Caso estudio 3:

EcoMarket SPA

Vicente Alfaro

Iván Morales

Gabriel Sanchez

**Presentación del caso:**

EcoMarket SPA es una empresa chilena en crecimiento que se dedica a la venta de productos ecológicos y sostenibles. Su primer punto de venta estuvo en el Barrio Lastarria, en Santiago, y debido a su éxito tanto en ventas como en su compromiso con la sostenibilidad, la empresa ha expandido su presencia con tiendas en Valdivia y Antofagasta. Con este crecimiento y el aumento de su base de clientes a nivel nacional, EcoMarket SPA necesita expandirse rápidamente. Sin embargo, su sistema monolítico actual está presentando problemas de rendimiento y disponibilidad, lo que está afectando tanto la eficiencia operativa como la experiencia del cliente. Estos inconvenientes representan un serio riesgo para la continuidad del negocio y la satisfacción de su clientela.

**Análisis de Requerimientos:**

Entrevistar a los diferentes perfiles de usuarios (administrador, vendedor, cliente) para

obtener una comprensión detallada de sus necesidades.

**1.- Requisitos Funcionales:**

**Administrador Del Sistema**:

* Gestion de Usuarios: Crear, actualizar, desactivar y eliminar cuentas de usuarios.
* Configurar Permisos: Asignar y modificar permisos de acceso a diferentes módulos.
* Monitorización Del Sistema: Visualizar el estado del sistema y recibir alertas sobre fallos.
* Respaldo y Restaurar Datos: Realizar copias de seguridad y restaurar datos en caso de pérdida.

**Gerente de Tienda**:

* Gestionar Inventarios: Agregar, actualizar y eliminar productos del inventario.
* Generar Reportes: Crear reportes de ventas, inventario y rendimiento de la tienda.
* Gestionar Tiendas: Configurar detalles específicos de la tienda (horarios, personal, políticas).
* Gestionar pedidos: Supervisar y autorizar pedidos de productos.

**Empleados de Ventas:**

* Registrar Ventas: Procesar transacciones de venta y aplicar descuentos.
* Atender Devoluciones y Reclamaciones: Procesar devoluciones y gestionar reclamaciones de clientes.
* Consultar Inventario: Verificar disponibilidad de productos en tiempo real.
* Generar facturas: Emitir facturas electrónicas y enviarlas por correo.

**Logística:**

* Gestionar Envíos: Crear, actualizar y seguir envíos de productos.
* Optimizar rutas: Planificar y optimizar rutas de entrega.
* Actualizar Estado De Pedidos: Cambiar el estado de los pedidos en el sistema.
* Gestionar Proveedores: Mantener y actualizar información de proveedores.

**Acciones De Los Clientes Vía Web:**

* Crear Cuentas: Registrarse en la plataforma web.
* Iniciar sesión: Acceder a la cuenta con credenciales.
* Navegar y Buscar Productos: Explorar el catálogo de productos.
* Agregar Productos Al Carrito: Seleccionar y añadir productos al carrito.
* Realizar pedidos: Completar el proceso de compra.
* Consultar Historial: Ver historial de compras y estado de pedidos.
* Gestionar Perfiles: Actualizar información personal y detalles de pago.
* Solicitar Soporte: Enviar consultas o problemas.
* Dejar Reseñas y Calificaciones: Evaluar productos y dejar comentarios.
* Aplicar Cupones y Descuentos: Ingresar códigos promocionales.

**2.- Requisitos No Funcionales:**

* Rendimiento**:** El sistema debe manejar al menos 1000 transacciones simultáneas sin degradar el rendimiento.
* Escalabilidad: Debe ser capaz de escalar horizontalmente para soportar un aumento en la carga de usuarios y transacciones.
* Seguridad: Implementar medidas de seguridad para proteger datos sensibles de usuarios y transacciones.
* Disponibilidad: El sistema debe estar disponible al menos el 99% del tiempo.
* Usabilidad:La interfaz debe ser intuitiva y fácil de usar para todos los perfiles de usuario.
* Mantenibilidad: El sistema debe ser fácil de mantener y actualizar sin interrumpir el servicio.

**Entrevista a perfiles de usuario:**

1. Entrevista al Administrador del Sistema
   1. ¿Cuáles son las principales funciones que realiza en el sistema actualmente?

R/ Principalmente gestiono usuarios, configuro permisos y monitorizo el rendimiento del sistema.

* 1. ¿Qué problemas ha encontrado en el sistema monolítico que afectan su trabajo diario?

R/ He notado que el sistema se vuelve lento durante las horas pico, lo que afecta la capacidad de respuesta.

* 1. ¿Cómo gestiona los permisos de acceso y qué dificultades ha enfrentado en este proceso?

R/ A veces es complicado asignar permisos específicos, ya que el sistema no permite configuraciones detalladas.

* 1. ¿Qué herramientas o funcionalidades le gustaría tener para mejorar la monitorización del sistema?

R/ Me gustaría tener un panel de control más intuitivo que me permita ver el estado del sistema en tiempo real.

* 1. ¿Cómo maneja las copias de seguridad y la restauración de datos? ¿Ha tenido problemas en este aspecto?

R/ Realizamos copias de seguridad manualmente, lo que puede ser arriesgado. Un sistema automatizado sería ideal.

1. Entrevista al Empleado de Ventas
   1. ¿Cómo es el proceso de registrar ventas en el sistema? ¿Es eficiente?

R/ El proceso es bastante sencillo, pero a veces se producen errores en los descuentos aplicados.

* 1. ¿Qué problemas ha encontrado al atender devoluciones y reclamaciones?

R/ El sistema no siempre refleja correctamente el estado de las devoluciones, lo que genera confusión.

* 1. ¿Cómo consulta el inventario y qué mejoras le gustaría ver en esta funcionalidad?

R/ A veces, la información no se actualiza en tiempo real, lo que me lleva a ofrecer productos que no están disponibles.

* 1. ¿Qué tan fácil es generar facturas y enviar correos electrónicos a los clientes?

R/ Emitir facturas es fácil, pero el envío por correo a veces falla y tengo que hacerlo manualmente.

* 1. ¿Qué sugerencias tiene para mejorar la experiencia del cliente durante el proceso de compra?

R/ Sería útil tener un sistema de chat en línea para resolver dudas de los clientes al instante.

1. Entrevista al Cliente (Usuario de la Plataforma Web)
   1. ¿Cómo fue su experiencia al crear una cuenta en la plataforma?

R/ Fue un proceso sencillo, aunque me gustaría que hubiera más opciones de personalización.

* 1. ¿Qué tan fácil fue navegar y buscar productos en el catálogo?

R/ La búsqueda es bastante buena, pero a veces los filtros no funcionan como deberían.

* 1. ¿Tuvo algún problema al agregar productos al carrito o realizar un pedido?

R/ No tuve problemas, pero el proceso podría ser más rápido.

* 1. ¿Cómo calificaría el proceso de consulta de su historial de pedidos?

R/ El historial es fácil de acceder, pero me gustaría ver más detalles sobre cada pedido.

* 1. ¿Qué mejoras le gustaría ver en la plataforma para facilitar su experiencia de compra?

R/ Sería genial tener recomendaciones personalizadas basadas en mis compras anteriores.

**Análisis del Sistema Actual:**

El sistema monolítico fue el elegido por la empresa en sus inicios debido a que es el más simple y barato, ideal para un emprendimiento nuevo que no dispone de los recursos necesarios para crear un sistema más complejo. Sin embargo, con el crecimiento exponencial que ha tenido la empresa, el sistema monolítico comenzó a presentar sus desventajas, como el mantenimiento complejo una vez que la aplicación crece o la dificultad para escalar horizontalmente. Por lo tanto, se vieron en la necesidad de buscar un sistema más eficiente para satisfacer las crecientes necesidades de la empresa.

**PRINCIPALES CAUSAS DE FALLOS**

Las principales causas de fallos y sobrecarga del sistema monolítico de EcoMarket SPA son: escalabilidad limitada, dependencias fuertes entre componentes, rendimiento deficiente de la base de datos, falta de tolerancia a fallos, baja disponibilidad y mantenimiento complejo, problemas en la gestión de recursos, implementaciones lentas y costosas, dependencia de tecnología obsoleta, problemas en la gestión de transacciones y falta de monitorización y visibilidad.

**Diseño de la Nueva Arquitectura:**

**Propuesta de Arquitectura Basada en Microservicios**

La nueva arquitectura se basará en la descomposición del sistema monolítico en varios microservicios independientes, cada uno encargado de una funcionalidad específica. Esto permitirá una mayor escalabilidad, flexibilidad y facilidad de mantenimiento. A continuación se detallan los microservicios propuestos:

* Microservicio de Gestión de Usuarios:

Funcionalidades:

* + - Registro y autenticación de usuarios.
    - Gestión de perfiles y permisos.

Tecnologías:

* + - Framework: Spring Boot (Java) o Express.js (Node.js).
    - Base de datos: MySQL.
* Microservicio de Gestión de Inventario:

Funcionalidades:

* + - Agregar, actualizar y eliminar productos.
    - Control de stock y alertas de bajo inventario.

Tecnologías:

* + - Framework: Django (Python) o Flask (Python).
    - Base de datos: MySQL.
* Microservicio de Gestión de Pedidos:

Funcionalidades:

* + - Procesar y gestionar pedidos de clientes.
    - Actualizar el estado de los pedidos.

Tecnologías:

* + - Framework: Spring Boot (Java) o Ruby on Rails.
    - Base de datos: MySQL.
* Microservicio de Gestión de Envíos:

Funcionalidades:

* + - Crear y seguir envíos de productos.
    - Optimizar rutas de entrega.

Tecnologías:

* + - Framework: Node.js o ASP.NET Core.
    - Base de datos: MySQL.
* Microservicio de Atención al Cliente:

Funcionalidades:

* + - Gestión de consultas y soporte al cliente.
    - Procesar devoluciones y reclamaciones.

Tecnologías:

* + - Framework: Flask (Python) o Express.js (Node.js).
    - Base de datos: MySQL.
* Microservicio de Reportes:

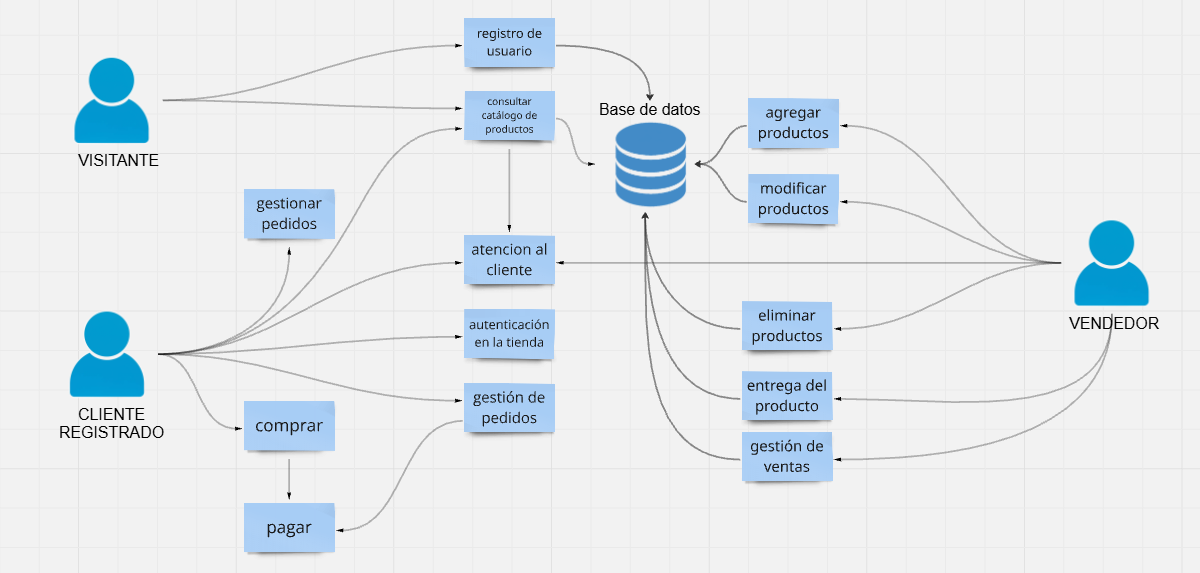
Funcionalidades:

* + - Generar reportes de ventas, inventario y rendimiento.

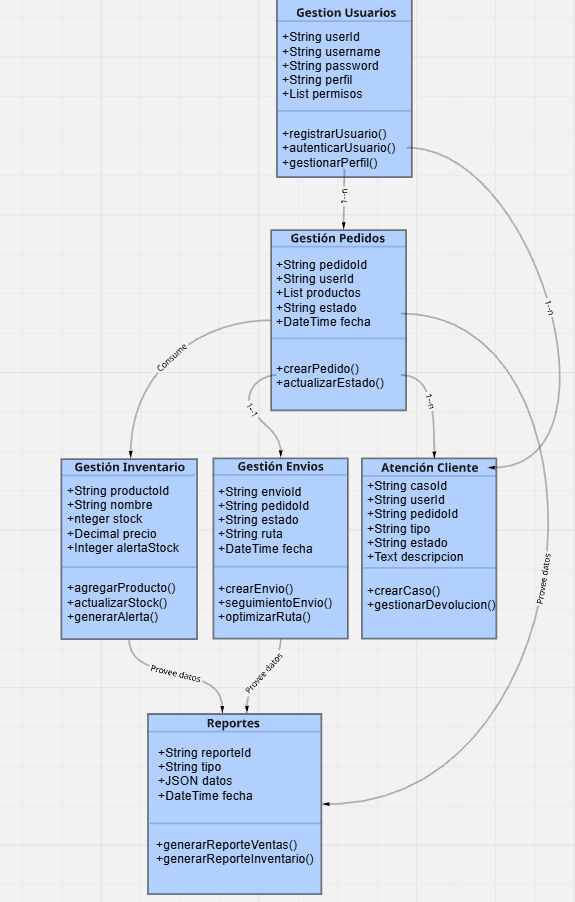
Tecnologías:

* + - Framework: Spring Boot (Java) o Django (Python).
    - Base de datos: MySQL.

**Diagrama casos de uso**

****

**Diagrama de clases**

****

### 

### Diagrama de Despliegue

### 

### Planificación de Migración

**Fase 1: Migración Gradual**

#### 1. Migración de Funcionalidades del Sistema Monolítico

* **Migración por fases**: No realizar la migración completa de una vez. Es recomendable migrar gradualmente los módulos del sistema monolítico a microservicios. Por ejemplo:  
  1. **Fase 1**: Migrar el microservicio de **gestión de productos**.
  2. **Fase 2**: Migrar el microservicio de **gestión de usuarios**.
  3. **Fase 3**: Migrar el microservicio de **gestión de pedidos**.
  4. **Fase 4**: Migrar el microservicio de **gestión de pagos**.
* **Integración continua**: Durante cada fase, se debe integrar y probar los microservicios con el sistema existente para garantizar que las funciones críticas sigan siendo accesibles.

#### 

#### 2. Implementación de un API Gateway

* **Desarrollo del API Gateway**: Implementar un API Gateway como punto de entrada para los usuarios. El Gateway se encargará de enrutar las solicitudes a los microservicios correspondientes.
* **Centralización de la autenticación y autorización**: El API Gateway también puede manejar la autenticación (JWT, OAuth) para que no sea necesario hacerlo en cada microservicio.

#### 3. Transición de Datos

* **Migración de bases de datos**: A medida que se migran las funcionalidades al nuevo sistema, los datos también deben ser transferidos gradualmente. Esto se puede hacer en paralelo, manteniendo sincronizadas las bases de datos del sistema monolítico y los nuevos microservicios.
* **Estrategia de sincronicidad de datos**: Asegurar que los datos del sistema monolítico y los microservicios estén sincronizados durante la transición. Se puede utilizar una estrategia de **event sourcing** o eventos para actualizar los datos en los microservicios de manera asíncrona.

### Fase 2: Despliegue y Monitoreo

#### 4. Despliegue Gradual de Microservicios

* **Despliegue en producción**: Después de realizar las pruebas y validaciones en un entorno de staging, se comienza a desplegar los microservicios de manera gradual en el entorno de producción, minimizando la posibilidad de interrupciones.
* **Despliegue en contenedores**: Usar **Docker** y **Kubernetes** para desplegar los microservicios de manera eficiente, con escalabilidad automática y manejo de fallos.

#### 5. Monitoreo y Logging

* **Implementar monitoreo**: Utilizar herramientas como **Prometheus** y **Grafana** para monitorear la salud de los microservicios y medir el rendimiento.
* **Implementación de logging**: Configurar herramientas como **ELK Stack** (Elasticsearch, Logstash, Kibana) o **Splunk** para la gestión de logs, permitiendo la trazabilidad y depuración de los servicios.

#### 

#### 

#### 6. Ajuste de Escalabilidad

* **Autoescalado**: Configurar **Kubernetes** para que los microservicios se escalen automáticamente en función de la carga de trabajo. Esto permite que el sistema maneje picos de tráfico sin problemas.
* **Optimización del rendimiento**: Evaluar el rendimiento de cada microservicio y ajustar su configuración (tamaño de instancias, recursos asignados) para garantizar la eficiencia operativa.

### Fase 3: Evaluación y Optimización

#### 7. Evaluación del Sistema

* **Pruebas post-despliegue**: Después de que todos los microservicios estén desplegados, realizar pruebas de carga, rendimiento y disponibilidad para garantizar que el sistema sea escalable y estable.
* **Revisión de la infraestructura**: Evaluar la infraestructura de red, bases de datos y otros componentes para identificar áreas de mejora.

#### 8. Optimización Continua

* **Refactorización y mejora de microservicios**: A medida que se adquieran nuevos conocimientos sobre el sistema, realizar mejoras en la implementación de los microservicios, optimizando el código y las bases de datos.
* **Automatización**: Asegurar que todos los procesos de integración, despliegue y monitoreo estén completamente automatizados, permitiendo una mejora continua del sistema.

**Riesgos de el plan de micro servicios**

### 1. Riesgo de Interrupción del Servicio

* Dado que la migración se realizará de forma gradual, existe el riesgo de que los servicios no migrados o parcialmente migrados presenten fallos o incompatibilidades con los nuevos microservicios. Esto podría resultar en interrupciones del servicio.
* Mitigación: Implementar pruebas exhaustivas en cada fase y asegurarse de que los microservicios y el sistema monolítico sean compatibles durante la transición.

### 2. Sincronización de Datos

* La migración gradual de datos y la necesidad de mantener los sistemas monolíticos y los microservicios sincronizados durante el proceso puede generar problemas de consistencia de datos, especialmente si se produce un fallo en la estrategia de sincronicidad de datos (como la integración de eventos o event sourcing).
* Mitigación: Establecer un mecanismo robusto de sincronización de datos en tiempo real entre el sistema monolítico y los microservicios, y realizar auditorías frecuentes para asegurar la integridad de los datos.

### 3. Complejidad en la Gestión de Múltiples Microservicios

* A medida que se añaden más microservicios, la complejidad general del sistema aumenta, lo que podría dificultar la gestión, el monitoreo y la resolución de problemas.
* Mitigación: Utilizar herramientas avanzadas de monitoreo (como Prometheus y Grafana) y logging (como ELK Stack) para gestionar la visibilidad de los servicios. También es fundamental que los equipos de desarrollo y operaciones estén bien entrenados en microservicios y en la gestión de sistemas distribuidos.

### 4. Riesgo de Desempeño y Escalabilidad

* Aunque la migración a microservicios tiene la ventaja de mejorar la escalabilidad, cada microservicio puede enfrentar problemas de desempeño si no se dimensiona correctamente o si no se optimiza adecuadamente.
* Mitigación: Configurar Kubernetes para autoescalar los servicios en función de la demanda y realizar pruebas de carga periódicas para evaluar la eficiencia y escalabilidad del sistema**.**

### 5. Gestión de Dependencias entre Microservicios

* La introducción de microservicios puede crear dependencias entre ellos que, si no se gestionan adecuadamente, pueden llevar a fallos en cadena o a un rendimiento deficiente.
* Mitigación: Definir claramente las dependencias entre microservicios y mantenerlas bajo control mediante contratos de servicios (por ejemplo, APIs bien definidas). Además, utilizar técnicas de orquestación y evitar las dependencias circulares entre microservicios.

### 6. Seguridad y Autenticación

* Con el uso de microservicios y la implementación de un API Gateway para manejar la autenticación y autorización, la seguridad podría volverse un punto débil si no se gestionan adecuadamente los tokens de autenticación o el manejo de credenciales.
* Mitigación: Asegurarse de que el API Gateway implemente una autenticación robusta (como JWT o OAuth) y utilizar prácticas de seguridad como la gestión centralizada de secretos y la auditoría de acceso.

### 7. Costo y Tiempo de Implementación

* La migración a microservicios puede ser costosa y llevar mucho tiempo, lo que puede generar una presión financiera y de recursos sobre la empresa. Además, el proceso de migración podría extenderse más de lo esperado.
* Mitigación: Establecer un presupuesto y cronograma realistas, con márgenes para imprevistos. También se deben priorizar las funcionalidades clave para asegurar que se logren los objetivos más importantes en las primeras fases.