| **DUOC UC - Escuela de informática y telecomunicaciones** |
| --- |
| Plan de Pruebas General |
| *Caso Semestral Proyecto: “Sistema Modular para Perfulandia SPA”* |
|  |
| **Revisión*: [ 01 ]*** |
| **[ 24 de junio de 2025 ]** |

| Principios y especificaciones de norma, estándar ISO/IEC 25000. |
| --- |

**Contenido**

[**Ficha del documento 3**](#_heading=h.g6eo3qbq28tz)

[**1. Introducción 4**](#_heading=h.njbc2thepigo)

[**2. Identificación del Software 4**](#_heading=h.cpr8hs1r212o)

[**3. Requisitos de Calidad (ISO/IEC 25010) 4**](#_heading=h.55e30j5lw077)

# Ficha del documento

| **Fecha** | **Revisión** | **Autor** | **Modificación** |
| --- | --- | --- | --- |
| *18-06-2025* | *—* | *—* | **Da comienzo al documento.**  **Estructurando el contenido.** |

Documento validado por las partes en fecha:

**Integrantes:**

| **Nombre Integrante del Equipo** | **Rol Definido** |
| --- | --- |
| Gabriel Toledo | * Líder de Proyecto * Analista de negocio * Arquitecto de sistemas * Desarrollador Backend |
| Camilo Tapia | * Analista de negocio * Desarrollador Backend * Desarrollador Frontend |
| Danae Collao | * Analista de negocio * Desarrollador Backend * Analista de calidad |

# Introducción

Este documento define el plan de pruebas para el sistema de gestión de operaciones empresariales de Perfulandia SPA, diseñado bajo una arquitectura de microservicios REST utilizando Java, Spring Boot y MariaDB. El objetivo es garantizar que el software cumpla con los requisitos funcionales y no funcionales, alineado con la metodología Waterfall y la norma ISO/IEC 25000.

Asimismo, este plan responde a las directrices establecidas en el apartado 13 del proyecto, las cuales solicitan:

* La implementación de pruebas alineadas a ISO/IEC 25000.
* Evidencias claras de calidad del software.
* Métricas de evaluación y un control de cambios documentado.

# Identificación del Software

* **Nombre del sistema**: Sistema Modular para Perfulandia SPA
* **Tecnologías**: Java, Spring Boot, Spring Security, REST, MariaDB
* **Arquitectura**: Microservicios + Arquitectura Hexagonal
* **Entorno**: Local (XAMPP + Postman), planificación para AWS

# Requisitos de Calidad (ISO/IEC 25010)

El modelo de calidad utilizado es el propuesto por ISO/IEC 25010, sub-norma perteneciente al marco ISO/IEC 25000 (SQuaRE). Se han considerado las siguientes características:

* **Funcionalidad**: Todos los endpoints deben cumplir los requisitos funcionales desde el RF01 al RF15.
* **Fiabilidad**: Tolerancia ante errores, validaciones, integridad de datos.
* **Usabilidad**: Interacciones comprensibles y retroalimentación clara.
* **Eficiencia**: El sistema debe mantener tiempos de respuesta menores a 2000 ms.
* **Seguridad**: Control de acceso mediante JWT, roles, y cifrado.
* **Mantenibilidad**: Estructura de código desacoplado, modular y legible.

**Definiciones, Acrónimos y Abreviaturas**

| **Término / Acrónimo** | **Definición** |
| --- | --- |
| API (Application Programming Interface) | Conjunto de funciones y protocolos que permite la comunicación entre diferentes componentes de software. En este sistema se utilizarán APIs REST para la interacción entre microservicios. |
| AWS (Amazon Web Services) | Plataforma de servicios en la nube que provee infraestructura tecnológica escalable. Se usará para desplegar los microservicios del sistema. |
| AWS Lambda | Servicio serverless de AWS que permite ejecutar código sin necesidad de administrar servidores. Los microservicios serán ejecutados como funciones Lambda. |
| JWT (JSON Web Token) | Estándar abierto para la transmisión segura de información entre partes como un objeto JSON firmado digitalmente. Será utilizado para autenticación y control de acceso. |
| REST (Representational State Transfer) | Estilo de arquitectura para el diseño de servicios web. Se usará en la definición de las APIs que comunican los microservicios. |
| Spring Boot | Framework de desarrollo basado en Java que simplifica la creación de aplicaciones empresariales y microservicios. Será la tecnología base del sistema backend. |
| Microservicios | Estilo arquitectónico donde las funcionalidades del sistema se dividen en servicios pequeños, independientes y desplegables por separado. |
| Arquitectura Hexagonal | Modelo de diseño de software que separa el núcleo de la lógica del negocio de sus interfaces externas, facilitando la mantenibilidad y escalabilidad del sistema. |
| Base de Datos Relacional | Sistema de almacenamiento estructurado de datos, basado en tablas con relaciones entre ellas. Cada microservicio tendrá su propia base de datos. |
| Hash Function | Función criptográfica que transforma una entrada en una cadena de caracteres fija, usada en este caso para asegurar y verificar contraseñas y datos sensibles. |
| Sistema monolítico | Enfoque tradicional de desarrollo de software en el que todos los componentes del sistema están integrados en una sola aplicación. Actualmente utilizado por Perfulandia SPA y reemplazado por la nueva arquitectura. |
| Serverless (Sin servidor) | Modelo de ejecución en la nube donde el proveedor gestiona automáticamente la infraestructura, permitiendo al equipo de desarrollo enfocarse en el código. |
| Objeto | Unidad básica de la programación orientada a objetos, que representa una entidad del mundo real con atributos y comportamientos. |
| Clase | Plantilla para la creación de objetos, definida por atributos (datos) y métodos (funciones). |
| UML (Unified Modeling Language) | Lenguaje estandarizado para el modelado de sistemas de software orientados a objetos, utilizado para representar la estructura y comportamiento del sistema. |
| ERS (Especificación de Requisitos del Software) | Documento que define en detalle los requisitos funcionales y no funcionales de un sistema, así como sus restricciones, actores y contexto de operación. |
| S.O.L.I.D. (Single Responsibility, Open /Closed, Liskov Substitution, Interface Segregation y Dependency Inversion) | Conjunto de principios fundamentales del diseño orientado a objetos que promueven la creación de software robusto, mantenible y escalable. |
| CI / CD (Integración Continua / Entrega Continua) | Prácticas de desarrollo de software que automatizan el proceso de construcción, pruebas y despliegue de aplicaciones. |
| HTTP / HTTPS (HyperText Transfer Protocol / Secure) | Protocolo de comunicación para transferencia de datos web sin cifrado / o con cifrado. |
| TLS (Transport Layer Security) | Protocolo criptográfico que proporciona seguridad y cifrado a las comunicaciones. |

**Referencias**

Documentos y recursos referenciados en el presente documento de Especificación de Requisitos del Software (ERS) para el proyecto de transformación digital de “Perfulandia SPA”:

1. IEEE Std 830-1998 – IEEE Recommended Practice for Software Requirements Specifications.
2. Documento (Anexo 3) de Análisis del Caso: *Transformación Digital de Perfulandia SPA*.
3. Manual de Arquitectura Hexagonal aplicada a Microservicios – Universidad Tecnológica de Chile (2024).
4. Guía de implementación de Spring Boot en AWS Lambda – AWS Developer Docs.
5. Recomendaciones de seguridad con JWT – OWASP JSON Web Token Cheat Sheet.
6. Buenas prácticas en desarrollo de Microservicios – Oracle Java Magazine (Ed. 2023).
7. Documentación oficial de Spring Boot –<https://spring.io/projects/spring-boot>
8. Documentación oficial de AWS Lambda –<https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>
9. Guía Académica de Especificación de Requisitos IEEE 830 – Departamento de Informática, Universidad (2023).

**Visión General del Documento**

* **Sección 1**: Introducción al proyecto, propósito, alcance y visión del producto.
* **Sección 2**: Descripción general del sistema, incluyendo la perspectiva del producto, funciones principales y características de los usuarios.
* **Sección 3**: Requisitos del sistema, tanto funcionales como no funcionales.
* **Sección 4**: Detalles sobre la planificación de la propuesta, Hitos, Tareas y carta gantt

* **Descripción General**

En esta sección se describirán aquellos factores que afectan al producto y el contexto de sus requisitos. Esto permitirá definir con detalle más adelante los requisitos específicos, haciendo que sean más entendibles.

**Perspectiva del Producto**

El “Sistema modular para Operaciones Empresariales de Perfulandia SPA” es una solución tecnológica orientada a reemplazar el sistema monolítico actual de Perfulandia SPA por una arquitectura moderna, escalable y desacoplada basada en microservicios. Esta transformación permitirá resolver las limitaciones técnicas y de crecimiento que presenta el sistema actual, mediante un diseño modular y orientado a objetos, que facilite la evolución y mantenimiento del software.

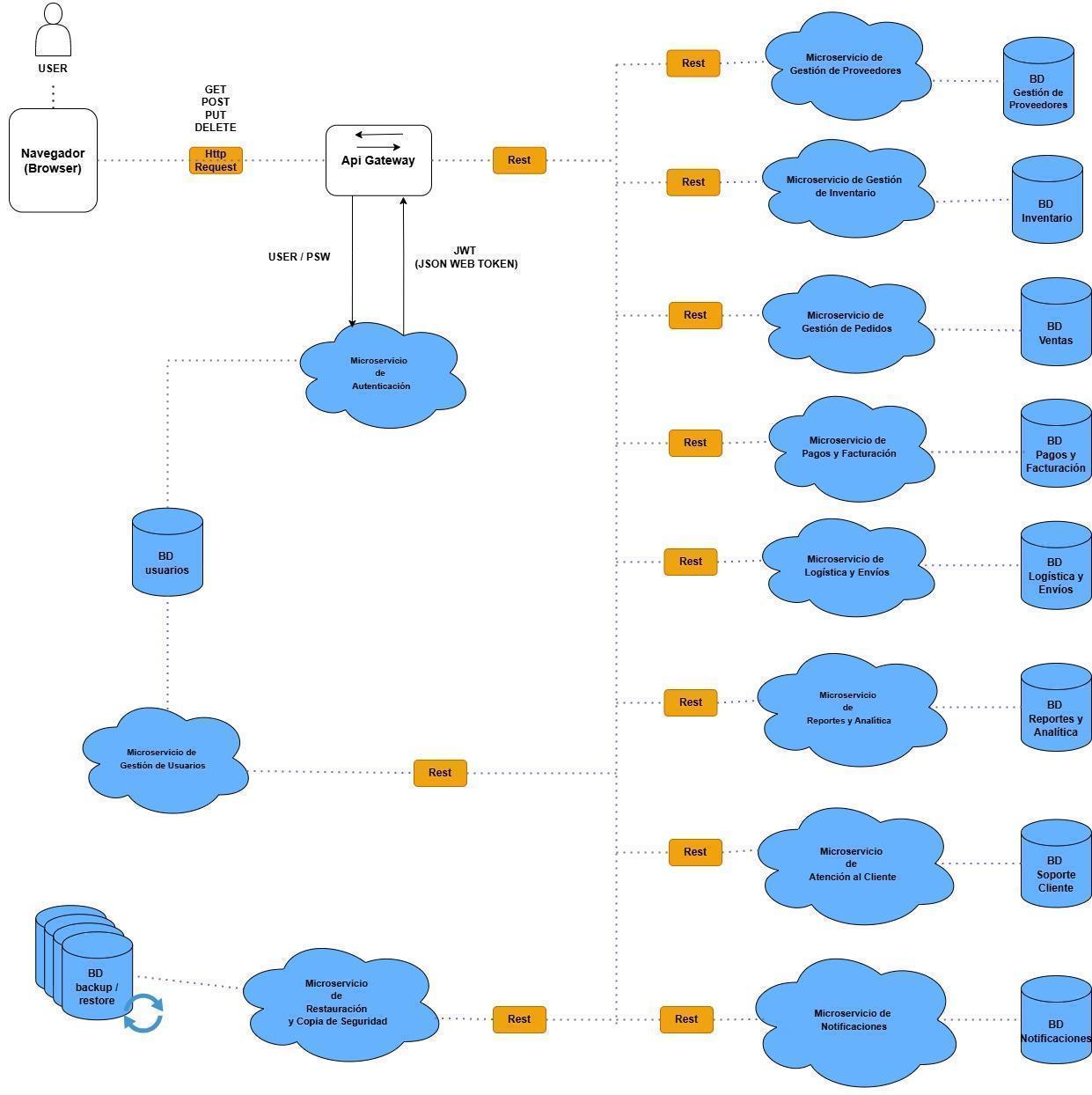
La nueva solución estará compuesta por microservicios independientes desplegados bajo un entorno serverless en la nube de Amazon Web Services (AWS Lambda). Cada microservicio será desarrollado en Java utilizando el framework Spring Boot, y se comunicará mediante APIs REST. Esta arquitectura se apoya en el patrón hexagonal, facilitando la separación entre el núcleo de negocio y sus interfaces externas.

La plataforma interactuará con servicios internos y externos a través de endpoints HTTP, y gestionará sus propias bases de datos relacionales, asegurando la independencia de cada componente. Además, incorporará mecanismos de seguridad robustos basados en tokens JWT y cifrado de datos mediante funciones hash, garantizando la protección de los datos y la autenticación de los usuarios.

**Funciones del Producto**

El “Sistema modular para Operaciones Empresariales de Perfulandia SPA” proporcionará un conjunto de funcionalidades organizadas en microservicios, cada uno con una responsabilidad bien definida, cubriendo todos los procesos operativos de Perfulandia SPA. A continuación, se describen las funciones clave del producto:

* **Autenticación y Autorización**: Permite la identificación segura de los usuarios mediante tokens JWT.
* **Gestión de Usuarios**: Permite el registro, modificación y gestión de usuarios internos y clientes.
* **Gestión de Inventario**: Administra productos, stock y disponibilidad en tiempo real.
* **Gestión de Pedidos**: Registra, procesa y da seguimiento a los pedidos realizados por los clientes.
* **Pagos y Facturación**: Procesa pagos y genera comprobantes y facturas electrónicas.
* **Logística y Envíos**: Coordina y monitorea el traslado de productos entre bodegas, sucursales y clientes.
* **Gestión de Proveedores**: Administra proveedores y pedidos de reabastecimiento.
* **Reportes y Analítica**: Genera informes operativos y de negocio, facilitando el análisis de datos.
* **Atención al Cliente**: Registra y gestiona consultas, reclamos, devoluciones y soporte.
* **Notificaciones**: Envia alertas y mensajes relevantes a usuarios del sistema.
* **Respaldo y Restauración**: Asegura la continuidad operativa mediante copias de seguridad y mecanismos de recuperación ante fallos.

**Diagrama conceptual de servicios de microservicios:**

**Características de los Usuarios**

En esta parte se describirán las características generales de los usuarios que interactuaron con el “Sistema modular para Operaciones Empresariales de Perfulandia SPA”, centrándose en su nivel educativo, experiencia profesional y competencia técnica. Además, se definen los tipos de usuarios y sus perfiles correspondientes, lo que permitirá comprender mejor sus necesidades y cómo interactúan con el sistema en función de sus roles y habilidades.

#### Tipos de Usuarios y Perfiles

1. **Administrador del Sistema**

**Educación**: Profesional técnico o universitario en informática o carreras afines.

**Experiencia**: Alta experiencia en gestión de sistemas, administración de usuarios, monitoreo de servicios y resolución de incidentes.

**Competencias Técnicas**: Manejo de herramientas de administración de sistemas, monitoreo en la nube (AWS), gestión de bases de datos y control de accesos.

**Rol**: Gestiona usuarios del sistema, supervisa el funcionamiento de los servicios, configura parámetros generales y realiza mantenimientos preventivos.

**Observaciones**: Requiere interfaces avanzadas, paneles de monitoreo y acceso a registros del sistema.

1. **Gerente de Sucursal**

**Educación**: Profesional con formación en administración, gestión comercial o carreras relacionadas.

**Experiencia**: Experiencia en gestión de equipos de ventas, seguimiento de objetivos comerciales y control de inventario a nivel local.

**Competencias Técnicas**: Uso intermedio de sistemas de gestión (ERP o CRM), interpretación de reportes y manejo de herramientas ofimáticas.

**Rol**: Supervisa las operaciones de la sucursal, visualiza reportes de ventas y desempeño, y autoriza solicitudes clave.

**Observaciones**: Requiere una interfaz amigable con acceso rápido a métricas, filtros por periodo y notificaciones.

1. **Empleado de Ventas**

**Educación**: Nivel técnico o enseñanza media completa con capacitación en atención al cliente.

**Experiencia**: Experiencia directa en procesos de venta, atención al público y uso de sistemas de punto de venta (POS).

**Competencias Técnicas**: Uso básico-intermedio de aplicaciones web o POS, ingreso de datos y uso de lectores de código de barras.

**Rol**: Registra ventas, consulta productos, realiza devoluciones o cambios, y atiende requerimientos de clientes.

**Observaciones**: Requiere una interfaz rápida y fluida, optimizada para puntos de atención con alto flujo.

1. **Encargado del Área de Logística**

**Educación**: Formación técnica o profesional en logística, operaciones o gestión de inventario.

**Experiencia**: Experiencia en control de stock, recepción y despacho de productos, y coordinación con bodegas o transporte.

**Competencias Técnicas**: Manejo de sistemas de inventario, escáneres de productos, y generación de guías de despacho.

**Rol**: Administra el inventario, registra movimientos, verifica stock crítico y gestiona pedidos de reposición.

**Observaciones**: Requiere una interfaz clara para el registro de entradas/salidas y alertas de stock mínimo.

1. **Cliente (Usuario vía Web)**

**Educación**: Variable, desde enseñanza básica a superior.

**Experiencia**: No se requiere experiencia técnica; se espera familiaridad con tiendas en línea.

**Competencias Técnicas**: Uso básico de internet, navegación en sitios web, compras en línea y pagos digitales.

**Rol**: Navega por el catálogo de productos, realiza compras, gestiona su cuenta, consulta pedidos y solicita soporte.

**Observaciones**: Requiere una experiencia de usuario intuitiva, compatible con dispositivos móviles, accesible y segura.

**Restricciones**

Durante el desarrollo del sistema, se deberán considerar las siguientes restricciones:

* **Políticas de la Empresa**: Cumplimiento con normativas internas, de seguridad y protección de datos.
* **Limitaciones del Hardware**: El sistema debe operar sobre la infraestructura de red y dispositivos existentes sin requerir inversiones adicionales importantes.
* **Interfaces con otras Aplicaciones**: Integración obligatoria con sistemas ERP, plataformas de pago y sistemas logísticos.
* **Operaciones Paralelas**: El sistema debe soportar concurrencia sin comprometer el rendimiento.
* **Lenguaje de Programación**: Uso exclusivo de Java con Spring Boot.
* **Protocolos de Comunicación**: Comunicación a través de APIs REST utilizando HTTP/HTTPS.
* **Funciones de Control**: Mecanismos para validación de acceso, control de permisos y auditoría.
* **Requisitos de Habilidad del Equipo**: Personal con experiencia en arquitectura de microservicios, Java, bases de datos y AWS.
* **Criticidad del Sistema**: Alta disponibilidad y tolerancia a fallos, dada su naturaleza central para las operaciones del negocio.
* **Seguridad**: Autenticación JWT, cifrado de datos, y protección contra ataques comunes.

**Suposiciones y Dependencias**

Durante el análisis y planificación del sistema se han considerado las siguientes suposiciones y dependencias:

**Suposiciones**:

* La estructura organizacional de Perfulandia SPA se mantendrá estable durante el desarrollo del sistema.
* AWS seguirá siendo la plataforma principal de nube utilizada por la empresa.
* No se anticipan cambios regulatorios importantes en materia de comercio electrónico o protección de datos.
* Los servicios externos utilizados (pagos, logística, notificaciones) mantendrán sus interfaces y disponibilidad.

**Dependencias**:

* Servicios de terceros como pasarelas de pago y herramientas de mensajería seguirán operativos.
* Disponibilidad de profesionales capacitados en tecnologías clave (Java, Spring Boot, AWS).
* Estabilidad de la red e infraestructura para garantizar operaciones en tiempo real.
* Compatibilidad con el sistema ERP y otras plataformas existentes.

**Requisitos Futuros**

Las futuras versiones del “Sistema modular para Operaciones Empresariales de Perfulandia SPA” podrían incluir mejoras y características adicionales que no están contempladas en el alcance inicial del sistema. Estas mejoras se identificarán y evaluarán a medida que la empresa crezca y sus necesidades evolucionen. Algunas de las áreas clave para futuros desarrollos incluyen:

1. **Integración de Inteligencia Artificial y Análisis Predictivo:** A medida que la empresa crezca, la incorporación de inteligencia artificial podría optimizar la predicción de demanda, la personalización de la experiencia del cliente y la optimización de inventarios, brindando mejores herramientas para la toma de decisiones estratégicas.
2. **Automatización Avanzada de Procesos:** Mejoras en la automatización de procesos como la reposición de stock, la asignación automática de tareas logísticas y la gestión de ventas, lo que permitirá mejorar la eficiencia operativa y reducir la intervención manual.
3. **Expansión de Capacidades de Reportes:** Se planea desarrollar herramientas avanzadas para la creación de informes personalizados y dashboards interactivos, proporcionando una mayor flexibilidad y visualización de datos en tiempo real para los usuarios y gerentes.
4. **Ampliación a Mercados Internacionales y Multimoneda:** La posibilidad de expandir el sistema a mercados internacionales, con soporte para múltiples idiomas, monedas y normativas fiscales locales, se evaluará a medida que Perfulandia crezca en nuevas regiones.
5. **Mejoras en la Experiencia del Cliente a través de Plataformas Móviles:** Se contempla la creación de una aplicación móvil o una interfaz web responsiva para que los clientes puedan acceder y gestionar sus pedidos, recibir notificaciones y acceder a información en tiempo real desde cualquier dispositivo móvil.
6. **Integración con Nuevos Canales de Ventas:** La integración con plataformas de comercio electrónico de terceros, redes sociales y marketplaces se evaluará para expandir los canales de ventas de Perfumalandia Bloom, lo que facilitará el acceso de la empresa a un público más amplio.
7. **Cumplimiento con Nuevas Normativas y Regulaciones:** A medida que evolucionan las leyes, se desarrollarán funcionalidades que aseguren el cumplimiento con normativas de privacidad, protección de datos (como GDPR) y otras regulaciones locales o internacionales que puedan surgir.

**3. Requisitos Específicos**

Esta sección contiene los requisitos a un nivel de detalle suficiente como para permitir a los diseñadores construir un sistema que satisfaga estos requisitos, y que permita al equipo de pruebas planificar y ejecutar las pruebas necesarias. Cada requisito describe comportamientos externos del sistema, perceptibles por usuarios, operadores o sistemas externos.

Se siguen principios de corrección, claridad, completitud, consistencia, clasificación, verificabilidad, modificabilidad y trazabilidad.

**3.1 Requisitos comunes de las interfaces**

**3.1.1 Interfaces de usuario**

El sistema contará con una interfaz web responsive, accesible desde navegadores modernos, diseñada para adaptarse tanto a equipos de escritorio como a dispositivos móviles.

* Los colores corporativos de Perfulandia SPA serán utilizados (verde esmeralda y blanco).
* Los formularios contarán con validaciones visuales (campos obligatorios, formatos correctos).
* Se utilizarán componentes como tablas dinámicas, filtros por fecha y ventanas modales para CRUDs.

**3.1.2 Interfaces de hardware**

El sistema no requiere interacción directa con hardware especializado. Solo se espera uso de teclado, mouse y pantalla estándar.

**3.1.3 Interfaces de software**

El sistema deberá integrarse mediante APIs RESTful con:

* Servicio de inventario
* Servicio de pagos
* Servicio de facturación
* Servicio logístico
* Proveedores externos (envíos, cupones)

Cada integración seguirá el formato JSON para el intercambio de datos. Autenticación JWT entre servicios.

**3.1.4 Interfaces de comunicación**

Todos los microservicios se comunicarán a través de HTTPS usando REST y JSON. Se aplicará control de acceso con OAuth2 y JWT para autenticación y autorización entre servicios.

**3.2 Requisitos funcionales**

**RF01 - Acceder al sistema/Inicio de sesión**

El sistema permitirá a los usuarios iniciar sesión ingresando sus credenciales en los campos “Usuario” (correo o RUT) y “Contraseña”.  
Al presionar el botón “Ingresar”, se validan las credenciales. Si son correctas, el usuario accederá a la pantalla correspondiente a su perfil (Administrador, Gerente, Empleado, cliente, etc.).  
 En caso de credenciales incorrectas, se mostrará un mensaje de error.  
 El sistema bloqueará la cuenta tras 5 intentos fallidos y registrará fecha/hora del último acceso.

**RF02 - Recuperación de contraseña**

El sistema permitirá a los usuarios recuperar su contraseña ingresando su correo registrado. Se enviará un enlace temporal para restablecer la clave.  
 El enlace será válido por 30 minutos y solo podrá usarse una vez.

**RF03 - Registro de clientes**

El sistema permitirá a los usuarios registrarse como clientes ingresando datos personales (nombre, correo, RUT, contraseña, dirección).  
 El sistema validará formato de correo y contraseña. Se enviará correo de activación para completar el registro.

**RF04 - Gestión de usuarios**

El Administrador podrá:

**RF04.1** - Crear cuentas de usuario (nombre, correo, perfil, contraseña temporal).

**RF04.2** - Editar datos del usuario.

**RF04.3** - Desactivar o reactivar cuentas.

**RF04.4** - Eliminar cuentas de forma permanente.

**RF05 - Configuración de permisos**

El sistema permitirá al Administrador asignar roles y permisos. Cada rol tendrá acceso definido a funcionalidades específicas.

**RF06 - Gestión de ventas**

El sistema permitirá registrar ventas seleccionando productos desde inventario, asignando cantidades y método de pago.  
 Al finalizar la transacción, se generará una factura y se actualizará automáticamente el stock del inventario.

**RF07 - Gestión de inventario**

El sistema permitirá:

**RF07.1** - Agregar productos (nombre, SKU, stock, precio).

**RF07.2** - Editar productos existentes.

**RF07.3** - Eliminar productos.

**RF07.4** - Generar alertas cuando el stock sea menor a un umbral definido.

**RF08 - Gestión de pedidos**

El Gerente de Sucursal podrá ver el estado de los pedidos (Pendiente, En camino, Entregado) y generar reportes de pedidos por rango de fechas.

**RF09 - Gestión logística**

El Encargado de Logística podrá:

**RF09.1** - Registrar rutas de envío.

**RF09.2** - Asignar pedidos a rutas.

**RF09.3** - Cambiar el estado del envío.

**RF09.4** - Integrar proveedores de transporte.

**RF10 - Emitir factura electrónica.**

Cada vez que se realice una venta, el sistema generará una factura con:

**RF10.1** - Datos del cliente

**RF10.2** - Productos vendidos

**RF10.3** - Total con impuestos

**RF10.4** - Método de pago Las facturas se enviarán por correo y quedarán disponibles para consulta posterior.

**RF11 - Gestión de reportes**

El sistema permitirá a usuarios autorizados generar reportes:

**RF11.1** - De ventas por rango de fechas

**RF11.2** - De productos más vendidos

**RF11.3** - De facturación por sucursal

**RF11.4** - De pedidos entregados

**RF12 - Atención al cliente**

El cliente podrá:

**RF12.1** - Enviar solicitudes de soporte mediante formulario.

**RF12.2** - Ver estado de sus solicitudes. El operador podrá responder consultas, escalar a soporte técnico y cerrar casos.

**RF13 - Búsqueda y compra de productos**

El cliente podrá:

**RF13.1** - Buscar productos por nombre, categoría o SKU.

**RF13.2** - Agregar productos al carro de compras.

**RF13.3** - Aplicar cupones de descuento.

**RF13.4** - Realizar el pedido finalizando con los datos de envío y forma de pago.

**RF14 - Consulta de historial de compras**

El cliente podrá consultar su historial de compras, con opción a ver detalles de cada pedido, reimprimir facturas y repetir pedidos.

**RF15 - Valoración de productos**

El cliente podrá calificar productos que haya comprado y dejar comentarios públicos sobre su experiencia.

**3.3 Requisitos no funcionales**

**3.3.1 Requisitos de rendimiento**

**RNF01** - **Escalabilidad:** El sistema debe poder manejar hasta 500 usuarios concurrentes sin degradación del rendimiento.

**RNF02** - **Rendimiento en transacciones:** El 95% de las transacciones deberán completarse en menos de 1 segundo, bajo carga media.

**RNF03** - **Rendimiento bajo carga:** El sistema deberá soportar un mínimo de 1000 peticiones por minuto, distribuidas entre los distintos microservicios desplegados.

**3.3.2 Seguridad**

**RNF04** - **Cifrado de datos:** Toda comunicación entre cliente y servidor debe estar cifrada mediante HTTPS utilizando TLS.

**RNF05** - **Acceso con autenticación:** El acceso a las funcionalidades deberá estar protegido mediante autenticación basada en tokens JWT.

**RNF06** - **Control de acceso:** Las funcionalidades deben estar segregadas por roles (Administrador, Vendedor, Cliente) mediante políticas RBAC.

**RNF07** - **Logs de actividad:** El sistema deberá registrar toda actividad crítica (inicio de sesión, cambios en el sistema, errores) en logs seguros y auditables.

**3.3.3 Fiabilidad**

**RNF08** - **Tolerancia a fallos:** El sistema deberá recuperarse automáticamente de fallos de servicios individuales mediante mecanismos de retry y circuit breakers.

**RNF09** - **Incidencias permitidas:** No se deberán registrar más de 2 fallos críticos mensuales en ambiente de producción.

**3.3.4 Disponibilidad**

**RNF10** - **Disponibilidad general:** El sistema debe estar disponible al menos el 99.5% del tiempo mensual.

**3.3.5 Mantenibilidad**

**RNF11** - **Mantenimiento evolutivo:** Las nuevas funcionalidades deben poder añadirse sin afectar al funcionamiento de los servicios existentes, mediante arquitectura de microservicios desacoplados.

**RNF12** - **Documentación técnica:** Todo el código deberá estar acompañado de documentación técnica y comentarios legibles.

**3.3.6 Portabilidad**

**RNF13** - **Independencia de plataforma:** El sistema deberá ser portable a distintos entornos cloud compatibles.

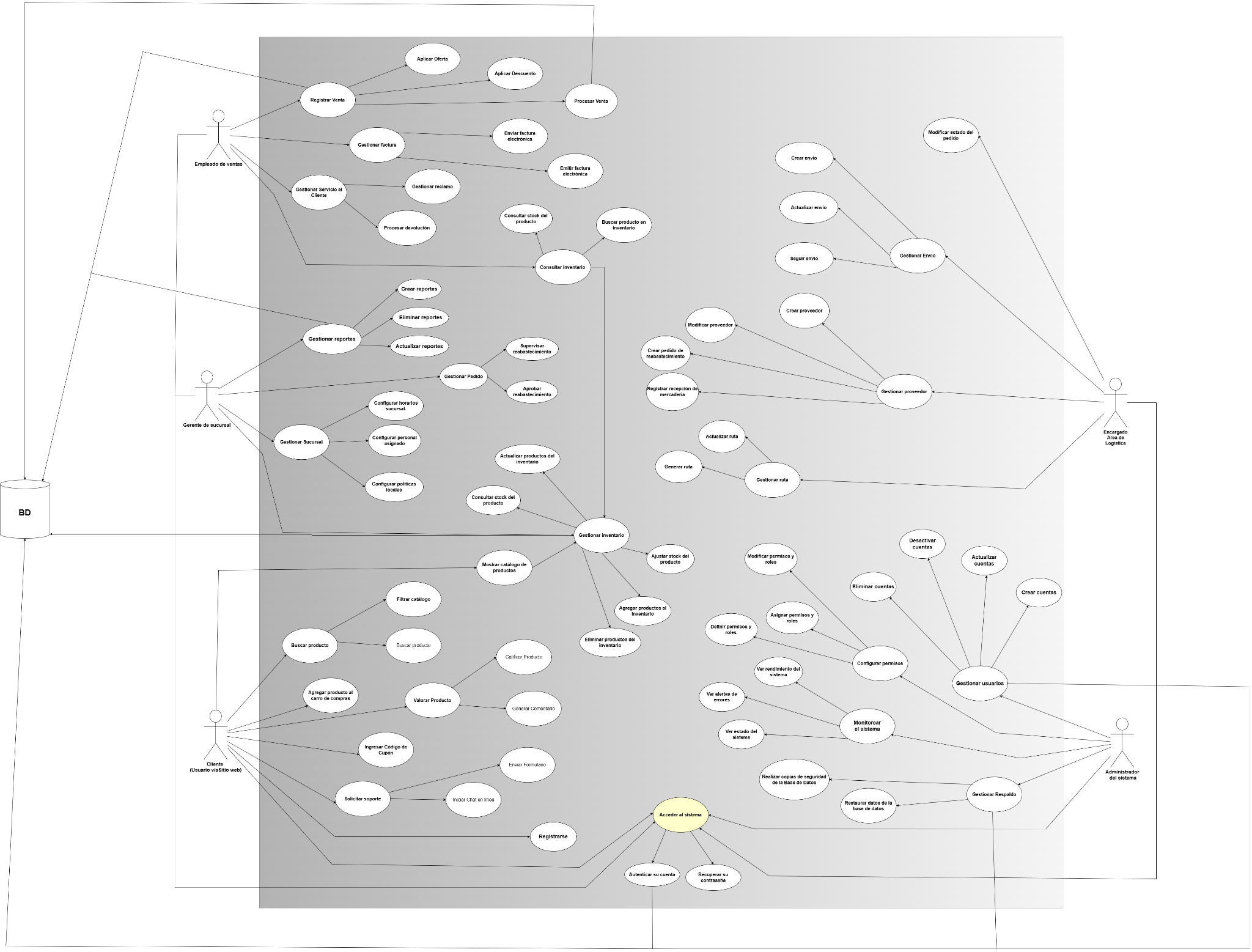
**RNF14** - **Compatibilidad de desarrollo:** Todo el código debe desarrollarse en Java 17 y Spring Boot, utilizando herramientas y librerías que aseguren la compatibilidad con entornos Linux.

**RNF15** - **Despliegue multiplataforma:** El sistema debe permitir su despliegue tanto en AWS Lambda como en máquinas virtuales estándar.

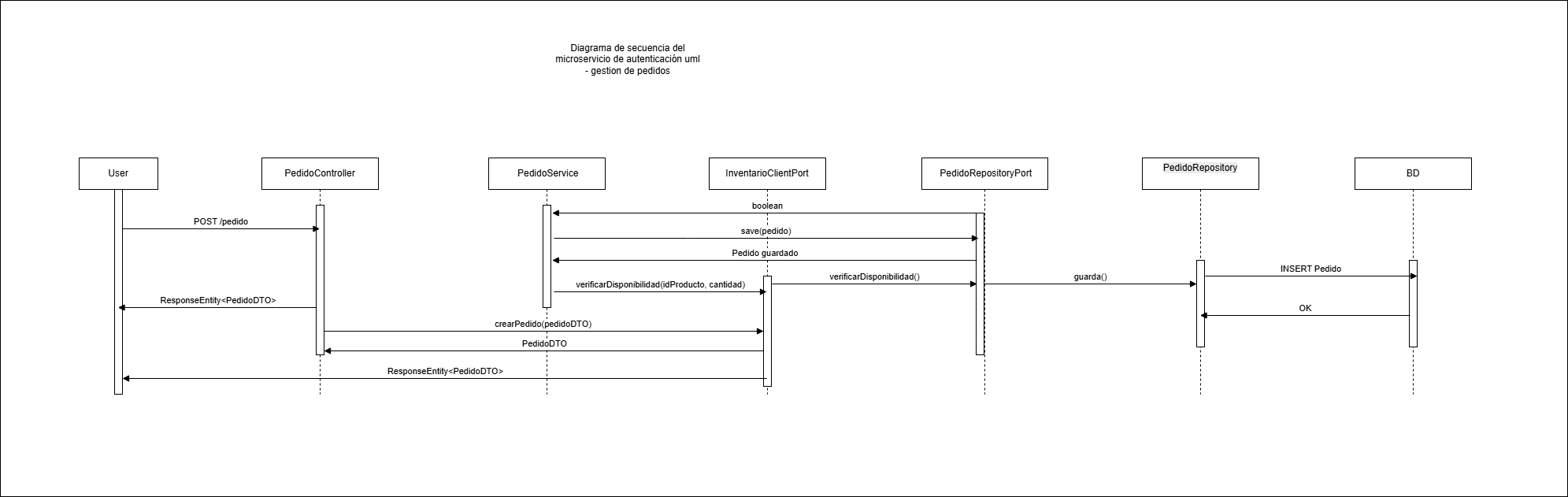
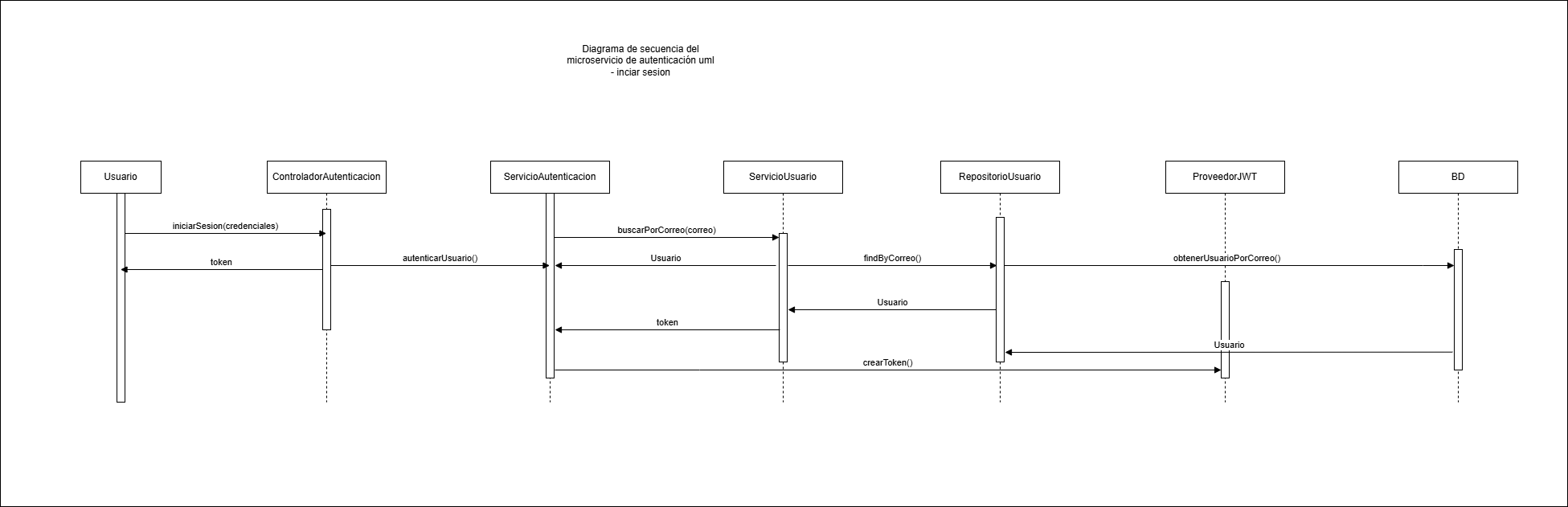
**3.4 Otros Requisitos**

**RNF16 - Mensajes de error usables:** Todos los mensajes de error mostrados al usuario deben ser claros, concisos y escritos en un lenguaje comprensible, sin mostrar detalles técnicos del sistema. Los mensajes deben ofrecer orientación sobre cómo proceder o corregir la acción realizada.

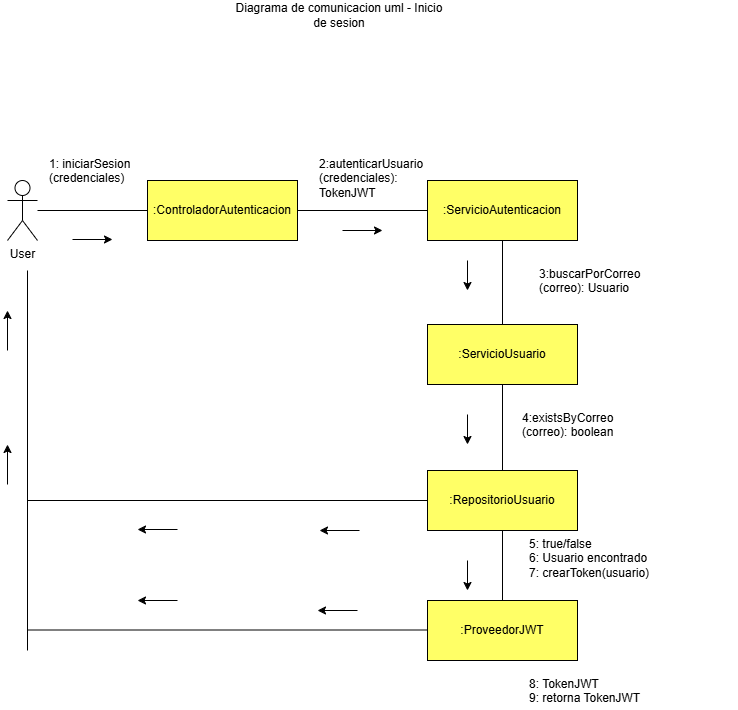
**4 Propuesta de Diseño y Arquitectura del Sistema**

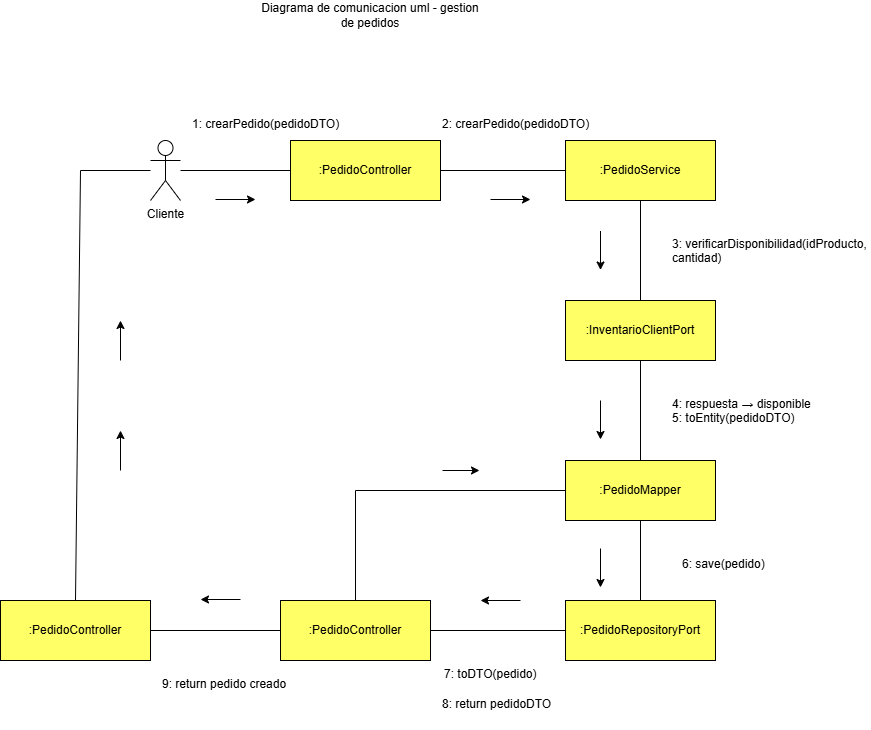
**4.1 Diagrama de caso de uso**

**4.2 Diagrama de secuencia**

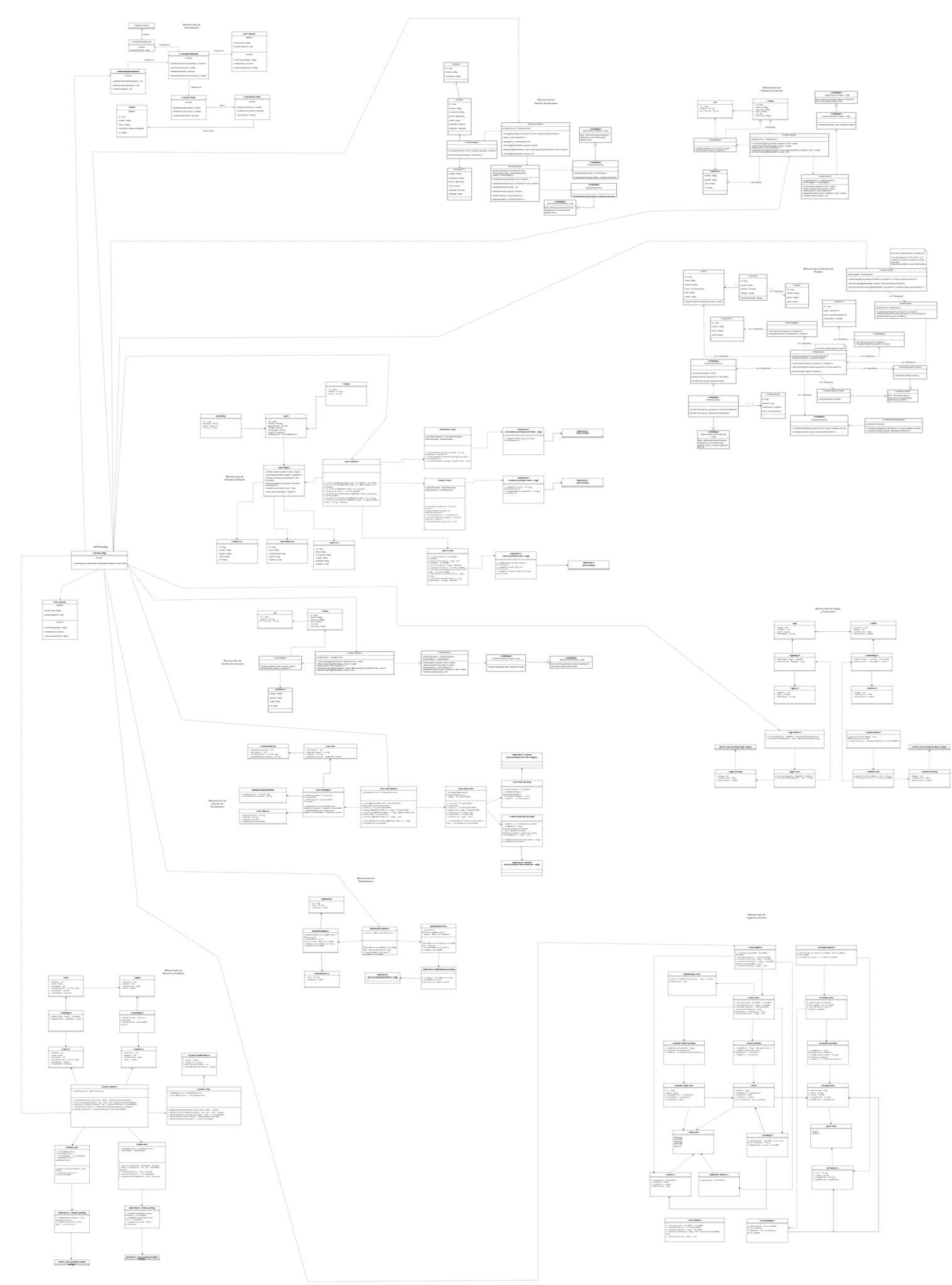


**4.3 Diagrama de comunicación**

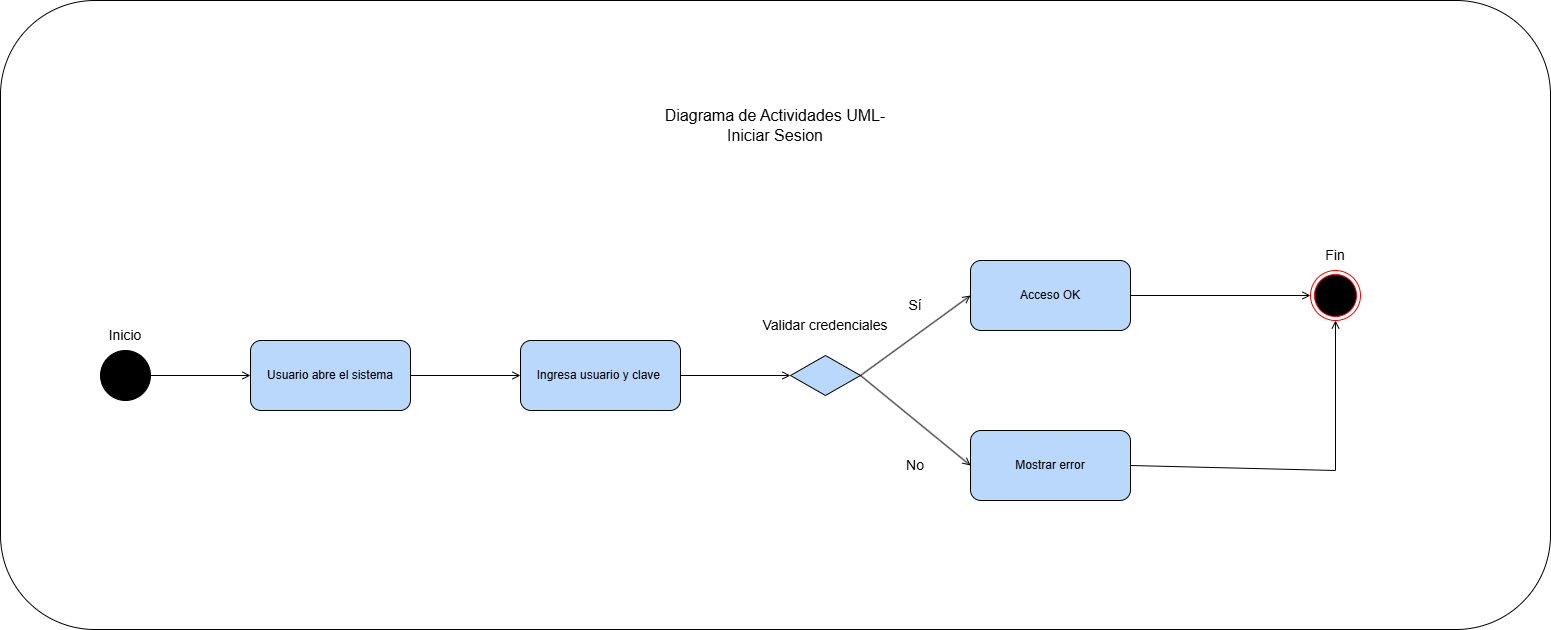
****

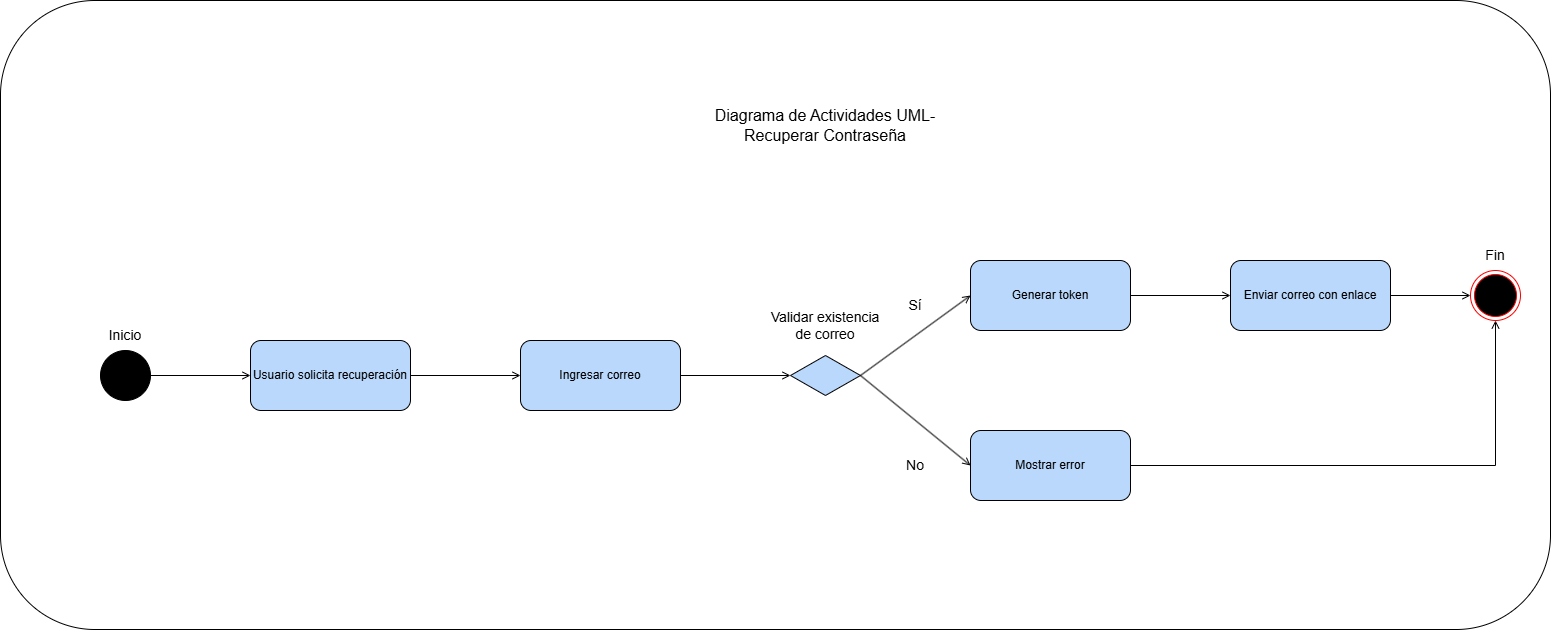
****

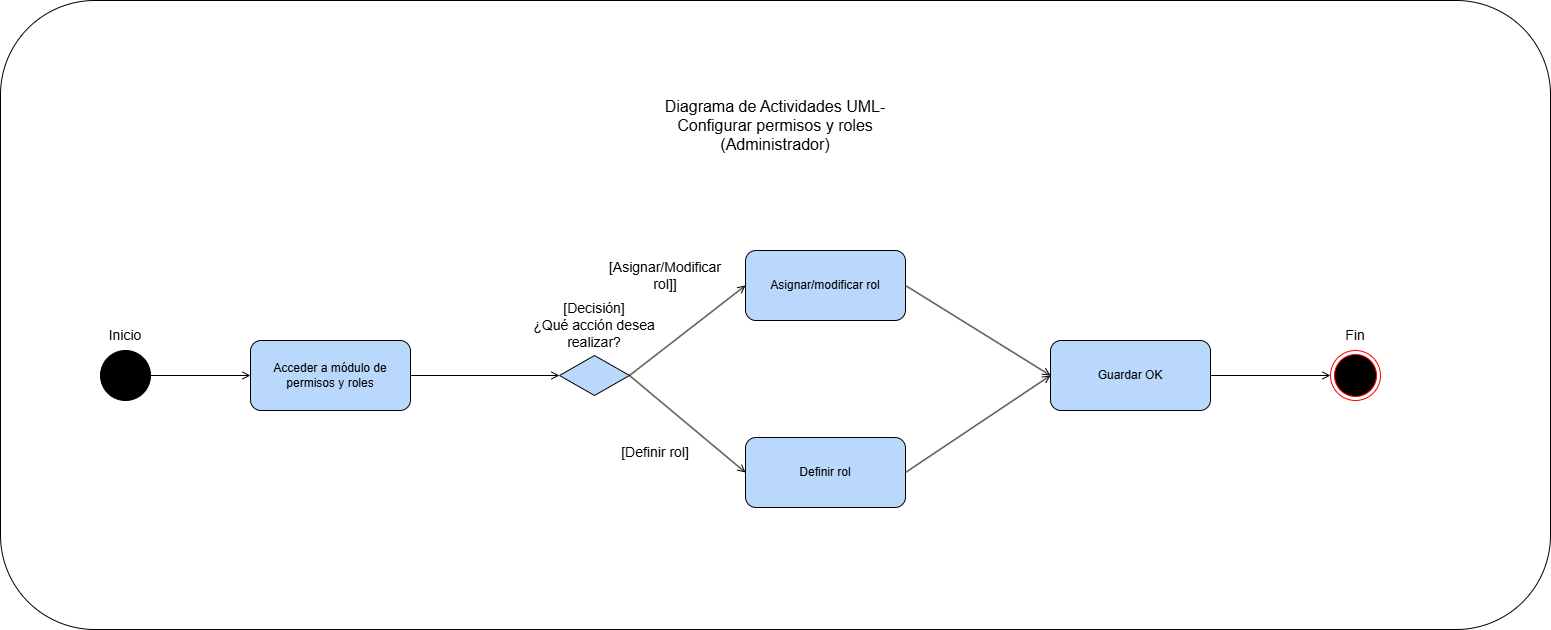
**4.4 Diagrama de clase**

****

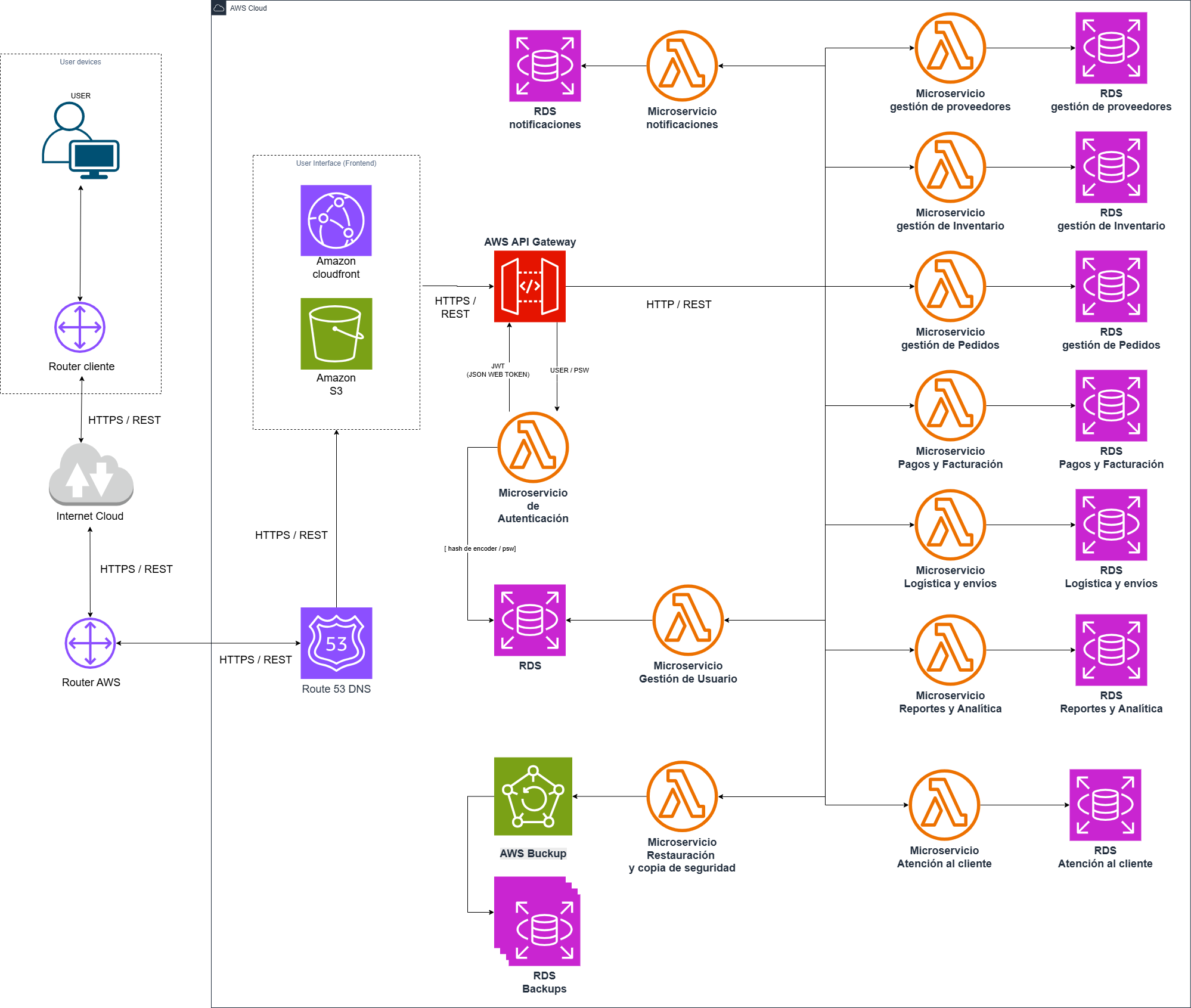
**4.5 Diagrama de actividad**

****

****

****

**4.6 Diagrama de despliegue**

****

**4.7 Propuestas de arquitectura de diseño**

La arquitectura de diseño es clave en el ciclo de desarrollo del software, ya que define cómo se estructurará internamente el sistema para cumplir con los requisitos técnicos, funcionales y de calidad. En esta sección se propondrán Arquitecturas que permitan abordar el proyecto con un enfoque modular, escalable y mantenible, aplicando buenas prácticas de diseño y principios de ingeniería de software. Estas propuestas consideran las necesidades específicas del sistema, tales como la separación de responsabilidades, la interoperabilidad entre componentes y la facilidad de pruebas y despliegues.

* **Arquitectura en Capas (Leyered Architecture)**

Esta arquitectura organiza la aplicación en capas separadas como Presentación, Lógica de negocio, Acceso a datos y Persistencia.

La aplicación de esta arquitectura nos facilita la separación de responsabilidades, mejorar la mantenibilidad del código y permitirá la reutilización de componentes. Es ideal para aplicaciones con lógica empresarial claramente estructurada, como servicios CRUD en Spring Boot.

* **Arquitectura Hexagonal (Ports and Adapters)**

Esta arquitectura separa el núcleo de la aplicación (dominio) de sus interfaces externas (bases de datos, APIs, usuarios).  
Nos permitirá construir software independiente de la infraestructura, facilitando pruebas y cambios en tecnología (sin afectar el núcleo). Útil cuando se quiere mantener la lógica de negocio desacoplada del framework o tecnologías externas (facilitando la migración).

* **Arquitectura Orientada a Servicios (SOA)**

Modelo basado en la exposición de funcionalidades como servicios independientes que se comunican entre sí a través de interfaces bien definidas.

Promueve la interoperabilidad entre sistemas heterogéneos, además cuando se integran múltiples sistemas o tecnologías, aunque con mayor complejidad que los microservicios modernos.

* **Arquitectura Basada en Microservicios**

Divide la aplicación en servicios pequeños e independientes que se comunican por API (REST, eventos).

Esta arquitectura es ideal para escalabilidad, despliegue independiente y desarrollo distribuido por equipos. Permite que cada microservicio evolucione por separado, lo cual se ajusta bien a proyectos grandes o modulares como una plataforma de ventas online cómo lo solicitado por Perfulandia.

**5. Propuesta de Planificación**

**5.1 Descripción general acerca de la Planificación**

La planificación del proyecto se basa en el enfoque tradicional de gestión **waterfall**, puesto que se tiene un entendimiento claro de los requerimientos y no se solicita innovar en el sistema, por lo que en cada fase del ciclo de vida del desarrollo de software (Requisitos, Diseño, Implementación, Verificación y pruebas, Despliegue y entrega, Mantenimiento y soporte ) es abordada secuencialmente. La migración del sistema monolítico actual hacia una arquitectura basada en microservicios se llevará a cabo durante un período estimado de **5 meses (20 semanas)**, considerando **actividades paralelas** para maximizar la eficiencia del equipo y reducir la duración lineal del proyecto.

Se propone una ejecución organizada mediante una **Estructura de Desglose del Trabajo (EDT)**, definiendo **fases, tareas específicas e hitos críticos**. Esta estructura considera además la implementación de **buenas prácticas en ingeniería de software**, tales como el uso de pruebas automatizadas, integración continua (CI/CD), principios SOLID en el diseño orientado a objetos aplicados sobre Spring Boot y herramientas de monitoreo y despliegue en la nube.

Se estiman **20 semanas de trabajo**, con un equipo técnico compuesto por **5 profesionales clave** que colaborarán de forma continua durante el desarrollo. Se fomentará una cultura DevOps que permita reducir los tiempos de integración y entrega, así como una estructura de control de versiones (Git) para garantizar trazabilidad y confiabilidad en cada avance del proyecto.

La planificación se alinea con los siguientes objetivos clave:

* **Reducción de riesgos** mediante pruebas tempranas e integración continua.
* **Aseguramiento de calidad** por medio de revisiones técnicas, validaciones funcionales y pruebas de rendimiento.
* **Escalabilidad técnica** gracias a una arquitectura desacoplada.
* **Disponibilidad y resiliencia** a través del despliegue en la nube y replicación de bases de datos.

**5.1.2 Definición del Equipo de Trabajo**

El equipo de trabajo estará conformado por los siguientes roles y responsabilidades principales:

| **Rol Profesional** | **Función Principal** |
| --- | --- |
| Líder de Proyecto | Gestión integral del proyecto, seguimiento de plazos y coordinación general. |
| Arquitecto de Software | Diseño de la arquitectura orientada a objetos, definición de microservicios y patrones de integración. |
| Desarrollador Backend | Implementación de microservicios, pruebas unitarias, configuración de APIs REST. |
| Ingeniero DevOps | Configuración de entornos en AWS, despliegue automatizado (CI/CD), monitoreo. |
| Analista QA | Definición de pruebas, ejecución de pruebas funcionales, carga y regresión. |

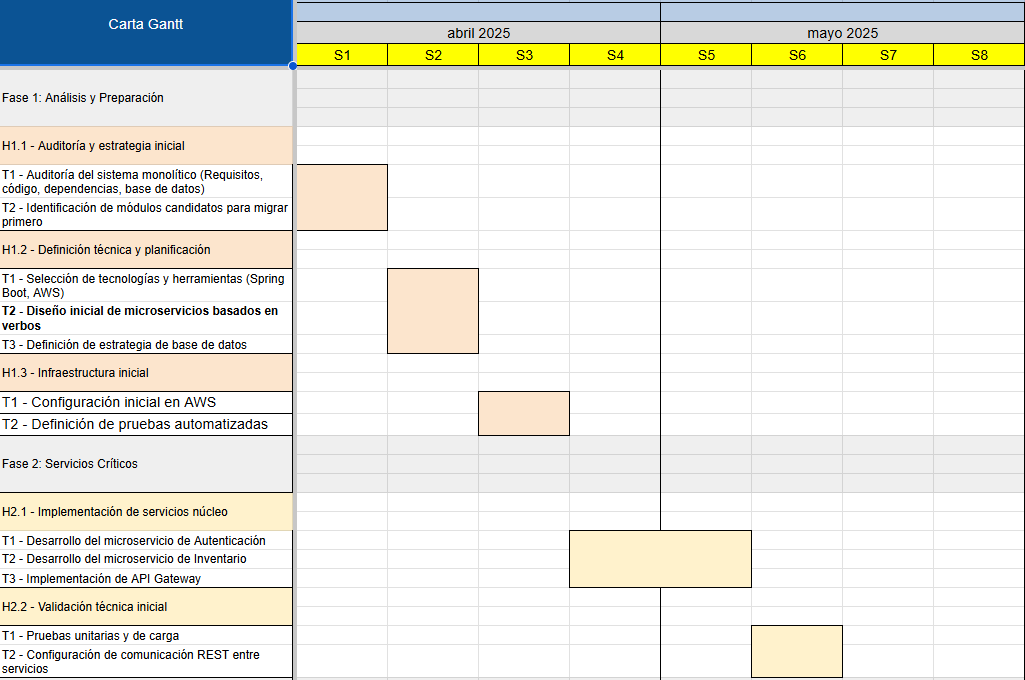
**5.1.3 Definición de Actividades principales del Proyecto**

Las actividades han sido organizadas bajo los estándares del PMI (Project Management Institute) para la gestión de la planificación, y se fundamentan en prácticas de ingeniería de software como análisis orientado a objetos, diseño basado en responsabilidades y principios de bajo acoplamiento entre componentes. Las fases principales son:

* Análisis y preparación del entorno
* Desarrollo inicial de microservicios críticos
* Migración de funcionalidades del sistema monolítico
* Optimización y servicios complementarios
* Desactivación del monolito y despliegue final

Cada fase incluye tareas específicas alineadas con los objetivos técnicos y organizativos del proyecto.

**5.1.4 Carta Gantt**

Link a Carta Gantt:  
[Organigrama de tareas\_Carta Gantt](https://docs.google.com/spreadsheets/d/1y-Qz9bB4pekQtYRnoqWo3gfY3IznRWcfEexHkSjtMAY/edit?usp=sharing)

**5.1.5 Resumen Costos del Desarrollo del Proyecto**

En esta sección se presenta una estimación de los costos que tendría el desarrollo del proyecto, considerando las distintas fases por las que se pasara. Para realizar esto, calculamos las horas-hombre (hh) que creemos necesarias para cada etapa, basándonos en una jornada laboral de 40 horas semanales.

También tomamos en cuenta un valor promedio por hora trabajada, pensando en lo que costaría un profesional con un perfil cercano al que se necesitará para las respectivas tareas. En la mayoría de las etapas usamos un valor de $30.000.- CLP por hora, excepto en la fase de mantenimiento, donde se reduce un poco el costo por la naturaleza del trabajo.

El proyecto está dividido en seis fases principales, que van desde el análisis hasta el apagado del sistema monolítico y su posterior mantenimiento. A continuación, se presenta el desglose estimado de costos por fase:

