Informe de Pruebas   
del   
Sistema Web Ferremas

Equipo de trabajo: Matias Rain  
Benjamín Celis   
Matias Sáez

Contenido

[Diseño del Plan de Pruebas 3](#_Toc201654680)

[ Verificar que los módulos del sistema Ferremas funcionen correctamente según los requerimientos establecidos. 3](#_Toc201654681)

[ Asegurar la correcta interacción entre los distintos componentes del sistema. 3](#_Toc201654682)

[ Detectar posibles errores o validaciones faltantes mediante pruebas manuales y herramientas externas. 3](#_Toc201654683)

[ Mejorar la calidad final del sistema antes de su entrega o despliegue. 3](#_Toc201654684)

[Tipos de pruebas realizadas: 3](#_Toc201654685)

[Herramientas utilizadas: 4](#_Toc201654686)

[Pruebas Documentada 4](#_Toc201654687)

[Pruebas Unitarias 5](#_Toc201654688)

[Pruebas de Integración 5](#_Toc201654689)

[Pruebas de API (Postman) 6](#_Toc201654690)

[Medición de Efectividad 23](#_Toc201654691)

[Resultados globales: 23](#_Toc201654692)

[Métricas: 23](#_Toc201654693)

[Conclusiones 24](#_Toc201654694)

Diseño del Plan de PruebasObjetivos:

* Verificar que los módulos del sistema Ferremas funcionen correctamente según los requerimientos establecidos.
* Asegurar la correcta interacción entre los distintos componentes del sistema.
* Detectar posibles errores o validaciones faltantes mediante pruebas manuales y herramientas externas.
* Mejorar la calidad final del sistema antes de su entrega o despliegue.

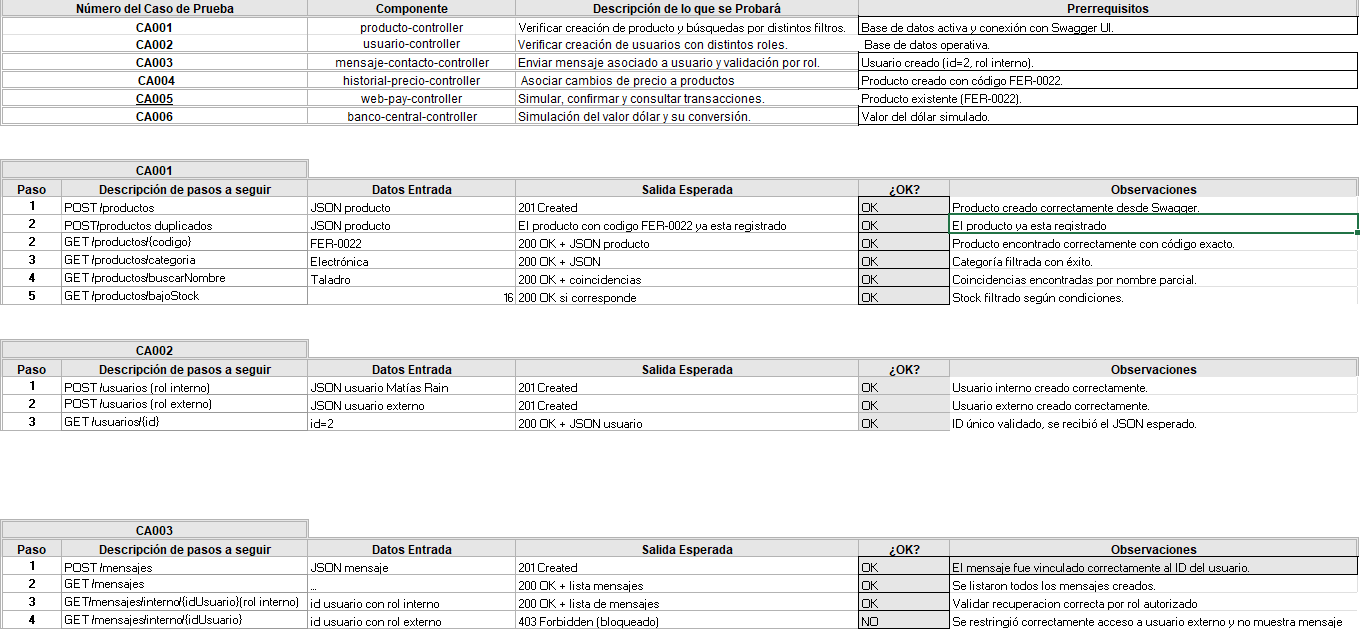
## Tipos de pruebas realizadas:

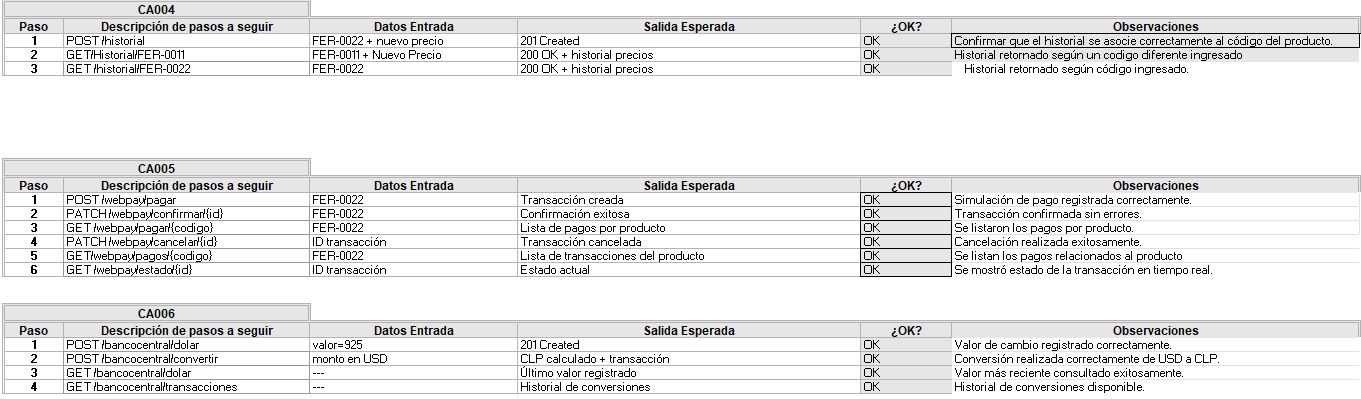
* Pruebas unitarias (manuales)
* Pruebas de integración
* Pruebas con herramientas externas (Postman)

## Herramientas utilizadas:

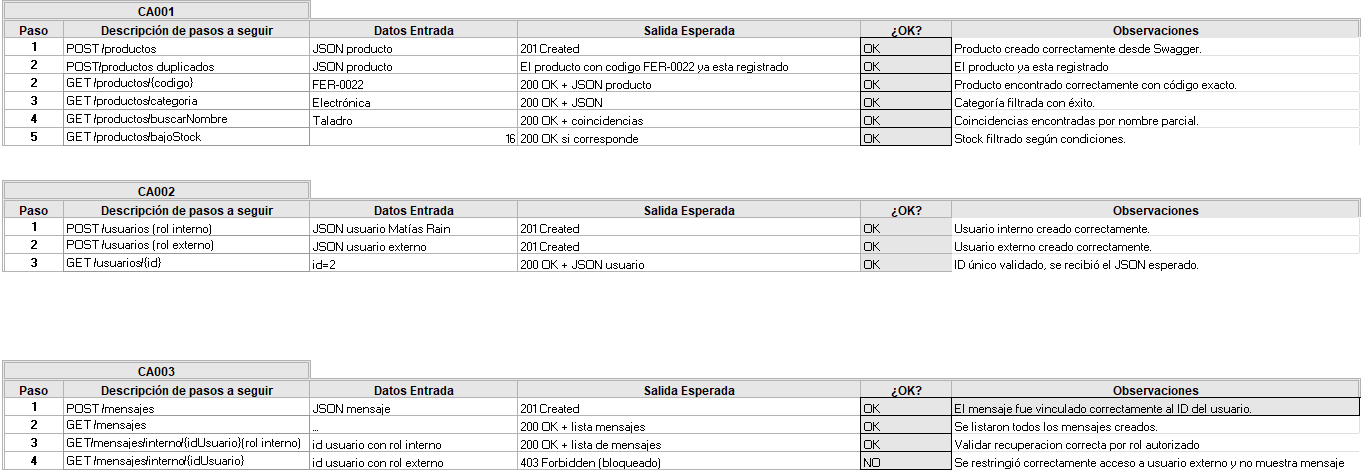
* Swagger UI (para explorar y validar endpoints directamente)
* Postman (para realizar pruebas externas desde el punto de vista de un cliente)

Pruebas Documentada





Pruebas Unitarias

****

Corresponden a pruebas sobre funcionalidades aisladas de cada controlador. Se ejecutaron desde Swagger para validar registros, consultas simples y comportamiento básico.  
Se trabajó sobre: productos, usuarios, historial, mensajes, y restricciones por código.  
Estas pruebas no dependen de ningún otro módulo para ejecutarse.

Pruebas de Integración Validan la interacción entre componentes del sistema: múltiples entidades, flujos de pago, conversiones, etc.



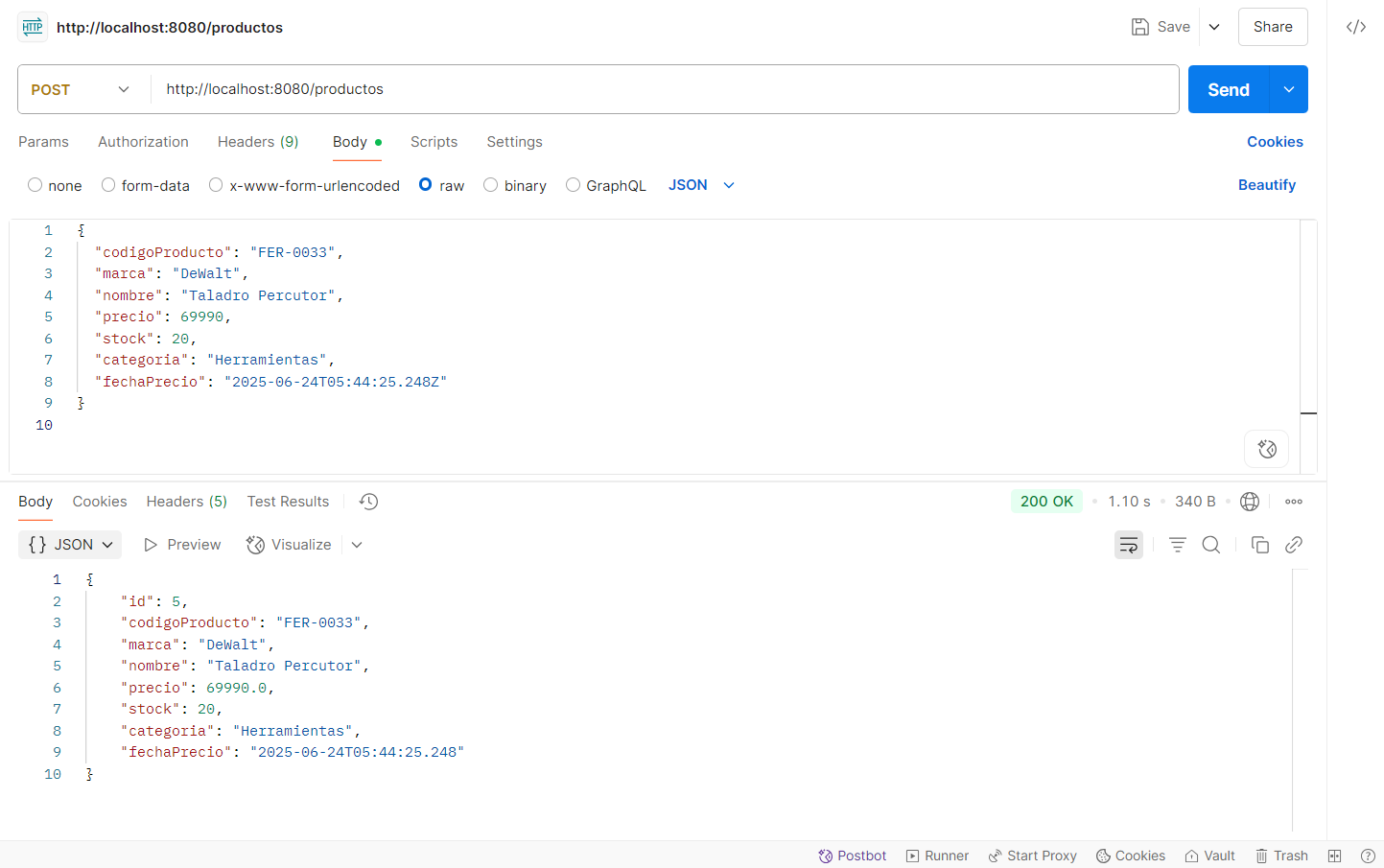
En estas pruebas se comprobó que el sistema pueda operar correctamente cuando distintos módulos se comunican entre sí.  
Por ejemplo:

* Registrar historial solo si el producto existe.
* Crear y confirmar pagos WebPay.
* Consultar transacciones de conversión después de simular una conversión.

Pruebas de API (Postman)

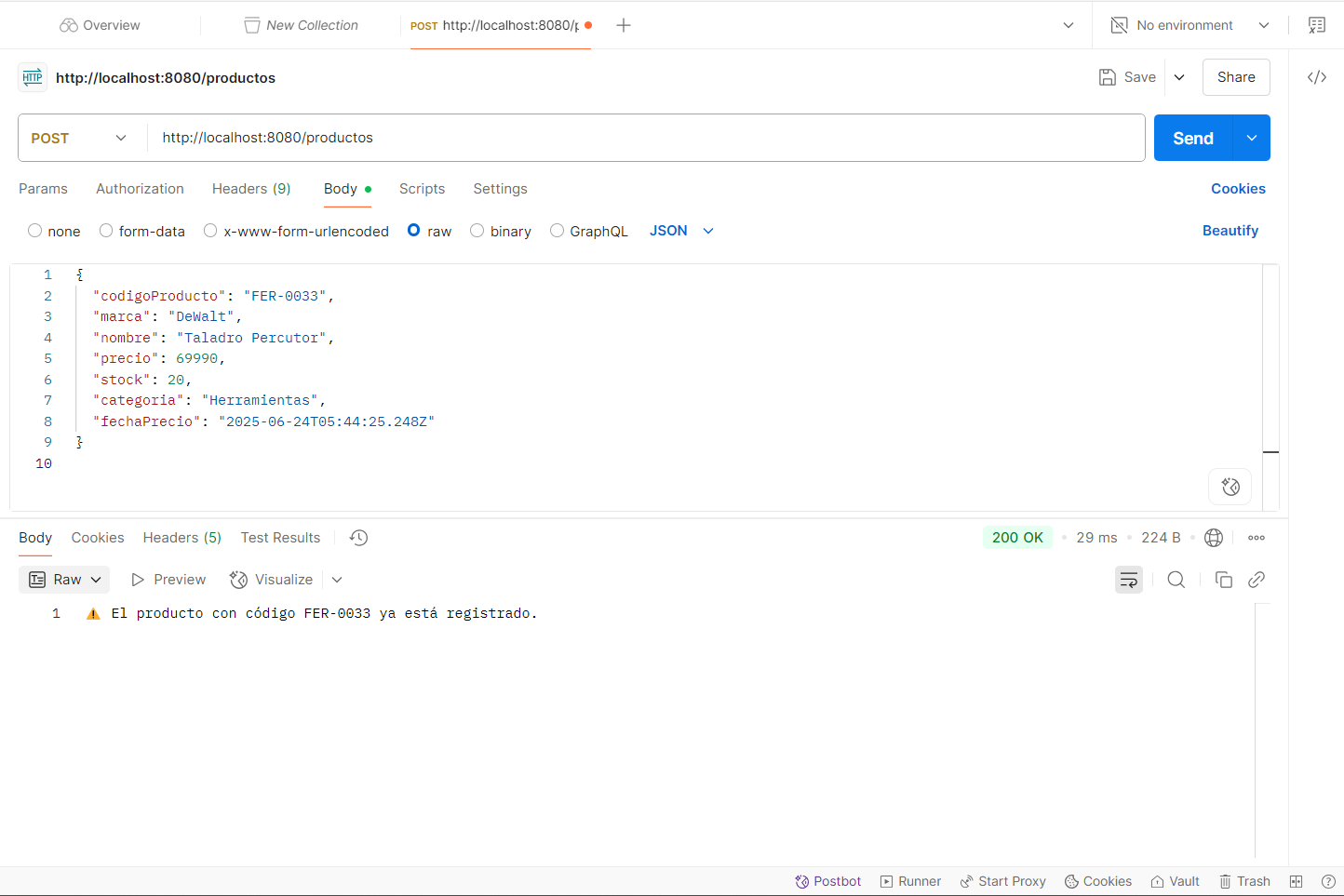
**Prueba 1 – Crear producto correctamente**

* **Método: POST**
* **URL: http://localhost:8080/productos**

****

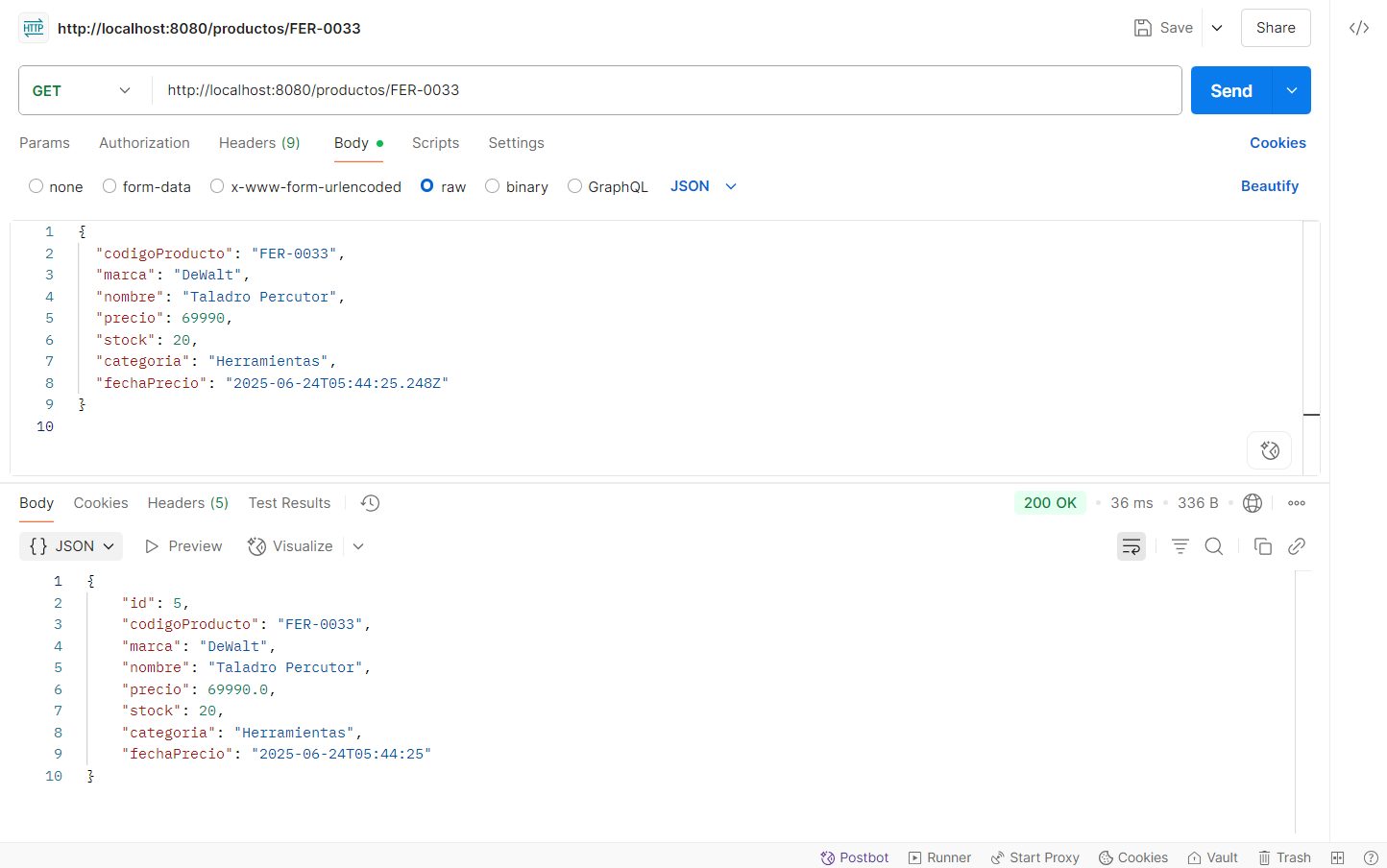
**Prueba 2 – Producto duplicado (por código)**

* **Método: POST**
* **URL: http://localhost:8080/productos**
* **JSON: (mismo que prueba 1)**

****

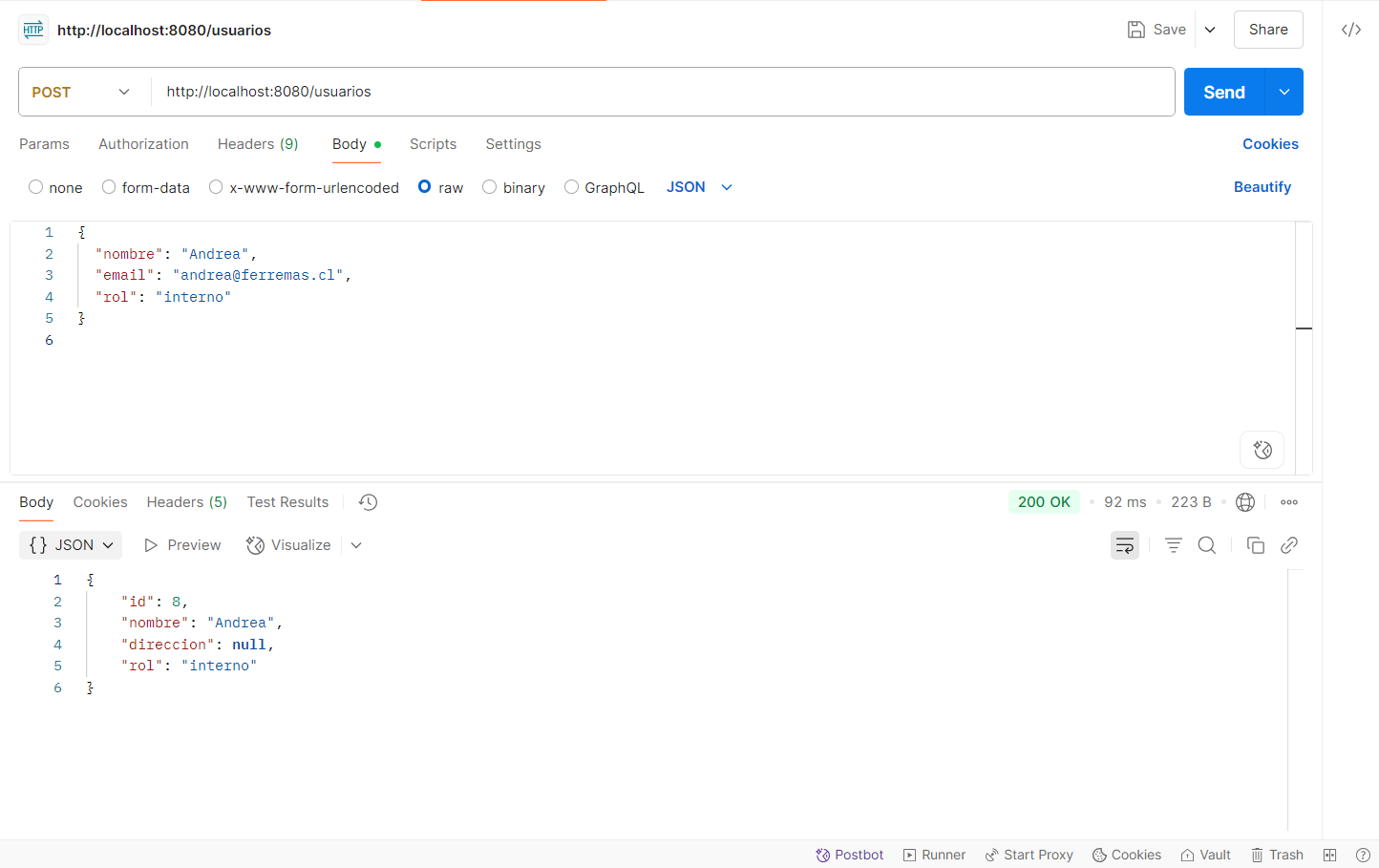
**Prueba 3 – Obtener producto por código**

* **Método: GET**
* **URL:** [**http://localhost:8080/productos/FER-0033**](http://localhost:8080/productos/FER-0033)

****

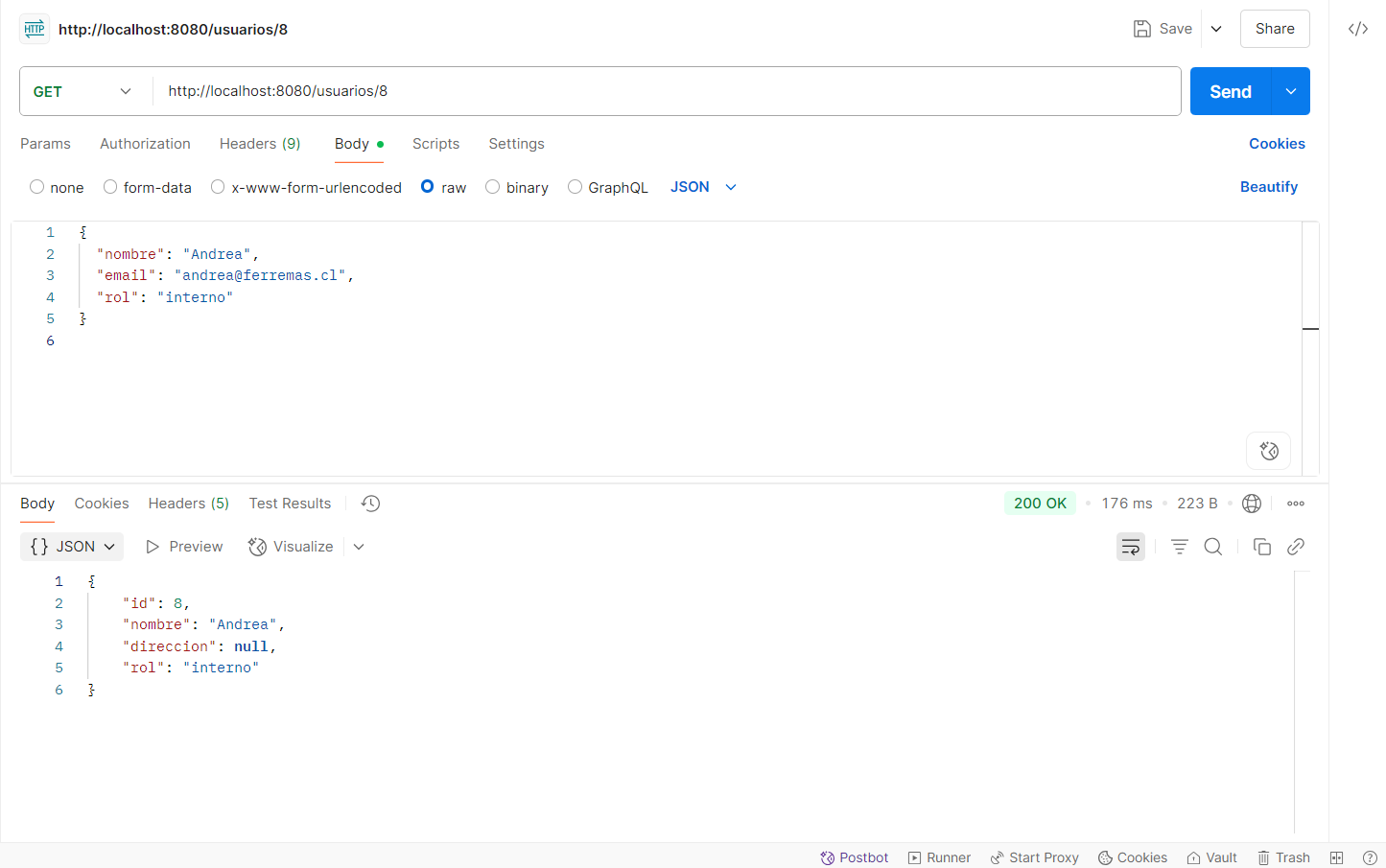
**Prueba 4 – Crear usuario interno**

* **Método: POST**
* **URL: http://localhost:8080/usuarios**

****

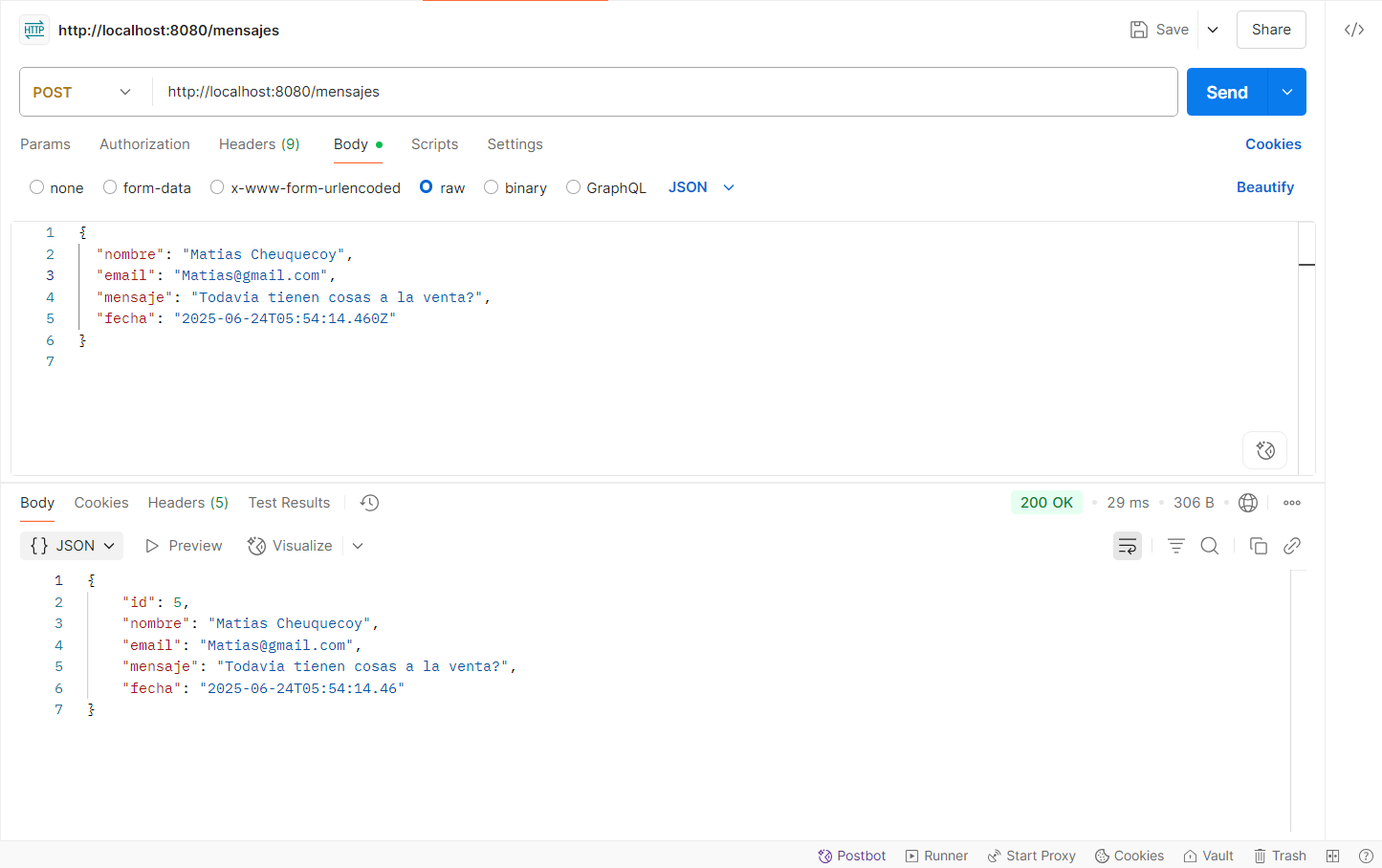
**Prueba 5 – Obtener usuario por ID**

* **Método: GET**
* **URL: http://localhost:8080/usuarios/8**

****

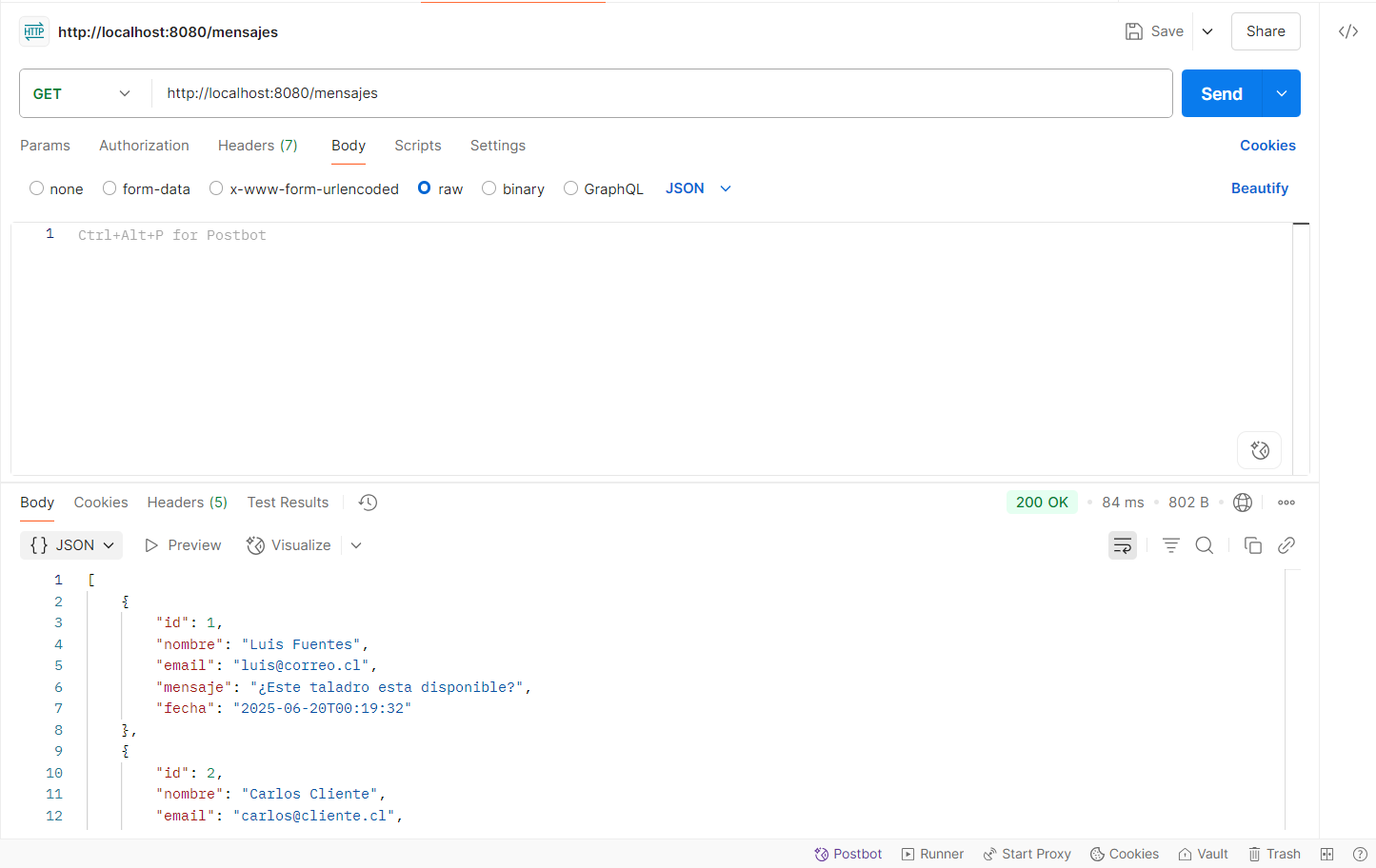
**🔹 Prueba 6 – Enviar mensaje de contacto**

* **Método: POST**
* **URL: http://localhost:8080/mensajes**

****

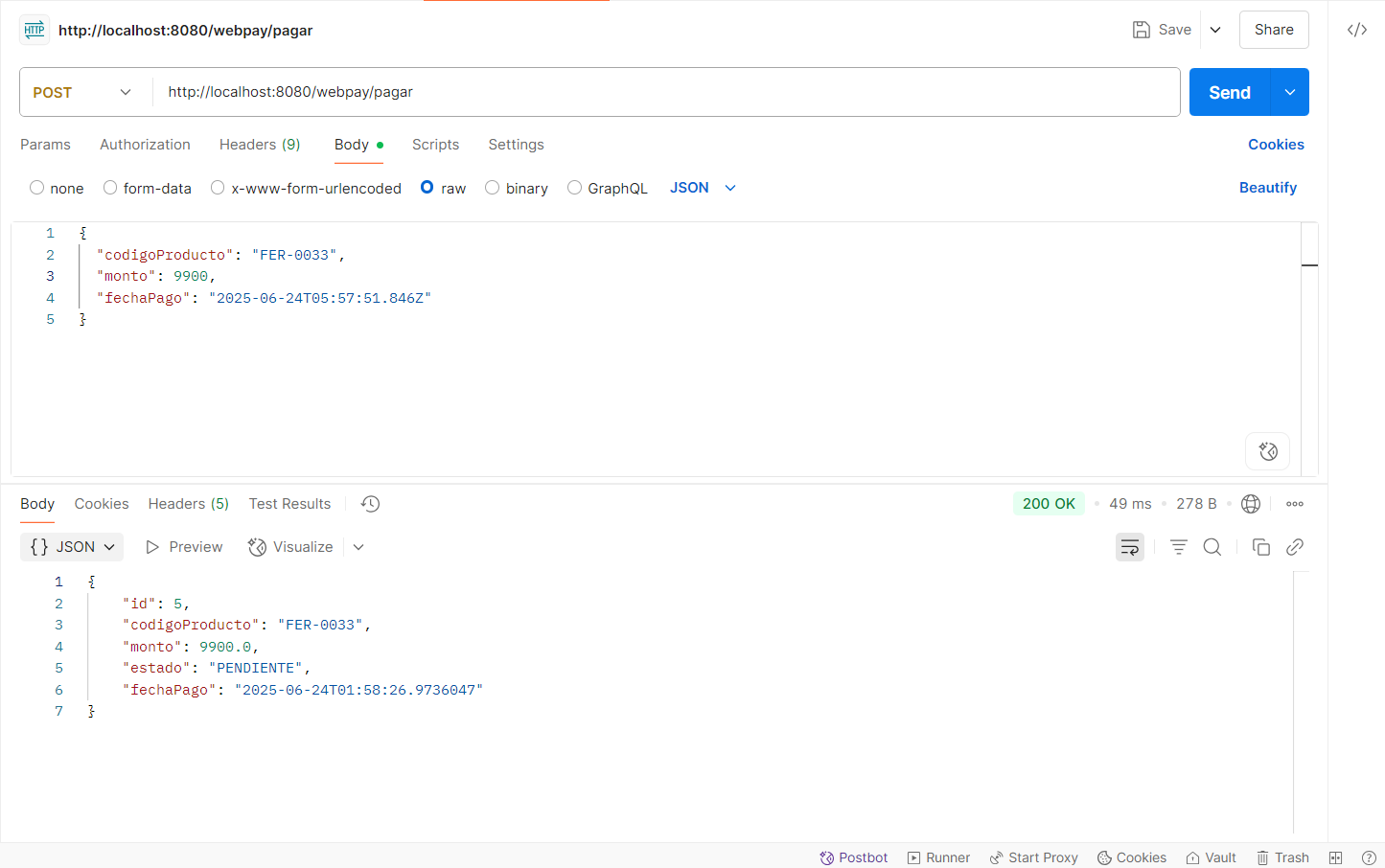
**🔹 Prueba 7 – Obtener todos los mensajes**

* **Método: GET**
* **URL: http://localhost:8080/mensajes**

****

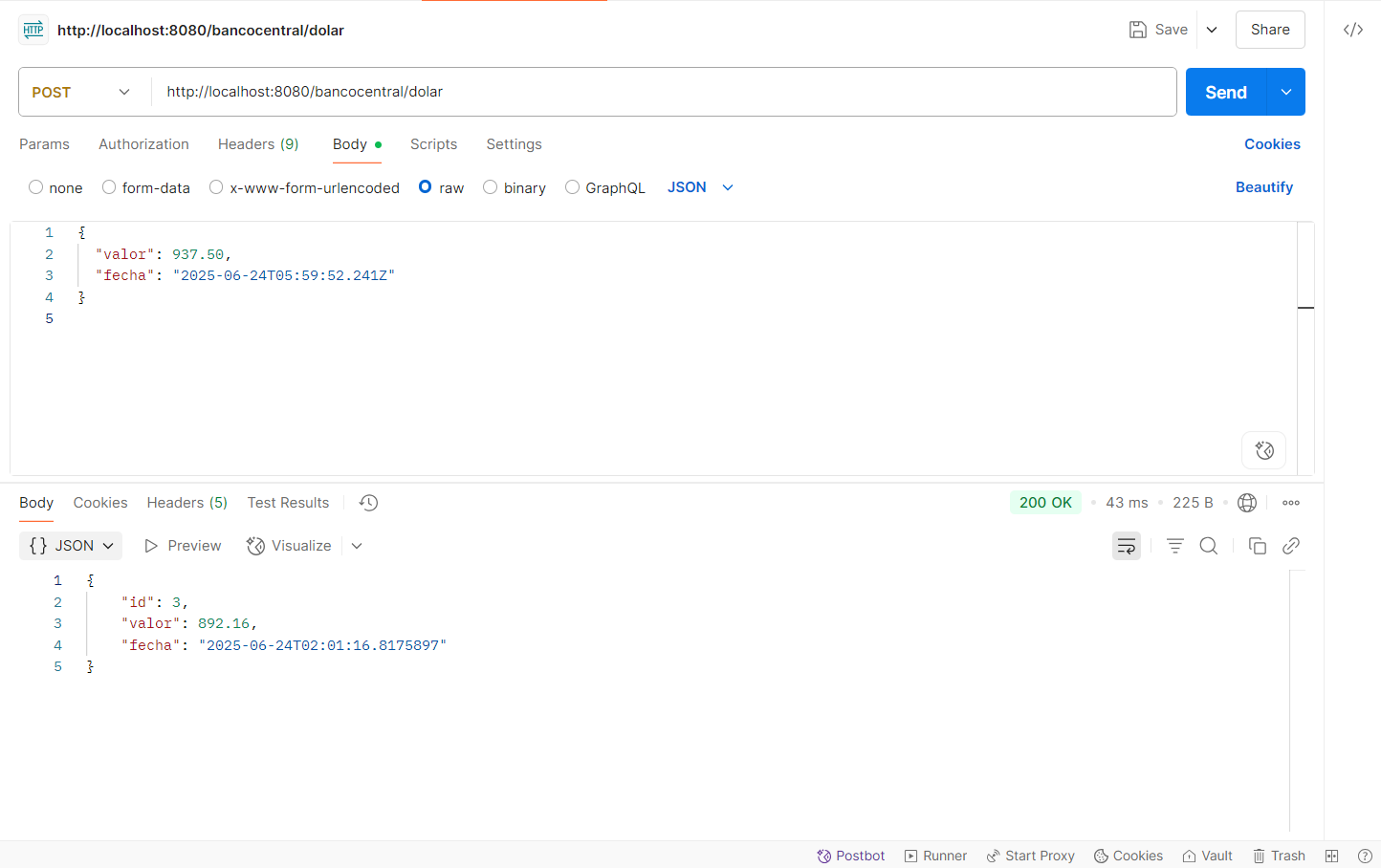
**🔹 Prueba 8 – Simular pago WebPay**

* **Método: POST**
* **URL: http://localhost:8080/webpay/pagar**

****

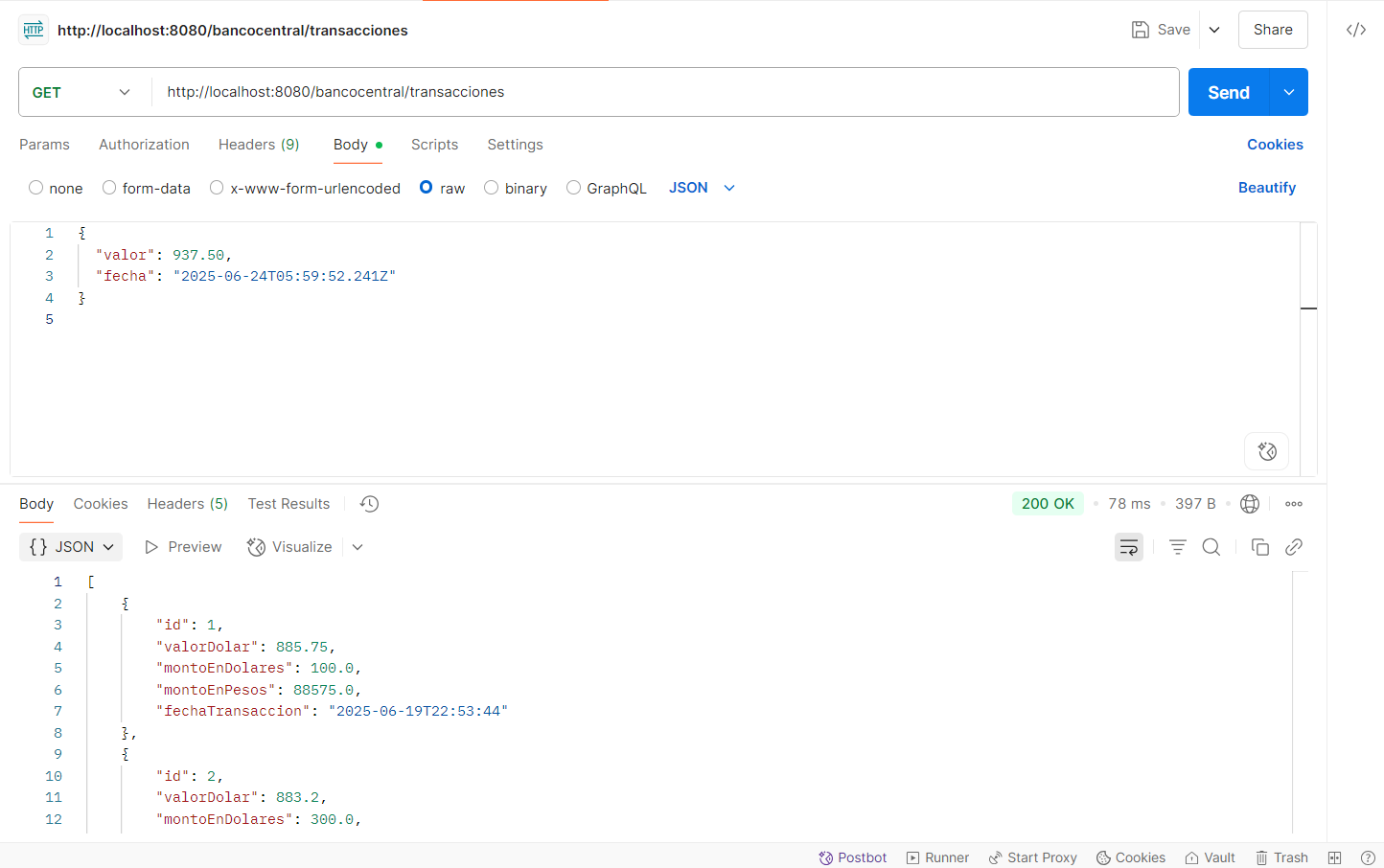
**🔹 Prueba 9 – Convertir 100 USD a CLP**

* **Método: POST**
* **URL: http://localhost:8080/bancocentral/convertir**

****

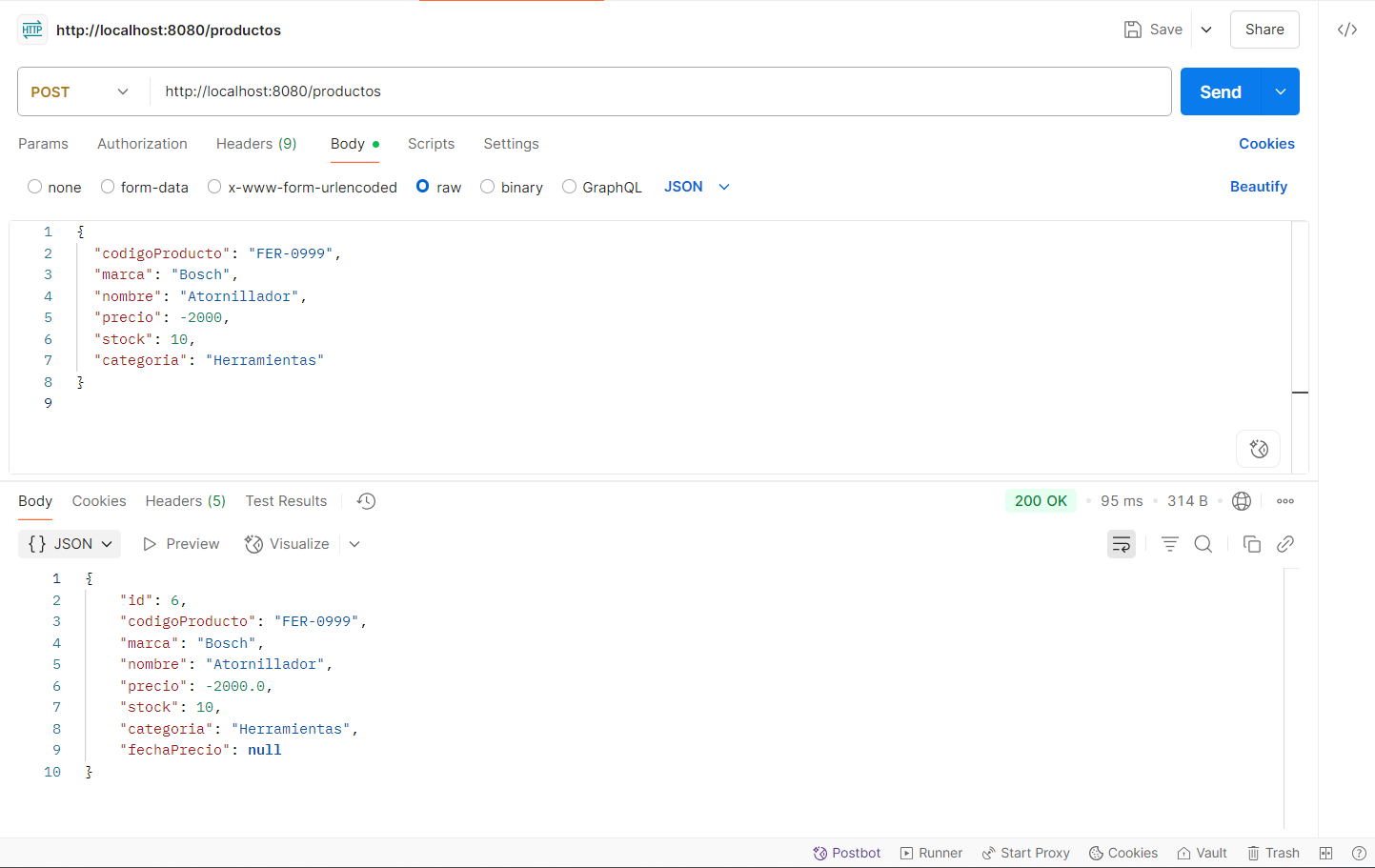
**🔹 Prueba 10 – Ver historial de conversiones**

* **Método: GET**
* **URL: http://localhost:8080/bancocentral/transacciones**

****

**11. Crear producto con precio negativo (validación)**

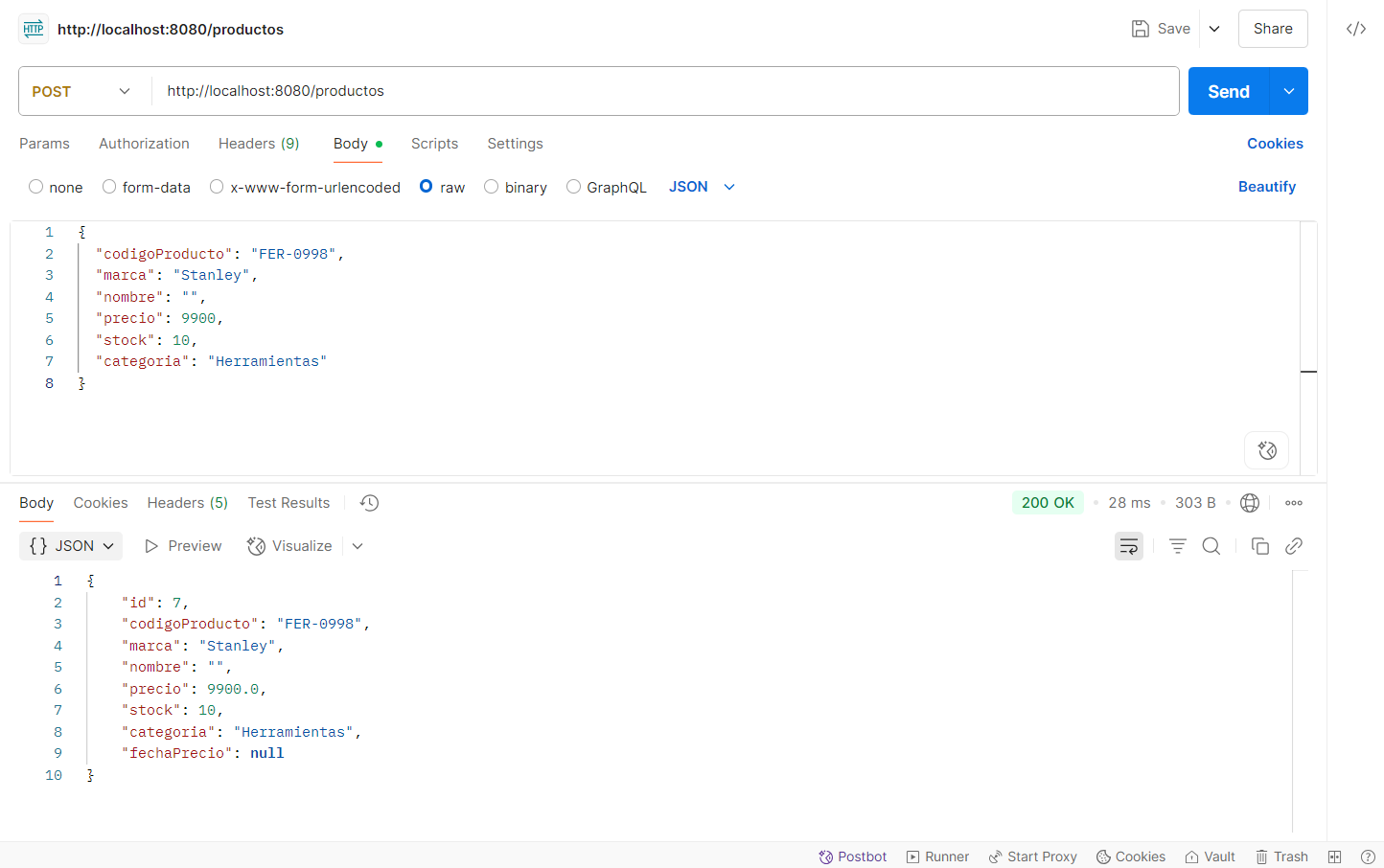
* **Método: POST**
* **URL: http://localhost:8080/productos**



Encontramos un error porque acepta la validación del producto a pesar de que el precio esta en negativo

**12. Crear producto sin nombre (campo vacío)**

* **Método: POST**
* **URL: http://localhost:8080/productos**

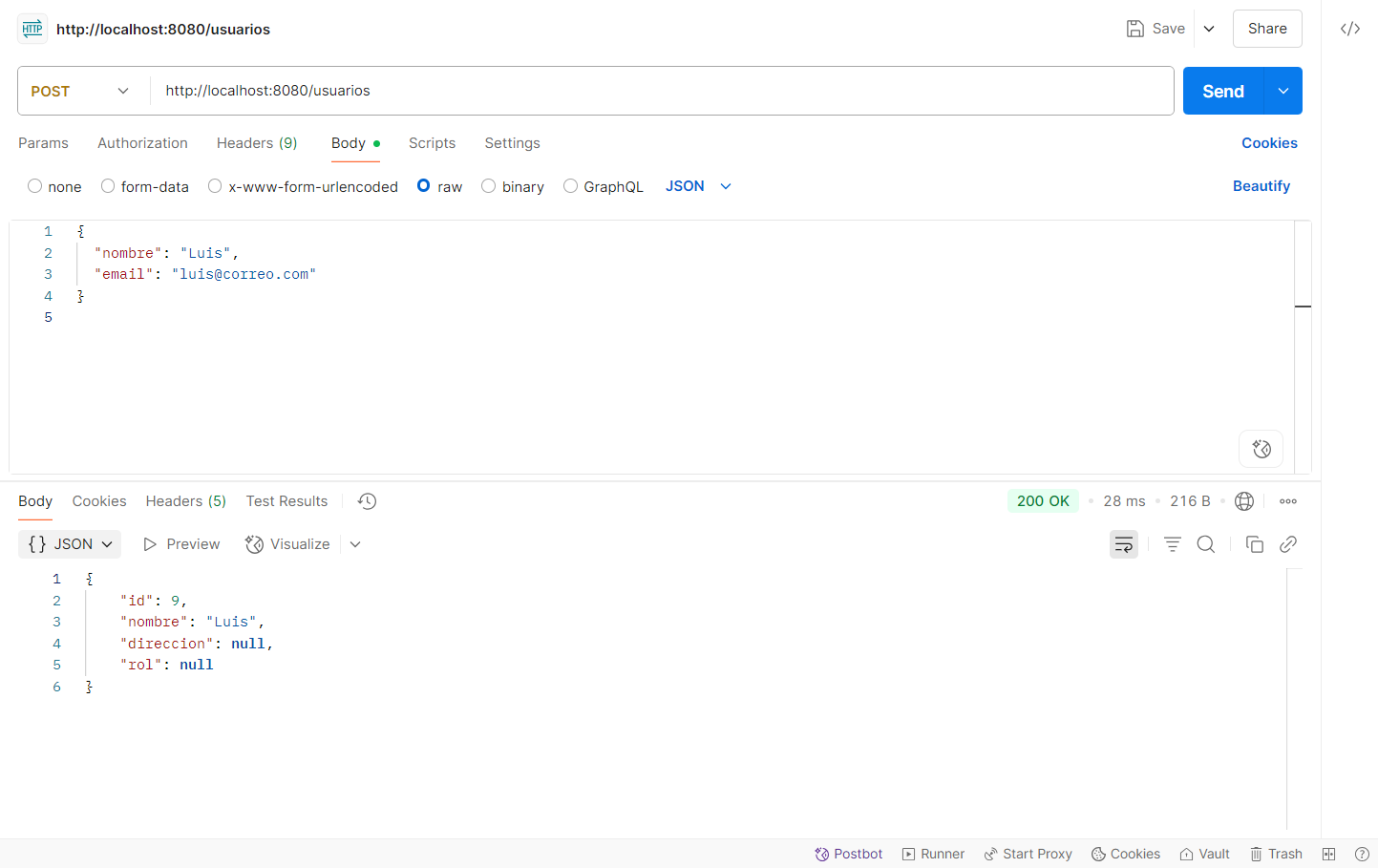
****

**Encontramos otro error por que acepta la validación a pesar que no tiene un nombre escrito**

**13. Crear usuario sin rol definido**

 **Método:** POST

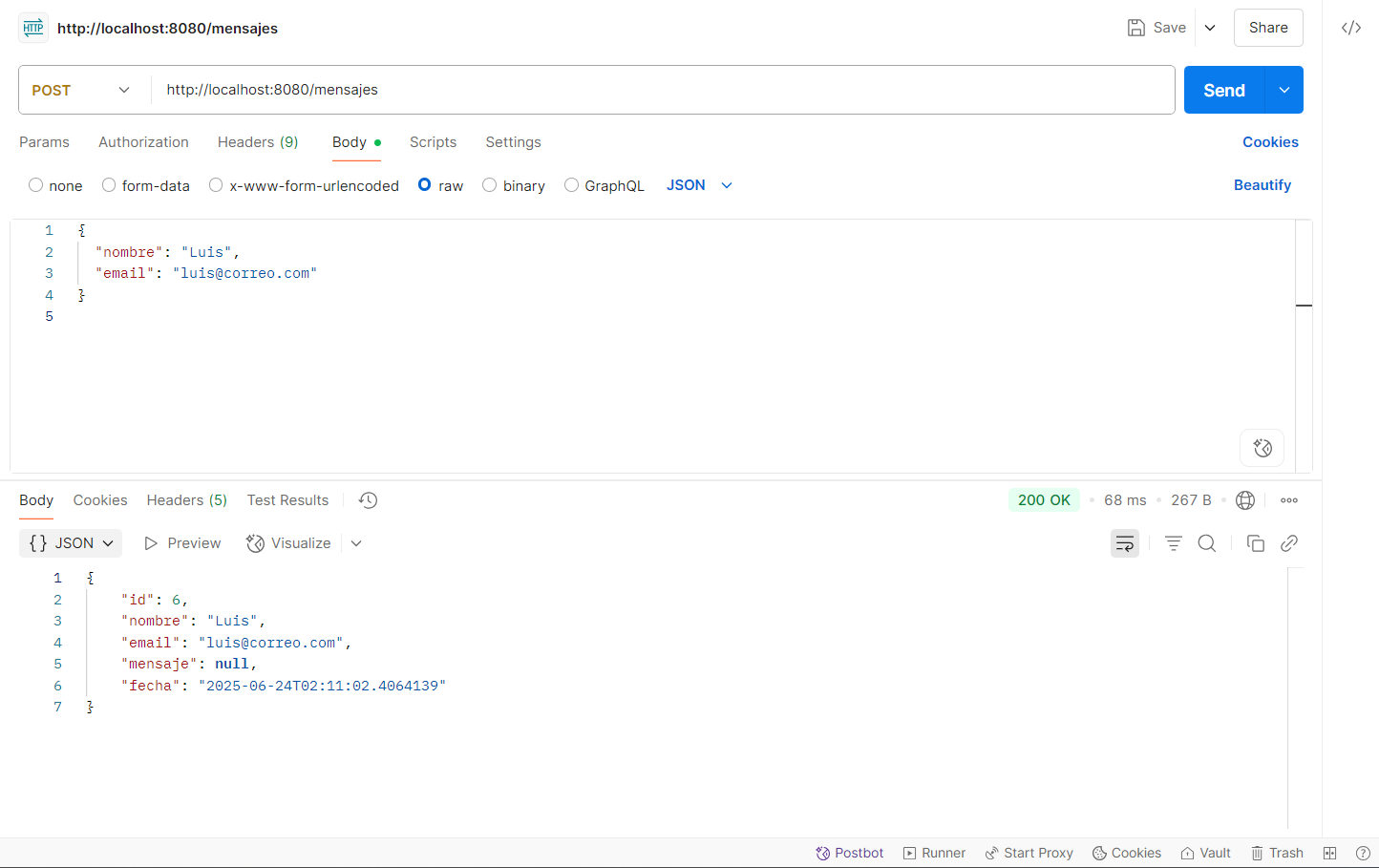
 **URL:** http://localhost:8080/usuarios

****

**Encontramos un error por que el Rol debe ser obligatorio, algo que se debe corregir**

**14. Enviar mensaje sin contenido**

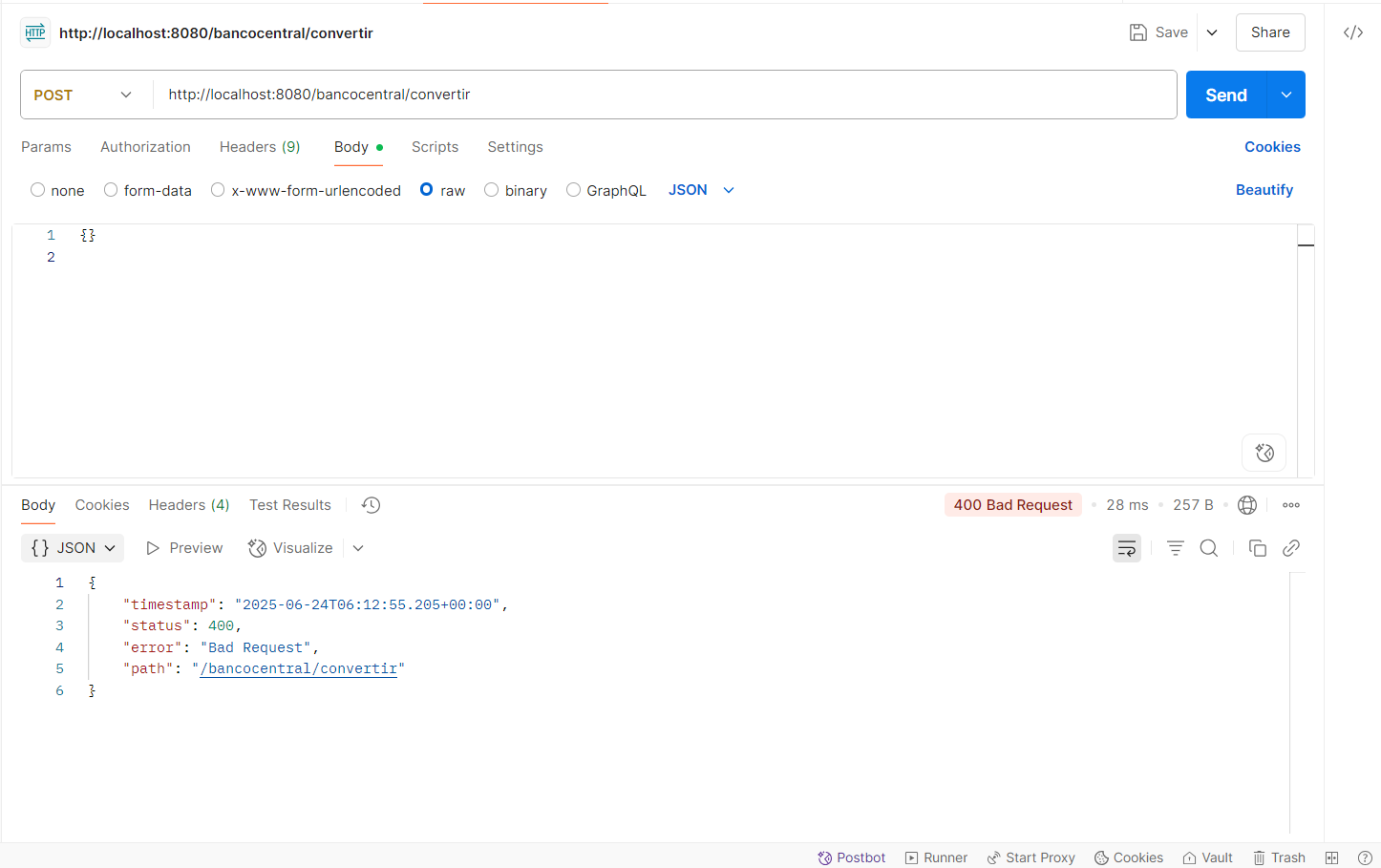
* **Método: POST**
* **URL: http://localhost:8080/mensajes**

****

No debería permitir enviar la validación sin un mensaje claro

**15. Convertir dólares sin monto**

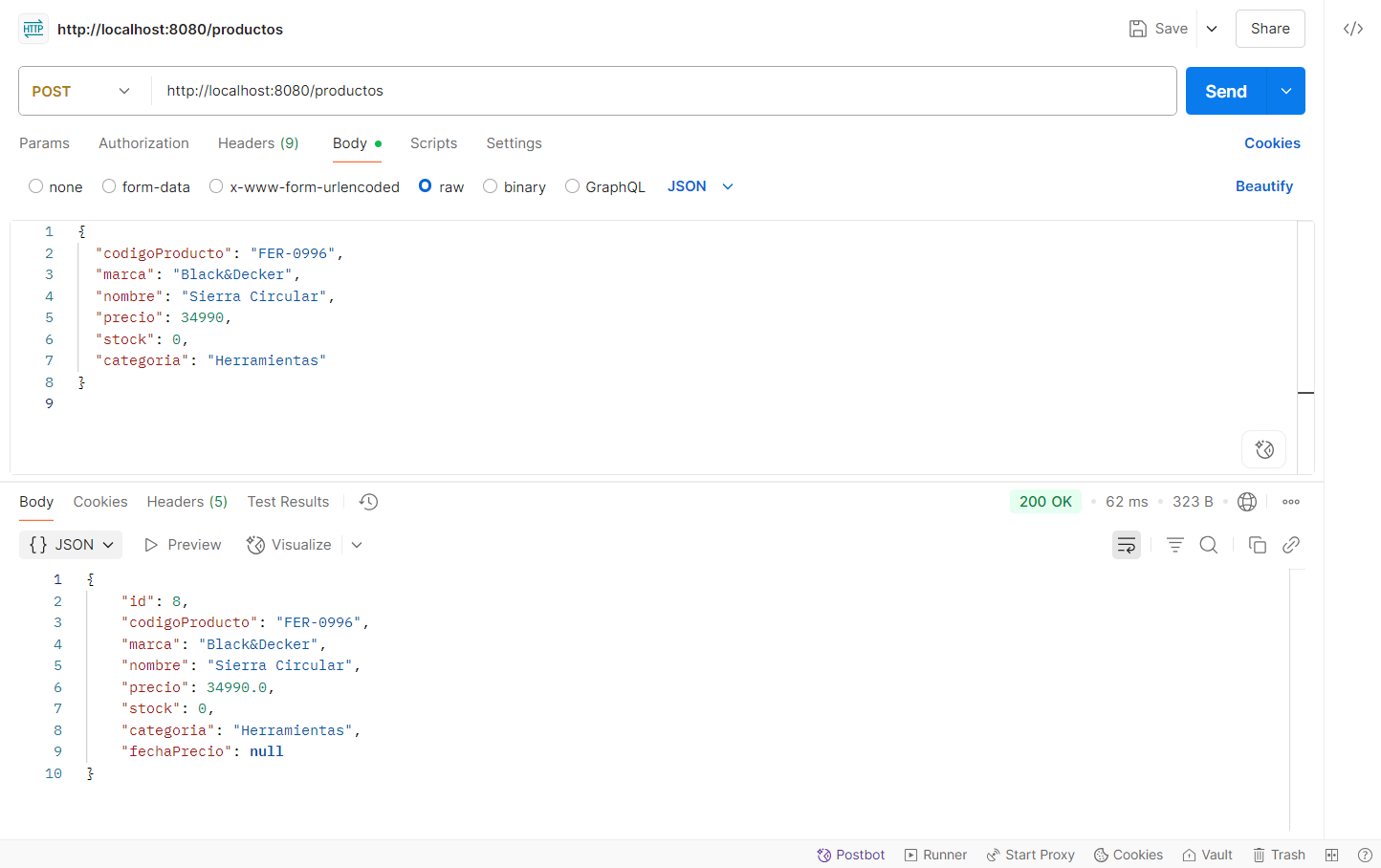
* **Método: POST**
* **URL: http://localhost:8080/bancocentral/convertir**



No permite validar sin un monto antes

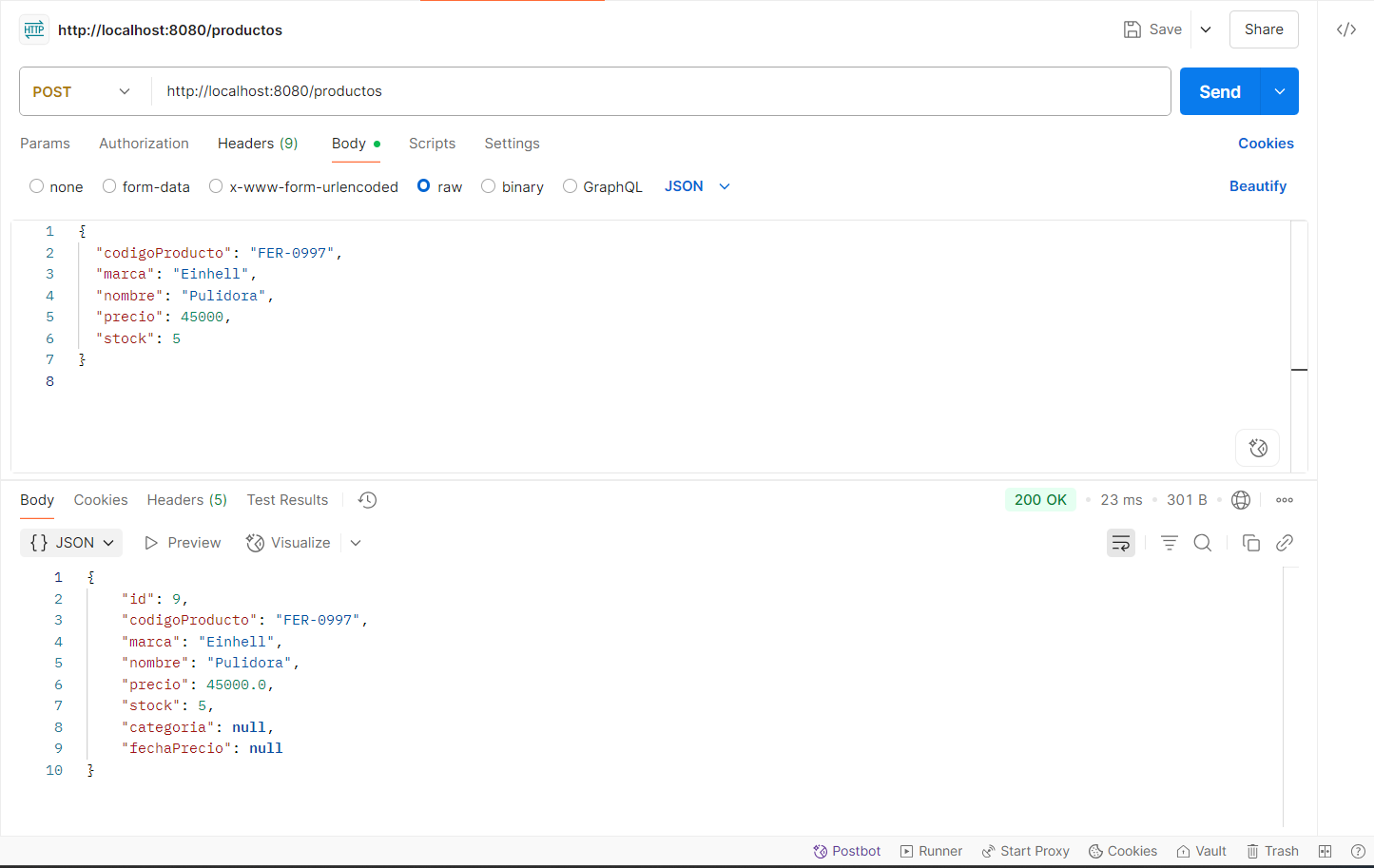
**16. Crear producto con stock = 0 (caso límite)**

* **Método: POST**
* **URL: http://localhost:8080/productos**

****

**16. Crear producto sin categoría (dato faltante)**

* **Método: POST**
* **URL: http://localhost:8080/productos**



No debería permitir la validación del producto sin categoría

Las pruebas realizadas en Postman sirvieron para validar el sistema desde un entorno externo, simulando cómo un cliente real consumiría la API REST.

Además de repetir algunos flujos principales, se diseñaron **pruebas nuevas no contempladas en Swagger**, principalmente para detectar errores de validación:

| **N°** | **Descripción breve** | **Resultado** |
| --- | --- | --- |
| 11 | Producto con precio negativo | ❌ Error: aceptado sin restricción |
| 12 | Producto sin nombre | ❌ Error: aceptado sin validación |
| 13 | Usuario sin rol | ❌ Error: se permite creación |
| 14 | Mensaje sin contenido | ❌ Error: se acepta mensaje vacío |
| 15 | Conversión sin montoUSD | ❌ Error: no se valida campo |
| 16 | Producto sin categoría | ❌ Error: se registra sin validación |
| 17 | Usuario externo accede a mensajes internos | ❌ Error: problema de rol detectado |

Estas pruebas fueron muy útiles para identificar debilidades en la capa de validaciones del backend.

# 

# Medición de Efectividad

## Resultados globales:

Durante la ejecución del plan de pruebas, se comprobó que el sistema Ferremas cumple correctamente con la mayoría de las funcionalidades esperadas. Sin embargo, al incorporar pruebas desde un entorno externo utilizando Postman, fue posible identificar fallos importantes en la validación de datos y seguridad.

**Principales observaciones:**

* El sistema **acepta entradas inválidas**, como precios negativos, campos vacíos y datos obligatorios faltantes.
* Se identificó una **falla de seguridad** crítica: un usuario externo pudo acceder a información restringida destinada solo a usuarios internos.
* No se están aplicando correctamente las validaciones en los controladores y modelos.

Estas fallas demuestran que, aunque la lógica funcional se cumple, aún es necesario reforzar las **validaciones internas y restricciones de acceso**, para garantizar la estabilidad, seguridad y calidad final del sistema.

## Métricas:

| **Indicador** | **Valor** |
| --- | --- |

|  |  |
| --- | --- |
| Total de pruebas ejecutadas | 41 |

|  |  |
| --- | --- |
| Pruebas unitarias exitosas | 12 |

|  |  |
| --- | --- |
| Pruebas de integración exitosas | 13 |

|  |  |
| --- | --- |
| Pruebas realizadas con Postman | 16 |

|  |  |
| --- | --- |
| Pruebas Postman exitosas | 9 |

|  |  |
| --- | --- |
| Pruebas fallidas (total) | 7 |

|  |  |
| --- | --- |
| Errores detectados mediante Postman | 6 |

|  |  |
| --- | --- |
| Porcentaje de éxito general del sistema | **82.9%** |

# Conclusiones

El sistema es funcional en sus principales tareas y relaciones entre módulos.  
Sin embargo, al ejecutar pruebas manuales y externas, fue posible descubrir errores que habrían pasado desapercibidos si solo se usaba Swagger.

**Postman resultó clave** para validar el sistema desde una visión de cliente real, encontrando 6 errores que no se veían desde Swagger.  
Con estas observaciones, el equipo concluye que el sistema cumple su propósito, pero debe reforzarse la validación de datos y seguridad por roles antes de salir a producción.