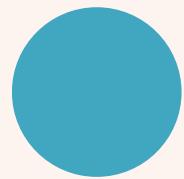


PERFULANDIA SPA

TRANSFORMACIÓN DIGITAL
MEDIANTE MICROSERVICIOS

Grupo 1: Gonzalo Navarrete -
Carla Prado - Fernando Zárate

INTRODUCCIÓN

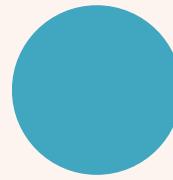


En el caso de estudio n°1, **Perfulandia SPA** es una empresa que ha crecido **rápida y exponencialmente**, lo que deja obsoleto su sistema de administración, hasta el punto en que es **inviable** escalar o actualizarlo.

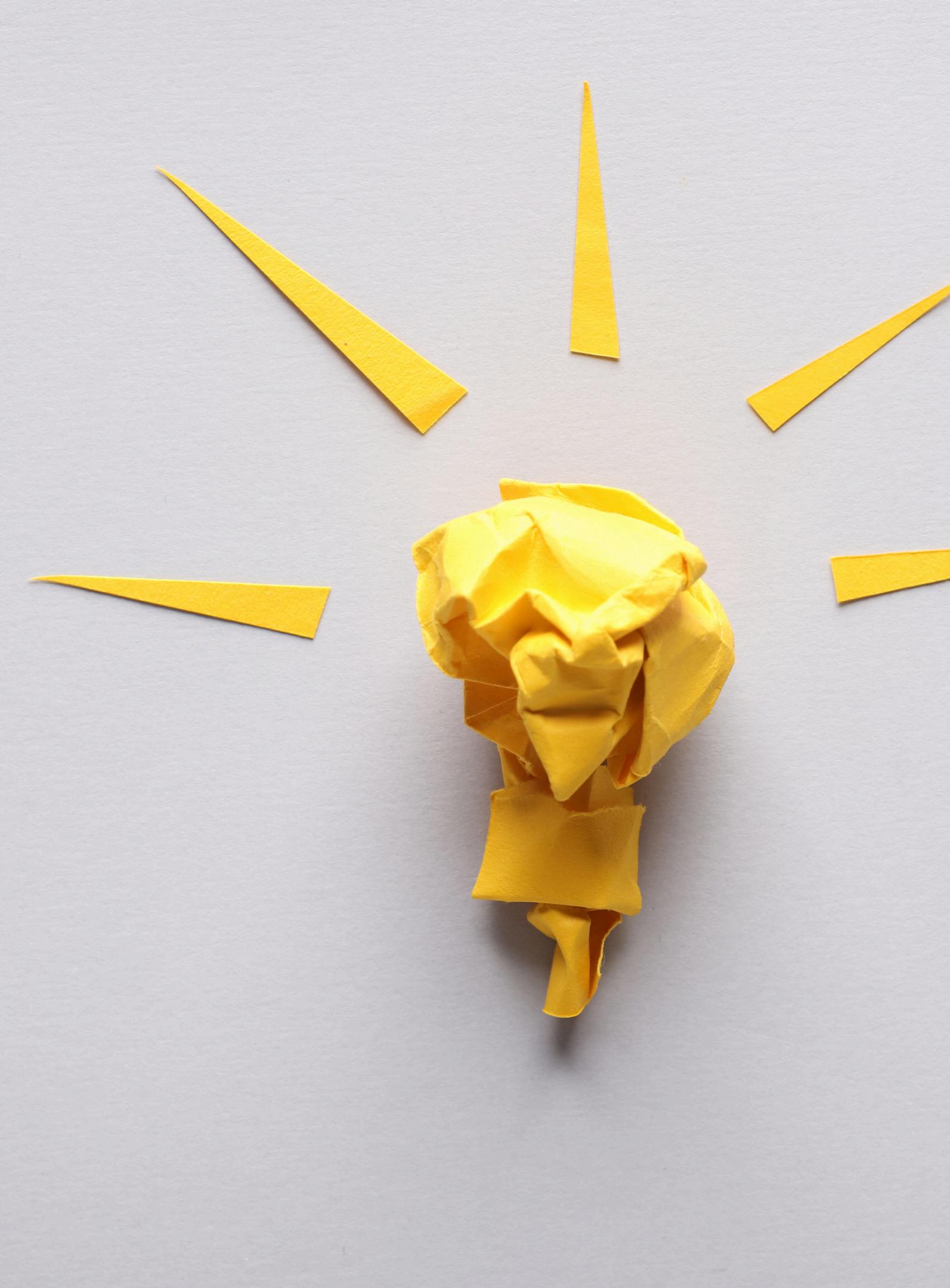
Se requiere una **transformación del sistema existente** mediante una **arquitectura más flexible, basada en microservicios**.

Tras un **exhaustivo análisis**, se entrega a continuación una **propuesta de diseño e implementación de un sistema nuevo** como solución al problema.

1. ANÁLISIS DE REQUERIMIENTOS



Los **requisitos funcionales** son **claros** en el caso de estudio y tienen un User Persona asociado, mientras que los **no funcionales** son **estándares** en la **industria del comercio**. Nacen a partir de **épicas de usuario** desarrolladas por el Equipo.



FUNCIONALIDADES DESPRENDIDAS DE ÉPICAS DE USUARIO

User Personas:

- **Administrador del Sistema**
 - Supervisar estado de todo el **sistema** en tiempo real
- **Gerente de Sucursal**
 - Registro de inventario p/análisis de datos
- **Empleado de Ventas**
 - Procesar ventas offline, integrar online
- **Área Logística**
 - Automatizar rutas de entrega + Seguimiento
- **Clients vía web**
 - Experiencia usuario amigable + recomendaciones

REQUISITOS FUNCIONALES

1. Gestión de personas - CRM (Registro y administración de empleados)
2. Gestión de inventario (Supervisión de pedidos y stock)
3. Registro y administración de ventas
4. Gestión de envíos (Planificación rutas, control proveedores)
5. UI Sitio Web (Front-end Registro clientes y experiencia de usuario)

REQUISITOS NO FUNCIONALES

- Escalabilidad: soportar el crecimiento exponencial
- Disponibilidad: evitar interrupciones en las operaciones.
- Rendimiento: Respuesta rápida del sistema
- Seguridad: Protección de datos sensibles
- Mantenibilidad: actualizaciones no afectan el todo del **sistema**.

2. ANÁLISIS SISTEMA ACTUAL

El sistema actual de Perfolandia SPA es una aplicación monolítica que integra todas las funcionalidades (ventas, inventario, logística, etc.) **en un solo código base**. Además, parece ser que funciona en un servidor centralizado que **soporta las tres sucursales y la plataforma web**.



DESVENTAJAS ARQUITECTURA MONOLÍTICA

Causas de fallos y sobrecargas

- El cambio más mínimo afecta a totalidad del sistema
- Baja tolerancia a Fallos - Difícil identificación.
- Cuellos de botella en servidor único
- Lentitud procesamiento
- Dificulta pruebas y actualizaciones

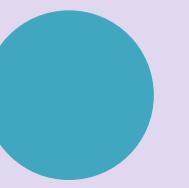
Consecuencias a nivel de negocio

- Pérdida de clientes
- Insatisfacción de los empleados.
- Potencial pérdida de datos

3. DISEÑO NUEVA ARQUITECTURA

La nueva propuesta es una **arquitectura basada en microservicios**, es decir, cada funcionalidad trabajará por separado con su **propia base de datos**, pero **de forma interconectada**, lo que busca resolver la **gran mayoría de los problemas descritos**.

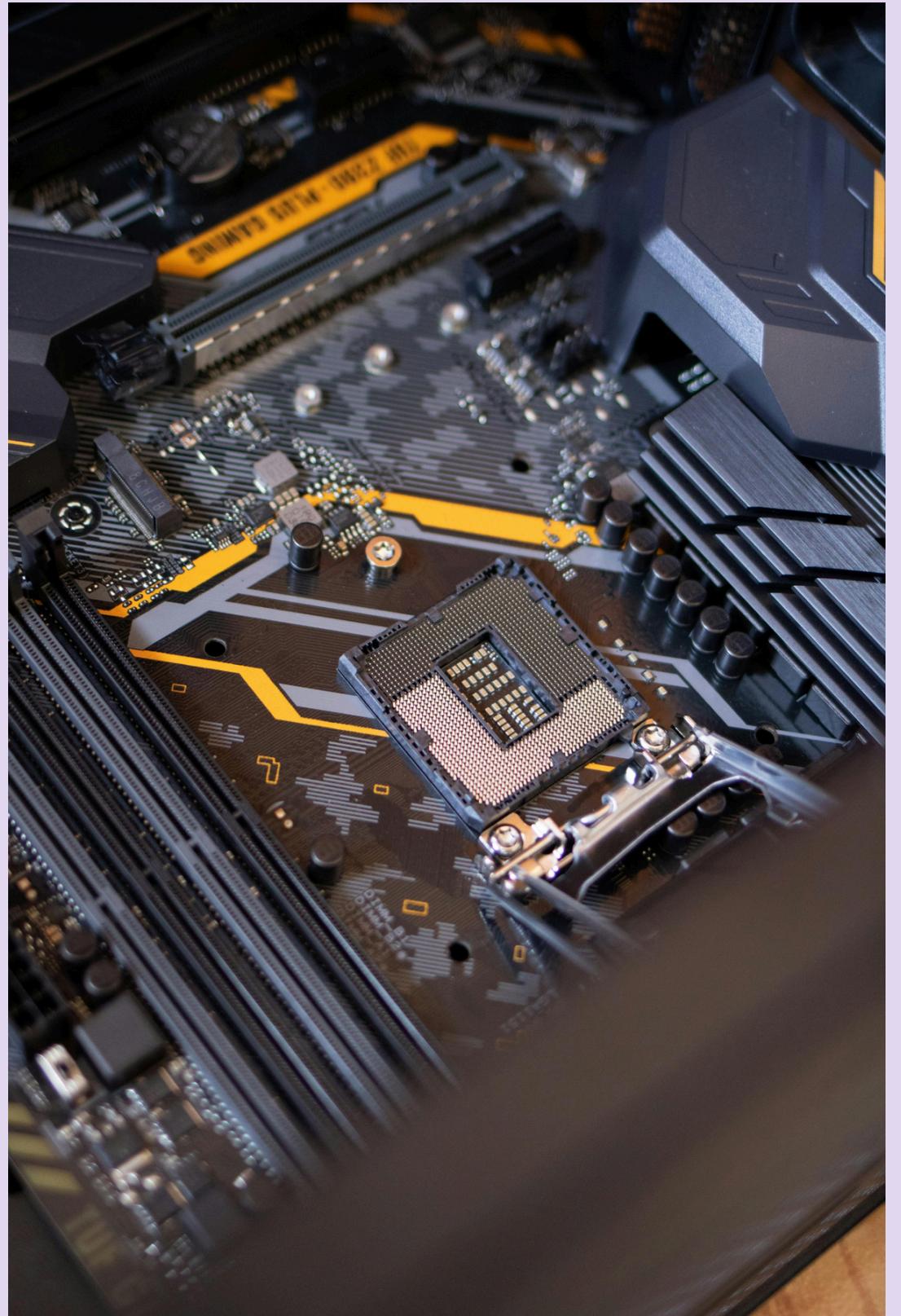
FACTORES EN LA DECISIÓN



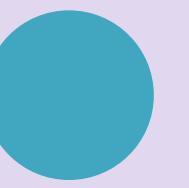
Se tomaron en cuenta una variedad de factores para justificar este enfoque. Entre ellos:

Ventajas de arquitectura basada en microservicios
(Solucionan la mayor parte de las problemáticas descritas anteriormente)

- Ofrece modularidad, evita que falle todo el sistema sólo por un error en un módulo.
- Facilita desarrollo, actualización y mantenimiento
- Procesamiento más rápido ante alta demanda
- Migración Sin Interrupciones en el servicio

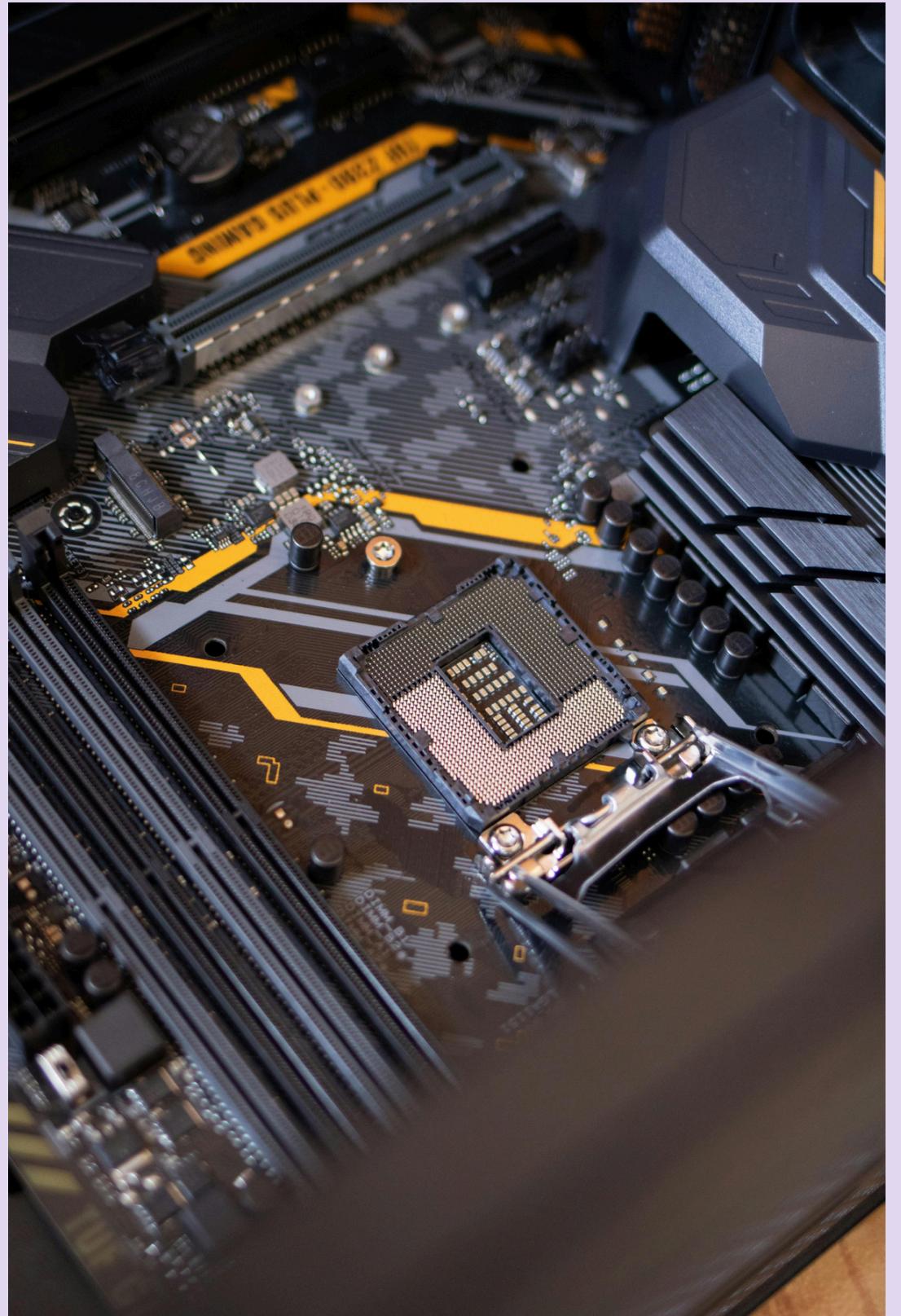


FACTORES EN LA DECISIÓN



Comparación con otras arquitecturas que priorizan escalabilidad

- SOA (Service Oriented)- Punto de fallo centralizado
- Serverless - Puede ser muy compleja y costosa
- Híbrido monolítico/microservicios - Sólo viable en transición



FACTORES EN LA DECISIÓN

CONSIDERACIONES ÉTICAS

- Privacidad de Datos: cumplimiento de normativas de cifrado estándar de la industria.
- Responsabilidad en Despliegue: Monitoreo proactivo para evitar caídas
- Impacto en el Empleo: Capacitación al personal + documentación
- Accesibilidad y Uso Ético de los Datos Analíticos: usar análisis solo para mejorar el servicio

Las herramientas a utilizar serán compatibles con este enfoque ético

Ej: Kubernetes Reduce el impacto ambiental al optimizar recursos, Spring Boot y MySQL facilitan auditorías de seguridad, HTML/CSS con diseño accesible aseguran inclusión.



MICROSERVICIOS PROPUESTOS

1. Core (Autenticación y Autorización - Config. del sistema)

- Tipo: Microservicio de seguridad y control de acceso
- Ejemplo: Netflix "Zuul" (un API Gateway)

2. Comercio (Inventario y Productos)

- Tipo: Microservicio de gestión de productos.
- Ejemplo: Amazon Multi-Channel Fulfillment (MCF)

3. Pedidos (Ventas y Facturación)

- Tipo: Microservicio transaccional.
- Ejemplo: Uber "Trip Execution Engine"

4. Logistica (Gestión de Envíos)

- Tipo: Microservicio de operación y despacho.
- Ejemplo: FedEx "Ship Manager"

5. Cliente (Interfaz y Experiencia de Usuario)

- Tipo: Microservicio de interacción con usuarios.
- Ejemplo: Spotify "Client services"

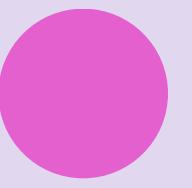
6. Notificaciones (Alertas del sistema)

- Tipo: Microservicio de comunicación.
- Ejemplo: Twitter "Event Processing System"

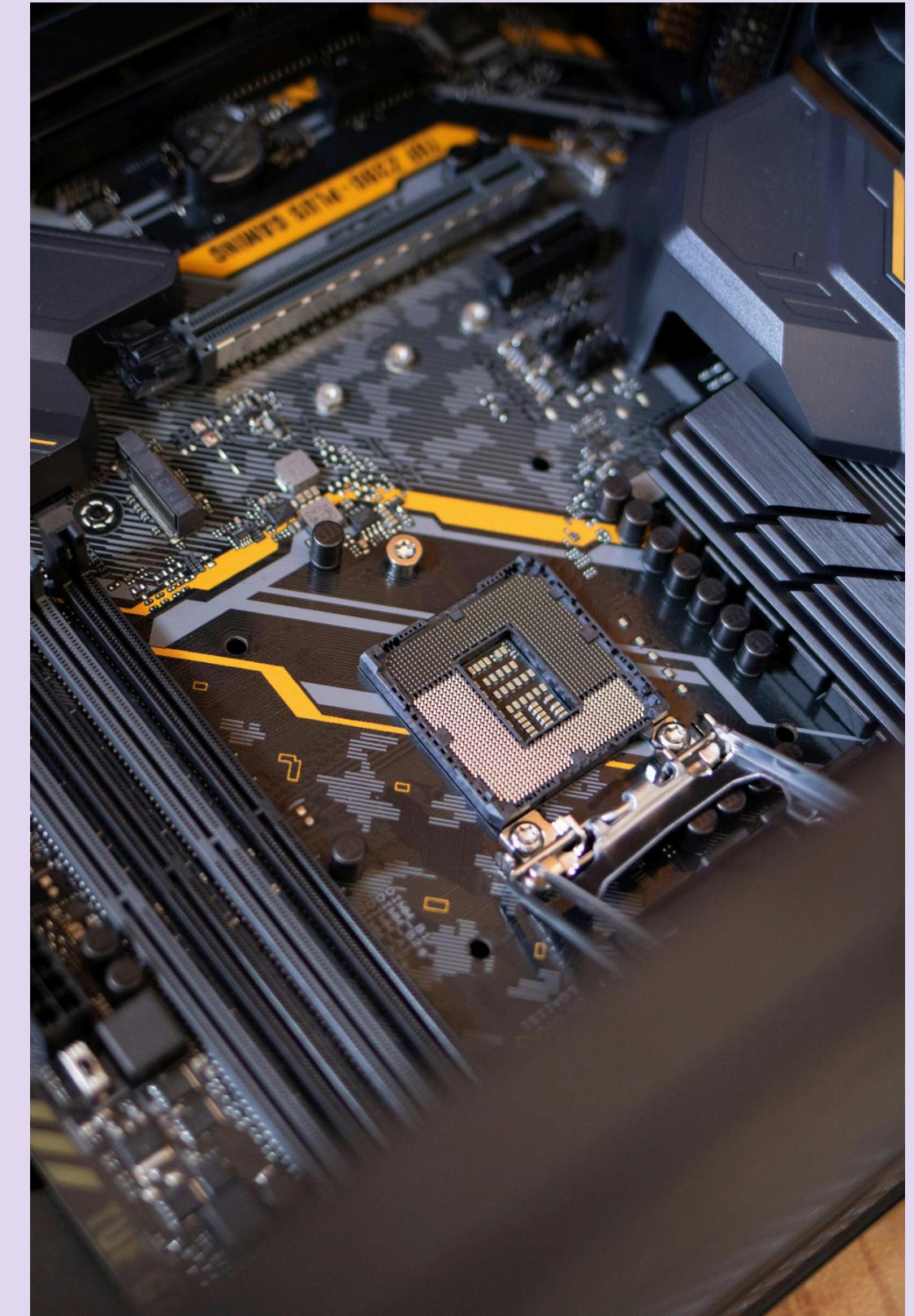
7. Reportes (Análisis y Estadísticas)

- Tipo: Microservicio de business intelligence.
- Ejemplo: "Airbnb Data Platform"

HERRAMIENTAS PROPUESTAS



- A continuación se describirán las herramientas propuestas para el desarrollo de los microservicios. Se decidieron tras haberlas comparado con otras opciones similares, con criterios como rapidez, flexibilidad y fiabilidad.



HERRAMIENTAS PROPUESTAS

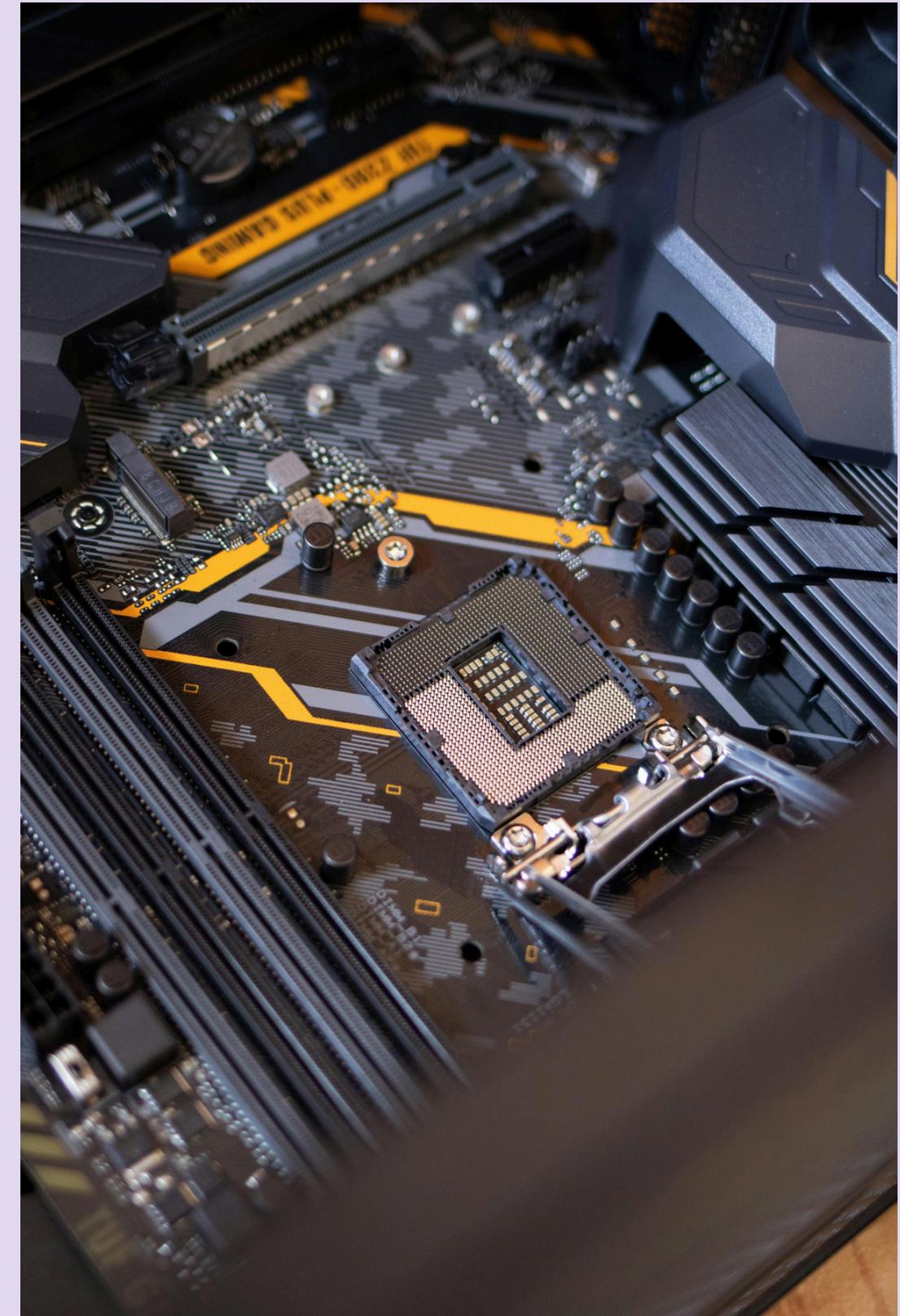
- Framework: Spring boot (Java) (Marco de trabajo, simplifica el desarrollo con configuraciones automáticas): más estructurado, alta compatibilidad, soporte API REST y MySQL (vs. Node.js)
- Base de Datos: MySQL (Almacena datos estructurados): - confiable, de código abierto, intuitivo, relacional (vs. PostgreSQL, MongoDB)
- Contenedores: Docker (Ejecución consistente de aplicaciones en cualquier entorno informático): - lo más rápido, popular y portátil. (vs. Podman, LXC)



HERRAMIENTAS PROPUESTAS



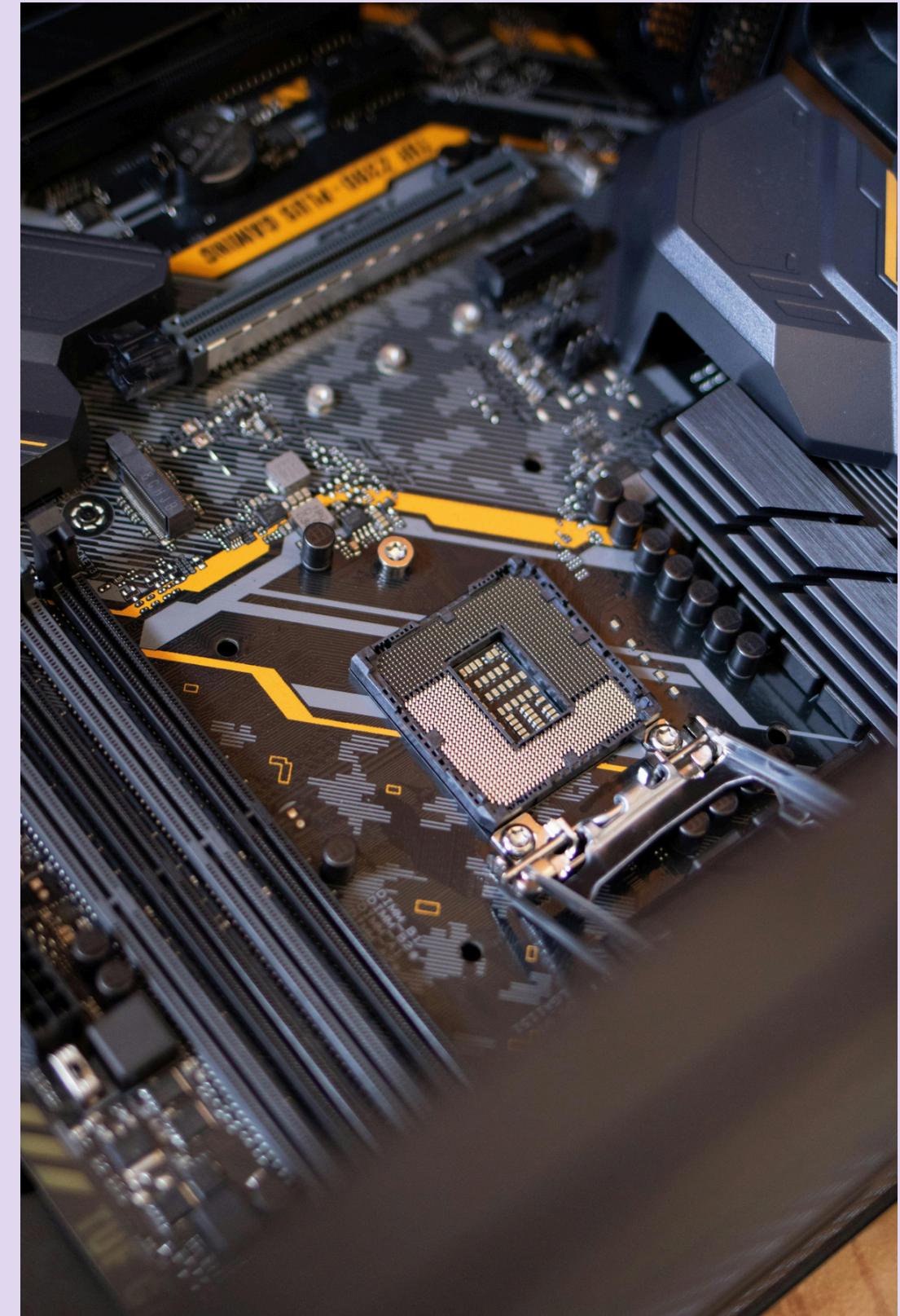
- Orquestación: Kubernetes (Escalar servicios según la demanda):- Líder en industria, potente, mayor soporte aunque complejo (vs. Docker Swarm, Nomad)
- Mensajería: RabbitMQ (comunicación entre microservicios, mejora la resiliencia y disponibilidad del sistema)- fiable, fácil de usar, rápido, poderoso (vs. Kafka, Redis)
- Frontend: React.js (Librería Interfaz de usuario)- interfaces dinámicas, rápidas (negocio lo requiere), mayor adopción empresarial, menor curva de aprendizaje, menos complejidad (vs. Angular, Vue.js)



HERRAMIENTAS PROPUESTAS

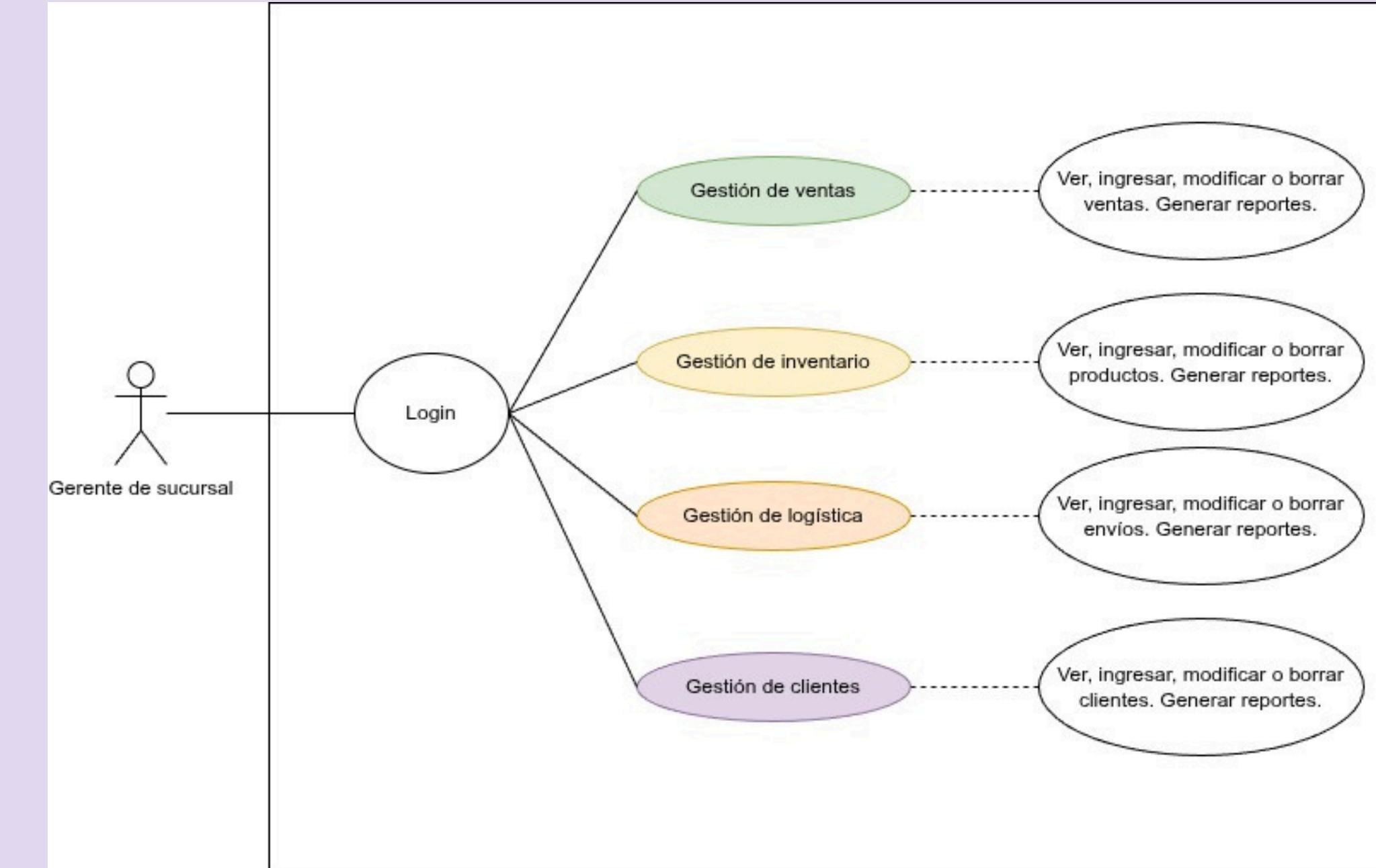
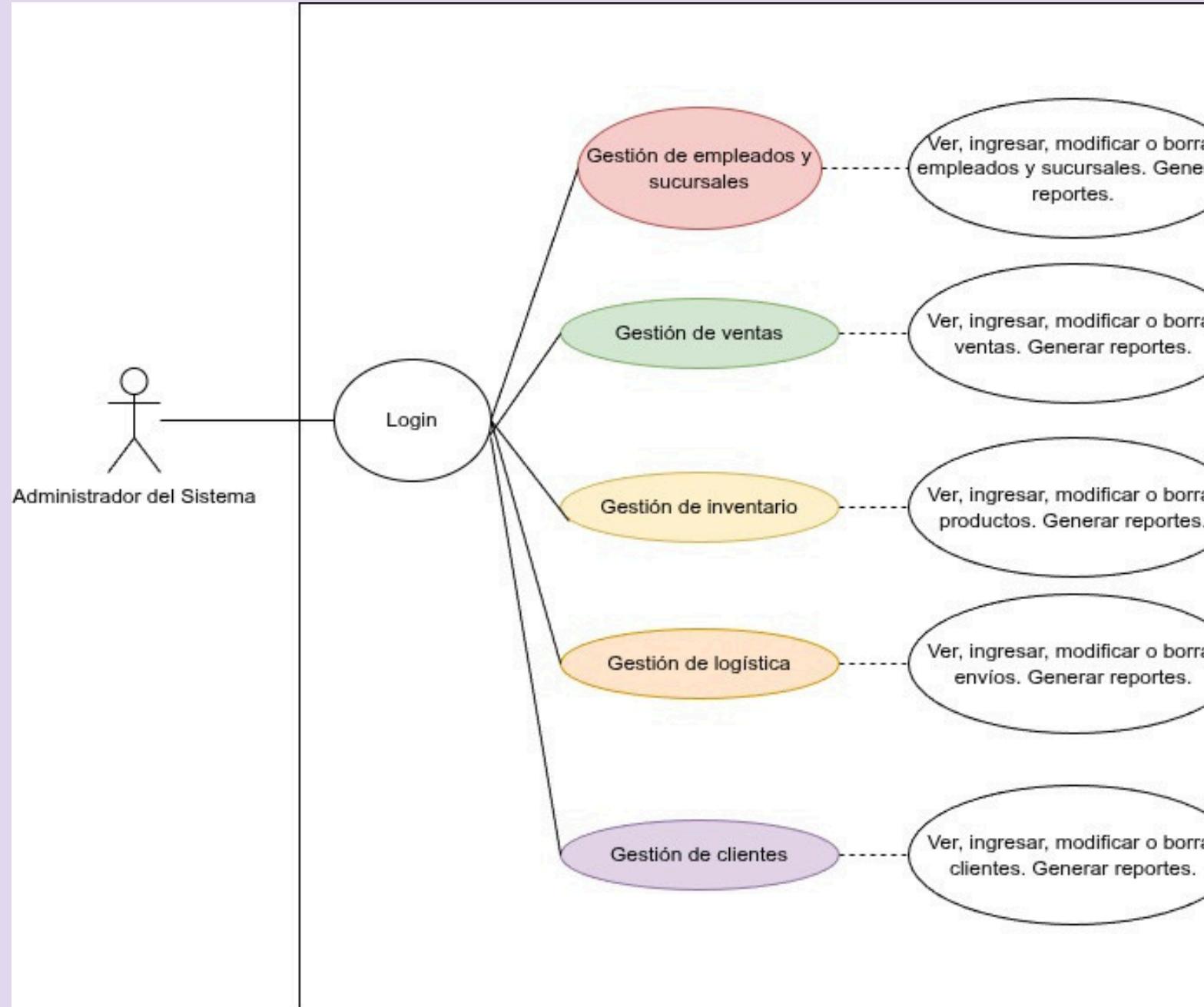


- Planificación y seguimiento de tareas:
Jira/Canva/Google Docs (Se utilizaron para desarrollar el informe y ppt, permitiendo edición de varias personas a la vez)
- Control de versiones y revisión de código:
GitHub
- Seguimiento: Grafana y Prometheus
(Monitoreo en tiempo real del sistema para evaluar rendimiento y detectar anomalías.) alta precisión, flexible, baja complejidad (vs. Nagios, Zabbix)



DIAGRAMAS DE CASOS DE USO

Detallan aún más las funcionalidades de los microservicios

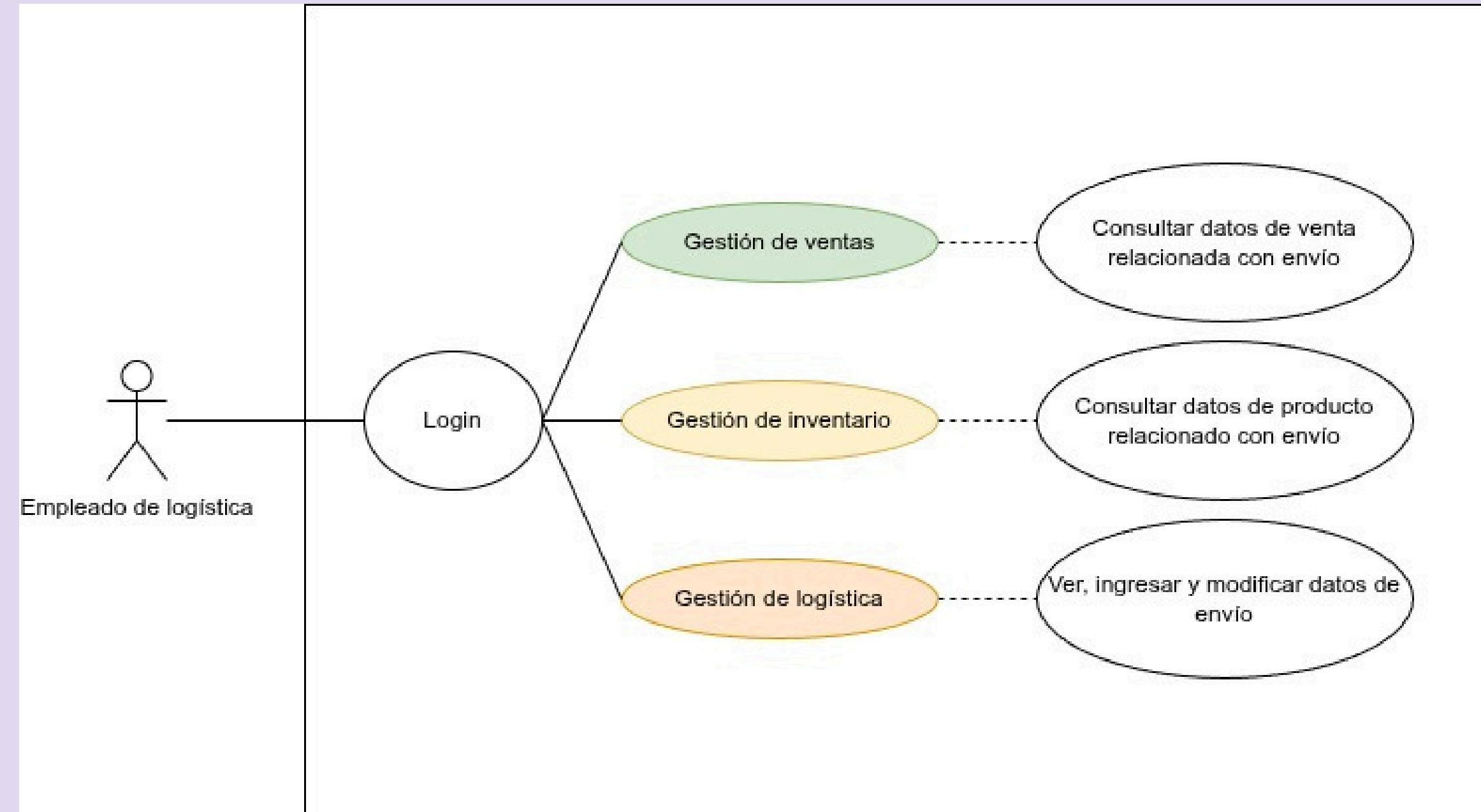


Empleados manejan sólo nivel administrativo

DIAGRAMAS DE CASOS DE USO



Detallan aún más las funcionalidades de los microservicios

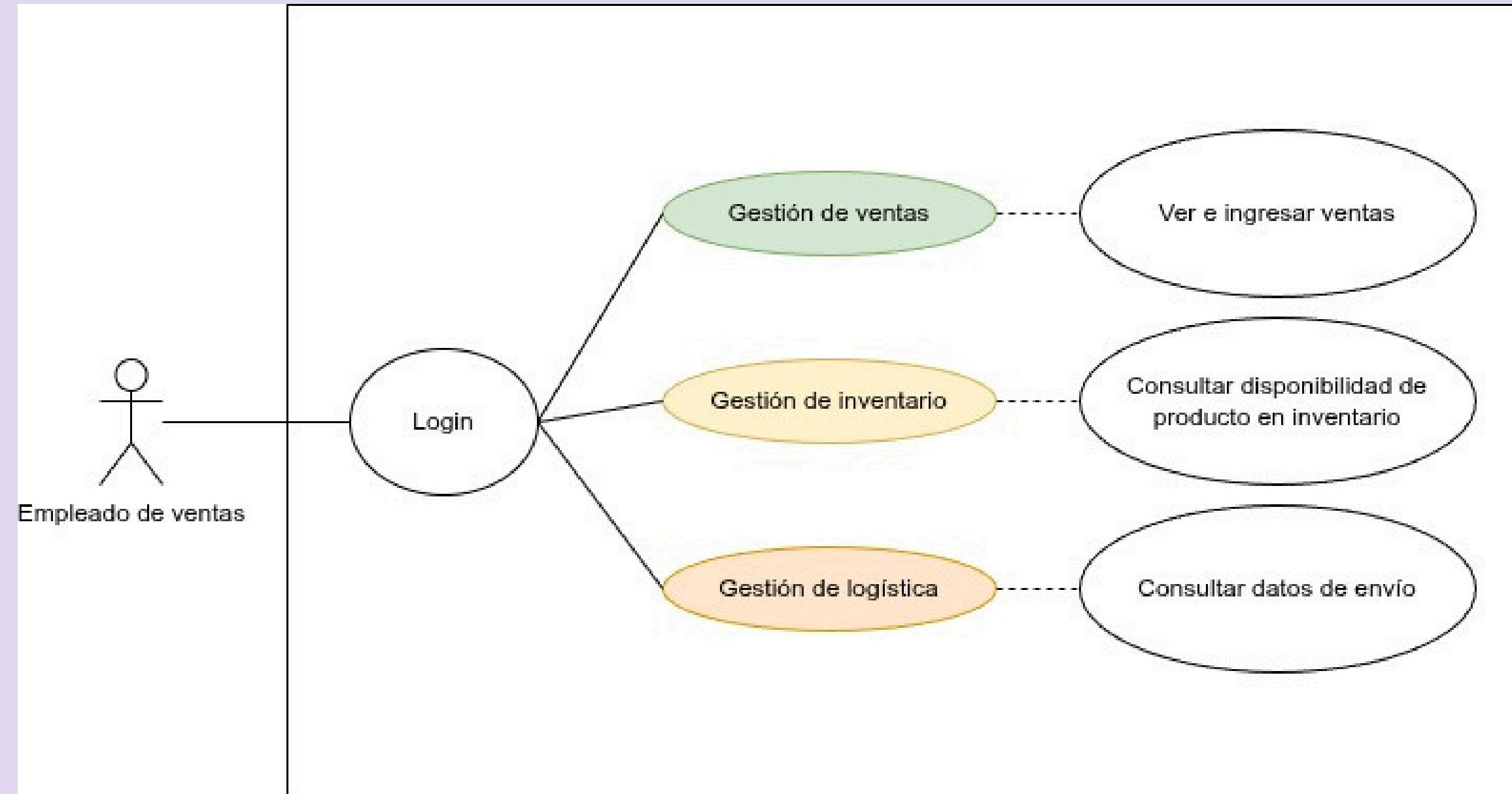


Empleados manejan sólo nivel administrativo

DIAGRAMAS DE CASOS DE USO

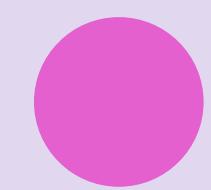


Detallan aún más las funcionalidades de los microservicios



Empleados manejan sólo nivel administrativo

DIAGRAMAS DE CASOS DE USO



Detallan aún más las funcionalidades de los microservicios

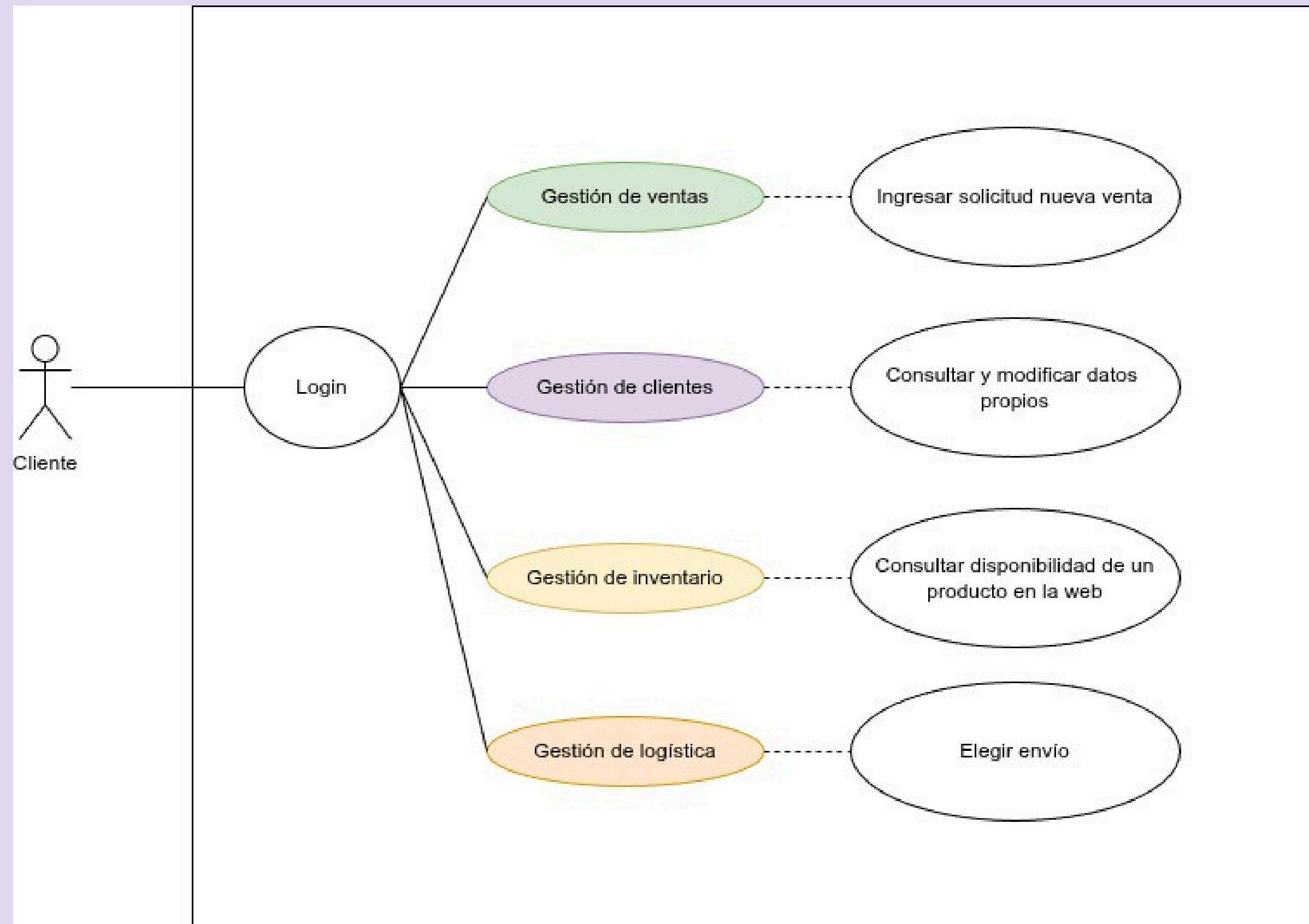


DIAGRAMA DE CLASES



Estructura y modelado del sistema a nivel Informático (Clases, sus atributos y métodos)

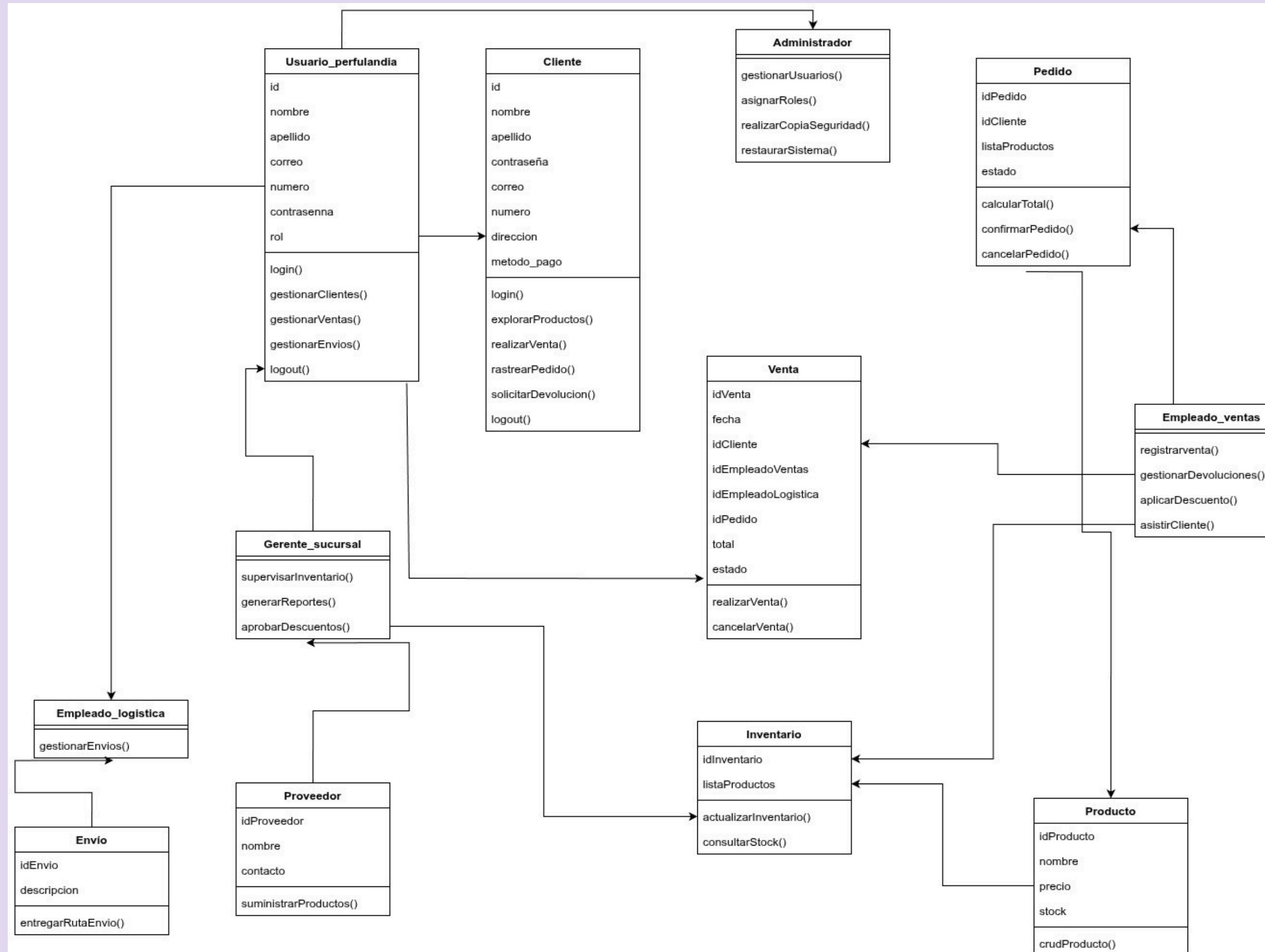
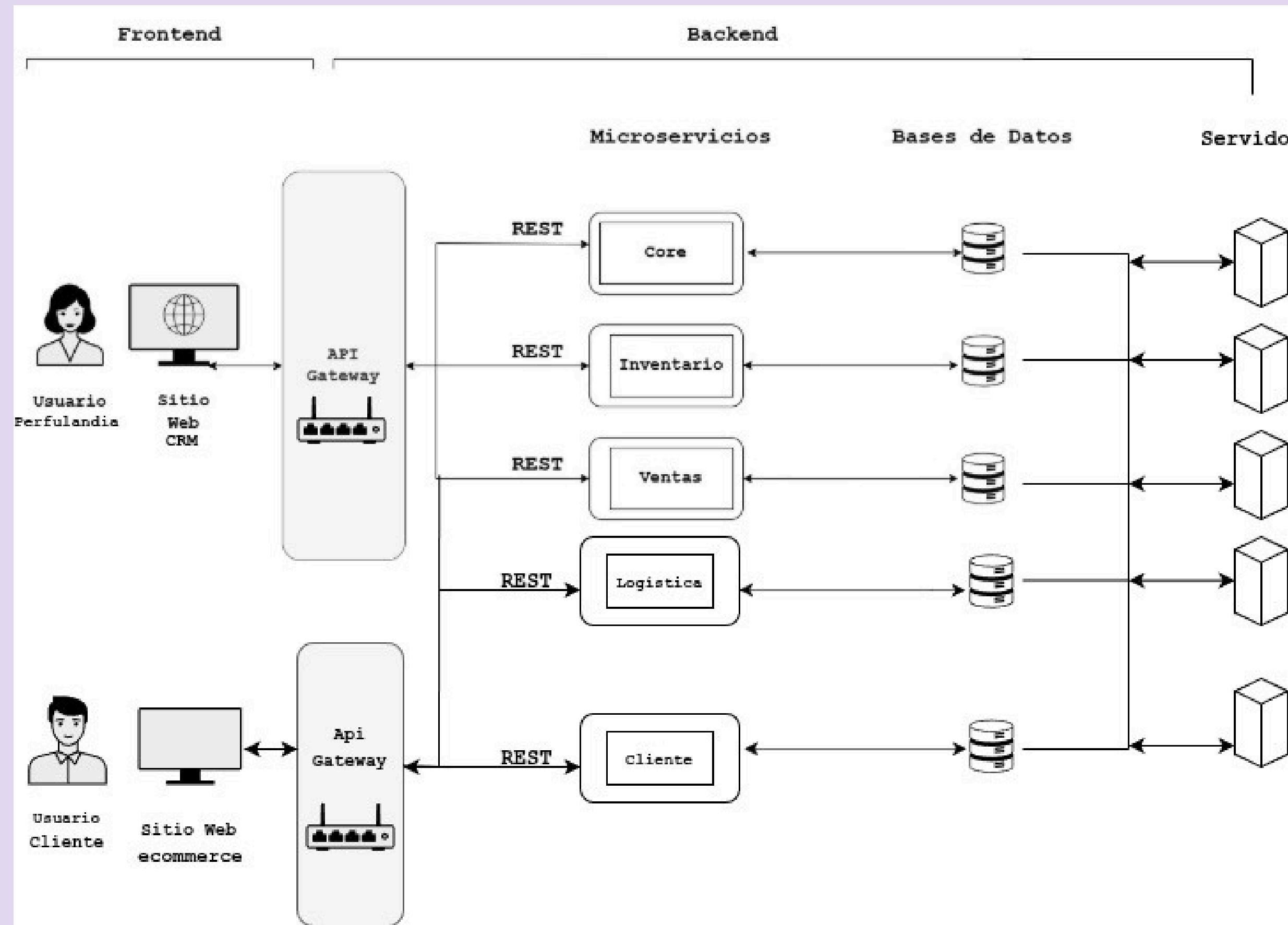


DIAGRAMA DE DESPLIEGUE



Estructura del sistema a nivel físico (hardware e infraestructura real)



4. PLANIFICACION DE LA MIGRACION

La migración del sistema de **monolítico** a **microservicios** busca garantizar una transición fluida y sin interrupciones en las operaciones, permitiendo la **convivencia temporal** entre el sistema antiguo y los servicios a medida que se van implementando.

Se analizaron **ventajas y desventajas** de las metodologías ágil y tradicional, y se decidió utilizar la ágil, pues permite **priorizar las tareas**, **adaptación a cambios inesperados** y retroalimentación constante.



OVERVIEW PLAN DE MIGRACIÓN



Fase 1: Preparación y Monitoreo Inicial

Duración estimada: 2-3 semanas

Acciones:

- Configurar infraestructura base (Kubernetes, Docker, RabbitMQ).
- Instalar y configurar Prometheus + Grafana.
- Auditar el sistema monolítico para identificar puntos de entrada (controladores REST, endpoints clave).
- Establecer un API Gateway que canalice todas las peticiones (p. ej., Zuul o Spring Cloud Gateway).
- **Se establece una capa de control que puede redirigir tráfico al monolito o a microservicios.**



OVERVIEW PLAN DE MIGRACIÓN



Fase 2: Extracción del Servicio Core (Usuarios y Sucursales)

Duración estimada: 3-4 semanas

Motivo: Es una base de control de acceso, no dependiente de lógica transaccional compleja.

Acciones:

- Crear microservicio "Core" (usuarios, roles, autenticación, sucursales).
- Implementar RBAC y autenticación JWT.
- Redirigir endpoints de login/registro desde el API Gateway.
- Actualizar el monolito para usar el nuevo servicio vía REST o colas.
- Gestión de usuarios y sucursales migrada; control de acceso centralizado.



OVERVIEW PLAN DE MIGRACIÓN



Fase 3: Extracción del Servicio de Clientes

Duración estimada: 3 semanas

Acciones:

- Crear microservicio "Clientes" con su propia base de datos.
- Manejar registro, login, perfiles y opiniones.
- Separar autenticación de clientes y empleados.
- Integrar frontend con este nuevo servicio.
- **Experiencia del cliente mejorada y desacoplada del backend de operaciones.**



OVERVIEW PLAN DE MIGRACIÓN



Fase 4: Migración de inventario (Microservicio de Comercio)

Duración estimada: 4 semanas

Acciones:

- Extraer lógica de productos e inventario.
- Asociar inventario a sucursales.
- Sincronizar stock con el microservicio "Ventas" y pedidos del monolito.
- Redirigir llamadas desde frontend/backend hacia este microservicio.
- Stock gestionado de forma independiente, mejora rendimiento en horas peak.



OVERVIEW PLAN DE MIGRACIÓN



Fase 5: Migración de Ventas y Facturación

Duración estimada: 4 semanas

Acciones:

- Crear microservicio "Pedidos" (ventas, pagos, devoluciones).
- Conectar con Inventario y Clientes mediante eventos (RabbitMQ).
- Redirigir flujo de ventas desde API Gateway.
- Validar facturación y reembolsos.
- **Ventas desacopladas, sistema más resiliente y trazable.**



OVERVIEW PLAN DE MIGRACIÓN



Fase 6: Extracción de Logística

Duración estimada: 4 semanas

Acciones:

- Extraer módulo de seguimiento de envíos y proveedores.
- Planificación de rutas y actualización de estado de pedidos.
- Integración con microservicios "Pedidos" y "Clientes".
- Probar con envíos simulados antes de cortar con el monolito.
- Entregas gestionadas eficientemente y en tiempo real.



OVERVIEW PLAN DE MIGRACIÓN



Fase 7: Monitoreo, Refactorización y Eliminación del Monolito

Duración estimada: 2 semanas

Acciones:

- Evaluar logs, rendimiento y errores usando Prometheus/Grafana.
- Apagar módulos del monolito ya reemplazados.
- Refactorizar microservicios según lo aprendido.
- Documentar servicios, endpoints y flujos de comunicación.
- **Monolito eliminado, sistema 100% basado en microservicios.**

MITIGACIÓN DE RIESGOS



- Errores en la migración
 - Pruebas de consistencia, estrategia de respaldo
- Fallas en la integración inicial
 - Pruebas constantes de integración
- Sobrecarga en horas peak
 - Autoescalado (Kubernetes)
- Brechas de seguridad
 - Implementar cifrado desde el inicio.
- Falta de experiencia del equipo
 - Capacitación
- Retrasos en el cronograma
 - Planear con tiempo de sobra, Priorización de microservicios críticos

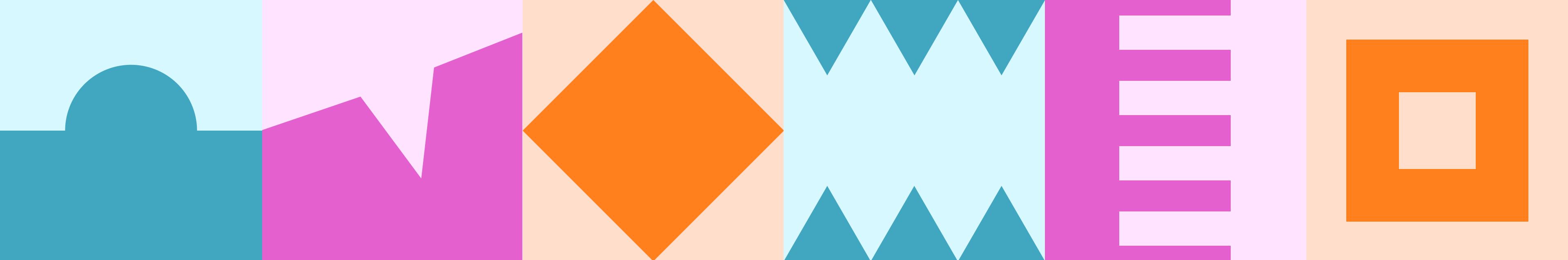
CONCLUSIÓN



- La arquitectura de microservicios **mejora el rendimiento y escalabilidad.**

- Importancia del Trabajo en Equipo: **Clave para el éxito del proyecto.**

Esta arquitectura no solo resuelve los desafíos actuales, sino que posiciona a Perfulandia SPA como una empresa competitiva y preparada para un crecimiento sostenible en el futuro.



MUCHAS GRACIAS

