



ESCUELA DE
INFORMÁTICA Y
TELECOMUNICACIONES



PERFULANDIA

**INFORME:
PERFULANDIA MANAGEMENT**

Nombre del Proyecto: "Perfulandia Management"
Integrantes: Martín Baza, Nicolás Bello, Rodrigo Vargas
Fecha de Entrega: 28-05-2025
Link Repositorio Github: [Perfulandia](#)

Índice

1. Introducción.....	4
2. Contexto de la problemática.....	4
3. Requisitos funcionales.....	5
3.1. Administrador del Sistema.....	5
3.2. Gerente de Sucursal.....	5
3.3. Empleado de Ventas.....	5
3.4. Logística.....	6
3.5. Clientes.....	7
4. Requisitos no funcionales.....	8
5. Entrevistas simuladas.....	9
5.1. Administrador del Sistema.....	9
5.2. Gerente de Sucursal.....	10
5.3. Usuario Frecuente.....	11
5.4. Area Logistica.....	11
5.5. Administración de Ventas.....	12
6. Descripción de Fallas.....	14
6.1. Fo1 - Caídas del sistema durante altas compras simultáneas.....	14
6.2. Fo2 - Lentitud por consultas simultáneas a una única base de datos.....	14
6.3. Fo3 - Fallos del sistema con accesos concurrentes desde múltiples departamentos..	14
6.4. Fo4 - Inhabilitación total ante caídas del sistema.....	14
6.5. Fo5 - Errores de stock por falta de sincronización entre sucursales.....	15
6.6. Fo6 - Reportes de ventas incompletos o desactualizados.....	15
6.7. Fo7 - Dificultad para escalar o actualizar módulos.....	16
6.8. Fo8 - Falta de respaldo de datos ante fallos o desastres.....	16
7. Principales Cuellos de Botella.....	17
7.1. CB01 - Sobrecarga del Servidor Único.....	17
7.2. CB02 - Interfaz Genérica para Todos los Departamentos.....	17
7.3. CB03 - Base de Datos Monolítica y Congestionada.....	17
7.4. CB04 - Arquitectura Monolítica Rígida e Inescalable.....	17
7.5. CB05 - Falta de Sincronización en Tiempo Real.....	17
7.6. CB06 - Procesamiento Secuencial en Transacciones.....	17
7.7. CB07 - Generación de Reportes Lenta y con Datos Incompletos.....	18
8. Causas Técnicas.....	19
8.1. CT01 - Arquitectura monolítica rígida.....	19
8.2. CT02 - Ausencia de pruebas y respaldo automático.....	19
8.3. CT03 - Riesgo en actualizaciones debido a arquitectura monolítica.....	19
8.4. CT04 - Base de datos centralizada y poco optimizada.....	19
8.5. CT05 - Problemas de rendimiento en horarios punta.....	19
8.6. CT06 - Sin separación entre capas (backend, frontend y base de datos).....	20
8.7. CT07 - Base de datos centralizada sin optimización.....	20
9. Causas Organizativas.....	21
9.1. CO01 - Falta de un equipo técnico especializado.....	21

9.2. CO02 - Procesos manuales de actualización y despliegue.....	21
9.3. CO03 - Falta de capacitación del personal en el uso del sistema.....	21
9.4. CO04 - Procesos no estandarizados entre sucursales.....	21
9.5. CO05 – No existe una cultura de mejora continua.....	22
10. Propuesta de Servicios.....	23
10.1. SO1 - Servicio de Seguridad y Autenticación.....	23
10.2. SO2 - Servicio de Inventario.....	23
10.3. SO3 - Servicio de Gestión de Ventas.....	23
10.4. SO4 - Servicio de Compras y proveedores.....	23
10.5. SO5 - Servicio de Reportes.....	23
10.6. SO6 - Servicio de Integración (mediante API).....	23
10.7. SO7 - Servicio de Gestión de Usuarios y Clientes.....	23
11. Diagramas.....	25
11.1. Diagrama de Casos de Uso.....	25
11.2. Diagrama de Clases.....	26
11.3. Diagrama de arquitectura de Microservicios.....	26
12. Plan de Migración.....	27
12.1. Fase 1: Análisis del Sistema Actual.....	27
12.2. Fase 2: Definición de Microservicios.....	30
12.3. Fase 3: Prueba de Concepto.....	31
12.4. Fase 4: Reestructuración Incremental.....	32
12.5. Fase 5: Despliegue en Contenedores.....	33
12.6. Fase 6: Integración Continua y Despliegue Continuo (CI/CD).....	33
12.7. Fase 7: Eliminación del Monolito.....	33
13. Configuración de Bases de Datos.....	34
14. Resultados de Pruebas con Postman.....	38
15. Conclusión.....	48

1. Introducción

En el presente documento se detallarán los diferentes puntos que hacen referencia al desarrollo de una solución informática para la empresa “Perfulandia SPA”. Esta empresa se ha desarrollado en un entorno competitivo durante los últimos años en su sede inicial la cual se encuentra en el Barrio Meiggs, Santiago. Sin embargo, el crecimiento y expansión que han experimentado durante este último periodo (los cuales incluyen nuevas sucursales en Concepción y Viña del Mar) han dejado en evidencia las limitaciones que se han ido acumulando debido a la forma en la que el actual sistema que se emplea ya no es capaz de mantener un nivel de rendimiento óptimo frente a la demanda.

2. Contexto de la problemática

En el caso específico de Perfulandia SPA, esta entidad ha surgido exitosamente dentro del mercado de perfumes, en el cual proveen a sus clientes de productos de alta calidad a precios altamente competitivos, lo cual los ha consolidado como una importante empresa a nivel nacional de este rubro, tanto así que supone una expansión urgente dentro de los sistemas con los que trabajan. Este crecimiento exponencial que ha sufrido la empresa ha dejado en evidencia las falencias de la estructura sobre la que se ha desarrollado su software monolítico, teniendo este fallas en aspectos no sólo de rendimiento, sino que adicionalmente la operatividad y estabilidad del mismo se han visto severamente afectadas, dejando a una multitud de clientes sin servicio por tiempos prolongados los cuales afectan directamente a la reputación de Perfulandia.

En síntesis, el software monolítico de Perfulandia presenta fallas por la creciente expansión debido al éxito y necesita ser reestructurado a una arquitectura basada en microservicios. Con este documento esperamos proveer una solución eficiente y duradera para que la empresa se mantenga en el tiempo y puedan brindar sus servicios de manera satisfactoria a sus clientes.

3. Requisitos funcionales

3.1. Administrador del Sistema

- **Gestión de Usuarios:** El sistema debe permitirle al administrador del sistema crear, actualizar, desactivar y eliminar cuentas de usuarios del sistema.
- **Configuración de Permisos:** El sistema debe permitir asignar y modificar permisos de acceso a diferentes módulos y funciones del sistema.
- **Monitorización del Sistema:** El sistema debe permitir visualizar el estado del sistema mediante una ventana que muestre el rendimiento del mismo, y recibir alertas sobre posibles fallos.
- **Respaldo y restauración de datos:** El sistema debe permitir realizar copias de seguridad periódicas y restaurar datos en caso de pérdida o fallo.

3.2. Gerente de Sucursal

- **Gestión de Inventario:** El sistema debe permitir al gerente agregar, buscar, modificar, y eliminar productos del inventario, así como ajustar el stock y sus respectivas cantidades.
- **Generación de reportes:** El sistema debe permitir al gerente realizar reportes de ventas, inventario y rendimiento de la sucursal en la que se desempeña.
- **Gestión de las sucursales:** El sistema debe permitir configurar detalles específicos de la sucursal, como horarios de apertura, personal asignado, y políticas locales.
- **Gestión de pedidos:** El sistema debe permitir supervisar y autorizar pedidos de productos para reabastecer el inventario de la sucursal.

3.3. Empleado de Ventas

- **Registro de las ventas:** El sistema debe permitir procesar transacciones de venta en el sistema, aplicando descuentos y ofertas cuando sea necesario.

- **Atención de devoluciones y reclamaciones:** El sistema debe permitir el procesamiento de devoluciones de productos y gestionar reclamaciones de clientes.
- **Consultar Inventario:** El sistema debe permitir la verificación de disponibilidad de productos en tiempo real y buscar productos en el inventario.
- **Generar Facturas:** El sistema debe permitir la emisión de facturas electrónicas y enviarlas por correo electrónico a los clientes.

3.4. Logística

- **Gestionar envíos:** El sistema debe permitir crear, actualizar y seguir envíos de productos desde la bodega hasta las sucursales y los clientes.
- **Optimizar rutas de entrega:** El sistema debe permitir planificar y optimizar rutas de entrega para los pedidos locales y regionales.
- **Actualizar estado de pedidos:** El sistema debe permitir cambiar el estado de los pedidos en el sistema, desde el procesamiento hasta la entrega final.
- **Gestionar proveedores:** El sistema debe permitir mantener y actualizar la información de proveedores, realizar pedidos de reabastecimiento y gestionar la recepción de mercancías.

3.5. Clientes

- **Creación de cuenta:** El sistema debe permitirle a los clientes registrarse en la plataforma web proporcionando la información necesaria como nombre, dirección de correo electrónico y dirección de envío.
- **Inicio de sesión:** El sistema debe permitir acceder a la cuenta utilizando las credenciales de usuario.
- **Navegación y búsqueda de productos:** El sistema debe permitir explorar el catálogo de productos, utilizando filtros de búsqueda para encontrar productos específicos.
- **Agregar productos al carrito:** El sistema debe permitir seleccionar productos y añadirlos al carrito de compras.
- **Realizar pedidos:** El sistema debe permitir completar el proceso de compra, proporcionando detalles de pago y seleccionando opciones de envío.
- **Consultar historial de pedidos:** El sistema debe permitir ver el historial de compras anteriores y el estado actual de los pedidos en curso.
- **Gestionar perfil:** El sistema debe permitir actualizar información personal, direcciones de envío y detalles de pago.
- **Solicitar soporte:** El sistema debe permitir enviar consultas o problemas a través de un formulario de contacto o chat en línea.
- **Dejar reseñas y calificaciones:** El sistema debe permitir evaluar productos comprados y dejar comentarios en la web.
- **Aplicar cupones y descuentos:** El sistema debe permitir ingresar códigos promocionales durante el proceso de compra para recibir descuentos.

4. Requisitos no funcionales

A continuación, se detallan los requisitos no funcionales a desarrollar durante esta primera versión del proyecto:

- **Escalabilidad:** El sistema debe permitir el aumento progresivo de clientes de todos los tipos sin sufrir pérdidas de rendimiento.
- **Rendimiento:** El sistema debe permitir el procesamiento de las ventas en un máximo de 2 segundos cuando la carga sea normal, es decir, no superior a 1000 transacciones simultáneas.
- **Seguridad:** Todos los datos sensibles, tanto credenciales de acceso de usuarios como de clientes, contraseñas, medios de pago, etc, deben estar cifrados para mantener su respectiva confidencialidad. Para complementar este requisito también debe cumplir con la normativa Chilena de protección de datos (Ley 19.628), lo que a su vez debe permitir auditorías de acceso.
- **Disponibilidad:** El sistema debe permitir su funcionamiento continuo garantizado en al menos un 99% en el horario correspondiente (en este caso horario comercial, el cual comienza a las 08:00hrs y culmina a las 20:00). Además, si hubieran fallas estas no deben suponer un tiempo de recuperación mayor a 15 minutos.
- **Integración:** El sistema debe permitir la interoperabilidad con sistemas de pago externos, como por ejemplo WebPay y/o TransBank.
- **Usabilidad:** La interfaz de usuario debe permitir la realización de una venta en menos de 3 pasos para usuarios frecuentes. Además, debe seguir el estándar UI/UX de entidades como Google para asegurar intuitividad.
- **Mantenibilidad:** El código debe estar documentado en su totalidad, bajo un estándar (en este caso Swagger dado que trabajaremos con APIs).
- **Compatibilidad:** El sistema debe ser compatible con al menos un par de navegadores actuales, tales como Opera GX y Chrome y sistemas operativos como lo serían Windows y Linux.
- **Resiliencia:** El sistema debe permitir operar en modo local (sin conexión) y sincronizar los datos con la unidad central al restaurar la conexión.

Todos estos requerimientos se han establecido con el fin de atacar directamente a los problemas que se han planteado en el contexto de Perfulandia, y responden netamente a las necesidades que se han detallado, aquellas como por ejemplo la expansión a nivel nacional, el cumplimiento legal y la satisfacción de los usuarios y clientes.

5. Entrevistas simuladas

5.1. Administrador del Sistema

1. ¿Qué tipo de problemas enfrentan actualmente al gestionar usuarios y sus permisos en el sistema?

R: Todos los usuarios de un mismo rol tienen exactamente los mismos accesos, sin poder personalizarlos según sus tareas. Además, la creación o desactivación de usuarios es lenta y no siempre refleja los cambios en tiempo real.

2. ¿Con qué frecuencia necesitas hacer respaldos, y cómo es el proceso actual de restauración de datos?

R: Hacemos respaldos manuales una vez por semana, lo cual no es ideal. La restauración también es manual y poco confiable.

3. ¿Tienes visibilidad en tiempo real del estado del sistema? ¿Cómo detectar fallas o caídas?

R: No, generalmente me entero de fallas cuando los usuarios comienzan a reportarlas.

4. ¿Qué tan flexible te resulta configurar nuevos perfiles de usuario o funciones específicas?

R: Muy limitado. Solo puedo usar los perfiles predefinidos, y no puedo crear subroles ni asignar permisos especiales.

5. ¿Qué tan seguro consideras que es el sistema actual frente a accesos no autorizados?

R: La seguridad es básica. Es altamente débil a ataques cuando hay alta demanda en la web.

5.2. Gerente de Sucursal

1. ¿Cómo gestionas actualmente el inventario y qué dificultades has tenido con el sistema?

R: El sistema tiene un módulo de inventario, pero tarda en actualizarse entre sucursales. Si una venta ocurre en otra ciudad, el stock no se refleja inmediatamente.

2. ¿Qué tipo de reportes necesitas generar con más frecuencia?

R: Reportes de ventas diarias y productos con bajo stock. El sistema los puede generar, pero toma mucho tiempo y en horas de alta demanda se vuelve lento.

3. ¿Cómo manejas los pedidos de reabastecimiento?

R: Desde el sistema puedo hacer solicitudes, pero no hay un flujo claro para seguimiento. A veces no queda claro si un pedido fue aceptado o si está en camino.

4. ¿Qué tanta autonomía tienes para configurar aspectos de tu sucursal?

R: Casi nula. La configuración está centralizada. Cualquier cambio, como horario o asignación de personal, depende del administrador del sistema.

5. ¿Hay alguna funcionalidad que desees y que el sistema actual no tenga?

R: Me gustaría poder ver en tiempo real el stock entre sucursales, tener alertas automáticas y más control local sobre nuestra propia operación.

5.3. Usuario Frecuente

1. ¿Cómo ha sido tu experiencia navegando y buscando productos en el sitio web?

R: Es buena, pero a veces el buscador no arroja resultados relevantes. Faltan filtros más específicos.

2. ¿Tuviste algún problema al registrarte o iniciar sesión?

R: El sistema suele tener problemas en horarios específicos, asumo que será por alta demanda de usuarios simultáneos.

3. ¿Qué tan claro y fácil te parece el proceso de compra y pago?

R: En general bien, pero sería útil tener más opciones de pago y una mejor confirmación después de la compra.

4. ¿Has utilizado cupones o descuentos? ¿El sistema los aplicó correctamente?

R: Sí, pero una vez el cupón no se aplicó correctamente y no hubo forma de corregirlo sin contactar soporte.

5. ¿Qué tan útil te parece la opción de consultar el estado de tus pedidos?

R: Me gusta, pero la información no siempre está actualizada y no incluye detalles del envío.

5.4. Area Logistica

1. ¿Cómo gestionan los envíos y qué tan automatizado es el proceso?

R: Los envíos se gestionan desde el sistema, pero el módulo es muy básico. No hay seguimiento real ni integración con rutas de entrega.

2. ¿Qué desafíos enfrentan al planificar rutas de entrega?

R: El sistema solo permite registrar direcciones, pero no tiene funciones para optimizar rutas o agrupar entregas por zona.

3. ¿Con qué frecuencia actualizas el estado de los pedidos, y cómo se refleja en el sistema?

R: Actualizamos los estados cuando el sistema responde, pero a veces se cae. Los estados no siempre se actualizan en el portal del cliente.

4. ¿Cómo es la comunicación con los proveedores?

R: El sistema permite registrar pedidos a proveedores, pero no tiene funciones para enviar notificaciones ni confirmar entregas. Todo eso lo seguimos haciendo por correo o teléfono.

5. ¿Crees que una mayor visibilidad del inventario en tiempo real facilitaría tu trabajo?

R: Sí, evitaría errores de envío y ayudaría a planificar mejor los despachos.

5.5. Administración de Ventas

1. ¿Qué tan fácil o difícil es procesar una venta en el sistema actual?

R: El sistema funciona, pero es pesado y poco intuitivo. A veces, durante promociones o días con alto flujo de usuarios, se pone muy lento o se cae.

2. ¿Has tenido problemas al emitir facturas electrónicas o al aplicar descuentos?

R: Sí, las funciones están dentro del sistema, pero son poco amigables. Si aplico un descuento mal, tengo que anular toda la venta.

3. ¿Con qué frecuencia necesitas consultar el inventario y cómo es la experiencia?

R: Todo el día. Pero a veces el stock no está actualizado o hay diferencias entre lo que muestra el sistema y lo que tenemos físicamente, al aplicar cambios muchas veces se hace de manera muy lenta o el sistema se cae.

4. ¿Qué tan rápido puedes gestionar una devolución o reclamación de un cliente?

R: El sistema tiene la opción, pero es muy rígida. No permite diferentes razones de devolución ni comentarios. Se siente como una solución genérica para un proceso complejo.

5. ¿Te gustaría que el sistema ofreciera sugerencias automáticas de productos relacionados o descuentos?

R: Claro. Actualmente no hay ninguna inteligencia o recomendación automática. Todo lo tengo que saber yo como vendedor.

6. Descripción de Fallas

6.1. Fo1 - Caídas del sistema durante altas compras simultáneas

El sistema se cae en los momentos en que se realizan muchas compras a la vez en las distintas sucursales.

- Usuario (Cajero o vendedor) : No puede realizar la compra o se demora, esto genera largas filas y descontento en los clientes.
- Sistema : No óptimo para múltiples operaciones simultáneas.

6.2. Fo2 - Lentitud por consultas simultáneas a una única base de datos

Todas las consultas se realizan a una misma base de datos, lo que provoca que la solicitud de datos en horas de alto flujo sea lenta.

- Usuario : Demora en terminar el trabajo, atrasos en todos los procesos posteriores.
- Sistema : No tiene redundancia ni mecanismos de recuperación ante fallos.

6.3. Fo3 - Fallos del sistema con accesos concurrentes desde múltiples departamentos

El sistema se cae en momentos en que varios departamentos interactúan con él y la base de datos.

- Usuario : Hacer procesos a mano, para resguardar el trabajo y no demorar los procesos.
- Sistema : No tiene redundancia ni mecanismos de recuperación ante fallos.

6.4. Fo4 - Inhabilitación total ante caídas del sistema

En momentos en que se cae el sistema, todos los usuarios quedan inhabilitados, desde los de venta hasta los administradores.

- Usuario : Hacer procesos a mano, para resguardar el trabajo y no demorar los procesos.
- Sistema : No tiene redundancia ni mecanismos de recuperación ante fallos.

6.5. F05 - Errores de stock por falta de sincronización entre sucursales

Error en stock total, al haber fallas en la sincronización entre sucursales.

- Usuario : Se promete un producto que ya no está disponible en esa sucursal.
- Sistema : No hay integración en tiempo real entre los sistemas de las tiendas.

6.6. F06 - Reportes de ventas incompletos o desactualizados

Reportes de ventas imprecisos.

- Usuario (administrador): El reporte no incluye las ventas del día completo.
- Sistema: El procesamiento de datos no es automático ni está en tiempo real.

6.7. Fo7 - Dificultad para escalar o actualizar módulos

Imposibilidad o es muy complejo de escalar el sistema, debido a que para agregar algo nuevo o actualizar algún módulo, hay que inhabilitar y modificar gran parte del mismo.

- Usuario (administrador): No podemos hacer mejoras sin afectar el sistema completo.
- Sistema: Arquitectura monolítica rígida y difícil de mantener.

6.8. Fo8 - Falta de respaldo de datos ante fallos o desastres

Sin respaldo automático, no hay resguardo de datos ante desastres.

- Usuario (Gerencia): Si perdemos datos, no tenemos cómo recuperarlos.
- Sistema: Falta un sistema de backups programados y recuperación ante fallos.

7. Principales Cuellos de Botella

7.1. CBo1 - Sobrecarga del Servidor Único

Todas las funciones y procesos se hacen, dependen de un único servidor, lo que produce sobrecargas frecuentemente.

7.2. CBo2 - Interfaz Genérica para Todos los Departamentos

El interfaz, diseño para todos los departamentos es el mismo, lo que lo hace menos intuitivo, y dificulta la diferenciación de trabajos.

7.3. CBo3 - Base de Datos Monolítica y Congestionada

Todos los datos se guardan en una misma base de datos haciendo que los procesos de guardado y consulta sean lentos o provoquen caídas del sistema.

7.4. CBo4 - Arquitectura Monolítica Rígida e Inescalable

Todo está construido como un solo bloque, desde el apartado de ventas, hasta el apartado de monitoreo y cambios a los usuarios, lo que trae como consecuencias que sea inviable la evolución tecnológica y limita la capacidad de escalar por módulos.

7.5. CBo5 - Falta de Sincronización en Tiempo Real

No existe sincronización automática y en tiempo real, lo que provoca diferencias de stock y datos desactualizados.

7.6. CBo6 - Procesamiento Secuencial en Transacciones

Cada transacción ejecuta múltiples procesos, esto en periodos de alto flujo provoca colapsos en el sistema, lentitud e incluso caídas.

7.7. CBo7 - Generación de Reportes Lenta y con Datos Incompletos

Los reportes de ventas, inventario y desempeño comercial se calculan desde la base de datos sin mecanismos de resumen, a veces inclusive con datos desactualizados.

8. Causas Técnicas

8.1. CTo1 - Arquitectura monolítica rígida

El sistema es monolítico (está construido y agrupado en un solo bloque vertical), lo que hace complicado o imposible separar las funcionalidades y escalarlos por módulos.

8.2. CTo2 - Ausencia de pruebas y respaldo automático

No hay pruebas y guardado automático, esto hace el sistema vulnerable ante desastres (situaciones inesperadas donde el sistema se cae o falla).

8.3. CTo3 - Riesgo en actualizaciones debido a arquitectura monolítica

No hay guardado automático, el sistema está construido y agrupado en un solo bloque, debido a esto hace complicado hacer mejoras y actualizaciones al sistema, dado que significa inhabilitar todo el sistema y cualquier fallo puede perder muchos datos pues el sistema no tiene respaldo de seguridad.

8.4. CTo4 - Base de datos centralizada y poco optimizada

Todos los datos y procesos se hacen hacia una misma base de datos, esto hace que los procesos sean lentos y poco óptimos.

8.5. CTo5 - Problemas de rendimiento en horarios punta

En tiempos de alto flujo de datos y procesos el sistema se cae, es lento o falla, esto debido a la falta de balanceo de carga (técnica que distribuye el tráfico entre varios servidores para mejorar el rendimiento de la red) y la incapacidad de hacerlo a su vez de manera eficiente.

8.6. CTo6 - Sin separación entre capas (backend, frontend y base de datos)

El sistema no cuenta con arquitectura en capas (Facilita la organización, el mantenimiento, y la escalabilidad de los sistemas), lo que complica la organización del código, diferenciación de la interfaz de trabajo, provoca o aumenta la probabilidad de que el sistema falle, pues todo se encuentra mezclado y agrupado.

8.7. CTo7 - Base de datos centralizada sin optimización

Todas las operaciones dependen de una única base de datos central, que no ha sido optimizada para el aumento de consultas simultáneas. Esto genera cuellos de botella y latencia en las operaciones y procesos.

9. Causas Organizativas

9.1. CO01 - Falta de un equipo técnico especializado

La empresa no cuenta con equipo de desarrollo especializado y dedicado, en backend e infraestructura, debido a que el crecimiento de la empresa es rápido y fue creado inicialmente pensando en solo una sucursal y con recursos limitados, por esto ahora no hay suficiente personal capacitado para mantenerlo, adaptarlo y escalar.

9.2. CO02 - Procesos manuales de actualización y despliegue

Las actualizaciones se hacen de manera manual, esto genera errores frecuentes, pérdidas constantes y hace riesgosa cualquier intervención técnica.

9.3. CO03 - Falta de capacitación del personal en el uso del sistema

El dejar de ser un sistema monolítico, y la apertura de las nuevas sucursales afectará en que muchos usuarios del sistema (vendedores, encargados de bodega, cajeros) no han sido capacitados adecuadamente.

9.4. CO04 - Procesos no estandarizados entre sucursales

Las sucursales de Santiago, Concepción y Viña del Mar trabajan de forma distinta, sin procesos claros y unificados. Esto dificulta la implementación de mejoras tecnológicas centralizadas o integradas.

9.5. CO05 – No existe una cultura de mejora continua

La empresa no cuenta con un enfoque sistemático de revisión y mejora de procesos. Los cambios, mejoras, actualizaciones se hacen “sobre la marcha”, lo que no permite poder prepararse para prevenir los posibles errores, además de que hacer cambios solo cuando lo necesitas es menos eficiente y poco óptimo.

10. Propuesta de Servicios

10.1. SO1 - Servicio de Seguridad y Autenticación

Maneja el inicio de sesión, registro de usuario y validación de tokens.

10.2. SO2 - Servicio de Inventario

Gestiona el stock entrante y saliente en tiempo real según corresponda, además, sincroniza datos con otros módulos tales como ventas, proveedores y compras.

10.3. SO3 - Servicio de Gestión de Ventas

Gestiona tanto pedidos como transacciones, además de integrarse con otros microservicios para mantener y garantizar la consistencia en operaciones.

10.4. SO4 - Servicio de Compras y proveedores

Gestiona órdenes de compra, y se integra con el microservicio de inventario para validar el stock según sea necesario.

10.5. SO5 - Servicio de Reportes

Generar reportes personalizados según las necesidades del usuario consultando datos de otros microservicios dentro del sistema.

10.6. SO6 - Servicio de Integración (mediante API)

Facilita la comunicación y conexión de manera segura entre este microservicio y sistemas externos como por ejemplo pasarelas de pago (para este proyecto usaremos WebPay y Transbank).

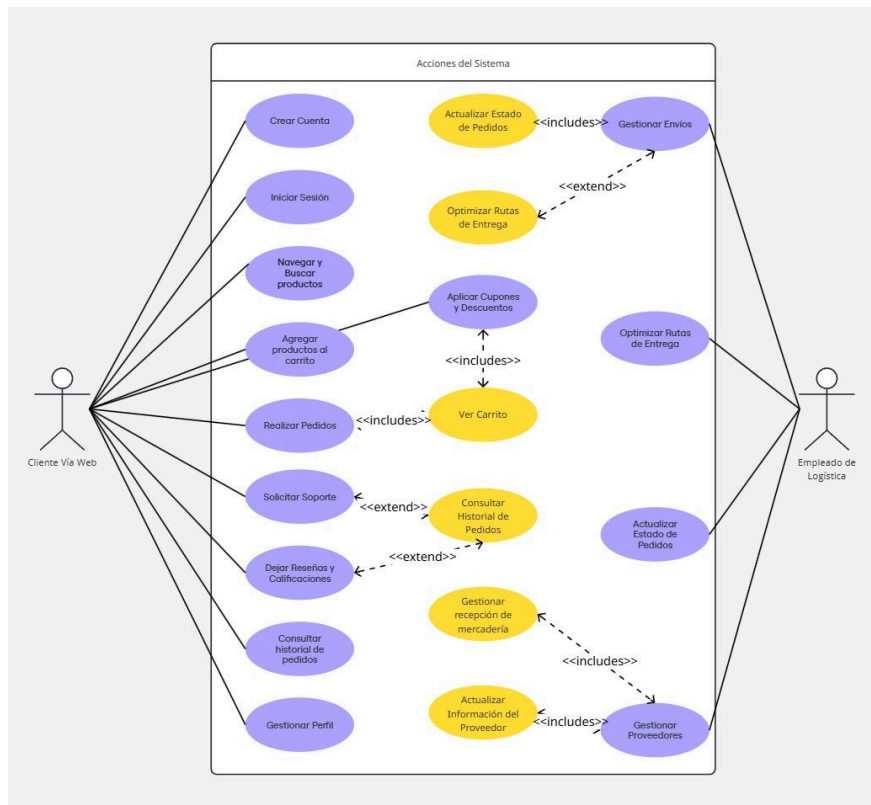
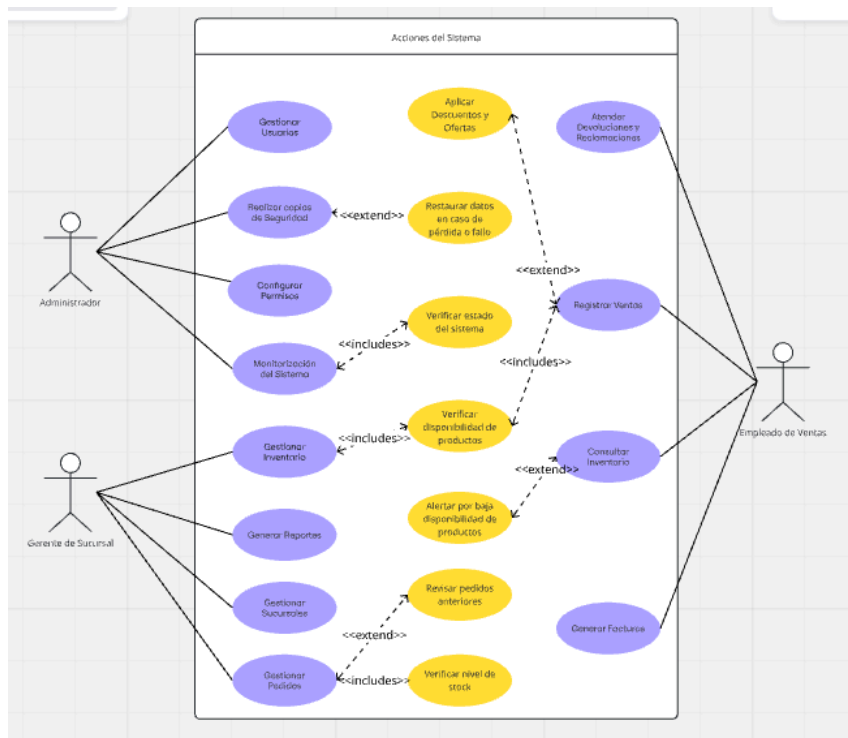
10.7. SO7 - Servicio de Gestión de Usuarios y Clientes

Gestiona todos los datos de usuarios y clientes, a su vez proporciona al administrador del sistema las facultades necesarias para otorgar permisos y roles por el lado de los usuarios del sistema. Por otra parte, proporciona APIs

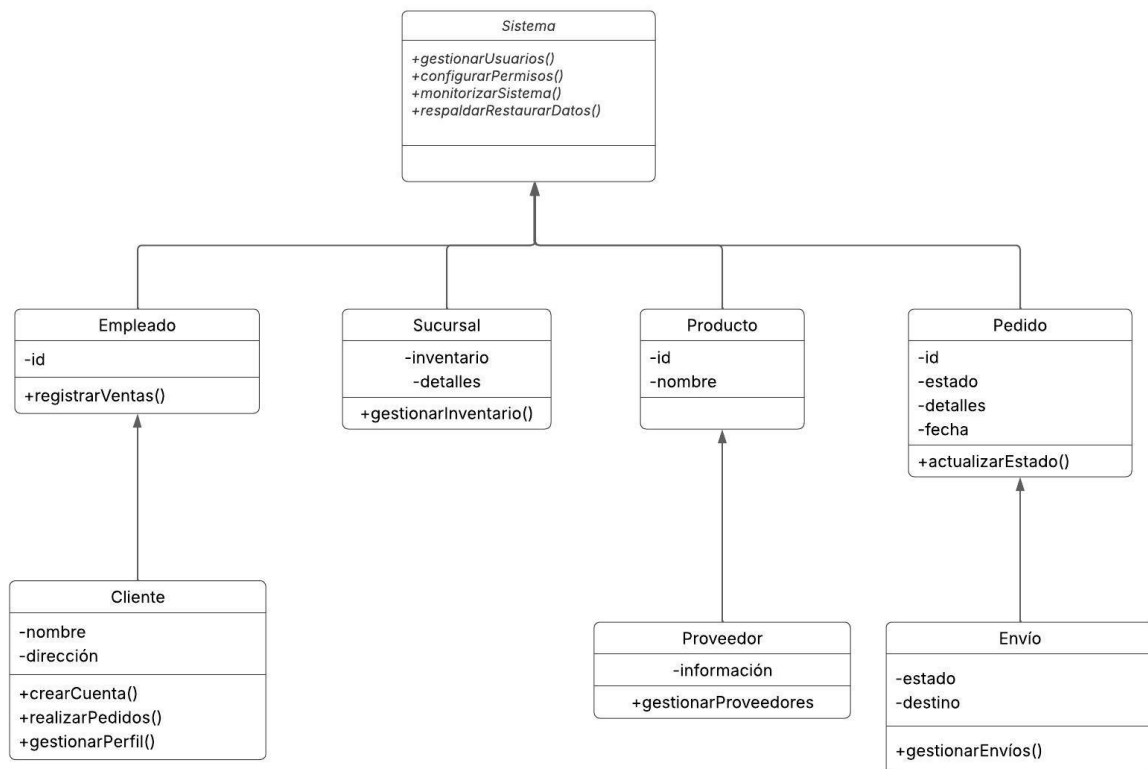
para su consumo con otros microservicios. Adicionalmente, centraliza aspectos tales como el registro y clasificación de datos de clientes.

11. Diagramas

11.1. Diagrama de Casos de Uso



11.2. Diagrama de Clases



11.3. Diagrama de arquitectura de Microservicios

12. Plan de Migración

12.1. Fase 1: Análisis del Sistema Actual

El sistema actual de Perfulandia SPA se estructura en un diseño monolítico, lo que ha producido problemas de distintas índoles a través del tiempo y la constante expansión de la empresa. Por lo tanto, a continuación se detallarán de manera concisa los distintos módulos que lo componen:

- **Módulo de Gestión de Ventas:** Procesa pedidos y ventas al por menor y por mayor. Este se relaciona con los módulos de clientes, inventario y reportes.
- **Módulo de Inventario:** Gestiona el stock que ingresa y sale de bodega por sucursal. Este se relaciona con los módulos de ventas, compras y proveedores además de reportes.
- **Módulo de Clientes:** Gestiona datos de todo tipo de clientes. Este se relaciona con los módulos de ventas y reportes.
- **Módulo de Compras y Proveedores:** Administra órdenes de compras a proveedores. Este se relaciona con los módulos de inventario y reportes.
- **Módulo de Reportes:** Genera reportes básicos de ventas, inventario y rentabilidad. Este se relaciona con todos los módulos para recopilar datos con el fin de realizar análisis.
- **Módulo de Seguridad y Autenticación:** Control de acceso al sistema. Al igual que el módulo de reportes, este también se relaciona con todos los demás para garantizar la seguridad y protección de los mismos.
- **Módulo de Integración:** Conexión con sistemas externos. Este módulo se relaciona con ventas e inventario.

A continuación detallamos la participación de cada módulo dentro del sistema, yendo desde un nivel de uso “Crítico” hasta uno bajo o netamente dependiente de terceros. (Crítico-Alto-Medio-Bajo-DE.Terceros)

- **Módulo de Gestión de Ventas:** Uso alto, corresponde al núcleo del negocio, usado diariamente. Dentro de sus posibles errores podemos mencionar el fallo a la hora de realizar cálculos de descuentos o impuestos, además de la lentitud en generación de facturas a horas de alta demanda.

- **Módulo de Inventario:** Al igual que el módulo anterior, este constituye un alto uso dado que se consulta en cada venta y compra. Dentro de los posibles errores que identificamos se encuentran la desincronización del stock y la falta de integración con sistemas externos.
- **Módulo de Clientes:** A diferencia de los módulos previos, el uso del módulo de clientes dependerá del volumen de clientes nuevos, pero se ha estimado que es de índole medio-alto por norma general. Dentro de los posibles errores que pueden ocurrir encontramos la inconsistencia de datos o la duplicidad de los mismos, y lentitud en las búsquedas.
- **Módulo de Compras y Proveedores:** Este módulo constituye un menor uso que sus predecesores, en este caso un uso medio. Posibles errores pueden ser la dificultad para comparar precios entre proveedores y el flujo manual de aprobación de órdenes de compra.
- **Módulo de Reportes:** Al igual que el módulo de compras y proveedores, este módulo posee un uso medio, más que nada utilizado por gerencia. Algunos problemas comunes son reportes estáticos sin personalización y consultas que al ser densas pueden ralentizar el sistema.
- **Módulo de Seguridad y Autenticación:** Nos encontramos con nuestro primer módulo con uso crítico, y es que todos los demás módulos dependen de la seguridad que nos brinda. Los posibles errores o problemas comunes que podemos encontrar en este son las vulnerabilidades y los roles o permisos mal configurados por parte del administrador del sistema.
- **Módulo de Integración:** Este último módulo tiene un uso que va a depender netamente de terceros, por lo que es difícil categorizar a qué tipo corresponde. Los problemas que podemos encontrar son fallos en conexiones con APIs y falta de “logs” (un log corresponde a un archivo que registra de forma cronológica las acciones que ocurren en un sistema informático) para realizar diagnósticos.

Finalmente, tomando en consideración el sistema monolítico que utiliza Perfulandia, y dado el análisis que hemos realizado previamente, podemos confirmar que estos son los módulos que pueden separarse más fácilmente en microservicios independientes:

- **Módulo de Seguridad y Autenticación:** Este será el módulo más recomendado a la hora de separarlo, dado que aunque todos los demás los usan, no depende de ellos.

- **Módulo de Reportes:** Como segundo módulo a separar se sabe que aunque este consume datos de otros módulos, no es crítico para operaciones en tiempo real y tiene un menor acoplamiento que sus pares.
- **Módulo de Integración:** Consideramos que el presente módulo puede ser el tercero en ser separado y es que los fallos en conexiones con terceros no deberían afectar al núcleo del sistema.

12.2. Fase 2: Definición de Microservicios

Para comenzar a realizar la migración de los módulos se ha determinado que la mejor manera de llevarlo a cabo es empezando por el servicio de reportes, dado que no es crítico para el funcionamiento del núcleo del negocio, además de operaciones en tiempo real. Este módulo puede fallar sin afectar directamente a los demás, lo cual lo vuelve el candidato perfecto para iniciar esta tarea. Como segundo candidato nos decantamos por el módulo de integración, ya que depende de terceros, por lo que sus fallos no afectan al sistema principal. Con esto culminaría la primera fase de migración de los módulos.

Avanzando hacia la segunda fase de migración, en este caso tomaremos al módulo de usuarios y clientes como tercer servicio, porque podemos manejar errores de duplicidad de datos y también puede operar con degradación si llegase a fallar. Para el cuarto módulo tomaremos al de compras y proveedores, considerando que su uso es medio y también es relativamente manejable en caso de fallas.

Por último, finalizamos la división de estos servicios con los que son altamente críticos y suponen un alto riesgo para el núcleo del negocio, comenzando por el módulo de seguridad y autenticación, ya que durante las fases 1 y 2 de la división fue probado y su independencia permite reemplazarlo sin romper otros servicios. Posteriormente, vamos a tomar el módulo de gestión de ventas que hemos dejado casi al final para no dañar a los demás. Finalmente, incluimos el módulo de inventario para el cual debemos tener disponible la realización de ventas en modo offline con colas.

12.3. Fase 3: Prueba de Concepto

Prueba de concepto del servicio de reportes: Este microservicio se encargaría exclusivamente de la realización de reportes con información de los demás módulos según las necesidades del usuario. Utilizará Kafka, Python y PostgreSQL. El monolito redirige las consultas de reportes al microservicio mediante un API Gateway. Se creará una base de datos independiente para gestionar el historial de informes y mediante el uso del API REST se consultarán los datos de cada módulo del cual se requiera generar un reporte. Posteriormente, redireccionaremos la consulta hacia este microservicio, para realizar las validaciones correspondientes. Realizaremos pruebas de estrés para demostrar que tan bien o mal responde a ciertas situaciones como las horas peak donde hay una mayor demanda por parte de los usuarios, entre otras, con el fin de verificar que cumpla su función correctamente y se justifique su separación.

12.4. Fase 4: Reestructuración Incremental

Con el fin de realizar una migración desde el modelo monolítico al modelo de microservicios de manera segura y eficiente, proponemos una transición gradual en base a lo detallado a continuación:

- Primero: Cada módulo existente se debe desacoplar como un microservicio que sea independiente, considerando una base de datos dedicada, una API rest propia (en esta parte nos referimos a una API rest cuando hablamos de comunicación entre distintos microservicios) y una contenedorización que sea también propia (aquí la palabra contenedorización hace referencia a agrupar todos los componentes de cada microservicio en, valga la redundancia, contenedores, en este caso de Docker).

Para complementar este primer punto veremos una tabla descriptiva la cual muestra los módulos que ahora pasarán a ser microservicios:

Módulo Original	Microservicio
Clientes	servicio-clientes
Compras y Proveedores	servicio-obtención
Gestión de Ventas	servicio-ventas
Integración	servicio-integración
Inventario	servicio-inventario
Reportes	servicio-reportes
Seguridad	servicio-seguridad

- Segundo: Estos microservicios se comunicarán e interactuarán mediante APIs REST.

12.5. Fase 5: Despliegue en Contenedores

Posterior a la reestructuración incremental, donde separamos los módulos en microservicios, y definimos de qué manera se comunicarán, procederemos a apoyarnos de Docker para empaquetar cada uno de ellos. Lo que hará Docker en este punto nos garantizará escalabilidad, consistencia, aislamiento y portabilidad, los cuales son aspectos clave a tener en cuenta en este proyecto.

12.6. Fase 6: Integración Continua y Despliegue Continuo (CI/CD)

Ahora nos compete una fase muy importante, y es que para cada cambio, actualización, mejora, integración y cualquier otro aspecto que involucre una edición en nuestro proyecto debemos ser capaces de testear previamente cada uno de estos, de manera que no afecte negativamente a ningún campo del mismo.

¿Cómo lo haremos? Concretamente nos apoyaremos en GitHub Actions, dado que es una herramienta muy útil para este fin y además está al alcance actual de nuestras capacidades, por lo que no supone una inversión de tiempo alta.

12.7. Fase 7: Eliminación del Monolito

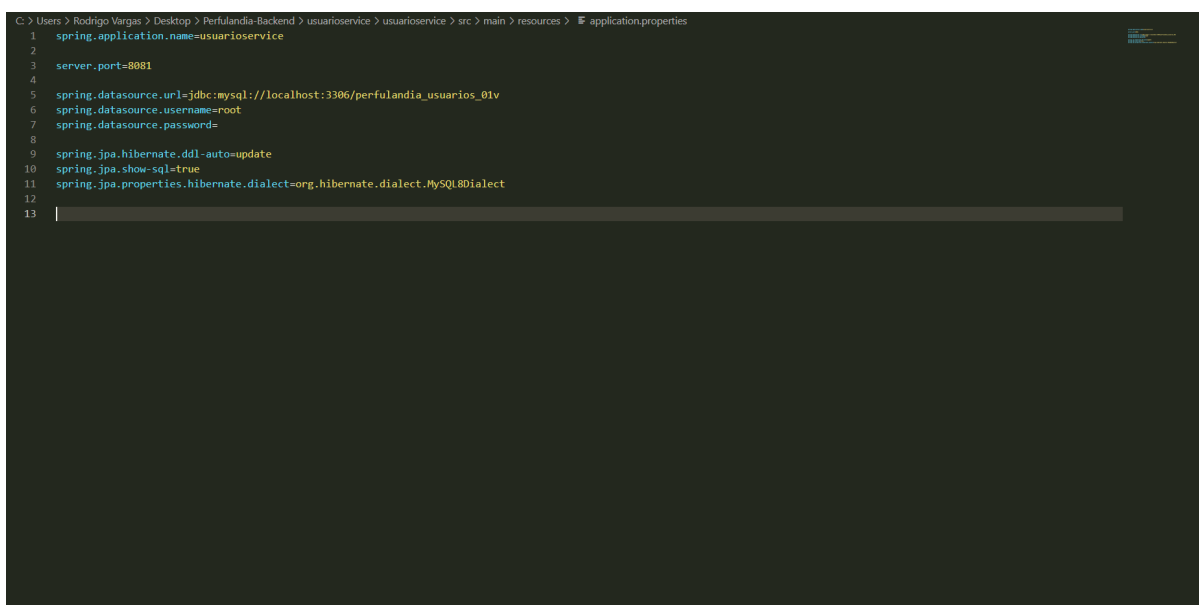
Hemos llegado a la fase final del proyecto, en este último paso nos corresponde eliminar el sistema monolítico anterior, para dar paso de manera total al nuevo basado en los microservicios que hemos realizado, puesto a prueba de manera aislada y posteriormente integrado ordenadamente al sistema previo. Con esto hecho podemos dar cuenta de que cada microservicio ahora nos brinda una gran suma de beneficios respecto del sistema anterior.

13. Configuración de Bases de Datos

Para la configuración de cada base de datos utilizamos Laragon en su versión 8.1.0.

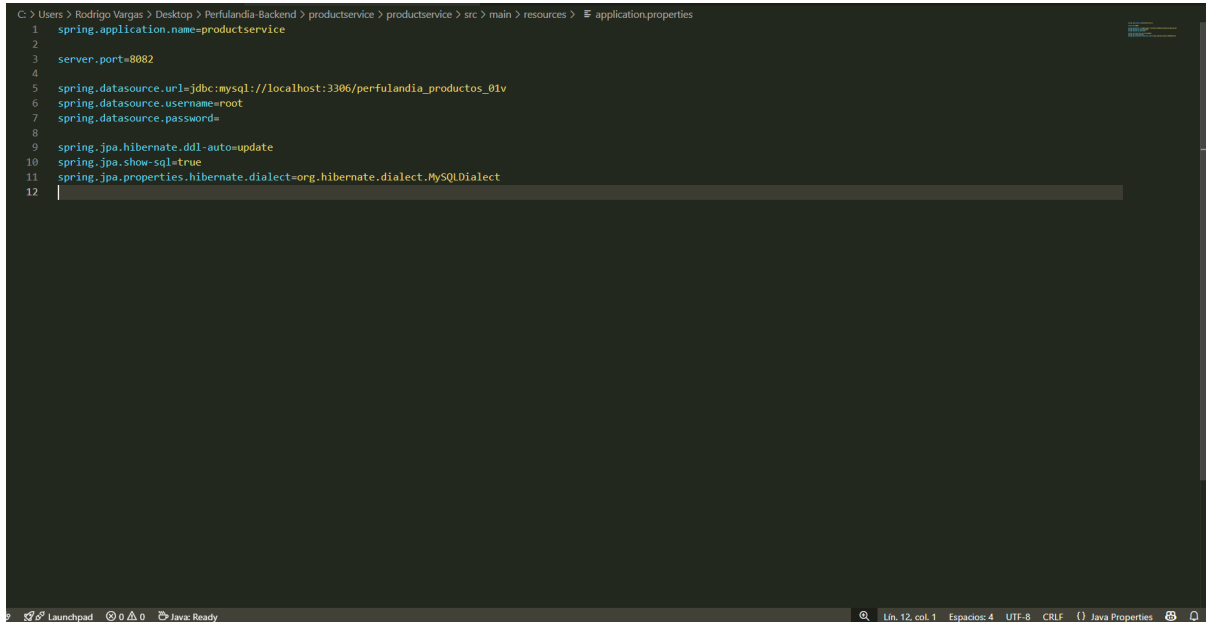
En las siguientes imágenes se muestran tanto la iniciación como las tablas para los primeros 2 microservicios creados en clases, los cuales corresponden a Producto y Usuario para Perfulandia.

1. Base de datos de productoservice (perfulandia_usuarios_01v): Esta BD se utiliza para almacenar productos (en este caso perfumes) con sus respectivos atributos, tales como id, nombre, precio y stock. Imagen referente al archivo de configuración application.properties:



```
C:\Users > Rodrigo Vargas > Desktop > Perfulandia-Backend > usuarioservice > usuarioservice > src > main > resources > application.properties
1  spring.application.name=usuarioservice
2
3  server.port=8081
4
5  spring.datasource.url=jdbc:mysql://localhost:3306/perfulandia_usuarios_01v
6  spring.datasource.username=root
7  spring.datasource.password=
8
9  spring.jpa.hibernate.ddl-auto=update
10 spring.jpa.show-sql=true
11 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
12
13
```

2. Base de datos de usuarioservice (perfulandia_productos_01v): Esta BD se utiliza para almacenar usuarios con sus respectivos atributos dentro del sistema, tales como id, nombre, correo y rol dentro del sistema. Imagen referente al archivo de configuración application.properties:



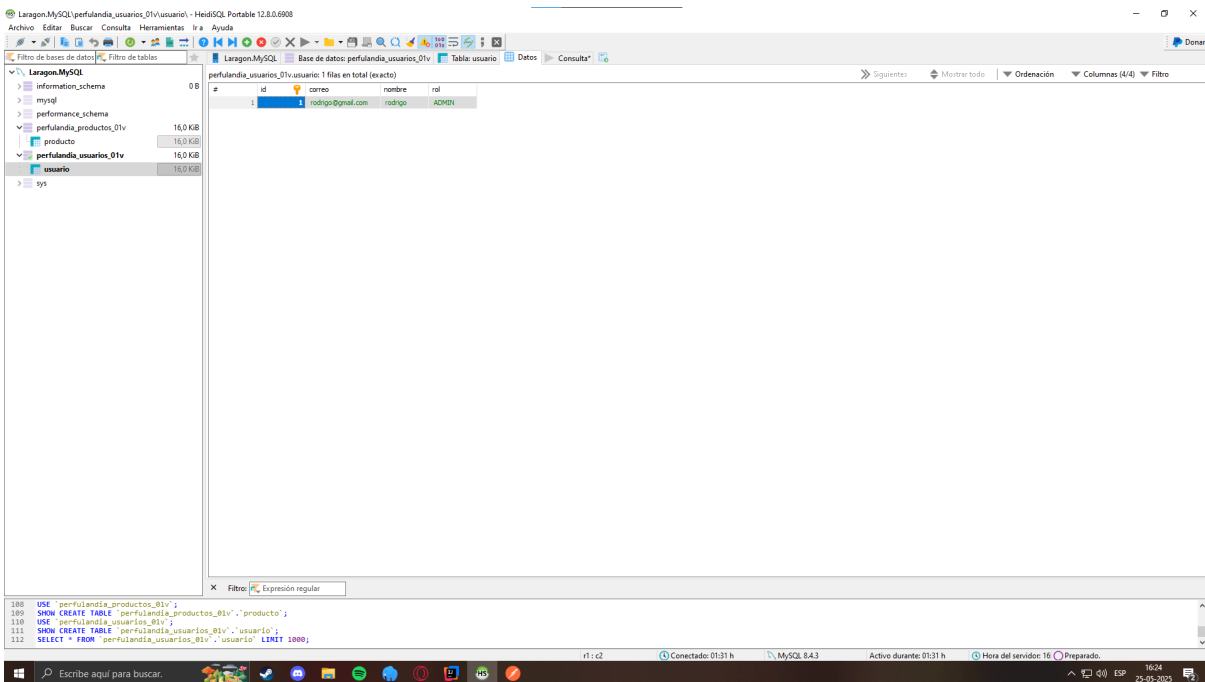
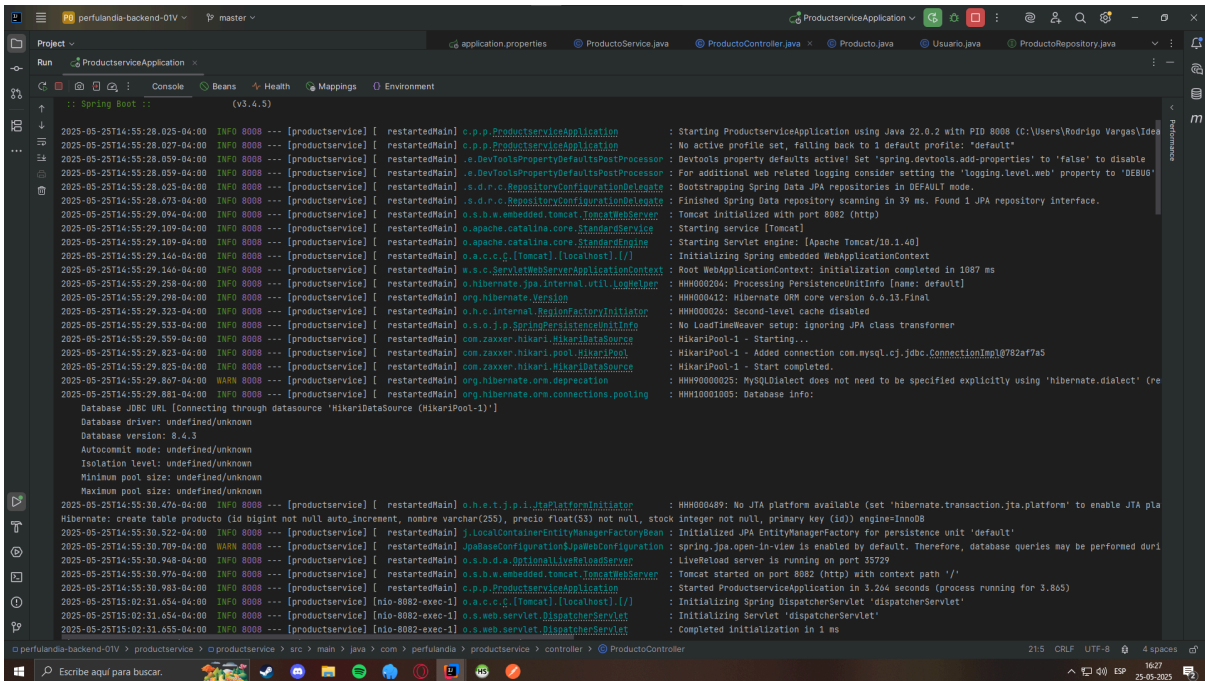
```
C:\Users > Rodrigo Vargas > Desktop > Perfulandia-Backend > productservice > productservice > src > main > resources > application.properties
1  spring.application.name=productservice
2
3  server.port=8082
4
5  spring.datasource.url=jdbc:mysql://localhost:3306/perfulandia_productos_01v
6  spring.datasource.username=root
7  spring.datasource.password=
8
9  spring.jpa.hibernate.ddl-auto=update
10 spring.jpa.show-sql=true
11 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
12
```

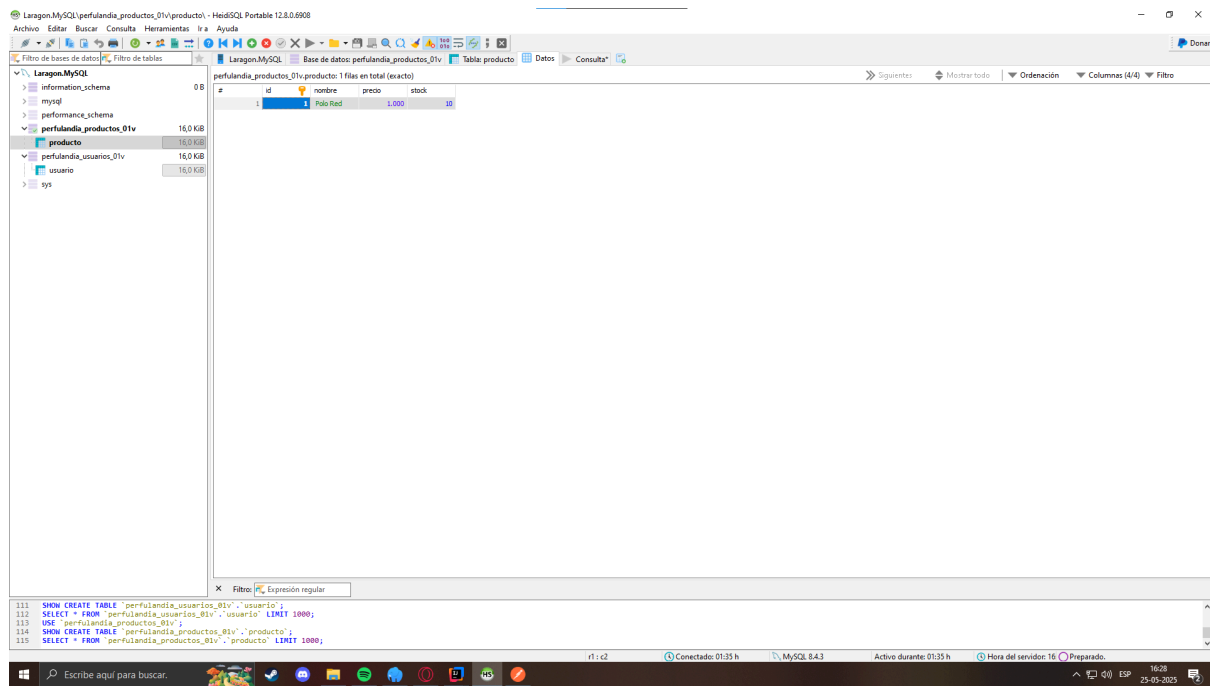
3. Base de datos de carritoservice (perfulandia_carritos_01v): Esta BD se utiliza para generar un carrito de compras mediante la comunicación con los 2 microservicios previamente mencionados. Crea un “carrito” con la id del mismo, el nombre del cliente, su id, la id del o los productos que se hayan agregado, y el total de la compra (en dólares). Imagen referente al archivo de configuración application.properties:



```
1  spring.application.name=carritoservice
2  productos.service.url=http://localhost:8082
3  usuarios.service.url=http://localhost:8081
4  server.port=8083
5
6  spring.datasource.url=jdbc:mysql://localhost:3306/perfulandia_carritos_01v
7  spring.datasource.username=root
8  spring.datasource.password=
9
10 spring.jpa.hibernate.ddl-auto=update
11 spring.jpa.show-sql=true
12 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

Inicialización de los servicios ya realizados para su verificación de manera local, en este caso resultó todo sin problemas, como podemos apreciar en las siguientes 3 imágenes:

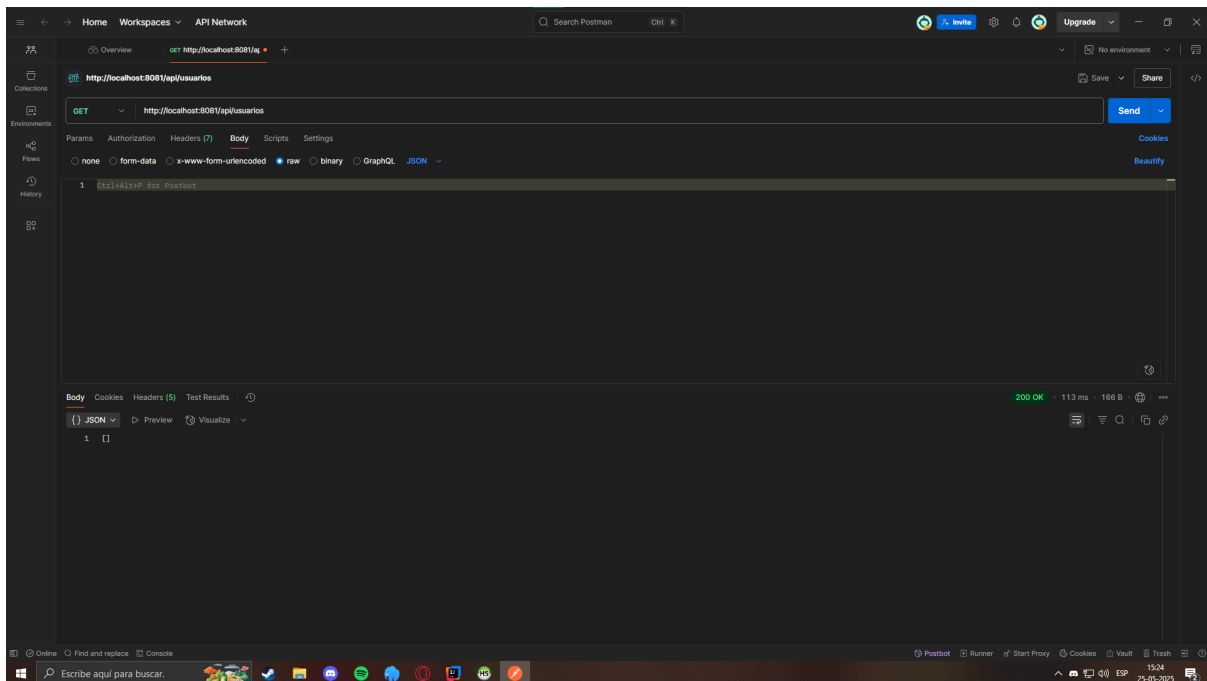




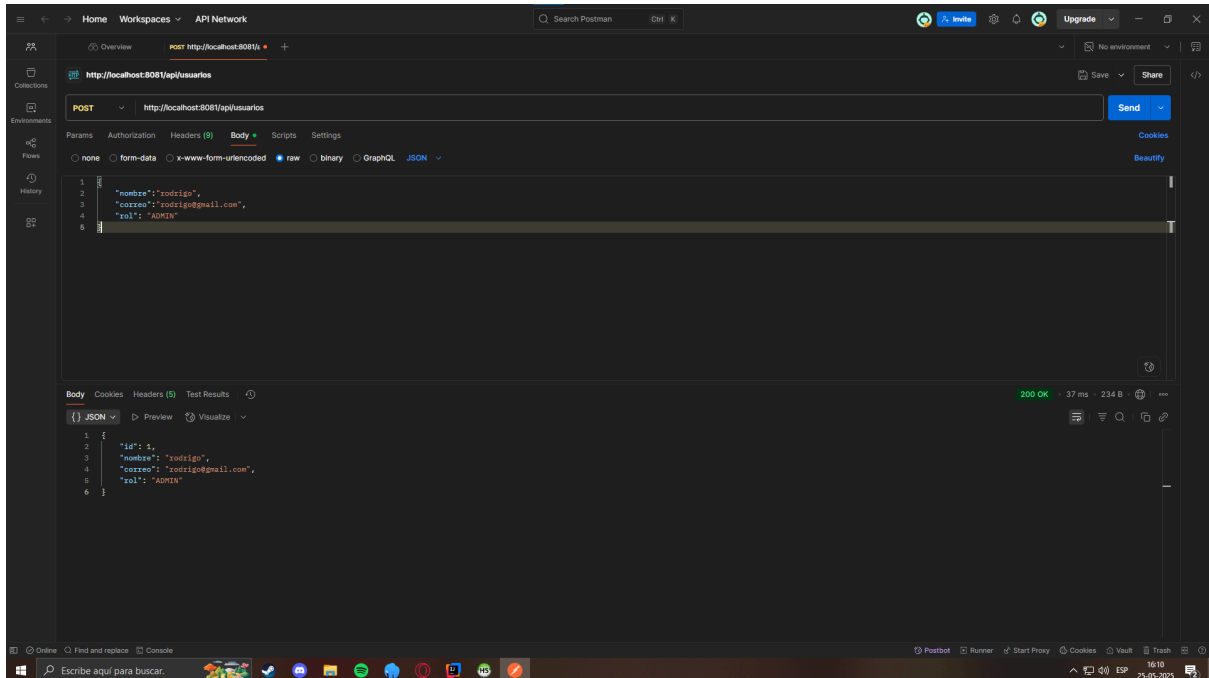
14. Resultados de Pruebas con Postman

Posterior a la realización y testeo de la base de datos realizamos pruebas mediante el uso de Postman para verificar el correcto funcionamiento de los microservicios Producto y Usuario, las cuales involucran métodos GET, POST y DELETE.

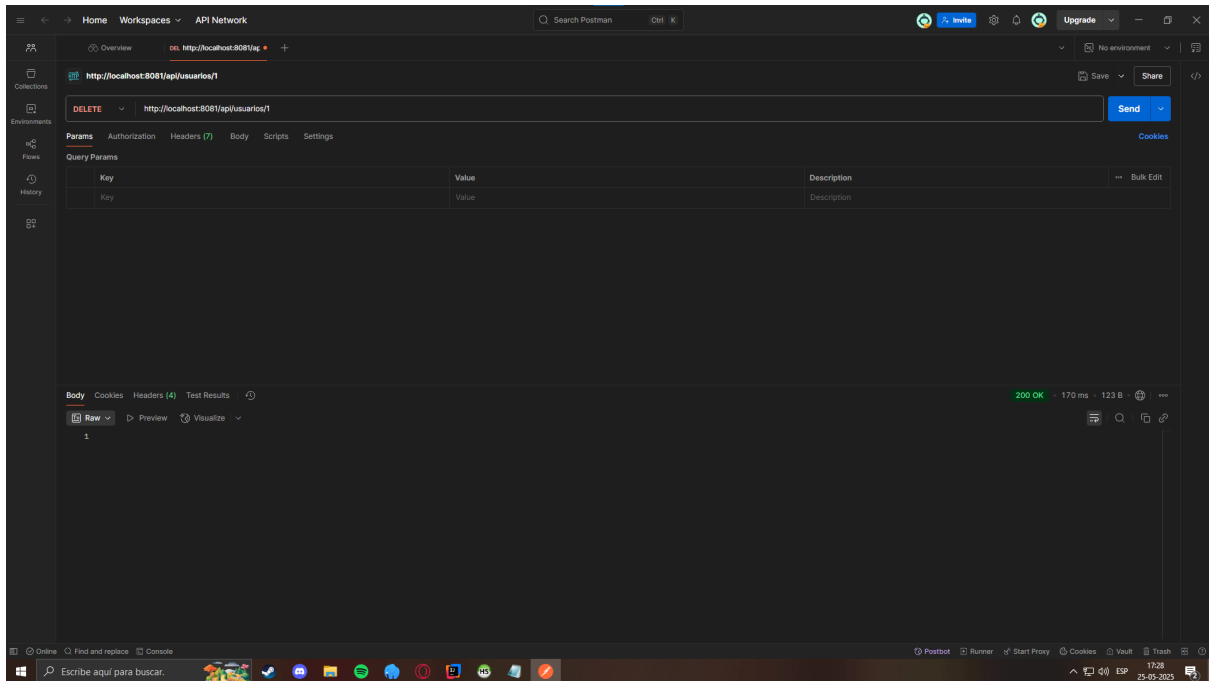
- Método GET para el microservicio de Usuario: Se espera que este método nos entregue todos los usuarios existentes en la base de datos. En el ejemplo no hay ninguno pero la respuesta es de código 200 por lo que podemos ver que la ejecución ha sido exitosa.



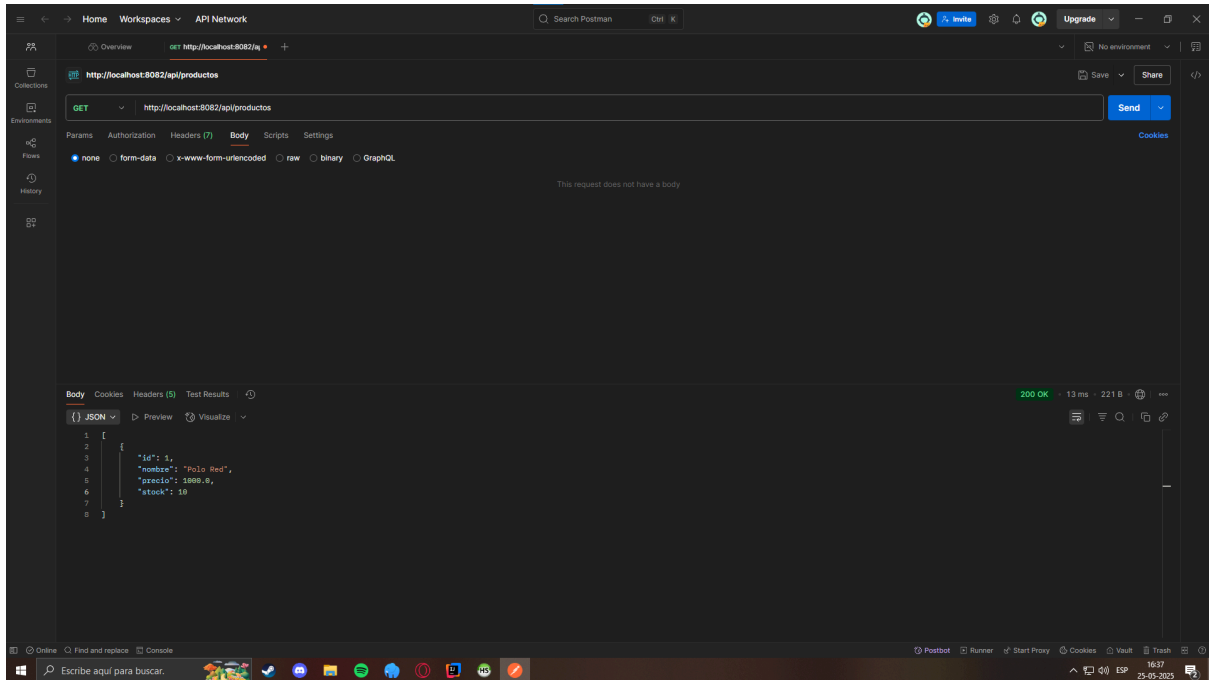
- Método POST para el microservicio de Usuario: En este caso ingresamos un nuevo usuario mediante este método y lo que se espera es que al momento de enviar los datos en formato JSON el sistema nos entregue un código 200 y además poder visualizar al usuario que acabamos de ingresar en la parte inferior de la ventana.



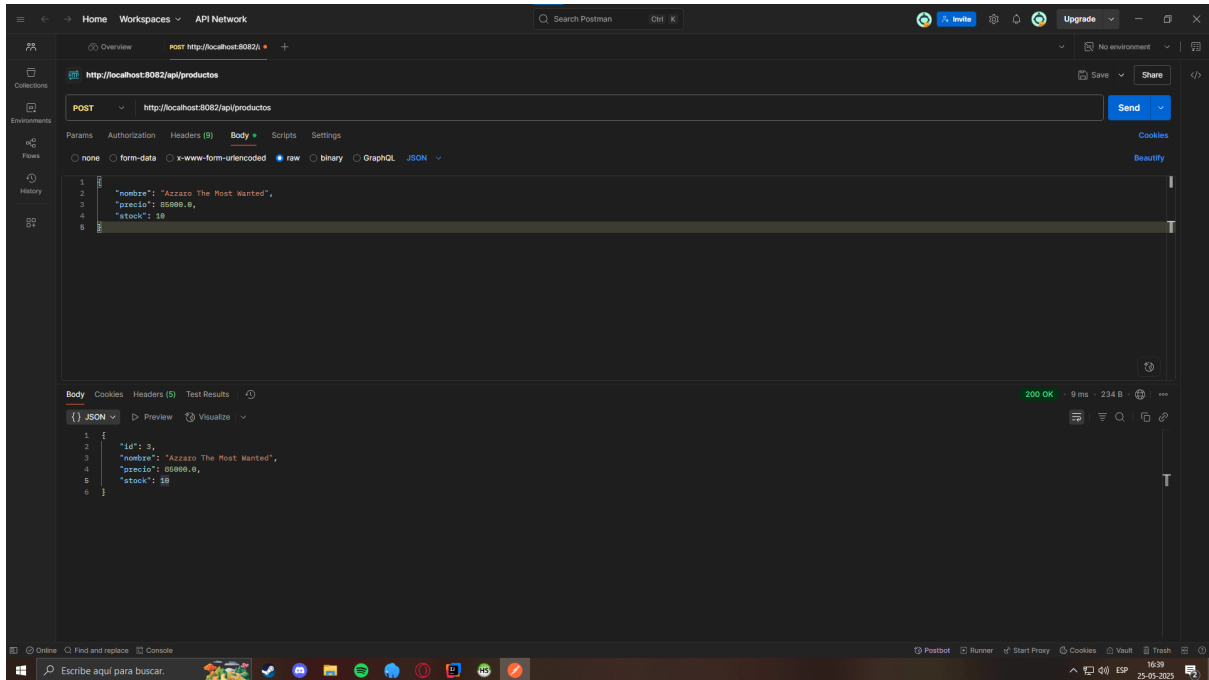
- Método DELETE para el microservicio de Usuario: A diferencia de los 2 métodos utilizados previamente, en la barra de ingreso de url debemos incorporar la ID del usuario que queremos eliminar de nuestra base de datos. En este caso esperamos que en la parte inferior de la ventana nos entregue un código 200 y que luego corroborando con el método GET que busca por ID no veamos a dicho usuario.



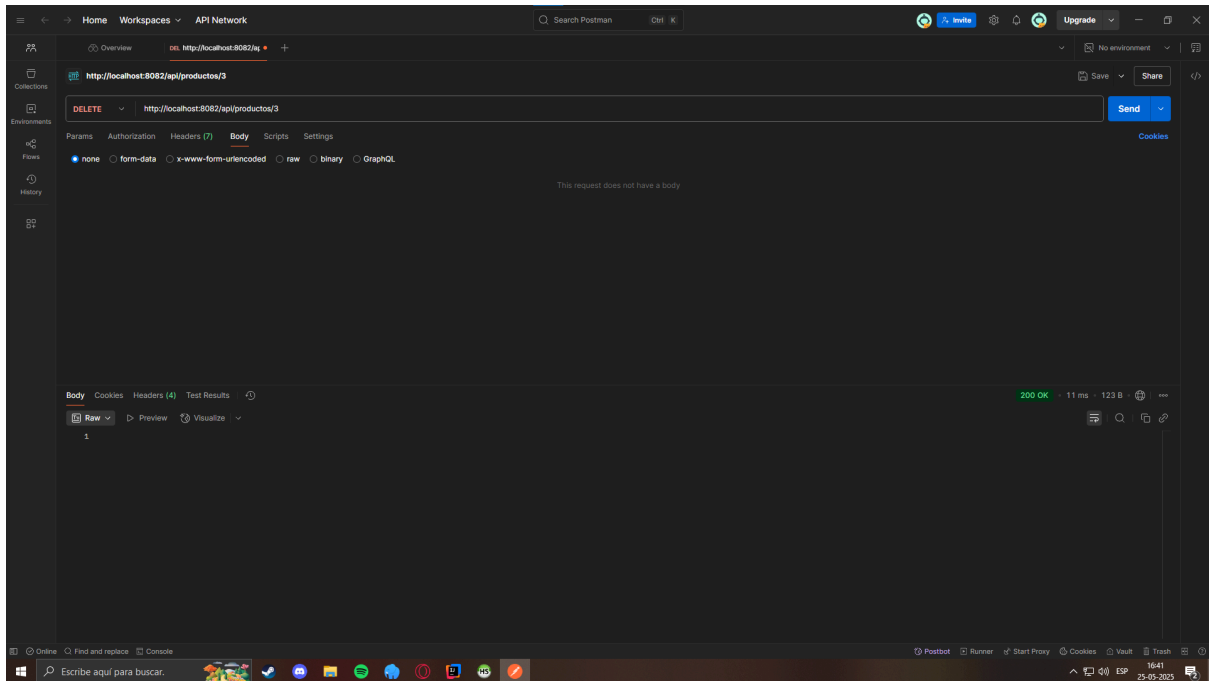
- Método GET para el microservicio de Producto: Método GET para el microservicio de Producto: Se espera que este método nos entregue todos los productos existentes en la base de datos. En el ejemplo hay uno, la respuesta es de código 200 y en la parte inferior de la ventana podemos ver dicho producto, por ende, podemos afirmar que la ejecución ha sido exitosa.



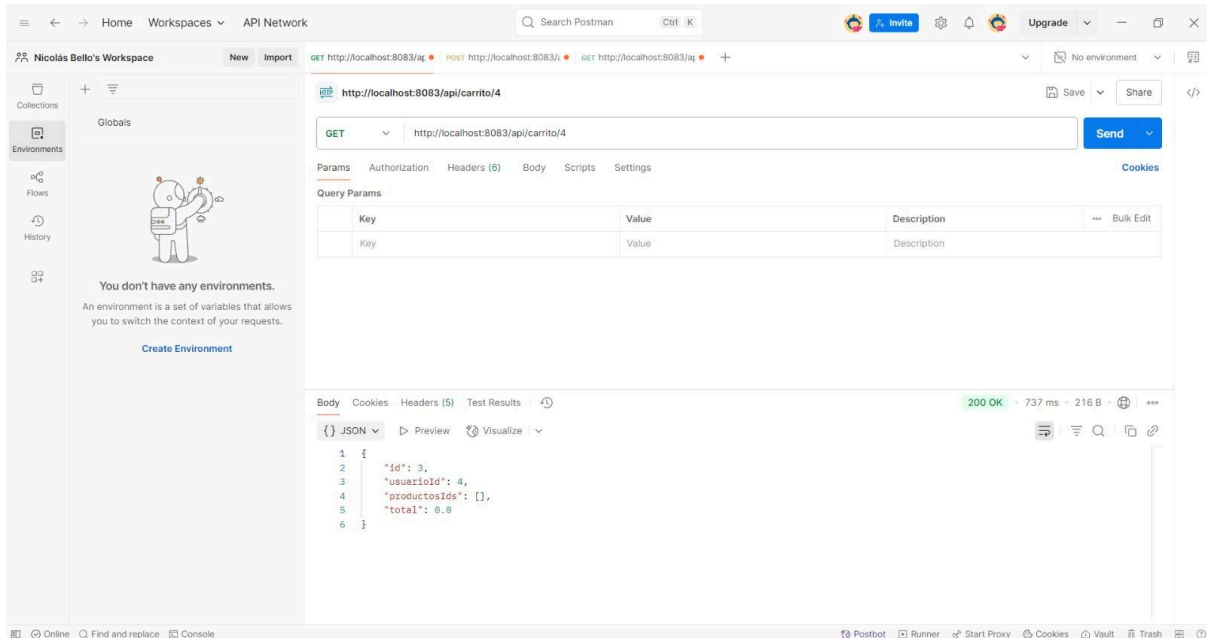
- Método POST para el microservicio de Producto: En este caso ingresamos un nuevo producto mediante este método y lo que se espera es que al momento de enviar los datos en formato JSON el sistema nos entregue un código 200 y además poder visualizar el producto que acabamos de ingresar en la parte inferior de la ventana.



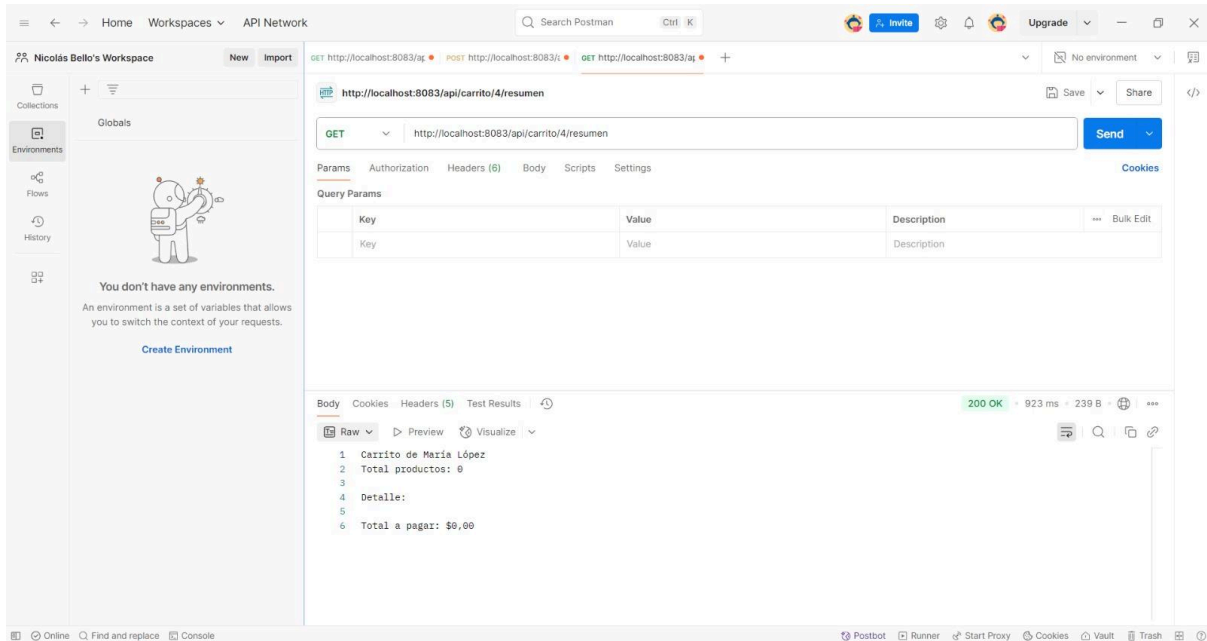
- Método DELETE para el microservicio de Producto: A diferencia de los 2 métodos utilizados previamente, en la barra de ingreso de url debemos incorporar la ID del producto que queremos eliminar de nuestra base de datos. En este caso esperamos que en la parte inferior de la ventana nos entregue un código 200 y que luego corroborando con el método GET que busca por ID no veamos dicho producto.



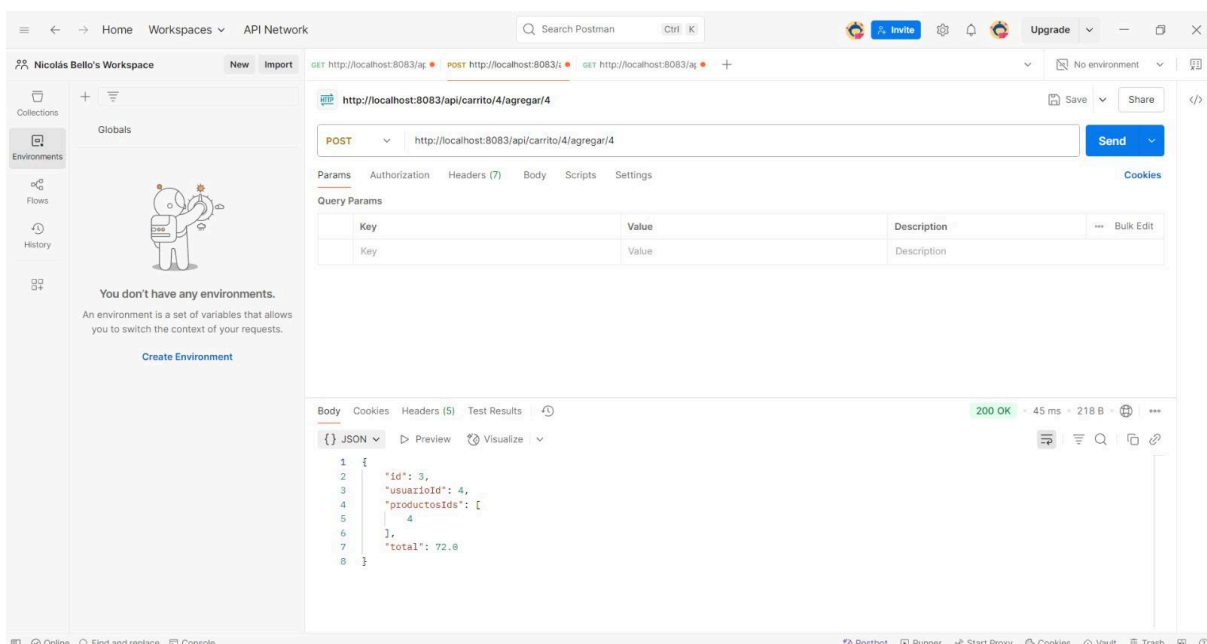
- Método GET para la creación de un carrito de compras: Esta función nos permite asignarle un carrito a un usuario en específico. Este carrito como tal posee ciertos atributos tales como la ID del usuario que lo posee, una lista con los productos que esté agregando y el total de la compra.



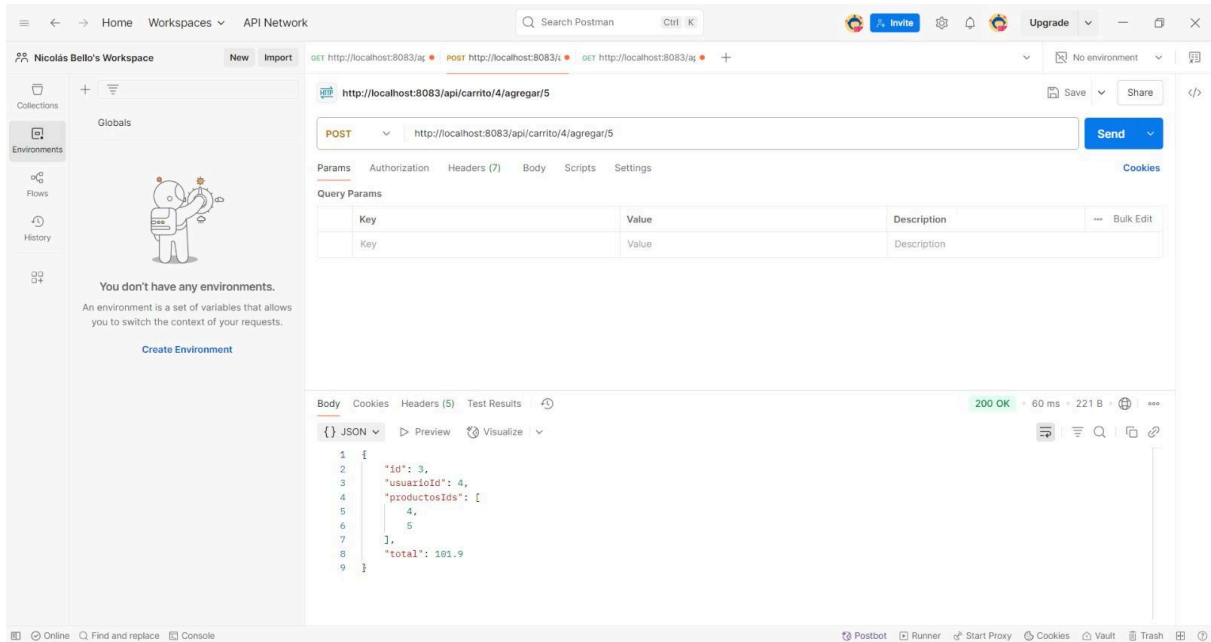
- Método GET para mostrar el resumen del carrito de compras de un usuario en específico: La función resumen nos permite ver el carrito de un usuario, mostrando su nombre, apellido paterno, el total de productos que contiene el carrito, y el detalle. Este último incluye netamente la cantidad total de dinero que el usuario debería pagar por ese carrito.



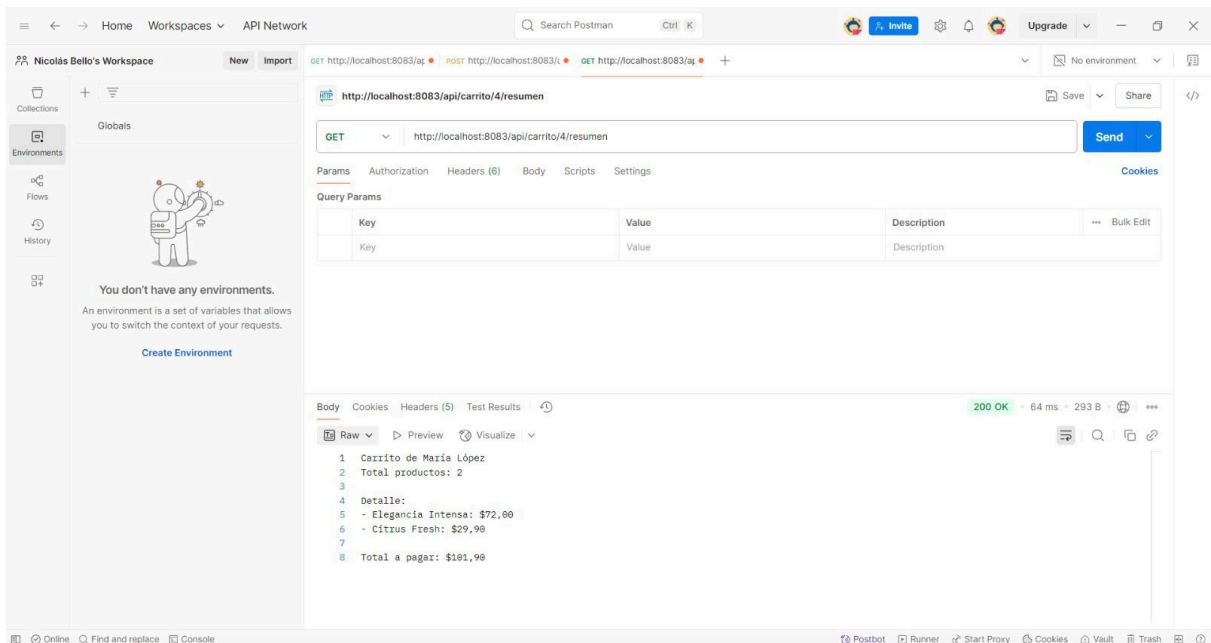
- Método POST para agregar un producto al carrito de compras de un usuario: En este caso, esta función nos permite agregar un producto mediante su ID al carrito de un usuario en específico. Mostrando posteriormente la información actualizada del mismo.



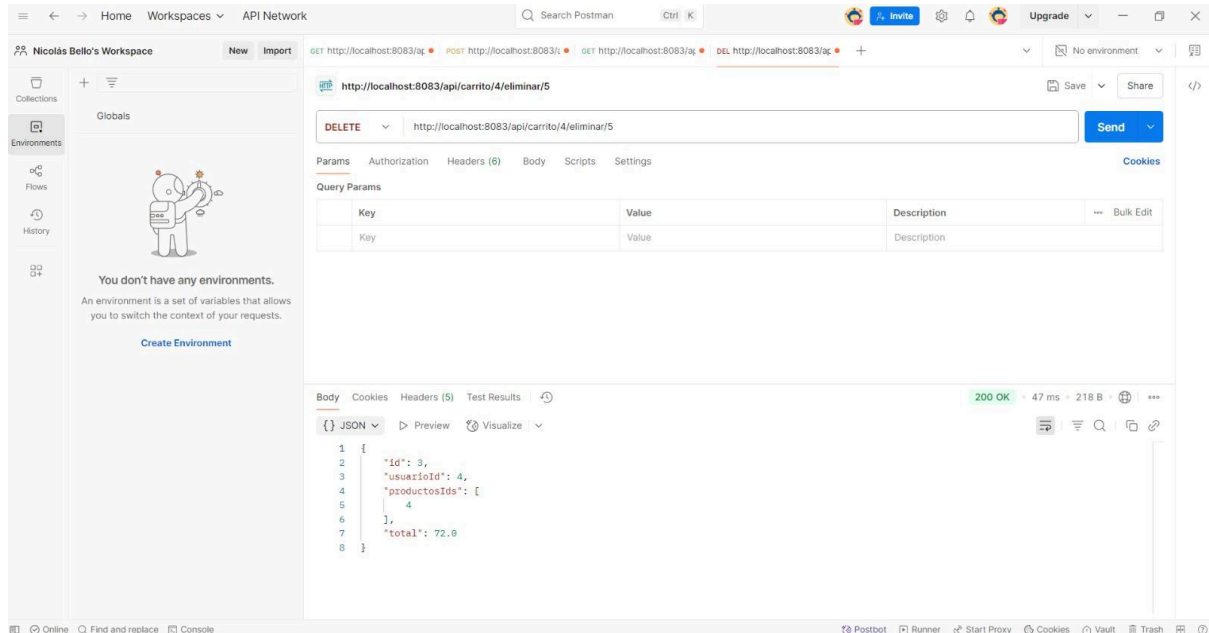
- Otro ejemplo del método anterior, esta vez agregando un producto distinto al carrito de compras. Podemos apreciar que se incorpora de manera satisfactoria un producto con ID 5 observando la parte inferior de la ventana.



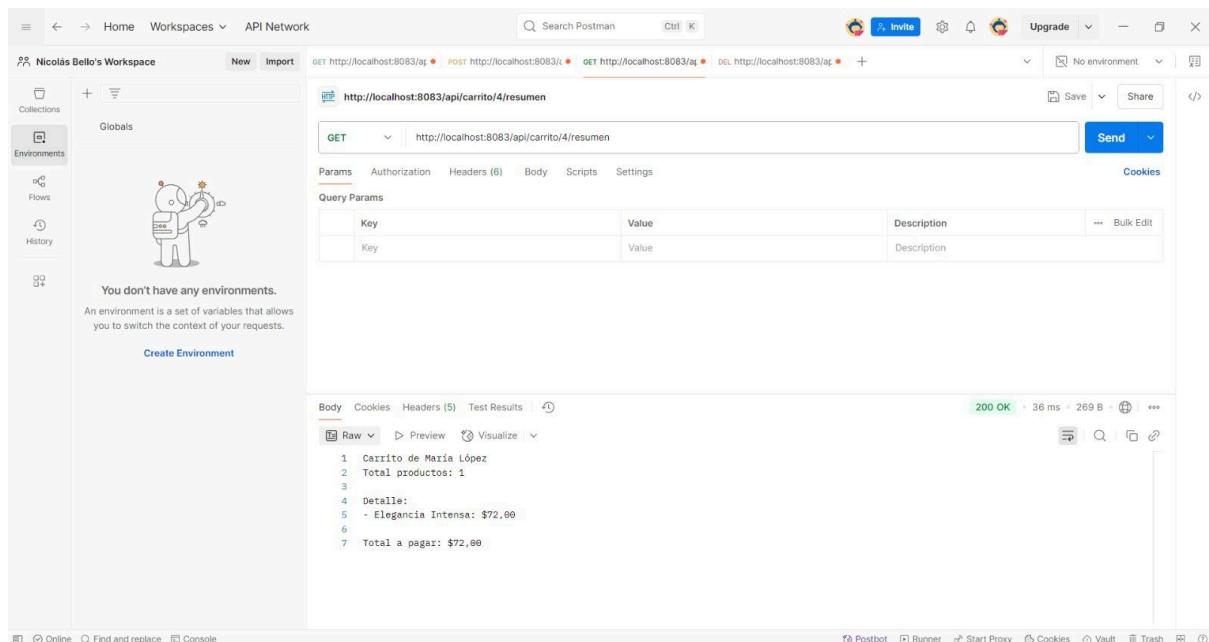
- Método GET para ver el resumen y resultado de la adición de 2 productos distintos al carrito de compras del usuario con ID 4.



- Método DELETE para eliminar un producto específico del carrito de compras de un usuario con ID 4: Esta funcionalidad nos permite buscar un producto dentro del carrito de compras de un usuario en específico y mediante la ID de ese producto eliminarlo del mismo.



- Método GET para mostrar el resumen final del carrito de compras del usuario con ID 4.



15. Conclusión

En este documento se ha detallado un problema que afecta a Perfulandia SPA, el cual se basa en las limitaciones de su sistema informático debido a múltiples factores. Ahora bien, se documentó cómo estos factores pueden ser solucionados mediante la transición gradual del sistema antiguo, el cual se basa en tener todas las funcionalidades juntas, a un nuevo sistema basado en microservicios. Estos microservicios se comunican entre sí mediante el uso de APIs REST para asegurar que cada uno de ellos opere de manera independiente y segura, con el fin de garantizar el desarrollo e intervención en caso de ser necesarios, y al mismo tiempo reducción de riesgos operacionales. La ya formulada solución permitirá a Perfulandia SPA mantenerse en el mercado de manera competitiva tal cual lo han hecho en el último tiempo, además de permitirles ir al top de su competencia, continuar expandiéndose, tener más clientes de manera simultánea, entre otros beneficios. Todo esto de manera segura, escalable, tolerante a fallos y alineada con estándares de calidad para brindar fiabilidad y una experiencia grata a todos sus clientes.