Migración a una Infraestructura de Microservicios: Caso Perfulandia

Integrantes:

- Hector Águila
- Fabián Lecaros
- Franco Ruz
- Cesar Veliz

Índice

Contexto

Perfulandia SPA es una empresa chilena dedicada a la venta de perfumes al por menor y mayor. Comenzó con una sucursal en el Barrio Meiggs en Santiago y, debido a su éxito comercial, expandió sus operaciones a Concepción y Viña del Mar. Si bien su rápido crecimiento genera expectativas prometedoras para la empresa, también ha puesto en evidencia las limitaciones de su sistema de software actual, el cual es monolítico y presenta fallos de rendimiento y disponibilidad. Estas deficiencias tecnológicas amenazan la continuidad operativa y la satisfacción de los clientes, por lo que la empresa busca modernizar su infraestructura tecnológica para apoyar una expansión sostenida.

1 . Introducción

El crecimiento acelerado de Perfulandia SPA, impulsado por la apertura de nuevas sucursales y el aumento sostenido de su base de clientes, ha puesto en evidencia las limitaciones de su infraestructura tecnológica actual. El sistema monolítico que ha acompañado a la empresa desde sus inicios ya no responde con eficacia a las demandas de escalabilidad, disponibilidad y rendimiento. Frente a este escenario, se hace necesario avanzar hacia una arquitectura basada en microservicios, que permita una gestión más flexible, modular y resiliente del sistema, asegurando así la continuidad operativa y el soporte a su proceso de expansión nacional.

2 . Requisitos Funcionales

A continuación se describen brevemente los requisitos funcionales del caso, agrupados por módulo.

2.1. Gestión de Usuarios

- RF01 – El sistema debe permitir a los usuarios registrarse mediante correo electrónico y contraseña.

- RF02 – El sistema debe permitir a los usuarios iniciar sesión con sus credenciales.
- RF03 – El sistema debe permitir al administrador crear, editar, desactivar y eliminar cuentas de usuario internas.
- RF04 – El sistema debe permitir asignar y modificar permisos a los usuarios según su rol.

2.2. Gestión de Inventario

- RF05 – El sistema debe permitir al gerente de sucursal agregar nuevos productos al inventario.
- RF06 – El sistema debe permitir actualizar y eliminar productos del inventario.
- RF07 – El sistema debe mostrar en tiempo real la disponibilidad de productos en cada sucursal.
- RF08 – El sistema debe permitir ajustar manualmente las cantidades de stock por parte del gerente.

2.3. Ventas y Facturación

- RF09 – El sistema debe permitir registrar ventas con productos seleccionados y descuentos aplicables.
- RF10 – El sistema debe generar facturas electrónicas automáticamente tras cada venta.
- RF11 – El sistema debe enviar por correo electrónico la factura electrónica al cliente.
- RF12 – El sistema debe permitir procesar devoluciones y aplicar reembolsos o notas de crédito.

2.4. Pedidos y Logística

- RF13 – El sistema debe permitir generar pedidos de productos para reabastecimiento entre bodega y sucursales.
- RF14 – El sistema debe permitir planificar y optimizar rutas de entrega.
- RF15 – El sistema debe actualizar automáticamente el estado de los pedidos (en preparación, en tránsito, entregado).

- RF16 – El sistema debe permitir registrar y mantener la información de proveedores.

2.5. Reportes y Monitoreo

- RF17 – El sistema debe generar reportes de ventas por día, semana, mes y por sucursal.
- RF18 – El sistema debe generar reportes del estado del inventario por producto y sucursal.
- RF19 – El sistema debe mostrar el estado de funcionamiento de los módulos del sistema.
- RF20 – El sistema debe emitir alertas ante fallos críticos de disponibilidad o rendimiento.

2.6. Portal Cliente Web

- RF21 – El sistema debe permitir al cliente crear una cuenta ingresando sus datos personales.
- RF22 – El sistema debe permitir al cliente buscar y filtrar productos en el catálogo.
- RF23 – El sistema debe permitir al cliente agregar productos al carrito de compras.
- RF24 – El sistema debe permitir al cliente realizar pedidos con selección de método de pago y envío.
- RF25 – El sistema debe permitir aplicar cupones o códigos promocionales durante la compra.
- RF26 – El sistema debe mostrar el historial de compras y estado de pedidos al cliente.
- RF27 – El sistema debe permitir al cliente editar sus datos personales y direcciones de envío.
- RF28 – El sistema debe permitir al cliente enviar consultas mediante formulario o chat.
- RF29 – El sistema debe permitir al cliente dejar calificaciones y comentarios sobre productos comprados.

3 . Requisitos No Funcionales

A continuación se describen los requisitos no funcionales, necesarios para el correcto funcionamiento del sistema.

3.1. Rendimiento

- RNF01 – El sistema debe garantizar un tiempo de respuesta menor a 2 segundos para las operaciones de búsqueda de productos.
- RNF02 – El sistema debe soportar al menos 500 usuarios concurrentes sin degradación significativa del rendimiento.
- RNF03 – El sistema debe permitir escalar horizontalmente cada microservicio de forma independiente.

3.2. Disponibilidad

- RNF04 – El sistema debe estar disponible al menos el 99.5% del tiempo mensual.
- RNF05 – El sistema debe recuperarse automáticamente ante la caída de un servicio crítico, sin intervención manual.
- RNF06 – El sistema debe contar con mecanismos de redundancia para garantizar la continuidad operativa en caso de falla.

3.3. Seguridad

- RNF07 – Todas las comunicaciones entre cliente y servidor deben estar cifradas mediante HTTPS.
- RNF08 – Las contraseñas deben almacenarse cifradas utilizando algoritmos seguros como bcrypt.
- RNF09 – Los usuarios deben autenticarse mediante credenciales únicas y seguras.
- RNF10 – Los accesos a funciones críticas deben estar restringidos mediante control de roles (RBAC).

3.4. Usabilidad

- RNF11 – La interfaz web debe ser intuitiva y accesible para usuarios sin conocimientos técnicos.
- RNF12 – El sistema debe estar adaptado para dispositivos móviles y tablets.
- RNF13 – El sistema debe contar con mensajes de error claros y amigables para el usuario final.

3.5. Mantenibilidad

- RNF14 – El sistema debe estar dividido en microservicios independientes para facilitar su mantenimiento y actualización.
- RNF15 – Cada microservicio debe tener una cobertura mínima del 70% en pruebas unitarias.
- RNF16 – La documentación técnica del sistema debe mantenerse actualizada y accesible para el equipo de desarrollo.

3.6. Portabilidad

- RNF17 – El sistema debe poder desplegarse en entornos basados en contenedores (por ejemplo, Docker).
- RNF18 – El sistema debe ser compatible con plataformas cloud como Oracle Cloud, AWS o Azure.
- RNF19 – Los datos del sistema deben poder migrarse a otra base de datos relacional sin pérdida de integridad.

4 . Entrevistas Simuladas

4.1. Tipos de Usuario

A continuación se describen los tipos de usuario, de los cuales posteriormente se simularán entrevistas.

4.1.1. Administrador del Sistema

- Rol: Técnico/superusuario con acceso completo.

- Funciones principales:
    - Gestión de usuarios y permisos.
    - Monitoreo del sistema.
    - Respaldo y restauración de datos.
- Acceso a: Panel de administración completo y consola técnica.

4.1.2. Gerente de Sucursal

- Rol: Encargado de una sucursal específica.
- Funciones principales:
    - Gestión del inventario local.
    - Generación de reportes de rendimiento.
      Configuración de parámetros locales (horarios, personal).
    - Supervisión y autorización de pedidos internos.
- Acceso a: Módulos de inventario, reportes y pedidos.

4.1.3. Empleado de Ventas

- Rol: Vendedor o cajero.
- Funciones principales:
    - Procesamiento de ventas.
    - Emisión de facturas.
    - Atención a devoluciones y reclamos.
    - Consulta de stock.
- Acceso a: Punto de venta, consulta de productos y gestión básica de pedidos.

4.1.4. Personal de Logística

- Rol: Encargado de envíos y proveedores.
- Funciones principales:
    - Gestión de envíos y rutas de entrega.
    - Actualización del estado de pedidos.
    - Administración de proveedores y recepción de productos.
- Acceso a: Módulo de logística y cadena de suministro.

4.1.5. Cliente Web

- Rol: Usuario final/comprador en línea.
- Funciones principales:
    - Registro e inicio de sesión.
    - Navegación y búsqueda de productos.
    - Realización de pedidos.
    - Seguimiento de compras y estado de envío.
    - Interacción con soporte y reseñas.
- Acceso a: Portal web responsivo, historial de pedidos y gestión de perfil.

4.2. Preguntas y Respuestas de Usuarios

4.2.1. Usuario: Administrador del Sistema

Pregunta: ¿Cuáles son los mayores desafíos al administrar el sistema día a día, especialmente durante períodos de mucha venta como Navidad o el Día de la Madre?.

Respuesta: El sistema se pone increíblemente lento en esas fechas, a veces hasta se cae por completo. Es un gran problema porque si una parte del sistema falla, a menudo todo se detiene. Nos cuesta darnos cuenta de los problemas a tiempo porque es difícil vigilar todo a la vez. Además, dar permisos a los usuarios es complicado; configurar accesos distintos para cada sucursal es riesgoso porque, por cómo está construido el sistema, un cambio puede afectar a todos lados. Y si el sistema se cae, levantarlo de nuevo toma demasiado tiempo.

4.2.2. Usuario: Gerente de Sucursal.

Pregunta: Pensando en tu trabajo diario, ¿puedes confiar siempre en las cifras de inventario que ves en el sistema? ¿Y qué tal es obtener los reportes de ventas al final del mes?

Respuesta: La confianza en el inventario es un tema. A veces vendemos algo y resulta que no quedaba, o al revés. El sistema no refleja los cambios al instante,

sobre todo cuando está muy ocupado. Y los reportes mensuales, como los de ventas, ¡pueden tardar horas! Especialmente a fin de mes. Eso nos retrasa mucho para saber qué pedir o cómo vamos.

4.2.3. **Usuario:** Empleado de Ventas.

Pregunta: Cuando estás en caja atendiendo, sobre todo si hay fila, ¿qué problemas te da el sistema al cobrar, buscar productos o hacer devoluciones?

Respuesta: Cuando hay mucha gente, el sistema de caja se congela seguido. Buscar un perfume específico o aplicar un descuento puede ser eterno. Y hacer una devolución es peor, a veces tenemos que anotarlo a mano y registrarlo después, lo que causa errores. Preguntar si queda stock en otra tienda es casi imposible hacerlo rápido mientras el cliente espera.

4.2.4. **Usuario:** Personal de Logística.

Pregunta: ¿Cómo organizan hoy las rutas para las entregas y cómo saben en qué estado está un pedido que va de la bodega a una sucursal o a un cliente?

Respuesta: La verdad es que planificar las rutas es muy manual, usamos planillas aparte porque el sistema no nos ayuda mucho con eso, o está muy lento. Actualizar si un pedido está "en camino" o "entregado" lo hacemos a mano y a veces se nos olvida o nos atrasamos, y ahí empiezan las llamadas preguntando por el pedido. Tampoco tenemos un lugar ordenado en el sistema para ver la información de los proveedores.

4.2.5. **Usuario:** Cliente Web.

Pregunta: ¿Cómo ha sido tu experiencia al usar la página web de Perfulandia para ver perfumes, comprar o revisar dónde viene tu pedido?

Respuesta: La página a veces es súper lenta para cargar, sobre todo las fotos de los perfumes. Una vez intenté usar un cupón de descuento y me dio un error raro, sin decir por qué. Y cuando miro mis compras anteriores, a veces la información no está actualizada y no sé bien si ya me enviaron el pedido o no. También creo que la búsqueda de productos podría ser más fácil. Es un poco frustrante a veces.

5 . Descripción de Fallas

Basado en el contexto, las entrevistas y el análisis del sistema actual, las fallas observables del sistema monolítico incluyen:

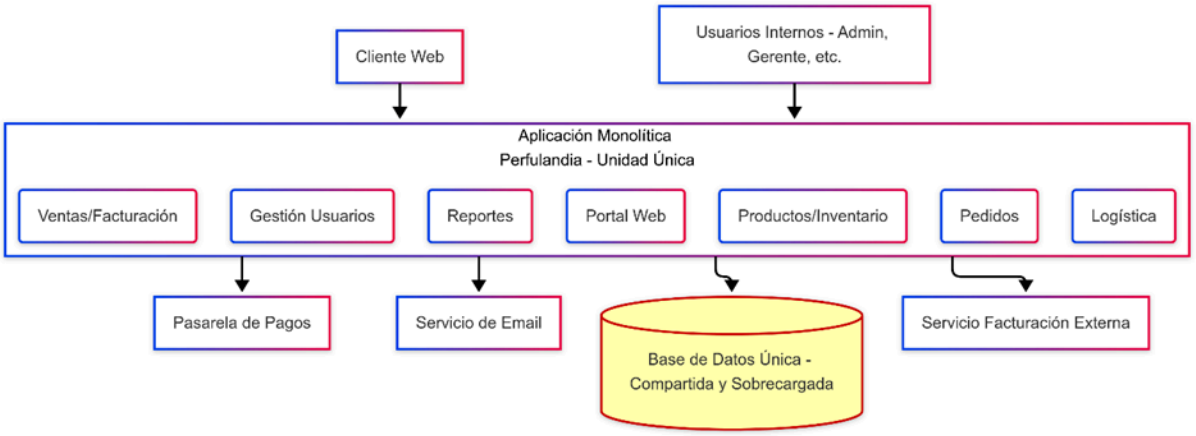
- **Caídas Frecuentes:** El sistema deja de responder completamente durante períodos de alta demanda, confirmando el problema de que “Falla un módulo afecta a todo el sistema”. Esto impacta todos los módulos: Ventas, Inventario, Pedidos, Logística, etc.
- **Lentitud Extrema:** Tiempos de respuesta muy altos (>10-15 segundos) en operaciones críticas como:
    - Procesamiento de pagos (Ventas y Facturación), validando la “Lentitud al procesar ventas en períodos de alta demanda”.
    - Búsqueda de productos (Gestión de Productos, Portal Cliente Web).
    - Generación de Reportes.
    - Actualización de stock (Gestión de Inventario).
- **Inconsistencia de Datos:** Discrepancias entre el stock registrado (Inventario) y el físico, probablemente debido a la sobrecarga y problemas de concurrencia en la base de datos compartida por el monolito.
- **Errores no Manejados:** Errores genéricos durante operaciones como aplicar descuentos (Ventas) o procesar devoluciones, relacionados con la “Complejidad al aplicar cambios o correcciones específicas sin riesgo general”.
- **Dificultad en la Recuperación:** Largos períodos de inactividad tras una caída, inherentes a la complejidad de reiniciar y verificar un sistema monolítico completo.
- **Problemas de Concurrencia:** Errores cuando múltiples usuarios acceden o modifican recursos compartidos (ej. Inventario de un producto) simultáneamente, exacerbado por la incapacidad de escalar módulos específicos como Ventas o Inventario de forma independiente.

6 . Principales Cuellos de Botella

Las fallas descritas son síntomas de cuellos de botella subyacentes en la arquitectura monolítica:

- **Base de Datos Centralizada y Sobrecargada:** Un único esquema o base de datos soporta todos los módulos (Usuarios, Productos, Inventario, Ventas, Pedidos, Logística, etc.). Consultas pesadas del módulo de Reportes impactan directamente el rendimiento del módulo de Ventas y Facturación, causando la lentitud observada.
- **Servidor de Aplicaciones Único/Limitado:** El monolito completo se ejecuta en una infraestructura que no puede escalar por partes. Un pico de carga en el Portal Cliente Web o Ventas consume recursos que afectan a Logística, Inventario y Reportes, reflejando la “Dificultad para escalar de forma modular”.
- **Gestión de Inventario en Tiempo Real:** La lógica para manejar el stock (Inventario) de forma concurrente entre sucursales y ventas es un punto crítico dentro del monolito, propenso a bloqueos.
- **Procesos de Reportes Intensivos:** El módulo de Generación de Reportes, al operar sobre la base de datos transaccional, genera carga significativa que ralentiza otras operaciones críticas como las Ventas. Esto apoya la idea de separarlo.
- **Código Altamente Acoplado:** La falta de separación clara entre los módulos (Ventas, Inventario, Pedidos, etc.) dentro del monolito hace que cualquier cambio sea riesgoso y complejo, apoyando la afirmación “Complejidad al aplicar cambios”.

7 . Causas Técnicas



Las causas técnicas raíz, que explican los problemas del monolito son:

- **Arquitectura Monolítica:** Esta es la causa fundamental. Impide escalar módulos como Ventas o Inventario independientemente y crea un único punto de fallo.
- **Tecnología Obsoleta:** El stack subyacente puede carecer de optimizaciones para la carga actual que manejan los módulos de Ventas e Inventario.
- **Gestión Ineficiente de Conexiones a la Base de Datos:** Agrava la sobrecarga de la base de datos compartida por todos los módulos.
- **Falta de Asincronismo:** Operaciones como la Generación de Facturas (parte de Ventas y Facturación) o notificaciones de Pedidos realizadas síncronamente bloquean recursos.
- **Ausencia de Caching:** Falta de caché para el Catálogo de Productos o datos de Inventario frecuentemente consultados aumenta la carga en la base de datos.
- **Infraestructura Insuficiente:** Hardware sub-dimensionado para soportar la carga combinada de todos los módulos del monolito.
- **Monitorización y Alertas Limitadas:** Dificultan identificar qué módulo (Ventas, Reportes, Inventario, etc.) está causando la degradación del rendimiento en un momento dado.

8 . Causas Organizativas

Más allá de los aspectos técnicos, ciertos factores organizativos contribuyeron significativamente a la situación actual del sistema y a los problemas de rendimiento y disponibilidad previamente detallados:

- **Crecimiento Rápido vs. Inversión Tecnológica:** La rápida expansión del negocio superó la capacidad original de la infraestructura y el software de soporte, ya que la inversión tecnológica necesaria no se realizó al mismo ritmo.
- **Enfoque en Funcionalidades sobre Estabilidad:** Se priorizó la entrega rápida de nuevas características requeridas por el negocio en expansión sobre la refactorización, optimización y aseguramiento de la estabilidad de la arquitectura del sistema existente.
- **Falta de Prácticas DevOps/SRE:** La ausencia de una cultura y herramientas modernas para la integración continua, el despliegue automatizado y la monitorización avanzada dificultó el mantenimiento eficiente y la evolución controlada del complejo sistema centralizado.
- **Deuda Técnica Acumulada:** Decisiones técnicas pasadas, a menudo orientadas a cumplir plazos cortos, resultaron en atajos y falta de refactorización, generando una deuda que hoy se manifiesta en la complejidad, fragilidad y dificultad de mantenimiento del sistema.
- **Silos de Conocimiento:** El conocimiento profundo sobre diferentes áreas funcionales críticas del sistema integrado tendía a concentrarse en pocas personas, lo que complicaba la resolución de problemas transversales y la evolución coherente de la aplicación.
- **Resistencia al Cambio:** Pudo existir una reticencia interna a adoptar enfoques arquitectónicos o tecnologías más modernas, a pesar de los evidentes y crecientes problemas de rendimiento, escalabilidad y disponibilidad que presentaba la estructura actual.

9 . Propuesta de Servicios

A continuación se describe una propuesta de microservicios para la resolución del caso. Se ha optado por un enfoque que combine robustez con sencillez, de forma que sean herramientas con las cuales el equipo tenga experiencia o que su aprendizaje pueda ser abordado con bajos costos de tiempo.

9.1. Tecnologías base

9.1.1. Backend: Spring Boot

Framework ligero de desarrollo en Java que permite construir microservicios de forma rápida y estructurada. Se complementa con:

- Spring Web: para crear APIs RESTful que permiten la comunicación entre servicios y clientes.
- Spring Data JPA: para acceder y manipular datos en bases de datos relacionales de forma simplificada usando repositorios.
- Spring Security: para gestionar la seguridad, incluyendo autenticación y control de acceso por roles.

9.1.2. Base de Datos: Oracle Database

Sistema de gestión de bases de datos relacional robusto y escalable. Puede utilizarse como una instancia común compartida entre los microservicios, o bien asignar un esquema por servicio, respetando el aislamiento lógico.

9.1.3. API Gateway (opcional): Spring Cloud Gateway

Herramienta que actúa como una puerta de entrada única para todas las peticiones a los microservicios. Facilita la gestión de rutas, la aplicación de filtros de seguridad y el monitoreo del tráfico, simplificando el acceso desde el frontend o clientes externos.

9.1.4. Descubrimiento de Servicios (opcional): Eureka Server

Componente de Spring Cloud que permite registrar y ubicar dinámicamente los microservicios dentro de un entorno distribuido. Es útil para simular escalabilidad y reemplazar direcciones fijas por nombres lógicos.

9.1.5. Autenticación: JWT con Spring Security

Se utiliza JSON Web Tokens (JWT) para autenticar a los usuarios de forma segura. El token se genera al iniciar sesión y se incluye en cada solicitud para validar permisos, evitando la necesidad de sesiones en el servidor.

9.1.6. Despliegue: Docker (opcional)

Plataforma que permite empaquetar cada microservicio en un contenedor independiente con su propio entorno. Esto facilita el despliegue, la portabilidad y la ejecución uniforme en distintos entornos, ideal para pruebas y simulaciones finales del proyecto.

9.2. Microservicios propuestos

Microservicio	Descripción	Tabla/s principales
Usuarios-Service	Gestión de usuarios, roles y autenticación	usuarios, roles
Productos-Service	Gestión del catálogo de productos	productos, categorías
Inventario-Service	Manejo de stock por sucursal	inventario, sucursales
Ventas-Service	Registro de ventas, facturas, descuentos	ventas, facturas, detalles_venta
Pedidos-Service	Pedido de clientes y reabastecimiento	pedidos, estado_pedido
Logística-Service	Envíos y planificación de rutas	envíos, rutas, proveedores

Reportes-Service	Generación de reportes (por ahora como endpoints simples)	(Consulta cruzada)
Clientes-Web-Service	Acciones del cliente en la web	(se conecta a ventas, productos, pedidos, auth)

9.3. Seguridad básica

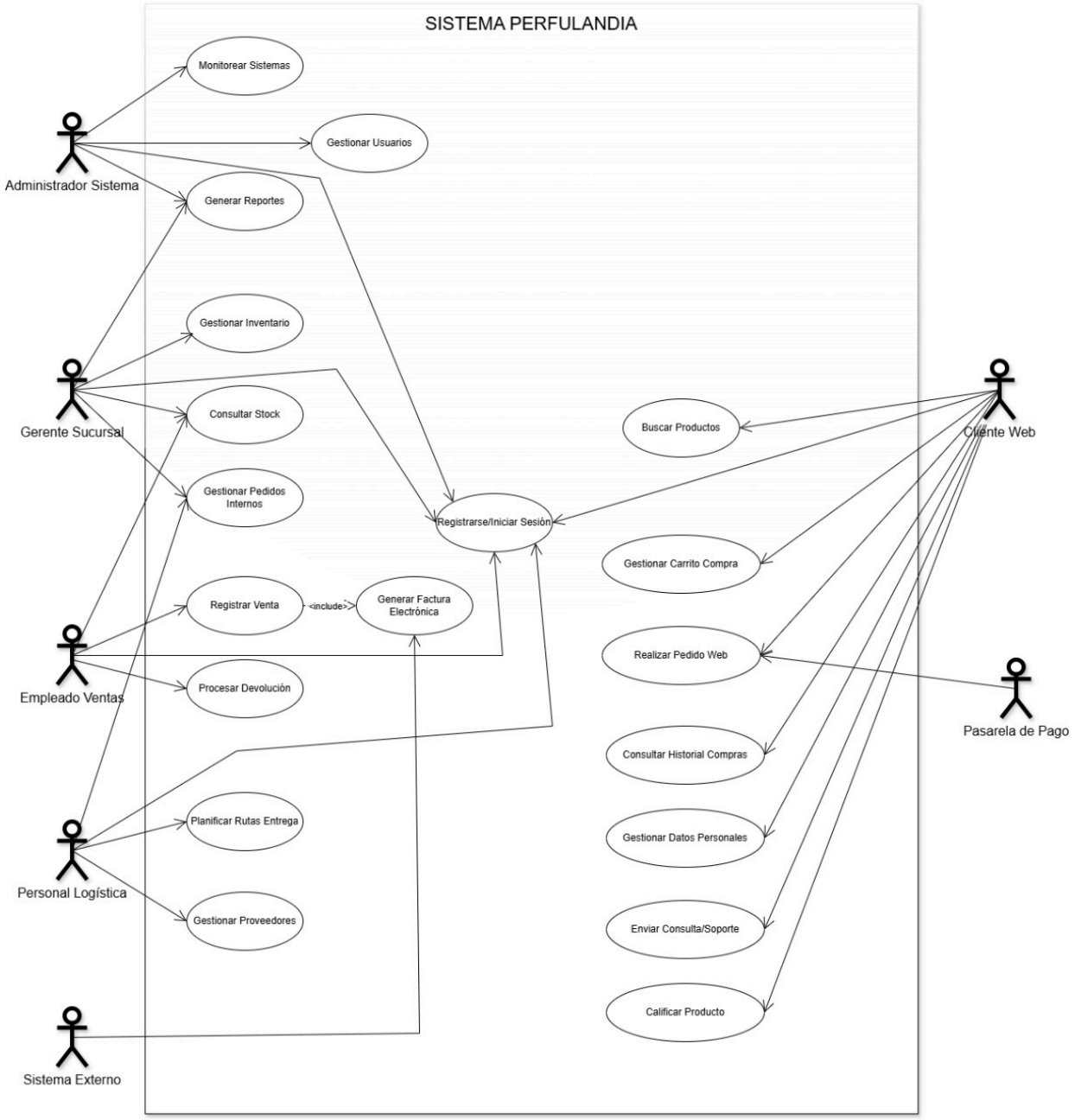
- Servicio de autenticación con login + generación de token JWT.
- Uso de roles (ADMIN, GERENTE, VENTAS, LOGISTICA, CLIENTE) para controlar el acceso desde Spring Security.
- Verificación del token desde un filtro global en cada microservicio.

9.4. Relación entre Tecnologías y Microservicios.

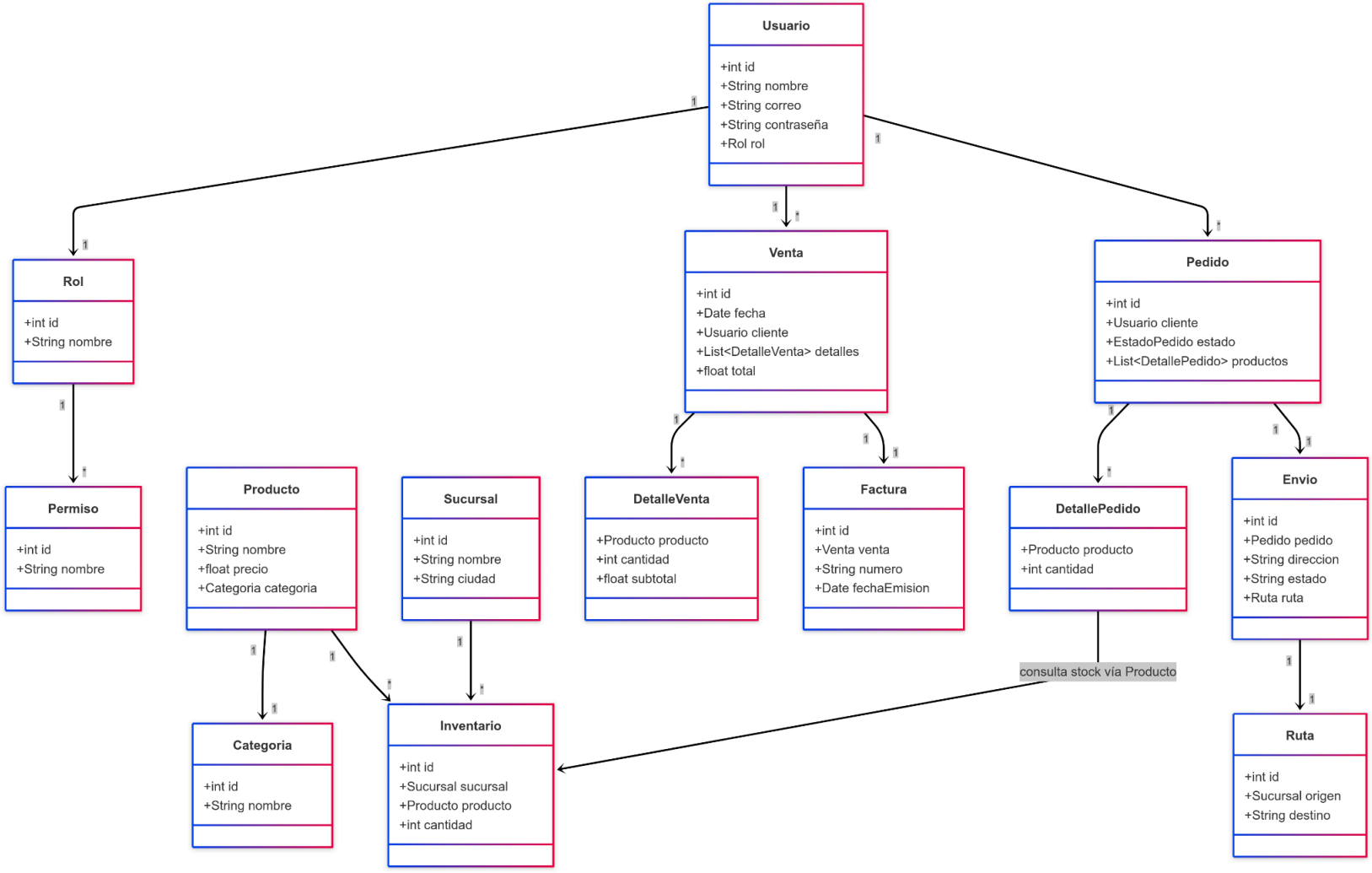
Tecnología	**Microservicios que la utilizan**	**Propósito principal**
Spring Boot (Web + JPA)	Todos los microservicios	Backend REST y acceso a datos
Spring Security + JWT	Usuarios-Service, Clientes-Web-Service, + validación en otros	Autenticación y control de acceso
Oracle Database	Todos los microservicios con datos persistentes	Almacenamiento relacional
Spring Cloud Gateway	Todos, especialmente Clientes-Web-Service	Centralización de rutas y seguridad
Eureka Server	Todos (opcional)	Descubrimiento de servicios
Docker	Todos	Contenerización y despliegue uniforme

10 . Diagramas

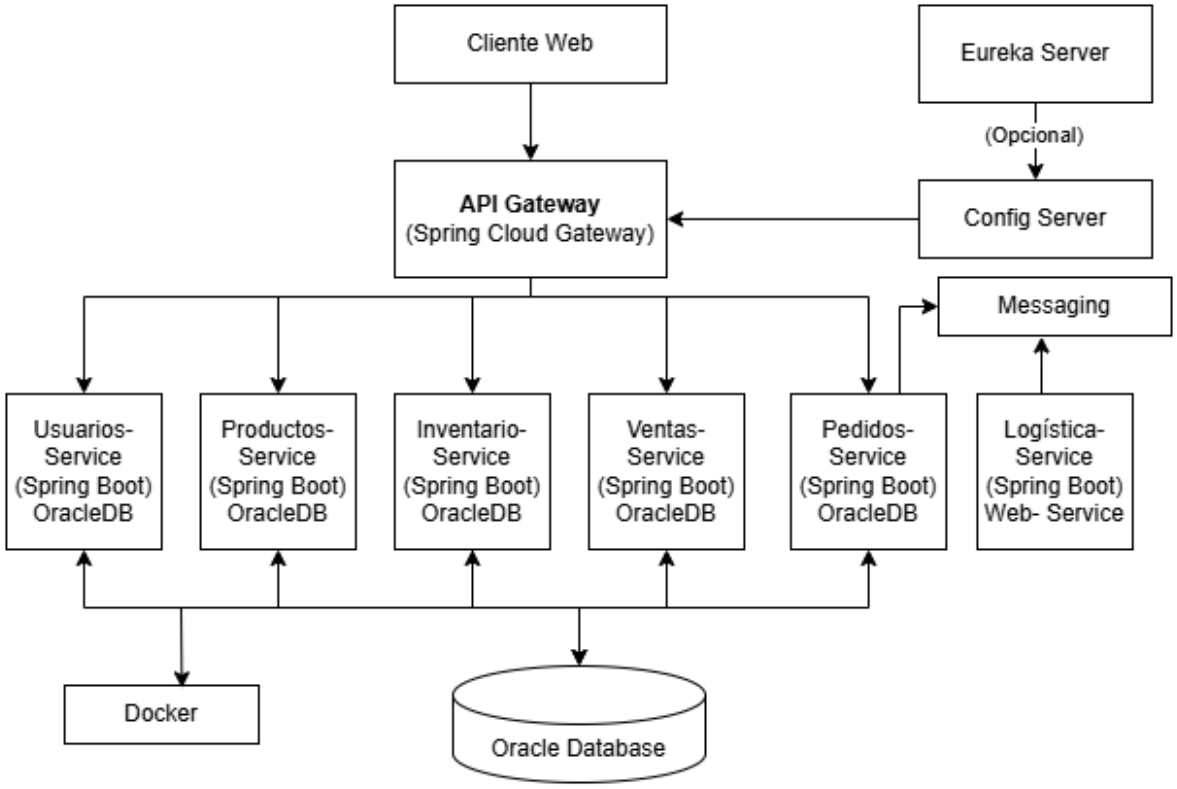
10.1. Diagrama de Casos de Uso



10.2. Diagrama de Clases



10.3. Diagrama de Arquitectura de Microservicios



11 . Plan de Migración

11.1. Análisis del Sistema Actual

El sistema actual de Perfulandia SPA es una aplicación monolítica que integra múltiples módulos funcionales en un único despliegue. Entre los principales módulos se encuentran:

- Gestión de Usuarios y Roles: Control de acceso, autenticación, creación de cuentas.
- Gestión de Productos e Inventario: Catálogo y stock por sucursal.
- Ventas y Facturación: Registro de transacciones, descuentos, emisión de facturas.
- Gestión de Pedidos: Recepción de pedidos desde sucursales o clientes web.
- Logística y Proveedores: Administración de rutas de entrega y abastecimiento.
- Generación de Reportes: Informes administrativos y de rendimiento.
- Portal Cliente Web: Interfaz de compra, consulta de productos y seguimiento de pedidos.

Los problemas principales del sistema monolítico actual incluyen:

- Dificultad para escalar de forma modular.
- Falla en un módulo afecta a todo el sistema.
- Lentitud al procesar ventas en períodos de alta demanda.
- Complejidad al aplicar cambios o correcciones específicas sin riesgo general.

Se identifican como candidatos de separación inicial los módulos de Reportes y Logística, por su bajo acoplamiento y menor criticidad.

11.2. Definición de Microservicios

Con base en el análisis anterior, el sistema se dividirá en los siguientes **microservicios**, desarrollados con Spring Boot, usando Oracle Database como backend común:

Microservicio	Descripción
Usuarios-Service	Autenticación, gestión de usuarios y roles.
Productos-Service	Administración del catálogo de productos y categorías.
Inventario-Service	Control del stock por sucursal.
Ventas-Service	Registro de ventas, facturas y aplicación de descuentos.
Pedidos-Service	Gestión de pedidos de clientes y sucursales.
Logistica-Service	Control de envíos, rutas de entrega y proveedores.
Reportes-Service	Generación de reportes simples (acceso solo lectura).
Clientes-Web-Service	Portal de cliente, que orquesta pedidos y navegación.

Se utilizarán JWT para autenticación segura y Spring Security para control de roles. Se recomienda Spring Cloud Gateway como punto de entrada y Eureka para simular descubrimiento de servicios.

11.3. Prueba de Concepto

La prueba de concepto inicial se centrará en dos microservicios de baja dependencia:

1. Logistica-Service
    - Permite registrar envíos, asociarlos a pedidos y actualizar su estado.
    - Utiliza tablas independientes (envíos, rutas, proveedores).

2. Reportes-Service

- Expone endpoints de lectura que consolidan datos de otros servicios.
- Simula integración a través de llamadas REST o datos simulados.

Estos microservicios se desarrollarán, probarán y desplegarán de forma aislada, sin modificar directamente el sistema monolítico.

11.4. Reestructuración Incremental

Una vez validados los microservicios iniciales, se migrarán gradualmente los módulos restantes:

1. Usuarios-Service: Para autenticación con JWT y gestión de roles.
2. Productos-Service: Separación del catálogo de productos.
3. Inventario-Service: Control del stock descentralizado por sucursal.
4. Pedidos-Service y Ventas-Service: Separados con coordinación vía APIs o colas de mensajes.

Durante esta etapa, los microservicios nuevos se comunicarán con el sistema monolítico por APIs REST o mediante mensajería asíncrona (RabbitMQ o similar).

11.5. Despliegue en Contenedores

Todos los microservicios serán empaquetados con Docker para asegurar consistencia y portabilidad.

Para la orquestación y el escalado, se utilizará Kubernetes, lo cual permite:

- Definir réplicas por servicio.
- Balanceo de carga automático.
- Reinicio de contenedores ante fallos.
- Escalabilidad horizontal por demanda.

Esta arquitectura simula un entorno productivo simplificado, ideal para prácticas académicas.

11.6. Integración y Despliegue Continuo (CI/CD)

Se configurará un pipeline de CI/CD básico usando GitHub Actions, con:

- Build y test automático en cada commit.
- Despliegue automatizado a un entorno de pruebas.
- Control de versiones y rollback ante fallas.
- Posibilidad de notificar cambios por correo o dashboard.

Esto permitirá a los estudiantes aplicar buenas prácticas en ciclos cortos de desarrollo.

11.7. Eliminación del Monolito

Una vez que todos los microservicios han sido implementados, probados e integrados correctamente:

- Se desconectarán los módulos equivalentes del monolito de forma progresiva.
- Se redirigirá el tráfico al conjunto de microservicios.
- Finalmente, se retirará el sistema monolítico, dejando solo la arquitectura distribuida operativa.

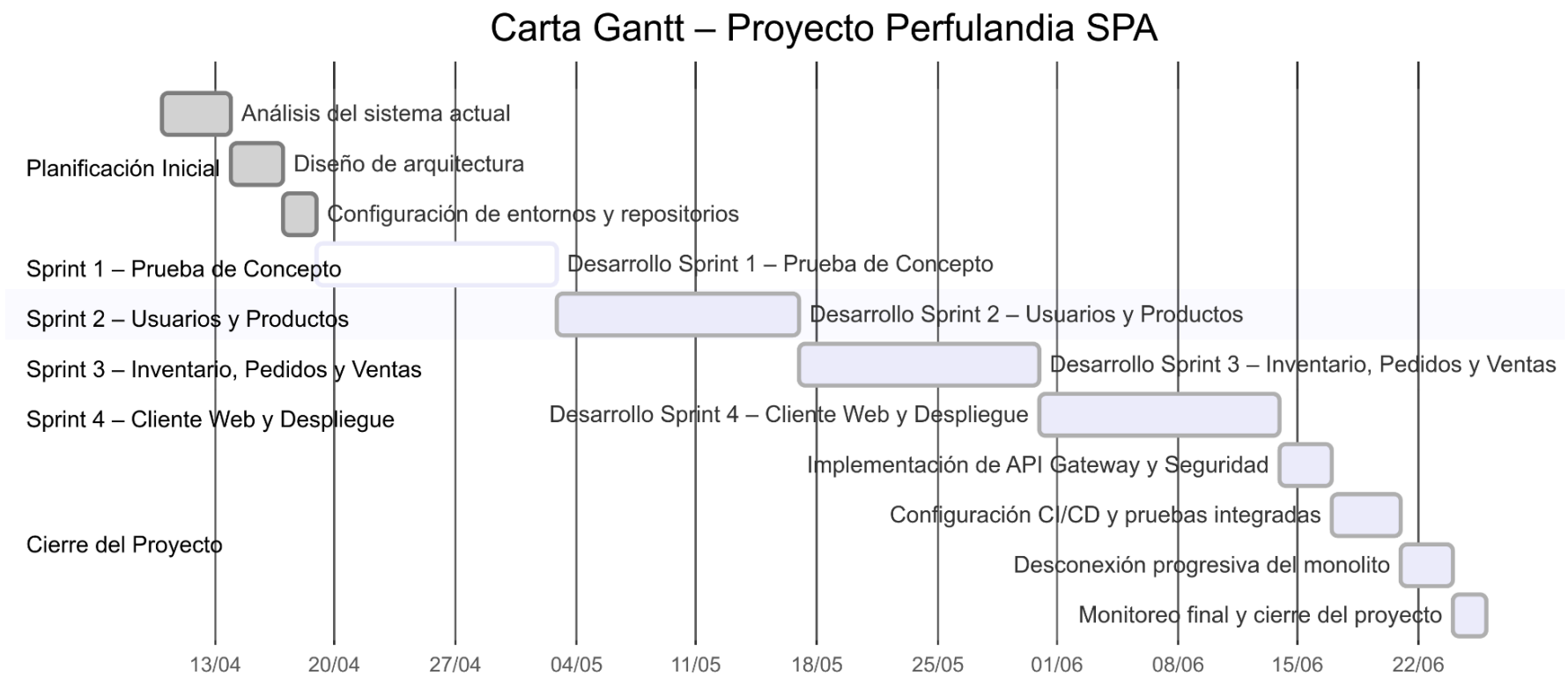
Esto se hará asegurando la mínima interrupción al negocio y manteniendo la trazabilidad completa del sistema.

11.8. Carta Gantt – Implementación por Plan de Migración

Esta carta Gantt sigue las fases del proyecto de migración propuesto.

Actividad	Fecha Inicio	Duración (días)	Fecha Fin
Análisis del sistema actual	2025-04-10	4	2025-04-13
Diseño de arquitectura	2025-04-14	3	2025-04-16
Configuración de entornos y repositorios	2025-04-17	2	2025-04-18
Desarrollo Sprint 1 – Prueba de Concepto	2025-04-19	14	2025-05-02
Desarrollo Sprint 2 – Usuarios y Productos	2025-05-03	14	2025-05-16

Desarrollo Sprint 3 – Inventario, Pedidos y Ventas	2025-05-17	14	2025-05-30
Desarrollo Sprint 4 – Cliente Web y Despliegue	2025-05-31	14	2025-06-13
Implementación de API Gateway y Seguridad	2025-06-14	3	2025-06-16
Configuración CI/CD y pruebas integradas	2025-06-17	4	2025-06-20
Desconexión progresiva del monolito	2025-06-21	3	2025-06-23
Monitoreo final y cierre del proyecto	2025-06-24	2	2025-06-25


Carta Gantt – Proyecto Perfulandia SPA
Planificación Inicial
Análisis del sistema actual
Diseño de arquitectura
Configuración de entornos y repositorios
Sprint 1 – Prueba de Concepto
Desarrollo Sprint 1 – Prueba de Concepto
Sprint 2 – Usuarios y Productos
Desarrollo Sprint 2 – Usuarios y Productos
Sprint 3 – Inventario, Pedidos y Ventas
Desarrollo Sprint 3 – Inventario, Pedidos y Ventas
Sprint 4 – Cliente Web y Despliegue
Desarrollo Sprint 4 – Cliente Web y Despliegue
Implementación de API Gateway y Seguridad
Configuración CI/CD y pruebas integradas
Cierre del Proyecto
Desconexión progresiva del monolito
Monitoreo final y cierre del proyecto
13/04
20/04
27/04
04/05
11/05
18/05
25/05
01/06
08/06
15/06
22/06

11.9. Product Backlog – Historias de Usuario

Este backlog incluye historias derivadas de los requisitos funcionales que serán consideradas para la etapa de desarrollo, organizadas por microservicio, considerando su nivel de prioridad y puntos de historia requeridos para su desarrollo.

ID	Historia de Usuario	Prioridad	Microservicio	Puntos de Historia
US01	Registrar usuario con correo y contraseña	Alta	Usuarios-Service	5
US02	Iniciar sesión con credenciales	Alta	Usuarios-Service	5
US03	Gestión CRUD de usuarios internos	Alta	Usuarios-Service	5
US04	Asignar y modificar roles	Alta	Usuarios-Service	5
US05	Agregar productos al inventario	Alta	Inventario-Service	5
US06	Actualizar y eliminar productos	Alta	Inventario-Service	5
US07	Mostrar disponibilidad por sucursal	Alta	Inventario-Service	5
US08	Ajuste manual de stock	Media	Inventario-Service	3
US09	Registrar ventas con descuentos	Alta	Ventas-Service	5
US10	Generar factura electrónica	Alta	Ventas-Service	5
US11	Enviar factura por correo	Media	Notificaciones-Service	3
US12	Procesar devoluciones y reembolsos	Media	Ventas-Service	3
US13	Generar pedidos de	Alta	Pedidos-Service	5

	reabastecimiento			
US14	Planificar rutas de entrega	Media	Logistica-Service	3
US15	Actualizar estado de pedidos	Alta	Pedidos-Service	5
US16	Registrar y mantener proveedores	Media	Logistica-Service	3
US17	Reportes de ventas por periodo y sucursal	Baja	Reportes-Service	2
US18	Reportes de inventario por producto y sucursal	Baja	Reportes-Service	2
US19	Monitoreo del estado de los módulos	Media	Reportes-Service	3
US20	Alertas ante fallos críticos	Media	Reportes-Service	3
US21	Registro de cuenta cliente	Alta	Clientes-Web-Service	5
US22	Buscar y filtrar productos	Alta	Clientes-Web-Service	5
US23	Agregar productos al carrito	Alta	Clientes-Web-Service	5
US24	Realizar pedido con pago y envío	Alta	Clientes-Web-Service	5
US25	Aplicar cupones en la compra	Media	Clientes-Web-Service	3
US26	Ver historial de pedidos	Alta	Clientes-Web-Service	5
US27	Editar datos personales y direcciones	Alta	Clientes-Web-Service	5
US28	Enviar consultas al soporte	Media	Clientes-Web-Service	3
US29	Dejar calificaciones y comentarios	Media	Clientes-Web-Service	3

11.10. Tablero Kanban por Sprint

Cada sprint representa una fase coherente con el plan de migración y detalla la etapa de desarrollo, partiendo con la prueba de concepto y terminando con el despliegue y cierre.

11.10.1. Sprint 1 – Prueba de Concepto

ID	Historia de Usuario	Puntos	Microservicio
US01	Registrar usuario con correo y contraseña	5	Usuarios-Service
US24	Realizar pedido con pago y envío	5	Clientes-Web-Service
US02	Iniciar sesión con credenciales	5	Usuarios-Service
US09	Registrar ventas con descuentos	5	Ventas-Service
US06	Actualizar y eliminar productos	5	Inventario-Service
US19	Monitoreo del estado de los módulos	3	Reportes-Service
US18	Reportes de inventario por producto y sucursal	2	Reportes-Service

11.10.2. Sprint 2 – Usuarios y Productos

ID	Historia de Usuario	Puntos	Microservicio
US13	Generar pedidos de reabastecimiento	5	Pedidos-Service
US23	Agregar productos al carrito	5	Clientes-Web-Service
US15	Actualizar estado de pedidos	5	Pedidos-Service
US03	Gestión CRUD de usuarios internos	5	Usuarios-Service
US11	Enviar factura por correo	3	Notificaciones-Service
US20	Alertas ante fallos críticos	3	Reportes-Service
US16	Registrar y mantener proveedores	3	Logistica-Service

US17	Reportes de ventas por periodo y sucursal	2	Reportes-Service

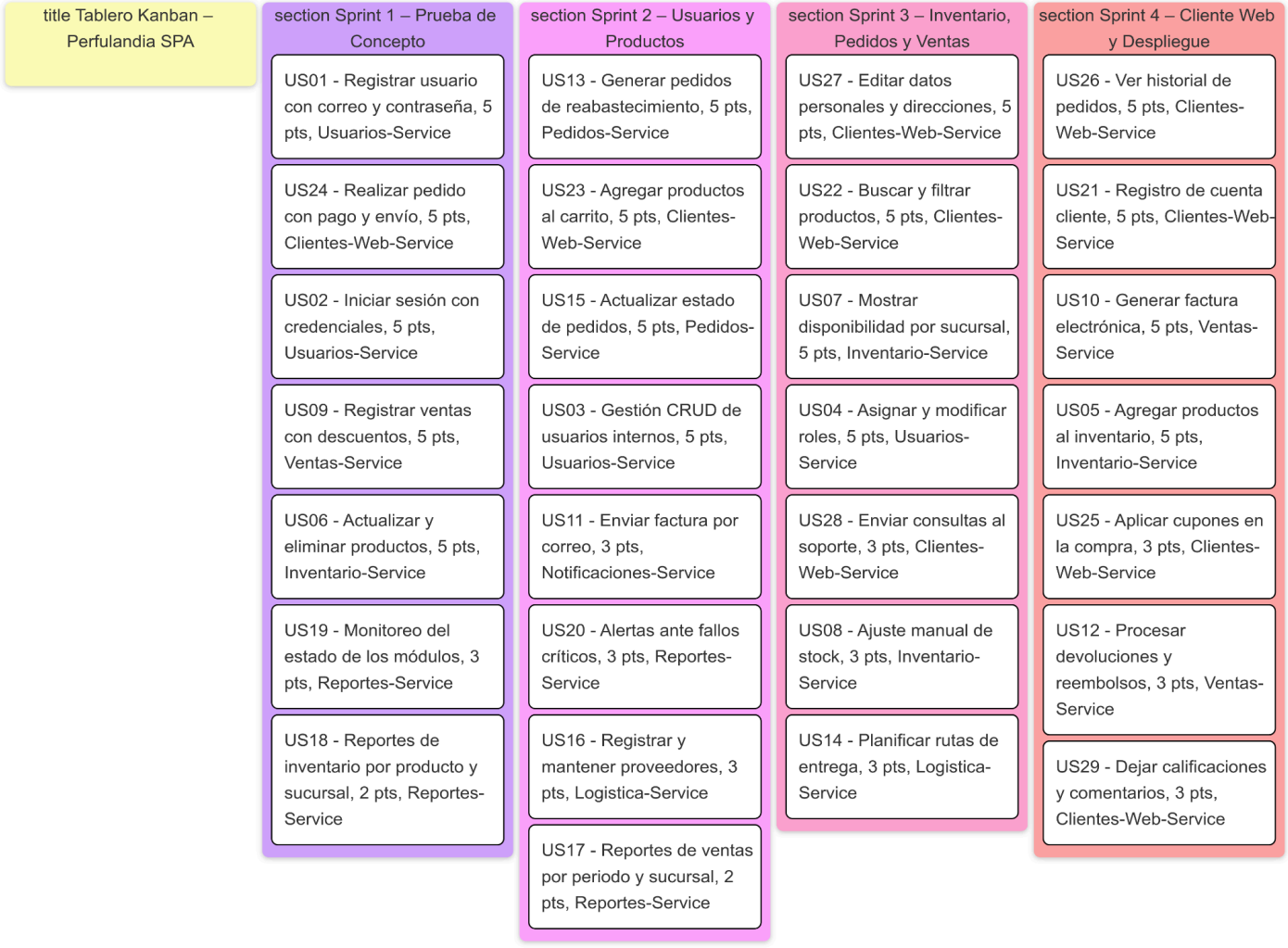
11.10.3. Sprint 3 – Inventario, Pedidos y Ventas

ID	Historia de Usuario	Puntos	Microservicio
US27	Editar datos personales y direcciones	5	Clientes-Web-Service
US22	Buscar y filtrar productos	5	Clientes-Web-Service
US07	Mostrar disponibilidad por sucursal	5	Inventario-Service
US04	Asignar y modificar roles	5	Usuarios-Service
US28	Enviar consultas al soporte	3	Clientes-Web-Service
US08	Ajuste manual de stock	3	Inventario-Service
US14	Planificar rutas de entrega	3	Logistica-Service

11.10.4. Sprint 4 – Cliente Web y Despliegue

ID	Historia de Usuario	Puntos	Microservicio
US26	Ver historial de pedidos	5	Clientes-Web-Service
US21	Registro de cuenta cliente	5	Clientes-Web-Service
US10	Generar factura electrónica	5	Ventas-Service
US05	Agregar productos al inventario	5	Inventario-Service
US25	Aplicar cupones en la compra	3	Clientes-Web-Service
US12	Procesar devoluciones y reembolsos	3	Ventas-Service
US29	Dejar calificaciones y comentarios	3	Clientes-Web-Service

11.10.5. Tablero Kanban, Representación Gráfica.



12. Conclusiones

La propuesta de migración de Perfulandia SPA hacia una arquitectura de microservicios representa una solución moderna y escalable frente a las limitaciones del sistema monolítico actual. A través del uso de tecnologías como Spring Boot, Oracle Database y JWT, se diseñaron ocho microservicios principales que abordan funciones clave del negocio, incluyendo usuarios, inventario, ventas, pedidos y logística. La implementación se estructura en cuatro sprints consecutivos, complementados con etapas de análisis, despliegue y cierre, abarcando un total de 77 días calendario. Esta estrategia progresiva permite una transición controlada, reduciendo riesgos y asegurando la continuidad operativa. En conjunto, el proyecto no solo mejora el rendimiento, disponibilidad y mantenibilidad del sistema, sino que también sienta las bases para una expansión tecnológica sostenible y adaptable a futuro.

13. Bibliografía y Referencias Web

- Spring Boot – https://spring.io/projects/spring-boot
- Docker – https://www.docker.com/
- Oracle SQL – https://www.oracle.com/database/
- JWT (JSON Web Tokens) – https://jwt.io/
- Spring Cloud Gateway – https://spring.io/projects/spring-cloud-gateway