ANÁLISIS DE ECOMARKET SPA Y SU MIGRACIÓN A MICROSERVICIOS

LUCIANO PEREIRA RODRÍGUEZ

SECCIÓN: 009D

DOCENTE: VÍCTOR ROSENDO

0. Introducción

En un entorno de trabajo en donde existe una constante evolución, se depende de sistemas que enfrentan desafíos significativos en términos de escalabilidad, mantenimiento y disponibilidad. A medida que crecen, aparecen problemas que pueden convertirse en un obstáculo para las operaciones de la empresa.

La migración de una arquitectura a otra se presenta la mayoría de las veces como una solución efectiva para mejorar el rendimiento y la modularidad de los sistemas. Teniendo este enfoque, se permite desarrollar, desplegar y escalar servicios de manera más independiente, optimizando la capacidad de respuesta y permitiendo la continuidad del negocio. Este documento establecerá los objetivos clave para llevar a cabo una transición exitosa, con una infraestructura más flexible y escalable.

1.Definición del Problema

–1.1 Descripción del Sistema Monolítico

ECOMARKET SPA es una empresa que se dedica a la venta de productos ecológicos y sostenibles. Debido al crecimiento exponencial de la empresa, se presentaron problemáticas relacionadas a las operaciones de compra y venta, como también con la base de datos de clientes.

–1.2 Identificación del Problema Principal

El principal problema que presenta la empresa ECOMARKET SPA, es que, al ser de una arquitectura del tipo monolítica, resulta poco escalable para soportar y llevar el crecimiento de la empresa consigo. Trae consigo problemas de rendimiento y eficiencia, afectando de diversas maneras dentro de las operaciones de la empresa.

–1.3 Impacto de dicho problema en el negocio/proyecto.

Estos problemas mencionados en el punto anterior, pueden verse reflejados en una mala experiencia de los clientes ante una baja de eficiencia operacional. Se podrían incluso dar resultados de pérdidas económicas, como también dar una mala imagen de la empresa. Para esto es necesario elaborar un plan de acción para poder mitigar este tipo de problemáticas.

–1.4 Objetivo general y objetivos específicos del proyecto.

El objetivo general de este proyecto está enfocado en migrar el sistema monolítico actual de ECOMARKET SPA a una arquitectura basada en microservicios para mejorar el rendimiento, disponibilidad, escalabilidad y mantenimiento del sistema, garantizando a los usuarios una experiencia óptima, y así crecer acorde a las necesidades que tenga la empresa.

–Objetivos Específicos.

* Principalmente se debe diseñar una arquitectura de microservicios que permite modularizar las funcionalidades principales del sistema, que pueden ser la gestión de ventas, inventario, emisión de boletas o facturación.
* La escalabilidad y disponibilidad deben mejorar a través de la implementación de una estructura más flexible, que permita escalar servicios de manera independiente.
* Facilitar el mantenimiento del sistema mediante la integración de principios de desarrollo ágil y desacoplamiento de componentes, permitiendo actualizaciones y mejoras sin tener que afectar la operación completa por sí misma.
* La seguridad e integridad de los datos se deben garantizar a través de protocolos de autenticación, autorización y cifrado, protegiendo así la información de las operaciones.
* Planificar una migración progresiva que reduzca el impacto en las operaciones de la empresa, garantizando una transición controlada.

2. Estrategia de Microservicios

2.1 Tipo de Estrategia Usada (Dominio o Funcionalidad)

Escoger una arquitectura que use microservicios por dominio ha sido la opción seleccionada para el presente proyecto, ya que garantiza mayor eficiencia y escalabilidad en comparación con los microservicios por funcionalidad.

2.2 Justificación técnica y de diseño de la estrategia elegida.

En contraste, los microservicios por funcionalidad pueden llevar a una fragmentación excesiva de las operaciones, pudiendo generar alta latencia por las múltiples comunicaciones entre servicios. Por esto, es que los microservicios por dominio resultan ser la mejor opción para este tipo de arquitectura.

Entre algunas de las otras razones por la cual los microservicios por dominio son la mejor opción, se puede mencionar que esta modalidad permite encapsular una lógica específica dentro de cada servicio, reduciendo la dependencia entre ellos; los dominios pueden escalarse de manera independiente, según el nivel de demanda, sin generar una conglomeración de microservicios pequeños que complejizan la operación; reflejan la estructura y las necesidades de negocio, lo que facilita la comunicación entre áreas de desarrollo y gestión.

2.3 Beneficios de adoptar esta estrategia en comparación al sistema actual

Algunos de los beneficios a mencionar al adoptar la estrategia elegida:

* Permite ajustar recursos según la demanda de cada dominio sin afectar la operación completa.
* Facilita actualizaciones y correcciones sin interrupciones.
* Cada dominio opera de forma independiente, reduciendo dependencias innecesarias.
* Minimiza la latencia al evitar excesivas llamadas entre servicios.
* Un fallo en un servicio no compromete la totalidad del sistema.

3. Herramientas utilizadas.

3.1 Listado de Herramientas (lenguajes, frameworks, bases de datos, plataformas cloud).

Para la migración del sistema monolítico a una arquitectura de microservicios por dominio, se seleccionaron herramientas que optimizan el rendimiento, la escalabilidad y la mantenibilidad del sistema:

* Lenguaje de programación:
  + Java
* Frameworks:
  + Spring Boot: Para el desarrollo de microservicios.
  + Spring Cloud: Para la gestión de comunicación, configuración y balanceo de carga.
  + Feign Client: Para la comunicación entre microservicios.
  + Lombok: Para simplificar el código, reduciendo la escritura de métodos repetitivos.
* Base de Datos:
  + MySQL: Base de datos relacional para almacenamiento estructurado.
  + Redis: Caché distribuido para mejorar la velocidad de consulta.
* Plataformas Cloud y Orquestación:
  + AWS (Amazon Web Services): Para despliegue en la nube

3.2 Funcionalidad específica de cada herramienta.

* Java + Spring Boot: Desarrollo de microservicios y escalabilidad.
* Spring Cloud: Gestión de balanceo de carga, configuración centralizada y descubrimiento de servicios.
* Feign Client: Facilita la comunicación entre microservicios.
* Lombok: Reduce la escritura de código repetitivo.
* MySQL: Almacenamiento estructurado de datos de negocio.
* Redis: Acelera consultas mediante caché distribuido.
* AWS: Infraestructura cloud para despliegue y escalabilidad.

3.3 Justificación de la elección de estas herramientas.

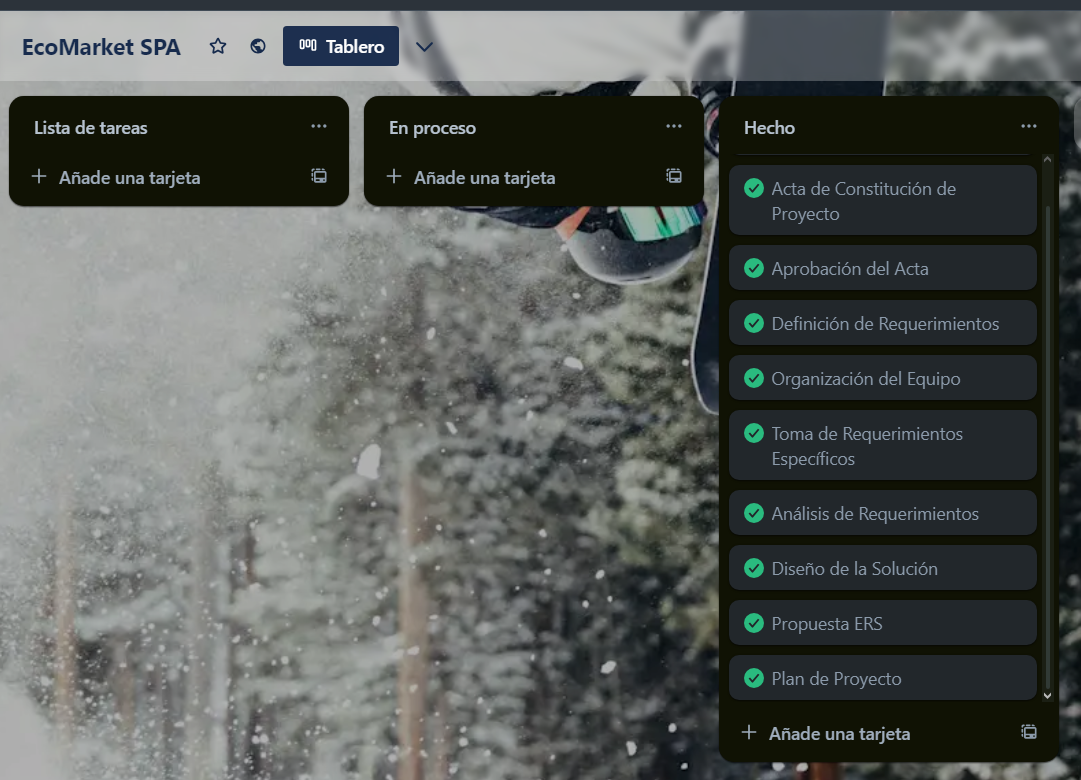
La combinación de herramientas elegidas asegura una infraestructura escalable, eficiente y mantenible, alineada con el crecimiento y necesidades de la empresa.

4. Herramientas de Trabajo Colaborativo.

4.1 Herramientas utilizadas

* **GitHub**: Se utilizará como herramienta para el desarrollo y resguardo del código de los microservicios construir.
* **Discord**: Como plataforma de reuniones y comunicación efectiva.
* **Trello**: Para organizar procesos y tareas relacionadas al proyecto.
* **LucidChart**: Para la construcción de diagramas de procesos.

4.2 Capturas de pantalla o enlaces a tableros como evidencia.



5. Enfoque Ético en el Desarrollo.

Adoptar una arquitectura basada en microservicios conlleva diversos desafíos técnicos, pero también posee desafíos éticos. Es imprescindible que el desarrollo y la migración se realicen con responsabilidad, garantizando el respeto por la privacidad, la seguridad, el bienestar de los colaboradores y el cumplimiento normativo.

5.1 Evaluación de los principales desafíos éticos.

Los principales desafíos éticos, son el manejo responsable de datos personales, la garantía de seguridad de la información, la responsabilidad en el uso de nuevas tecnologías y el impacto que estos cambios pueden tener sobre los puestos de trabajo existentes.

5.2 Privacidad de los datos

Se deberá garantizar la protección de datos personales de los clientes y empleados. Esto requiere implementar mecanismos que aseguren la confidencialidad, integridad y trazabilidad de la información, además de limitar el acceso a los datos más sensibles mediante roles y permisos bien definidos.

5.3 Seguridad.

Para la nueva arquitectura se deberán adoptar medidas de seguridad sólidas, bajo el enfoque de Security by Design. Esto refiere a el uso de cifrado, autenticación robusta entre servicios, control de acceso basado en roles, y monitoreo para detectar accesos indebidos o patrones sospechosos. Cada servicio se aislará, de forma que una brecha de seguridad no llegue a comprometer al sistema completo.

5.4 Responsabilidad en el despliegue.

Se establecerán prácticas y pruebas antes de cada despliegue, con especial foco en el entorno de producción. Además, se mantendrán mecanismos de trazabilidad para identificar el origen de errores y garantizar la rendición de cuentas. Todos los cambios pasarán por revisiones de código y pruebas automatizadas.

5.5 Impacto en los puestos de trabajo.

La transición puede implicar cambios en los roles del personal. El proyecto se compromete a abordar este impacto de forma ética, capacitando a los equipos de trabajo sobre los cambios y las nuevas tecnologías que se pudiesen ir implementando.

5.6 Soluciones y buenas prácticas adoptadas.

Se necesitará la implementación de buenas prácticas dentro del proceso de migración, como lo son la transparencia en la recolección y uso de datos como también la documentación clara del comportamiento del servicio. Se implementará de igual manera una gobernanza de datos que permita a los usuarios ejercer sus derechos digitales, como el acceso, modificación o eliminación definitiva de su información.

5.7 Consideraciones de cumplimiento normativo o legislativo.

El proyecto será implementado en conformidad de la normativa y legislación vigente en Chile, particularmente con la Ley 19.628, que dicta sobre la protección de la vida privada. Asimismo, se integrarán prácticas internacionales como el GDPR (General Data Protection Regulation) por su significado en español, el Reglamento General de Protección de Datos de la Unión Europea, y habla sobre cómo las empresas y organizaciones deben trabajar los datos personales de las personas.

6. Análisis de Requerimientos.

La migración de EcoMarket SPA hacia una arquitectura de microservicios requiere la integración de tanto las necesidades funcionales del negocio como las exigencias técnicas del sistema.

6.1 Requisitos Funcionales

* Administrador del Sistema
* Gestión de cuentas de usuario.
* Asignar y modificar permisos de acceso.
* Monitorear estado del sistema y rendimiento.
* Realizar respaldos automáticos y restauración de datos.
* Gerente de Tienda
  + Gestión de inventario
  + Configurar y establecer datos de la tienda.
  + Generar reportes de ventas e inventario.
  + Autorizar y gestionar pedidos de reabastecimiento.
* Vendedor
  + Procesar ventas
  + Gestionar devoluciones y reclamos.
  + Verificar disponibilidad de productos.
  + Emitir y enviar boletas y facturas electrónicas.
* Equipo de Logística
  + Crear y actualizar envíos.
  + Cambiar estado de pedidos.
  + Gestionar datos de proveedores y recepción de productos.
* Cliente Vía Web
  + Registrarse, iniciar sesión, actualizar datos personales.
  + Buscar y filtrar productos.
  + Agregar productos al carrito y realizar pedidos.
  + Aplicar cupones y ver historial de compras.
  + Consultar estado de pedidos
  + Contactar soporte o dejar reseñas.

6.2 Requisitos No Funcionales

* Escalabilidad
  + El sistema debe ser capaz de escalar servicios de forma independiente para soportar picos de tráfico.
* Seguridad
  + Cifrado de datos en tránsito y en reposo.
  + Autenticación robusta.
  + Gestión segura de credenciales y datos sensibles de clientes.
* Disponibilidad y Rendimiento.
  + Alta disponibilidad (99,9% o superior).
  + Respuesta del sistema en menos de 2 segundos para acciones clave (búsquedas y pagos).
* Tolerancia a fallos
  + Los fallos en un microservicio no deben impactar al sistema completo.
  + Mecanismos de recuperación ante errores.
* Mantenibilidad
  + Modularidad y bajo acoplamiento entre servicios.
  + Soporte para actualizaciones.
* Auditoría y Trazabilidad
  + Registro de acciones clave. (Ventas, accesos, flujo de inventario, etc.)
  + Logs centralizados y accesibles para análisis.
* Usabilidad
  + Interfaz web intuitiva y responsiva para clientes.
  + Experiencia de usuario fluida durante navegación y compra.
* Cumplimiento legislativo
  + Alineamiento con la Ley sobre protección de datos personales en Chile.

6.3 Síntesis de necesidades de cada perfil.

* **Administrador de Sistemas**: Garantizar el funcionamiento seguro, estable y controlado de toda la infraestructura de microservicios.
* **Gerente de Tienda**: Optimizar la operación diaria de la tienda física y tomar decisiones basadas en datos.
* **Empleado de Ventas**: Brindar una experiencia fluida al cliente en el punto de venta y resolver problemas en el acto.
* **Equipo de Logística**: Asegurar una cadena de suministro eficiente y trazable, desde la bodega al cliente.
* **Cliente**: Tener una experiencia cómoda, personalizada y segura.

7. Análisis en el Sistema Actual.

7.1 Arquitectura general del sistema monolítico.

El sistema actual de EcoMarket SPA utiliza una arquitectura monolítica. Esto indica que toda la lógica de negocio, la interfaz de usuario, el acceso a datos y las integraciones con servicios externos están unificados en un solo bloque de código o aplicación, desplegada en un único entorno.

Todo el código de los distintos módulos del sistema comparten la misma base de datos, lo cual puede comprometer el funcionamiento de la operación completa del sistema ante cualquier cambio o mejora que se quiera implementar si es que no se interviene adecuadamente.

Este tipo de arquitectura es útil en las etapas iniciales del negocio, pero a medida que el crecimiento de la empresa se comienza a evidenciar, deja de tener practicidad en las operaciones críticas del negocio.

7.2 Puntos débiles detectados.

Dentro de los puntos débiles identificados en la arquitectura actual de EcoMarket SPA, podemos enlistar los siguientes:

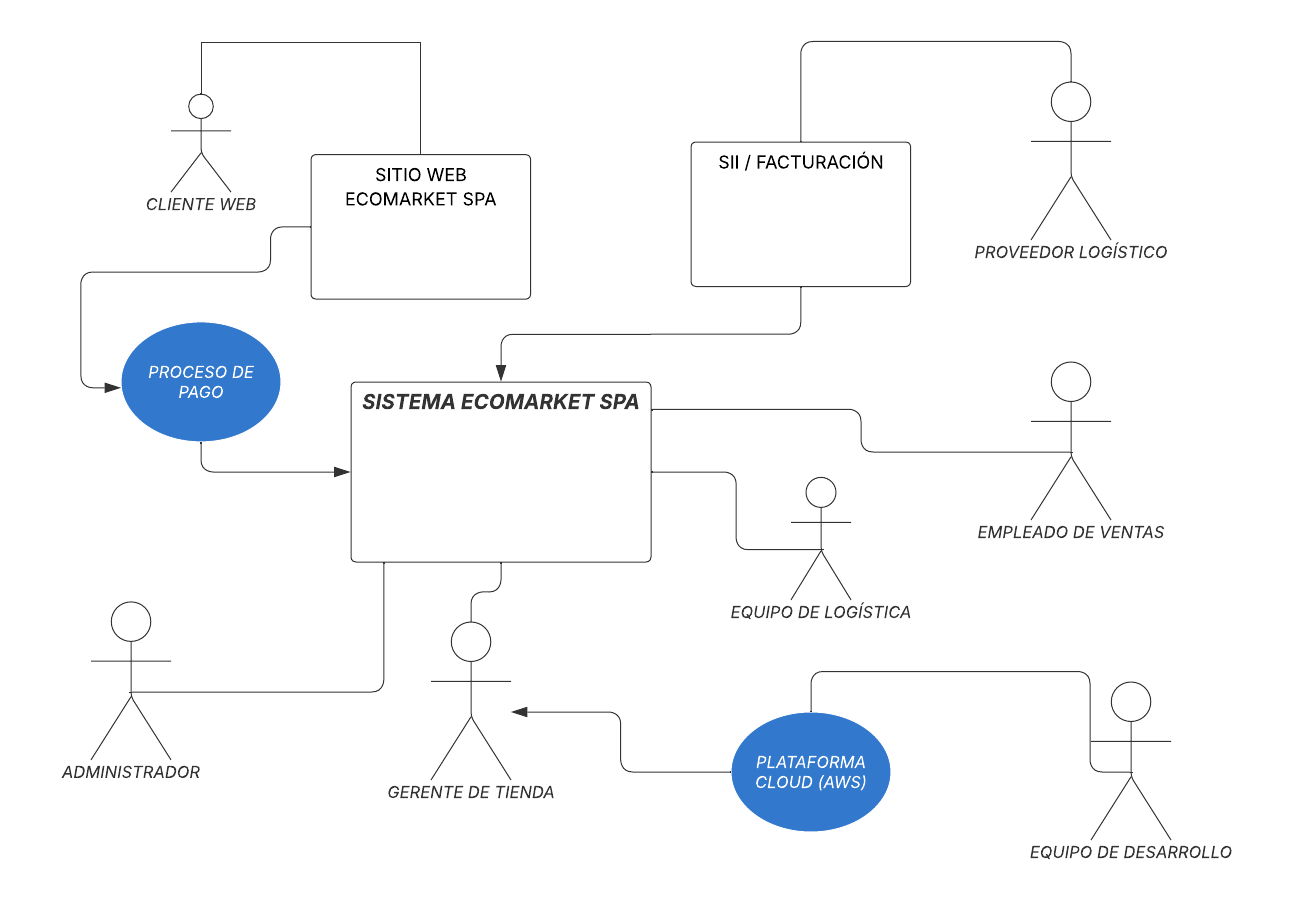
* **Poca Escalabilidad**: No es posible escalar módulos individuales de acuerdo al nivel de demanda.
* **Poca Flexibilidad para el desarrollo**: Es complejo incorporar nuevas funcionalidades o tecnologías, además de resultar en procesos de implementación lentos y riesgosos, al comprometer la funcionalidad completa del sistema.
* **Tiempo de despliegue elevado**: Cualquier cambio requiere volver a compilar, probar y desplegar la aplicación completa.
* **Mantenimiento**: A medida que las necesidades de la empresa crecen, el código también se ve en la necesidad de hacerlo. Es por esto que al momento de tener que realizar cualquier tipo de mantenimiento o testeo, se vuelve complejo incluso entender el código.
* **Disponibilidad Limitada**: Las caídas que puede tener este tipo de arquitectura son críticas, ya que no existe el aislamiento de servicios.
* **Poca Adaptabilidad a las necesidades de Negocio**: La empresa siempre puede estar abierta a cambios, por lo que necesita un sistema que se pueda adaptar a esos cambios. La arquitectura actual no tiene la agilidad suficiente para integraciones externas, abarcar nuevas tiendas, o adaptarse a nuevas campañas de negocio.

8. Diseño de la Nueva Arquitectura.

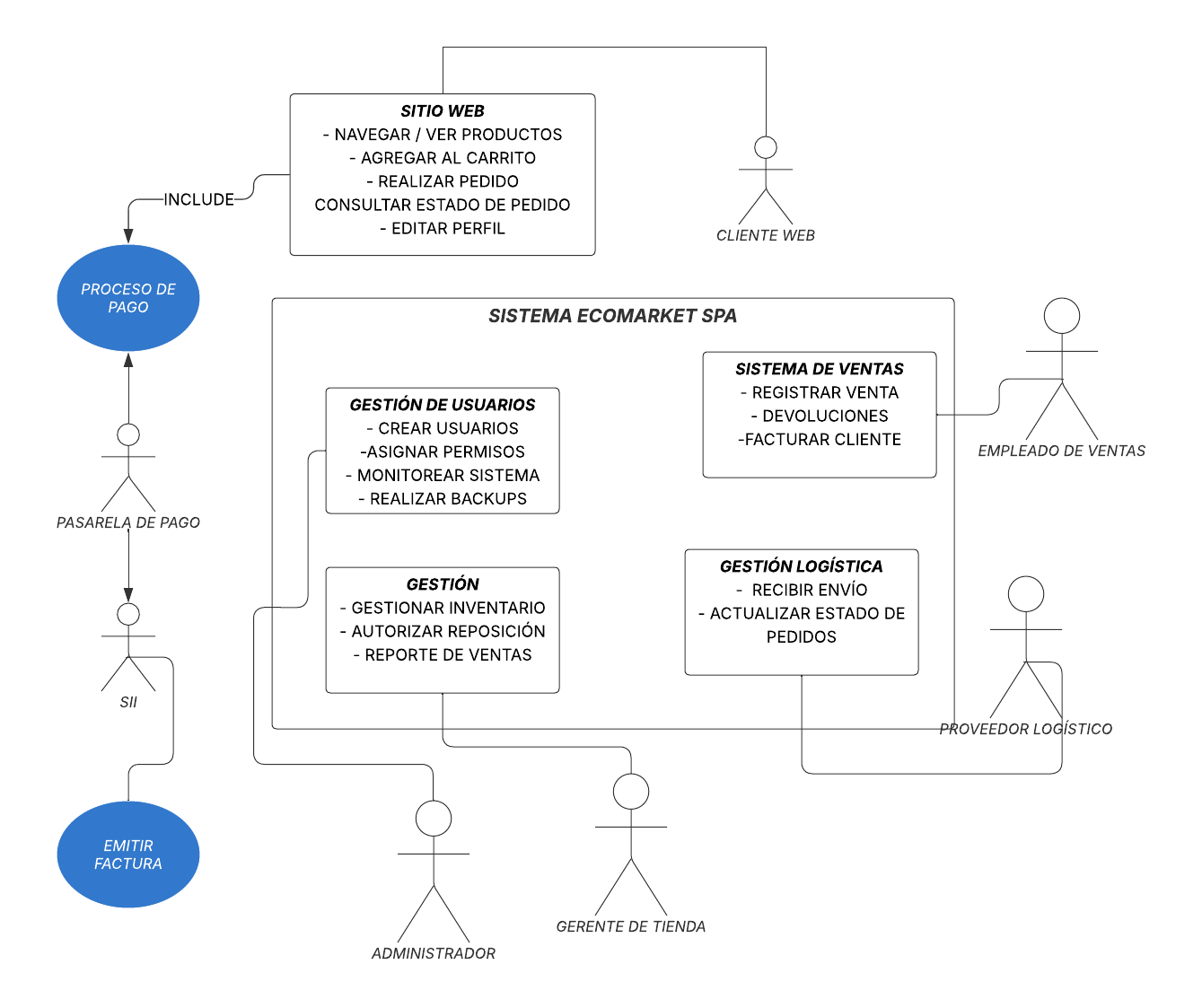
8.1 Descripción general de la arquitectura basada en microservicios

La arquitectura propuesta como solución para EcoMarket SPA se basa en el modelo de microservicios por dominio, lo que implica dividir el funcionamiento del sistema en varios servicios independientes, cada uno orientado a una función determinada del negocio. Esto permite que cada funcionalidad del negocio se desarrolle, despliegue, escale y mantenga de forma autónoma, convirtiéndose en una solución más flexible, escalable y resiliente.

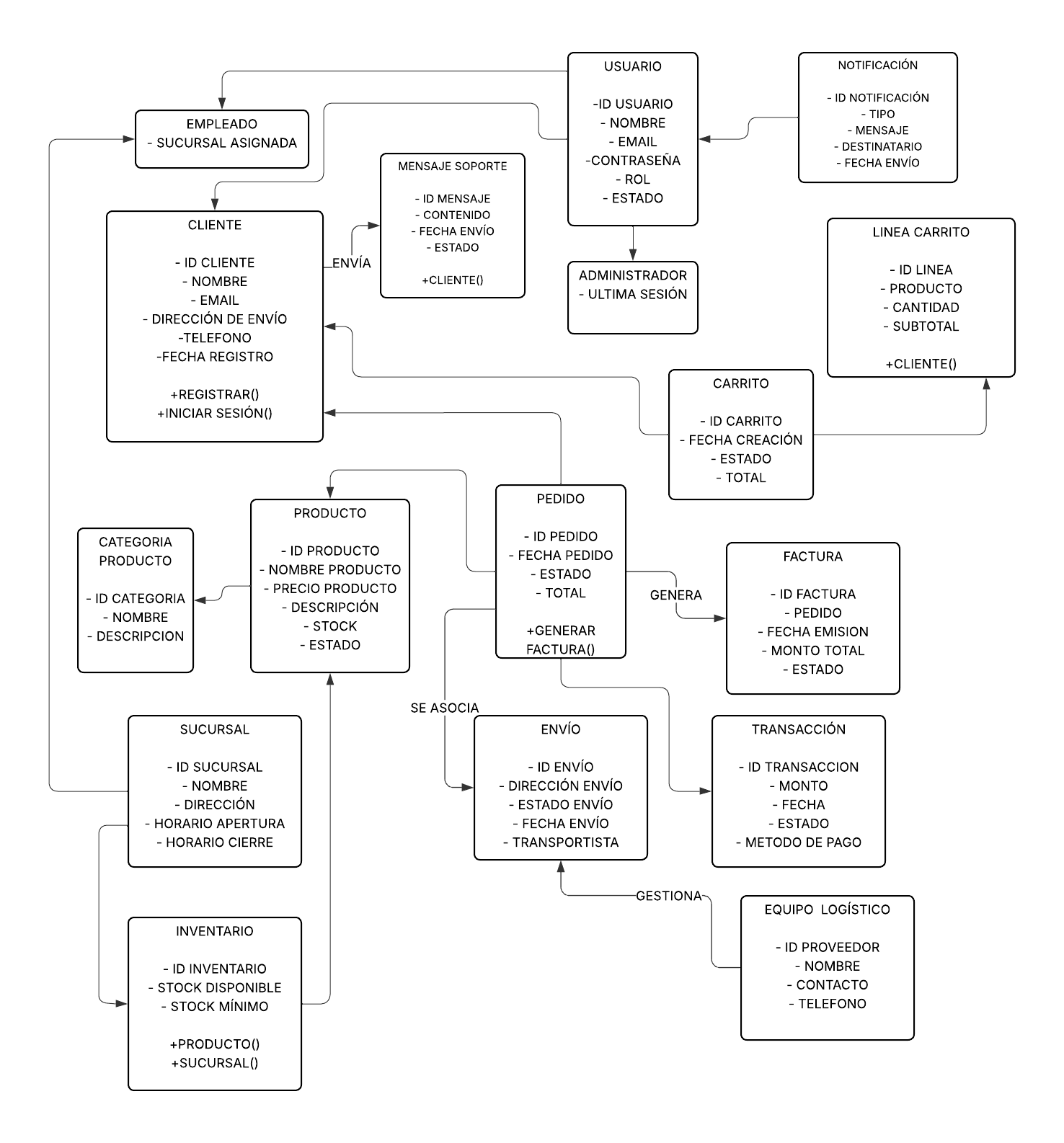
8.2 Diagrama de Actores de Alto Nivel



8.3 Diagrama de Casos de Uso



8.4 Diagrama de Clases



9.Planificación de la Migración.

La migración desde una arquitectura monolítica a una de microservicios por dominio se debe realizar de manera gradual y controlada, para que las operaciones de la empresa no se vean afectadas en su funcionamiento. Este proceso de migración debe garantizar la continuidad del negocio y la integridad de datos.

9.1 Fases de Migración

* **Fase 1: Evaluación y Diseño**:
  + Se debe analizar la arquitectura actual y sus limitaciones.
  + Se deben identificar los dominios funcionales independientes.
  + Se debe diseñar la nueva arquitectura enfocada a microservicios por dominio.
  + Se deben definir las herramientas y tecnologías con las que se trabajarán, así como sus lineamientos técnicos.
* **Fase 2: Preparación del Entorno**:
  + Se debe preparar el entorno de desarrollo y pruebas
  + Se debe configurar la infraestructura en la nube (AWS)
  + Se debe implementar logging, monitoreo y control de versiones.
* **Fase 3: Migración Progresiva**:
  + Se deben migrar primero los servicios de bajo riesgo y alta independencia funcional.
  + Se deben integrar los microservicios con el sistema monolítico existente.
  + Exposición de servicios vía API.
  + Se deben realizar pruebas funcionales y de carga por cada microservicio implementado.
* **Fase 4: Refactorización e Integración Completa**:
  + Se deben eliminar gradualmente las funcionalidades del monolito que hayan sido reemplazadas por los microservicios.
  + Se debe consolidar la comunicación entre servicios.
  + Se deben optimizar las redundancias y eliminar las dependencias que no sean necesarias.
* **Fase 5: Transición Final**:
  + Se debe desactivar el sistema monolítico.
  + Se debe revisar el desempeño global del nuevo sistema.
  + Se debe capacitar el equipo técnico y operativo de la empresa.
  + Se debe monitorear el sistema intensivamente post-despliegue.

9.2 Consideraciones de compatibilidad.

Durante el proceso de migración, se debe garantizar que el nuevo sistema:

* Sea compatible con los datos existentes.
* Permita la coexistencia del monolito con microservicios en la transición.
* Mantenga integraciones actuales con sistemas externos.
* Conserve la experiencia de usuario final.

9.3 Identificación de riesgos y su plan de mitigación.

| RIESGO | PLAN DE MITIGACIÓN |
| --- | --- |
| Pérdida de Datos | Testeos previos en entornos de backups completos. |
| Inestabilidad Operativa | Despliegue gradual, monitoreo activo |
| Sobrecarga del equipo técnico | Capacitación continua, adopción incremental, soporte externo si se necesita. |
| Duplicidad de Funcionalidades | Gestión de versiones y control estricto de endpoints activos |
| Problemas de Integraciones con sistemas externos | Uso de adaptadores, pruebas de integración desde etapas tempranas |

10. Planificación y Cronograma.

10.1 Breve explicación del enfoque ágil usado (SCRUM)

Para llevar a cabo la migración con un mayor margen de éxito, se optará por usar el marco de trabajo SCRUM, una metodología ágil que permite entregar valor al producto de forma incrementable, iterativa y continua.

SCRUM nos permite dividir el proyecto en ciclos cortos llamados SPRINTS, los cuales tienen una duración de 2 semanas usualmente. También es un elemento fundamental que permite priorizar funcionalidades clave de las operaciones de la empresa desde etapas tempranas del desarrollo, así como también permite incorporar mejoras continuas mediante revisiones y retrospectivas.

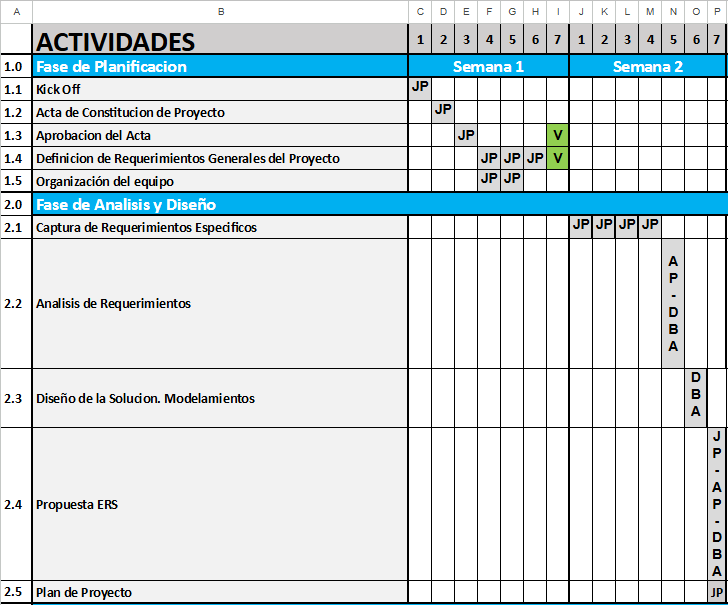
Cada sprint contempla las siguientes etapas:

Planificación → Desarrollo → Revisión → Entrega → Retrospectiva

10.2 Roles asumidos por los integrantes

| ROL | INTEGRANTE |
| --- | --- |
| PRODUCT OWNER | CLIENTE |
| SCRUM MASTER | LUCIANO PEREIRA |
| EQUIPO DE DESARROLLO | LUCIANO PEREIRA |
| STAKEHOLDERS | GERENCIA, DEV. TEAM, SCRUM MASTER, PRODUCT OWNER, ADMINISTRADOR DEL SISTEMA, GERENTE DE TIENDA, EMPLEADOS DE VENTA, CLIENTES FINALES, PROVEEDORES, PROVEEDORES TECNOL., CONSULTORES. |

10.3 Carta Gantt con cronograma de implementación y revisión



11. Conclusiones y Recomendaciones.

11.1 Lecciones aprendidas

Como equipo de trabajo y desarrollo, podemos mencionar algunas lecciones aprendidas en el término de la fase de la primera parte del desarrollo final del presente proyecto de migración:

* Planificar una migración gradual es un punto crítico para poder mantener la continuidad del negocio en su conjunto, para así evitar cualquier tipo de interrupción en las operaciones de la empresa.
* La arquitectura actual de EcoMarket SPA, al ser una de tipo monolítica, tiene limitaciones de su propia naturaleza, que, en un contexto de expansión y crecimiento empresarial, afecta la eficiencia operativa y la experiencia de los clientes.
* La metodología SCRUM permitió organizar el trabajo de forma iterativa, flexible y enfocada en las entregas continuas.
* El diseño de microservicios por dominio ofrece ventajas notables en comparación con el sistema actual, como lo son la modularidad, independencia, escalabilidad, y mejor alineación con las necesidades del negocio

11.2 Potencial Escalabilidad del Sistema

La nueva arquitectura de EcoMarket SPA, basada en microservicios, permitirá:

* Escalar servicios de manera más independiente según los niveles de demanda.
* Adaptación a nuevas funcionalidades o integraciones, sin comprometer el funcionamiento completo del sistema.
* Distribuir la carga entre varios servidores, lo que abre la posibilidad de expansión, con mayor seguridad y rendimiento.

11.3 Posibles mejoras futuras.

* Implementar arquitectura orientada a eventos para poder reducir los acoplamientos y mejorar el flujo de procesos como pedidos, pagos y notificaciones.
* Adoptar contenedores y orquestación para poder simplificar despliegues y escalar horizontalmente.
* Incorporar Inteligencia Artificial para el análisis de comportamientos financieros, de clientes, recomendaciones de productos y optimización de inventario.
* Automatizar el monitoreo, incluyendo métricas de negocio en tiempo real.
* Desarrollar una app móvil propia que se comunique directamente con los microservicios existentes.

CONCLUSIÓN

La transformación del sistema de EcoMarket SPA desde una arquitectura monolítica hacia una basada en microservicios representa un paso estratégico y necesario para sostener su crecimiento y adaptarse a las exigencias del mercado actual. Esta nueva arquitectura permite una plataforma más escalable, flexible y robusta, capaz de soportar una operación nacional distribuida y de responder con agilidad a los cambios del negocio.

Gracias a una planificación estructurada, el uso de metodologías ágiles y una visión centrada en la continuidad operativa, se sientan las bases no solo para resolver los problemas actuales de rendimiento y disponibilidad, sino también para impulsar futuras innovaciones tecnológicas que fortalezcan la competitividad y sostenibilidad de la empresa.

ANEXOS

Tablero de Planificación:

[TRELLO](https://trello.com/invite/b/67e943a7484bb2dedbd43e45/ATTI2757b7c49736f4eec11907bfc16bfbff5ED9F04E/ecomarket-spa)

Código:

[Repositorio GitHub](https://github.com/anhedoniasnm/ECOMARKET-SPA.git)