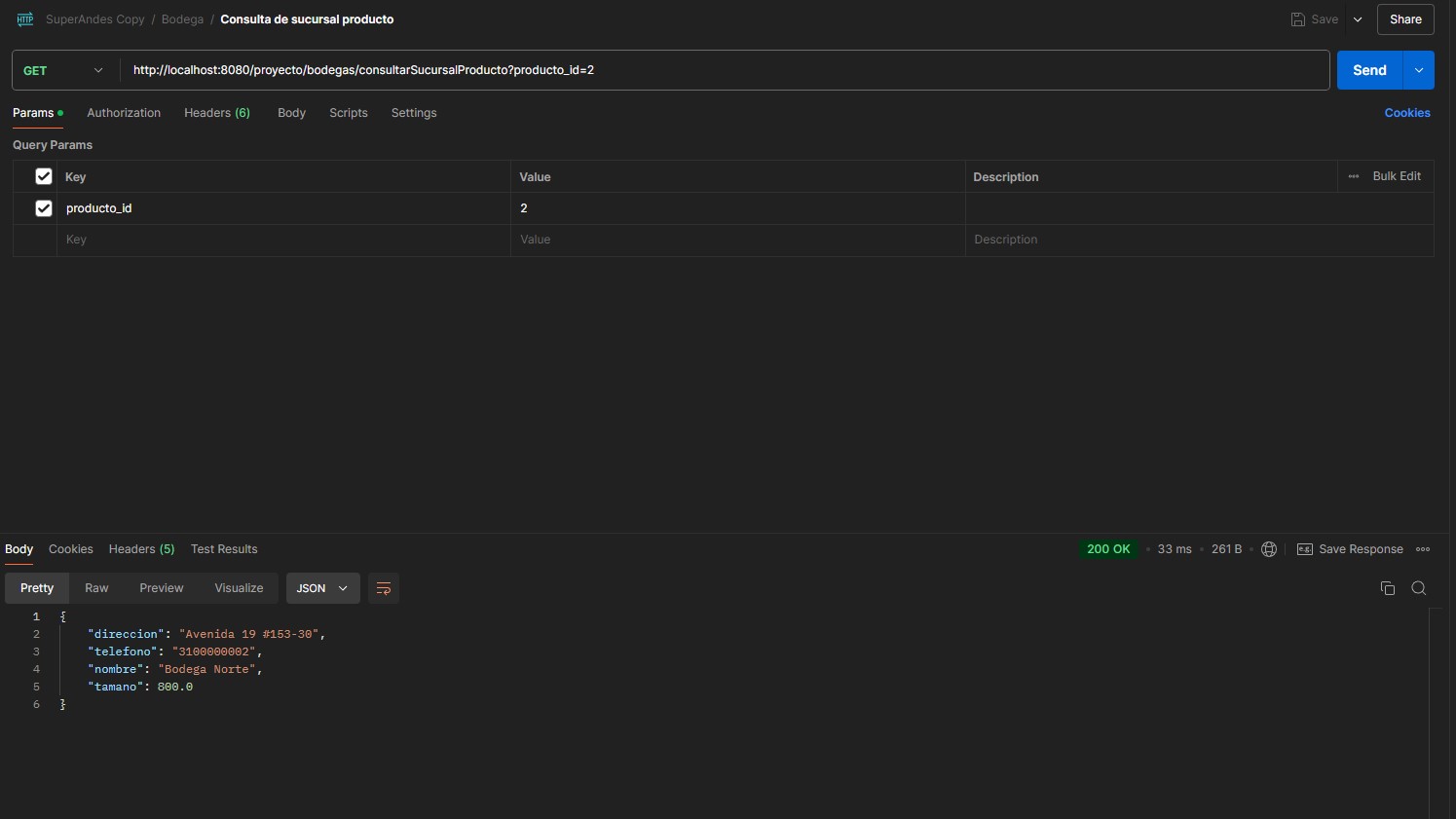
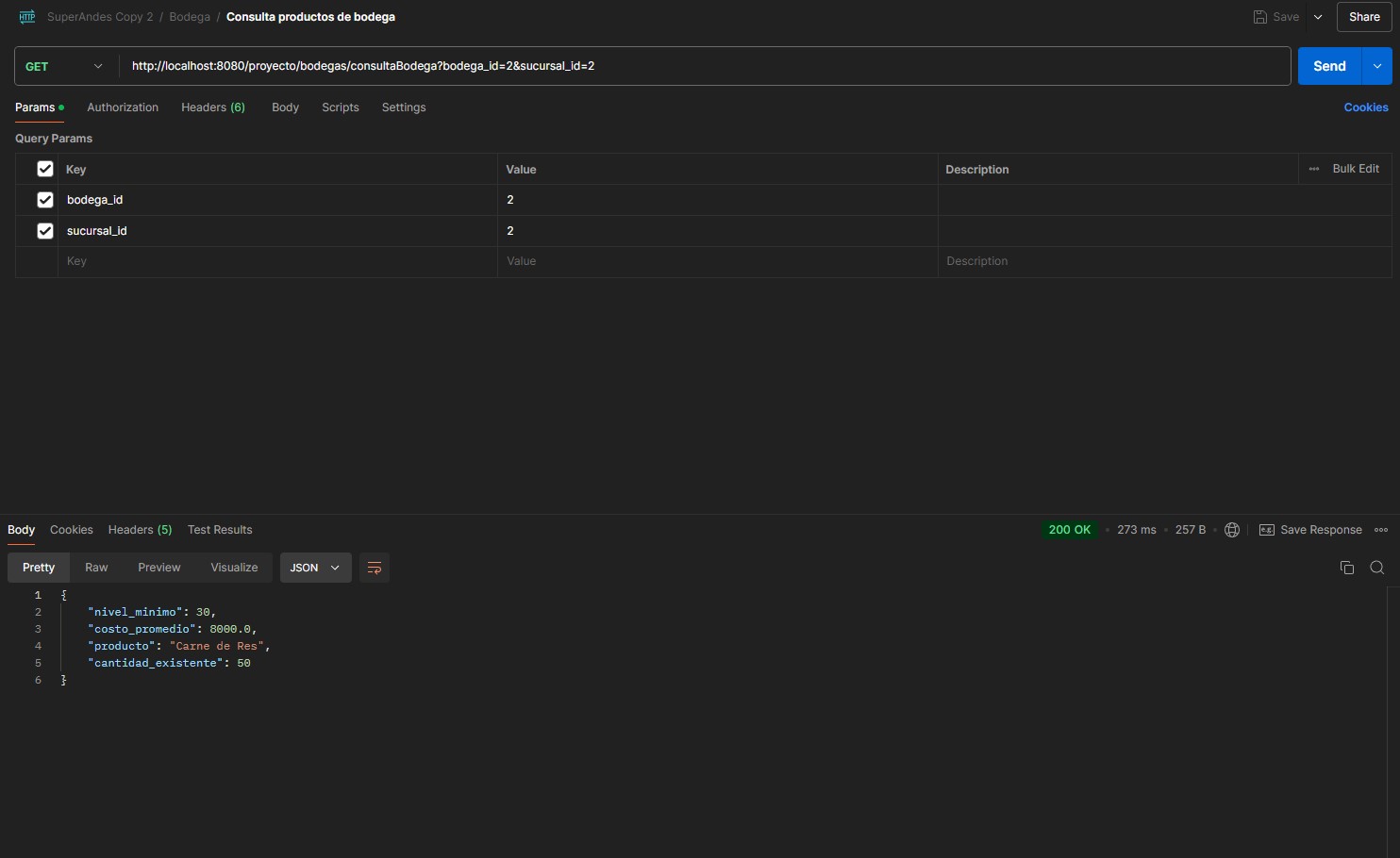
**Mostrar índice de ocupación de cada una de las bodegas de una sucursal: -Requerimiento funcional de consulta 2:**

**Mostrar los productos que cumplen con cierta característica:**

**-Requerimiento funcional de consulta 3:**

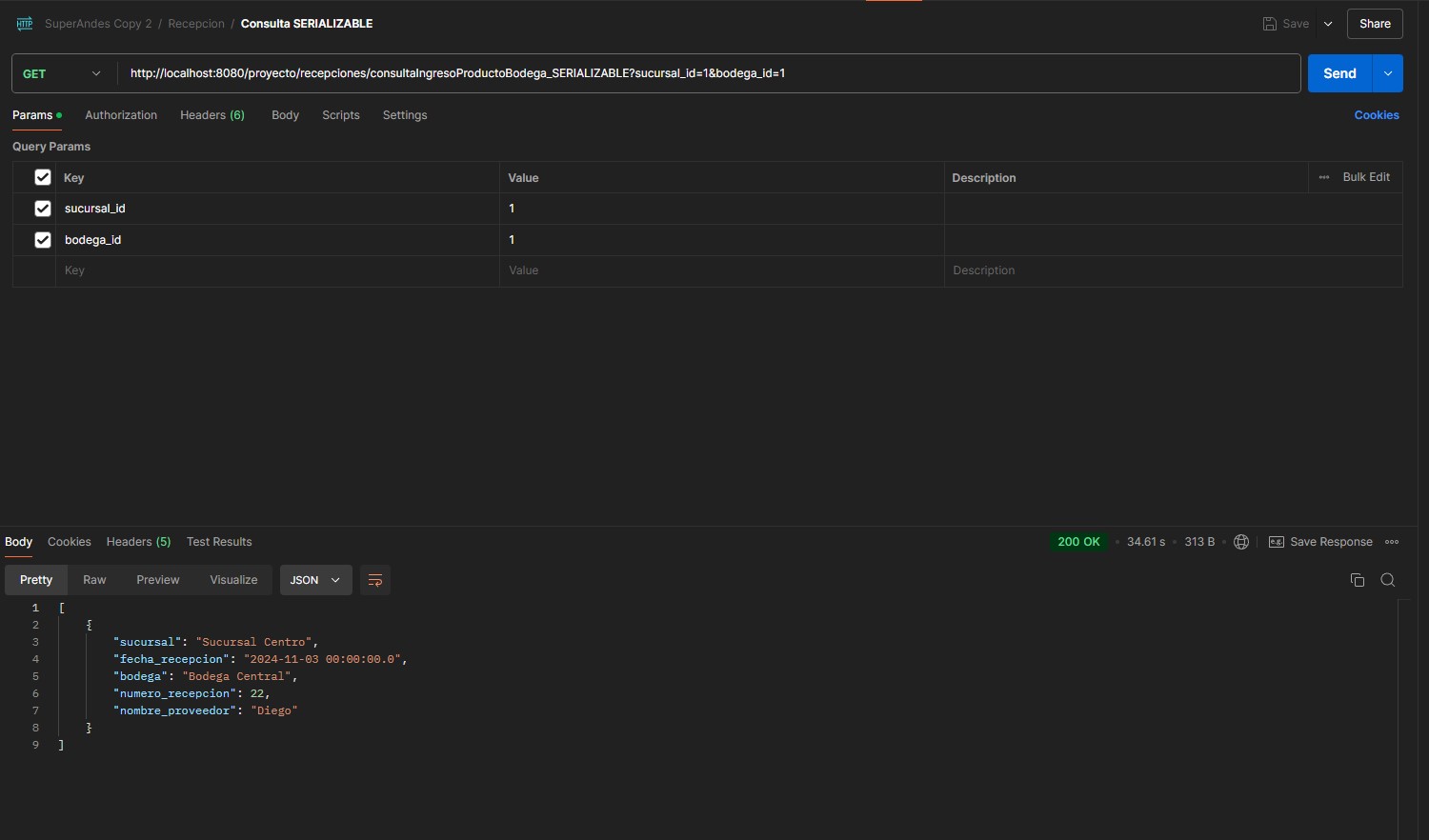
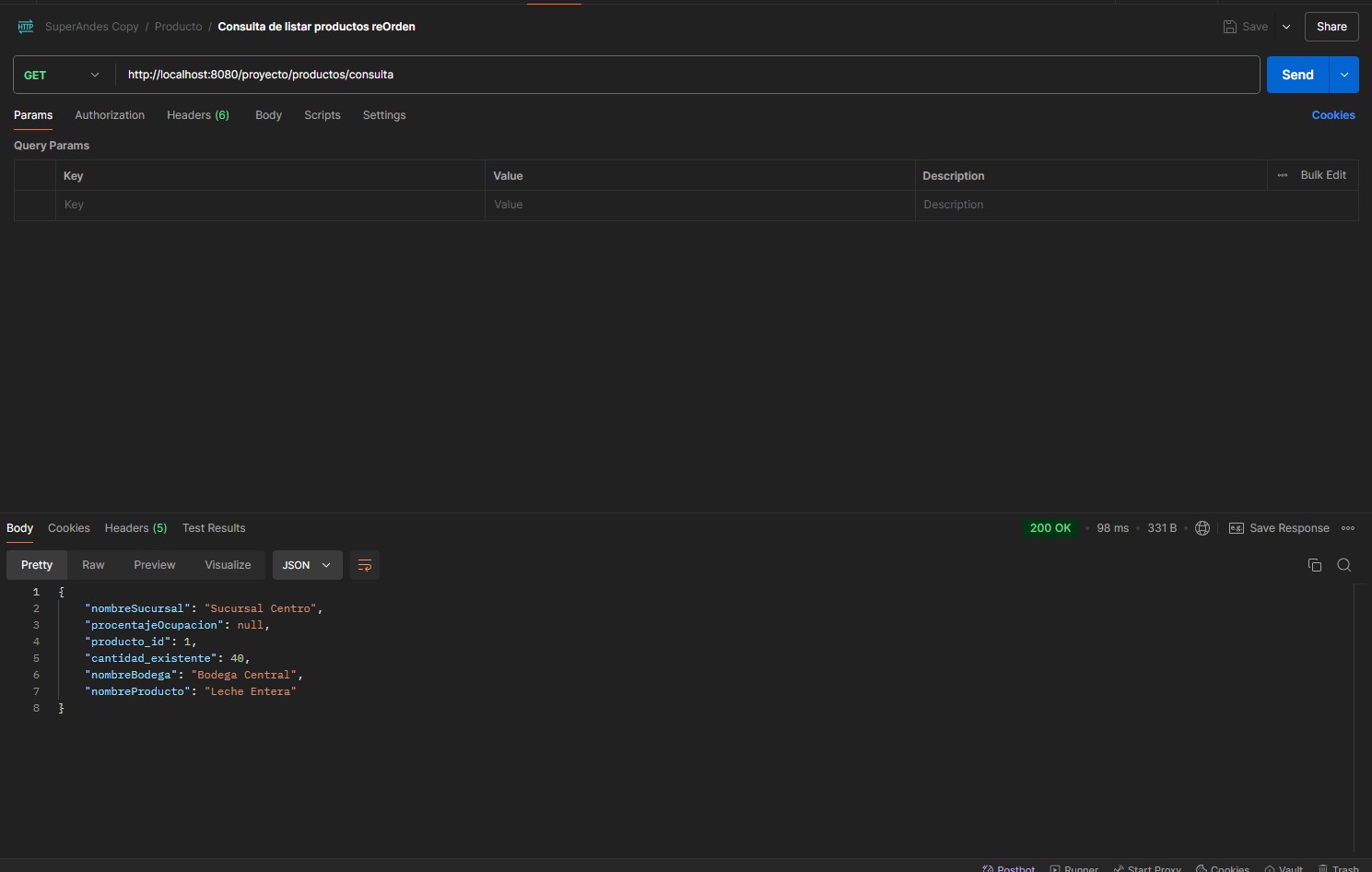
**Inventario de productos en bodega:**

**-Requerimiento funcional de consulta 4:**

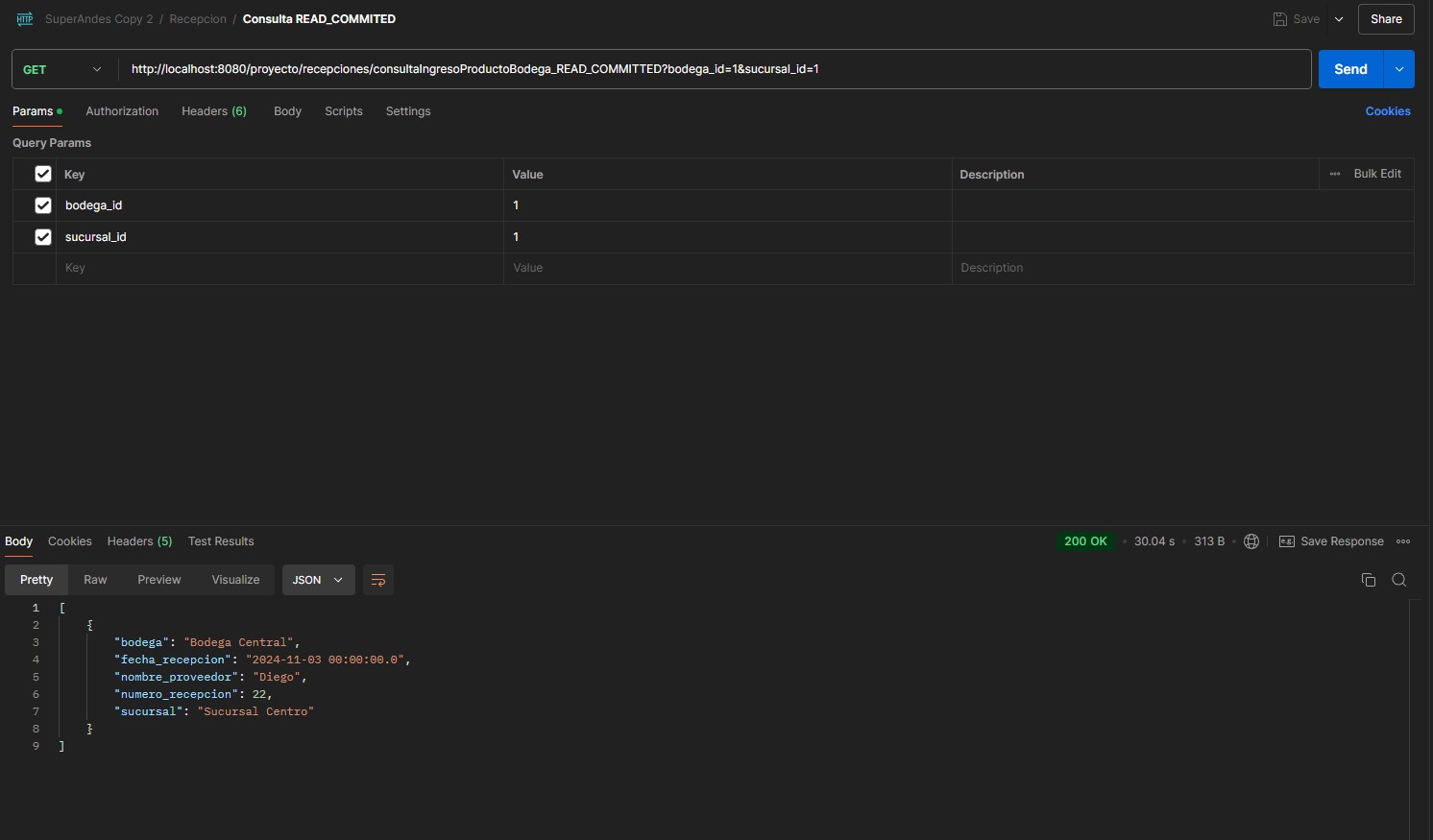
**Mostrar sucursales en las que hay disponibilidad de un producto:**

**-Requerimiento funcional de consulta 5:**

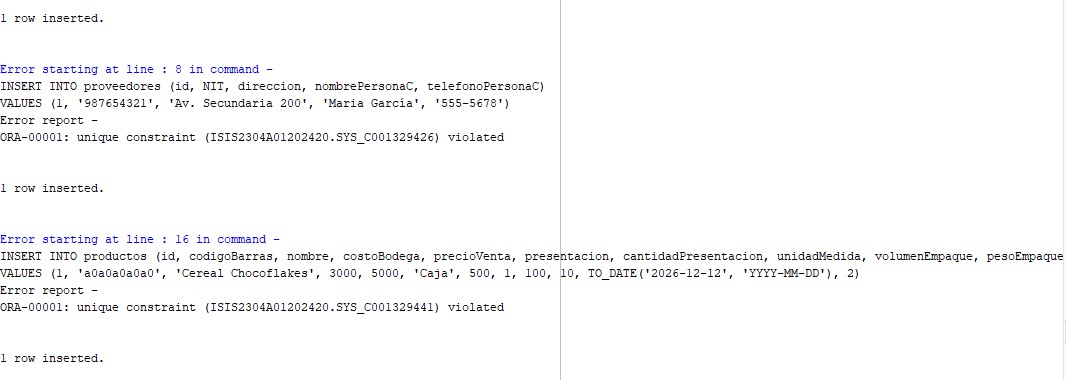
**Mostrar todos los productos que requieren una orden de compra: -Requerimiento funcional de consulta 6:**

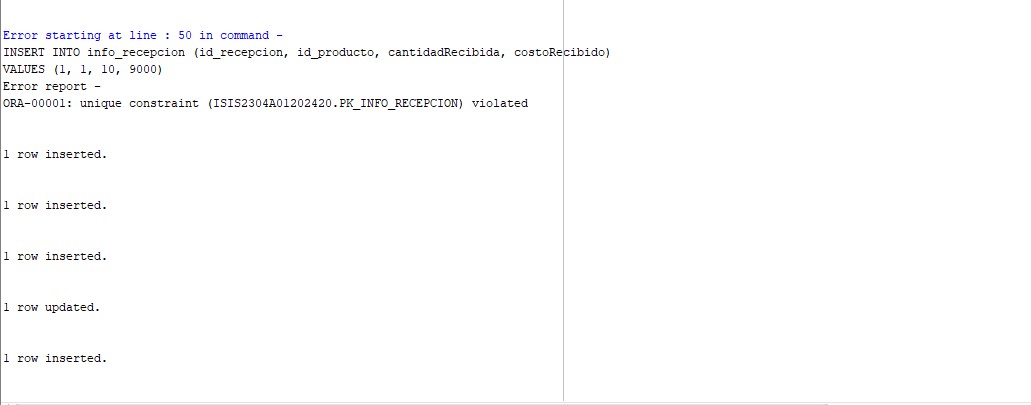
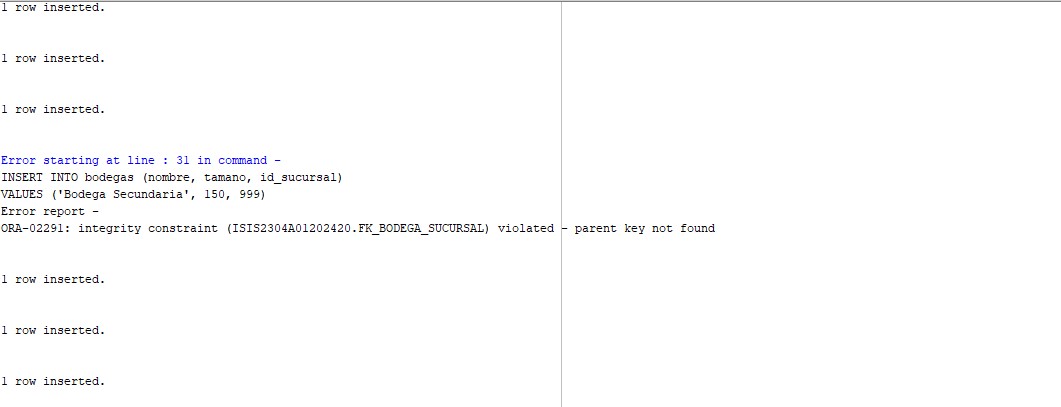
**Consulta de documentos de ingreso de productos a bodega - Serializable:**

**-Requerimiento funcional de consulta 7:**

**Consulta de documentos de ingreso de productos a bodega – Read Commited:**

1. **Escenarios de prueba**





**ESCENARIOS DE PRUEBAS DE CONCURRENCIA:**

**Escenario 1**

**Para el escenario 1 llegamos a las siguientes conclusiones:**

Se lanza la consulta consultaIngresoProductoBodega\_SERIALIZABLE, que busca obtener los documentos de ingreso de productos en una bodega específica.

Esta operación es transaccional y se ejecuta en un nivel de aislamiento SERIALIZABLE, lo cual es el nivel más estricto. Este nivel de aislamiento garantiza que no habrá interferencias de otras transacciones y asegura la consistencia en los datos durante la ejecución de RFC6.

La consulta realiza una primera obtención de los datos (almacenados en respuestaConsultaIgreso), espera 30 segundos (Thread.sleep(30000)), y luego vuelve a realizar la consulta en el repositorio antes de retornar los datos.

**Ejecución Concurrente de RF10:**

Antes de que transcurran los 30 segundos de espera en RFC6, se ejecuta la operación registroProductoBodega (RF10), cuyo propósito es registrar el ingreso de productos en la bodega. Esta operación también es transaccional, con un nivel READ COMMITTED.

RF10 busca actualizar el estado del inventario en la bodega y marcar la orden de compra como "ENTREGADA".

**Paso a Paso en la Línea de Tiempo**

T=0 segundos:

RFC6 comienza a ejecutarse y realiza la primera consulta en el repositorio para obtener los documentos de ingreso de productos. Como está en modo SERIALIZABLE, se "bloquea" un conjunto de datos, evitando que otros procesos modifiquen el mismo conjunto durante la transacción.

T=15 segundos:

Se lanza RF10 mientras RFC6 aún está en espera. Debido a la serialización de RFC6, RF10 no puede modificar o registrar ningún ingreso de producto en la bodega hasta que RFC6 termine su ejecución.

T=30 segundos:

RFC6 realiza la segunda consulta en el repositorio y finaliza, liberando el bloqueo serializable.

T=31 segundos:

RF10 puede ahora registrar los productos en la bodega y finalizar su proceso.

**Resultado y Observaciones**

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

**Descripción de lo sucedido:**

El componente RF10 no pudo registrar el ingreso de productos en la bodega hasta que finalizó la ejecución de RFC6. Esto se debe a que el nivel de aislamiento SERIALIZABLE de RFC6 bloquea las modificaciones concurrentes de los datos que está leyendo.

Al estar en aislamiento serializable, RFC6 garantiza que los datos que leyó inicialmente permanecen inalterados hasta el final de la transacción. RF10, que requiere modificar estos datos, tiene que esperar a que RFC6 libere el bloqueo.

**Escenario 2**  
**1. Pasos para la Ejecución Concurrente de RFC7 y RF10 a través de la Línea de Tiempo**

• Paso 1: Se inició RFC7 con el nivel de aislamiento READ COMMITTED, lo cual comenzó una

consulta de documentos de ingreso de productos a bodega, incluyendo un temporizador

de 30 segundos.

• Paso 2: Mientras RFC7 estaba en ejecución, se inició RF10 para registrar un nuevo ingreso

de producto en la bodega.

• Paso 3: RF10 finalizó rápidamente y confirmó la transacción de inserción, mientras RFC7

aún estaba en el proceso de espera.

• Paso 4: Al finalizar los 30 segundos, RFC7 completó su consulta y devolvió los documentos

de ingreso de los últimos 30 días.

**2. Descripción de lo Sucedido**

• ¿RF10 Debió Esperar a RFC7?

No, RF10 no tuvo que esperar a que terminara RFC7. Dado que el nivel de aislamiento de

RFC7 es READ COMMITTED, no bloquea la escritura concurrente de RF10. Esto significa

que RF10 pudo registrar el ingreso del producto de manera simultánea sin interferir con la

consulta en curso de RFC7.

**3. Resultado Presentado por RFC7**

• ¿Apareció el Nuevo Documento de Ingreso?

Sí, el nuevo documento de ingreso creado por RF10 apareció en los resultados de la

consulta de RFC7. Esto se debe a que RFC7, con el nivel de aislamiento READ

COMMITTED, puede ver las transacciones confirmadas durante su ejecución. Al completar

el temporizador de 30 segundos, RFC7 incluyó el nuevo documento de ingreso registrado

por RF10 en sus resultados.

**Conclusión**

El nivel de aislamiento READ COMMITTED permitió que la consulta de RFC7 incluyera el

documento de ingreso de productos realizado por RF10 de manera simultánea, ya que la

transacción de RF10 se confirmó antes de que finalizara la ejecución de RFC7. Esto

demuestra que READ COMMITTED permite ver los datos confirmados sin bloquear las

transacciones de escritura concurrentes.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente  
Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente