DuocUC® | ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES

SkillSwap

INFORME DE FUNCIONAMIENTO Y ASPECTOS ÉTICOS DE LA ARQUITECTURA DE MICROSERVICIOS DE SKILLSWAP

Integrantes:

- ***Diego Herrera***
- ***Esteban Bravo***
- ***Sebastian Carrera***
- ***Luis Rosales***

Índice

Problema del Caso

La empresa Skill Swap es una empresa Chilena emergente que se define a sí misma como una red social que permite a profesionales ofrecer y solicitar intercambios de habilidades, fomentando el aprendizaje colaborativo.

Inicialmente, la empresa comenzó con una extensión de FaceBook, para luego expandirse a un híbrido entre FaceBook y LinkedIn, pero su éxito en usuarios y los pagos por publicidad la forzaron a pensar en independizarse y formar su propia plataforma. La empresa ahora planea continuar su expansión debido a su crecimiento exponencial y el aumento de nuevos clientes a nivel nacional. Sin embargo, este rápido crecimiento ha revelado las limitaciones de su actual sistema de software monolítico. El sistema ha comenzado a fallar, presentando problemas de rendimiento y disponibilidad que ponen en riesgo las operaciones diarias y la satisfacción del cliente.

Análisis del Problema Actual de la Empresa

La arquitectura actual no soporta el crecimiento exponencial de usuarios y funcionalidades, afectando la experiencia del usuario, la operación del negocio y su proyección futura. En resumen, debido a la arquitectura monolítica que posee la empresa, ha comenzado a mostrar limitaciones críticas como:

- Caídas frecuentes del sistema bajo alta demanda.
- Dificultad para escalar componentes de manera independiente.
- Bajo rendimiento en transacciones y navegación del usuario.
- Baja disponibilidad y confiabilidad en momentos de alta concurrencia.
- Dificultades en el mantenimiento, actualización y despliegue continuo.

Requerimientos del Proyecto

Para el desarrollo del sistema, es fundamental identificar y establecer los requerimientos necesarios que permitirán dar solución a las problemáticas actuales y cumplir con el objetivo principal de mejorar la experiencia de los clientes.

Los requerimientos se dividen en dos grandes categorías: requerimientos funcionales, que definen las funcionalidades que el sistema debe ofrecer, y requerimientos no funcionales, que establecen las características de calidad, rendimiento y condiciones generales que debe cumplir la solución.

Requisitos Funcionales

Los requisitos Funcionales para el caso que debe poseer el nuevo sistema son:

1. Gestión de Usuarios:

a. RF 1.1: Crear, actualizar, desactivar y eliminar cuentas de usuarios.
b. RF 1.2: Asignar y modificar los permisos de acceso a diferentes módulos y funciones del sistema.
c. RF 1.3: Monitorear el estado del sistema en tiempo real.
d. RF 1.4: Recibir alertas de fallos del sistema.
e. RF 1.5: Realizar copias de seguridad periódicas y restaurar datos cuando sea necesario.

2. Gestión de Publicidad:

a. RF 2.1: Agregar, actualizar y eliminar anuncios.
b. RF 2.2: Configurar la cantidad de repeticiones de anuncios.
c. RF 2.3: Generar reportes de ventas de publicidad y rendimiento.
d. RF 2.4: Configurar núcleos locales, como modalidades de intercambio: en línea, eventos presenciales, cafés, seminarios, etc.
e. RF 2.5: Supervisar y autorizar pedidos de publicidad.

3. Gestión de Transacciones y Reclamos:

a. RF 3.1: Procesar transacciones de Skill Swap
b. RF 3.2: Gestionar devoluciones y reclamaciones
c. RF 3.3: Consultar inventario de intercambios
d. RF 3.4: Emitir y enviar facturas electrónicas

4. Funcionalidades para Clientes Vía Web:

a. RF 4.1: Crear y gestionar cuentas de usuarios.
b. RF 4.2: Iniciar sesión en la plataforma.
c. RF 4.3: Buscar y explorar Skill Swap con filtros y búsqueda avanzadas.
d. RF 4.4: Agregar productos al carrito de compras, para aquellos usuarios de publicidad.
e. RF 4.5: Realizar pedidos y completar el pago, para aquellos usuarios de publicidad.
f. RF 4.6: Consultar historial de pedidos, para aquellos usuarios de publicidad.
g. RF 4.7: Actualizar el perfil, direcciones y métodos de pago.
h. RF 4.8: Enviar consultas y/o problemas mediante formulario o chat en línea.

5. Gestión de Eventos de Skill Swap:

a. RF 5.1: Crear y administrar eventos tipo TED.
b. RF 5.2: Gestionar inscripción a eventos.
c. RF 5.3: Integración con TicketMaster para la venta de entradas.
d. RF 5.4: Gestionar permisos con organizadores y entidades gubernamentales.

Requisitos No Funcionales

1. **Rendimiento y Escalabilidad:**
    a. RNF 1.1: El sistema debe soportar varios usuarios de manera simultánea sin degradación del rendimiento. (VERIFICAR EL DATO)
    b. RNF 1.2: La infraestructura debe permitir escalabilidad horizontal y vertical para soportar el crecimiento de usuarios.
    c. RNF 1.3: Las consultas a la base de datos no deben exceder los milisegundos en promedio (VERIFICAR EL DATO)
2. **Seguridad:**
    a. RNF 2.1: Los datos de los usuarios deben almacenarse encriptados.
    b. RNF 2.2: Implementación de autenticación de dos factores para usuarios administrativos.
    c. RNF 2.3: Uso de sistema de autenticación segura (VERIFICAR EL DATO)
    d. RNF 2.4: Protección contra ataques comunes. (VERIFICAR EL DATO)
3. **Disponibilidad y Confiabilidad:**
    a. RNF 3.1: La plataforma debe estar disponible la mayor parte del tiempo para los usuarios. (VERIFICAR EL DATO)
    b. RNF 3.2: Implementación de balanceo de carga para evitar fallos por alto tráfico.
    c. RNF 3.3: Mecanismos de recuperación ante desastres, con respaldo automático de datos. (VERIFICAR EL DATO)
4. **Usabilidad y Experiencia de Usuario:**
    a. RNF 4.1: La plataforma debe ser responsive y accesible desde diferentes dispositivos (móviles, escritorios, tablets, etc).
    b. RNF 4.2: El diseño debe cumplir con estándares de accesibilidad.
    c. RNF 4.3: La interfaz debe ser intuitiva.
    d. RNF 4.4: Dejar reseñas y calificaciones sobre Skill Swap.
5. **Integración y Compatibilidad:**
    a. RNF 5.1: Integración con pasarelas de pago como MercadoPago, TransBank, WebPay, Kishki etc.
    b. RNF 5.2: API para integración con plataformas externas como TicketMaster.
    c. RNF 5.3: Compatibilidad con navegadores modernos.

Perfiles de Usuarios

1. Usuarios Regulares (Profesionales)

Estos son los usuarios principales de Skill Swap, profesionales que buscan intercambiar habilidades con otros. Pueden crear un perfil detallando sus áreas de expertise (ej: diseño, programación, marketing) y buscar oportunidades de aprendizaje colaborativo. Sus funciones incluyen publicar solicitudes de intercambio, unirse a eventos (presenciales o virtuales), gestionar transacciones de habilidades y calificar experiencias. Por ejemplo, un desarrollador web podría ofrecer mentoría en JavaScript a cambio de aprender diseño UX/UI de otro usuario.

2. Clientes de Publicidad (Anunciantes)

Son empresas, emprendedores o instituciones que utilizan la plataforma para promocionar servicios, cursos o eventos pagados. Tienen acceso a herramientas de gestión de anuncios, donde configuran audiencias, presupuestos y frecuencia de visualización. También pueden generar reportes de desempeño (ej: clics, conversiones). Un caso típico sería una escuela de idiomas anunciando talleres de inglés en la sección de eventos de Skill Swap.

3. Administradores de la Plataforma

Equipo interno de Skill Swap encargado de garantizar el correcto funcionamiento del sistema. Supervisan métricas de rendimiento, resuelven fallos técnicos, realizan respaldos de datos y gestionan permisos de usuarios (ej: suspender cuentas inapropiadas). También autorizan la publicación de anuncios y eventos para cumplir con políticas de la plataforma. Su rol es crítico para mantener la seguridad y escalabilidad del servicio.

4. Organizadores de Eventos

Usuarios con permisos especiales para crear y gestionar eventos colaborativos, como charlas TED, workshops o meetups. Configuran detalles (fechas, modalidad, aforo), gestionan inscripciones y se integran con servicios como TicketMaster para venta de entradas. Por ejemplo, un experto en sostenibilidad podría organizar un seminario sobre energías renovables, atrayendo a profesionales interesados en el tema.

5. Entidades Gubernamentales o Institucionales

En casos de eventos masivos o alianzas, organismos públicos o educativos pueden participar como colaboradores. Gestionan permisos legales (ej: habilitaciones para eventos presenciales) o promueven programas de capacitación. Por ejemplo, una municipalidad podría asociarse con Skill Swap para ofrecer talleres gratuitos de empleabilidad.

Entrevistas

Para diseñar un sistema que cumpla con las expectativas de los usuarios, es clave realizar entrevistas estructuradas que capturen sus necesidades, dolores y expectativas. A continuación, se presenta un formato adaptable para cada perfil:

Entrevista al administrador de la plataforma

¿Qué buscamos?

R: *Identificar necesidades técnicas, de seguridad y gestión del sistema.*

Gestión de usuarios:

¿Qué problemas enfrenta actualmente al crear/desactivar usuarios?
R: *Actualmente, el proceso es manual y consume mucho tiempo. Por ejemplo, al desactivar un usuario, no se revierten automáticamente sus permisos en eventos o anuncios, lo que genera inconsistencias. Además, no hay notificaciones a los usuarios afectados por cambios en sus cuentas.*

¿Necesita integración con otros sistemas (ej: SSO, LDAP)?
R: *Sí, sería ideal tener SSO (Single Sign-On) para usuarios empresariales que ya usan Google Workspace o Microsoft 365. También necesitamos integrarnos con LDAP para validar credenciales de empleados internos.*

Seguridad y permisos:

¿Qué niveles de acceso requiere (ej: roles para gerentes, moderadores)?
R: *Necesitamos al menos estos roles:*

- *Superadmin (acceso total),*
- *Moderador (gestionar usuarios y contenido, pero no configuraciones del sistema),*
- *Auditor (solo lectura para reportes y logs).*
  Problema actual: Los permisos son estáticos y no se pueden personalizar por módulo (ej: un moderador que solo gestione eventos).

¿Cómo manejan actualmente las brechas de seguridad o accesos no autorizados?
R: *Hoy usamos logs manuales y revisamos IPs sospechosas, pero no hay alertas automáticas. Por ejemplo, si un usuario intenta acceder 10 veces en 5 minutos, no se bloquea temporalmente. Necesitamos un SIEM (como Splunk) para detectar patrones de ataque.*

Entrevista al Gerente de Ventas de Publicidad

¿Qué buscamos?

R// *Comprender necesidades comerciales, reportes y gestión de campañas.*

Gestión de anuncios:

¿Cómo priorizan los anuncios (ej: por categoría, ubicación geográfica)?

R: *Actualmente, los anuncios se priorizan manualmente según:*

- *Categoría (ej: habilidades técnicas tienen mayor prioridad que hobbies).*
- *Ubicación geográfica (ej: anuncios para eventos en Santiago tienen más peso que regiones).*
- *Presupuesto del anunciante (los clientes premium aparecen primero).*
  Problema: No hay un sistema de subasta automatizado (como Google Ads) para optimizar ingresos.

¿Necesitan herramientas de drag-and-drop para editar banners?
R: *¡Absolutamente! Hoy dependemos del equipo de diseño para cada cambio (ej: ajustar texto o imágenes), lo que retrasa las campañas. Una herramienta drag-and-drop con plantillas pre aprobadas aceleraría todo.*

Reportes y analytics:

¿Qué datos son clave para tomar decisiones (ej: CTR, conversión por rubro)?
R: *Los métricas críticas son:*

- *CTR (Click-Through Rate) por tipo de anuncio (historia vs. post).*
- *Conversión a "skill swaps" (cuántos clicks generan intercambios reales).*
- *ROI por cliente (comparar ingresos vs. costo de la campaña).*
  Dato faltante: No podemos medir el engagement post-intercambio (ej: si los usuarios repiten).

¿Prefieren dashboards interactivos o reportes en PDF/Excel?
R: *Dashboards en tiempo real (como Power BI) para monitorear campañas activas, pero también reportes semanales en Excel para análisis históricos con el equipo financiero.*

Núcleos locales (eventos presenciales):

¿Cómo gestionan actualmente los eventos (ej: integración con Ticketmaster)?
R: *Hoy usamos planillas Excel y emails para coordinar con Ticketmaster:*

- *La venta de entradas es externa (no hay API integrada).*
- *Los asistentes no se sincronizan automáticamente con nuestra base de datos.*
  Necesidad: Una integración que muestre en nuestro sistema: entradas vendidas, asistentes registrados y check-in en el evento.

¿Requieren validación automática de permisos municipales?
R: *Sí. Hoy esto lo hace manualmente un asistente, revisando webs municipales (y a veces se pierden plazos). Una API con gobiernos locales para validar permisos (ej: capacidad del lugar, horarios) sería un game-changer.*

Entrevista al Encargado de Skill Swap

¿Qué buscamos?

R// *Optimizar procesos operativos (ventas, reclamaciones).*

Procesamiento de transacciones:

¿Qué métodos de pago deben soportarse (ej: WebPay, PayPal)?
R: *Actualmente sólo aceptamos transferencias bancarias y WebPay, pero perdemos clientes por no tener opciones como:*

- *PayPal (para usuarios internacionales).*
- *Tarjetas de crédito/débito (con cuotas, clave para eventos premium).*
- *Wallet interno (ej: saldo recargable para intercambios rápidos).*

¿Necesitan facturación automática con SII?
R: *¡Urgentemente! Hoy generamos facturas manualmente en el sistema del SII, lo que causa errores y retrasos. Necesitamos:*

- *Integración directa con el SII para boletas y facturas electrónicas.*
- *Alertas por documentos rechazados (ej: RUT inválido).*

Atención al usuario:

¿Cómo gestionar reclamaciones hoy? ¿Requieren un sistema de tickets (ej: Zendesk)?
R: *Usamos Gmail y planillas compartidas, ¡es un caos! Los casos se pierden o duplican. Requerimos:*

- *Un sistema de ticketing (como Zendesk) con:*
  - *Categorías predefinidas (ej: "fraude", "falla técnica").*
  - *Asignación automática a agentes según especialidad.*
- *Chat integrado en la plataforma para resolver problemas en tiempo real.*

¿Deben los usuarios poder cancelar un skill swap por 24 horas?
R: *Sí, pero con condiciones:*

- *Cancelación libre en 24h solo para intercambios no presenciales.*
- *Para eventos presenciales, hasta 48h antes (por logística).*

- *Penalización por cancelación recurrente (ej: 3 cancelaciones = suspensión temporal).*

Entrevista a Clientes (Usuarios Web)

¿Qué buscamos?

R// *Mejorar la experiencia de usuario (UX) y fidelización.*

Registro y perfil:

¿Qué información les parece invasiva al pedir en el registro?
R: *Me molesta que pidan datos personales como RUT o dirección exacta solo para registrarme. Con nombre, email, profesión y ubicación (comuna) debería bastar. Lo demás podría completarse después.*

¿Les gustaría loguearse en las redes sociales?
R: *¡Sí! Es más rápido y seguro que con Google o LinkedIn. Además, así podrían importar mi red de contactos para recomendaciones.*

Búsqueda y filtros:

¿Qué filtros usan más?
R: *Los imprescindibles son:*

- *Ubicación (ej: "Santiago Centro" o radio de 5 km).*
- *Valoración (mínimo 4 estrellas).*
- *Tipo de intercambio (ej: "clases presenciales" vs. "asesorías online").*
- *Precio (si aplica, que permita filtrar por rangos).*

¿Prefieren recomendaciones personalizadas?
R: *Sí, pero que sean relevantes. Por ejemplo:*

- *Que consideren mis habilidades (si soy diseñador, que me sugieran proyectos de marketing).*
- *Que eviten repetir ofertas que ya rechacé.*

Eventos (TED-like):

¿Qué funcionalidades esperan?
R: *Sería ideal:*

- *Recordatorios automáticos (email/app) una semana y un día antes.*
- *Networking virtual previo al evento (ej: sala de Zoom para participantes).*
- *Material descargable (ej: PDFs con resúmenes de charlas).*

¿Pagarían por acceso premium a eventos?
R: *Depende. Pagaría por:*

- *Eventos con expertos reconocidos (ej: un taller de un CEO de Silicon Valley).*
- *Beneficios como certificados o acceso a grabaciones exclusivas.*

Soporte:

¿Prefieren chat en vivo, email o WhatsApp para soporte?
R: *Chat en vivo (integrado en la app) para problemas urgentes (ej: no puedo unirme a un evento). Para temas complejos, email con respuesta en 24h. ¡Nada de WhatsApp! No quiero dar mi número personal.*

¿Algunas sugerencias ?

- *Perfil poco visible: No hay forma de "destacar" mi perfil (ej: como "Profesional Verificado" o "Top 10 en Diseño").*
- *Problemas con pagos: Si ofrezco servicios premium, quiero retirar mi dinero en 24-48h, no en 15 días.*
- *Falta de protección: Necesito un sistema de reviews justas (que eviten calificaciones falsas por competidores).*

Análisis del sistema monolitico actual

El sistema actual de SkillSwap es limitado. No facilita la escalabilidad y la disponibilidad del servicio a la velocidad de su crecimiento. En un sistema monolítico como el que se usa actualmente no permite implementar nuevas funcionalidades o actualizar ciertos servicios sin que se vea afectado el rendimiento o la disponibilidad del sistema, pues un cambio en alguna parte del software implicaría un shut-down y un deployment del sistema. El aumento de la cantidad de usuarios mensuales también es un problema para un diseño que no es flexible a la hora de pensar en aumentar la capacidad y rendimiento de los componentes de hardware. Un sistema así de limitado, está predispuesto a una alta sensibilidad a los fallos, pues en caso de que falle una porción del sistema, se vería afectado el sistema en su totalidad o al menos en gran parte, y mucho más si la demanda del servicio está creciendo cada vez más por el crecimiento exponencial de nuevos clientes.

El sistema actual monolítico de SkillSwap falla porque no puede seguir el ritmo exponencial de crecimiento de la empresa. Estos problemas no son solo técnicos, pues también afecta a innovación, la agilidad de negocio de la empresa y la experiencia del usuario. Si bien el sistema monolítico actual pudo haber sido una ventaja en los inicios del desarrollo del software, ahora se requiere migrar a un sistema diferente que soporte la complejidad de un sistema en crecimiento con potencial de crecimiento tecnológico y de usuarios

Resumen del análisis:

- Escalabilidad limitada.
- Poca disponibilidad y alta sensibilidad a fallos.
- Tiempos de respuesta lentos.
- Dificultad para mantenimientos y despliegues.
- Demasiada dependencia entre módulos.
- Dificultad para implementar nuevos requerimientos.

Diseño de la nueva Arquitectura

La arquitectura propuesta para SkillSwap es la arquitectura EVENT-DRIVEN. Esta es una arquitectura basada en eventos, un modelo de integración creado alrededor de la publicación, captura, procesamiento y almacenamiento de aplicaciones o servicios, por lo cual es una excelente opción para un servicio como SkillSwap y su solución de tipo web.

La propuesta de microservicios para la nueva arquitectura del sistema de Skill Swap tiene como objetivo abordar los problemas de escalabilidad, rendimiento y disponibilidad identificados en el sistema monolítico actual. Cada microservicio ha sido diseñado considerando la separación de responsabilidades y la independencia de despliegue.

Microservicios propuestos:

1. Auth Service – Gestión de Autenticación y Usuarios

Responsable de la autenticación, registro y administración de cuentas de usuario.
Incluye la asignación de roles como Administrador, Gerente, Encargado y Usuario.

- Autenticación mediante OAuth 2.0 con JWT.
- Registro, edición y eliminación de cuentas de usuario.
- Asignación y gestión de roles de acceso.
- Base de datos: PostgreSQL.

2. Admin Service – Administración de Plataforma

Permite a los administradores gestionar y supervisar el estado general del sistema.

- Monitorización del sistema y visualización de alertas.
  Funcionalidades de respaldo y restauración de datos.
- Almacenamiento de logs y trazabilidad de operaciones.
- Base de datos: MongoDB (orientado a auditoría y registros históricos).
- Depende de: auth-service

3. Ads Service – Gestión de Publicidad y Anuncios

Encargado de la creación y administración del inventario de anuncios publicitarios.

- Creación, edición y eliminación de campañas publicitarias.
- Configuración de visibilidad por secciones: publicaciones, historias, perfiles, etc.
- Generación de reportes de rendimiento y ventas.
- Base de datos: PostgreSQL.
- Depende de: auth-service, reports-service

4. Events Service – Coordinación de Eventos y Núcleos Locales

Gestiona la planificación y organización de eventos tanto presenciales como virtuales.

- Configuración de eventos en espacios físicos (plazas, cafés, seminarios) o en línea.
- Integración con Google Calendar y plataformas externas de eventos (como Ticketmaster).
- Control de participantes y permisos asociados.
- Base de datos: MongoDB.
- Depende de: auth-service

5. Swap Service – Gestión de Intercambios de Habilidades

Administra las operaciones relacionadas con el intercambio de habilidades.

- Procesamiento de transacciones de intercambio (skill swap).
- Manejo de devoluciones y reclamos.
- Invitación de profesionales y coordinación de disponibilidad.
- Base de datos: PostgreSQL.
- Depende de: auth-service, inventory-service

6. Reports Service – Analítica y Reportes

Genera métricas de rendimiento y reportes estadísticos sobre el uso de la plataforma.

- Análisis de actividad y ventas publicitarias.
- Evaluación del comportamiento del usuario y tendencias de uso.
- Visualización de datos para la toma de decisiones.
- Base de datos: Elasticsearch.
- Depende de: eventos de todos los servicios

7. Notification Service – Sistema de Notificaciones en Tiempo Real

Gestiona la entrega de notificaciones y alertas a usuarios a través de múltiples canales.

- Notificaciones de eventos relevantes: coincidencias de habilidades, mensajes, recordatorios.
- Integración con Kafka para procesamiento en tiempo real.
- Métodos de envío: WebSockets, Firebase Push Notifications y correo electrónico.
- Base de datos: Redis.
- Depende de: Todos los servicios mediante comunicación Kafka

8. Inventory Service – Gestión de Inventario de Habilidades

Supervisa la oferta y disponibilidad de servicios y habilidades dentro de la plataforma.

- Seguimiento de oferta y demanda de habilidades disponibles.
- Registro actualizado de profesionales habilitados para realizar intercambios.
- Base de datos: PostgreSQL.
- Depende de: auth-service, swap-service

Posibles dependencias externas

- Almacenamiento de archivos (Fotos, PDF’s, etc): AWS S3
- Geolocalización para núcleos/eventos: Google Maps Platform
- Mensajería Interna: WebSockets

La arquitectura propuesta se alinea con los principios de desacoplamiento y escalabilidad de los microservicios, permitiendo a Skill Swap crecer de forma sostenible sin repetir los errores del sistema monolítico anterior.

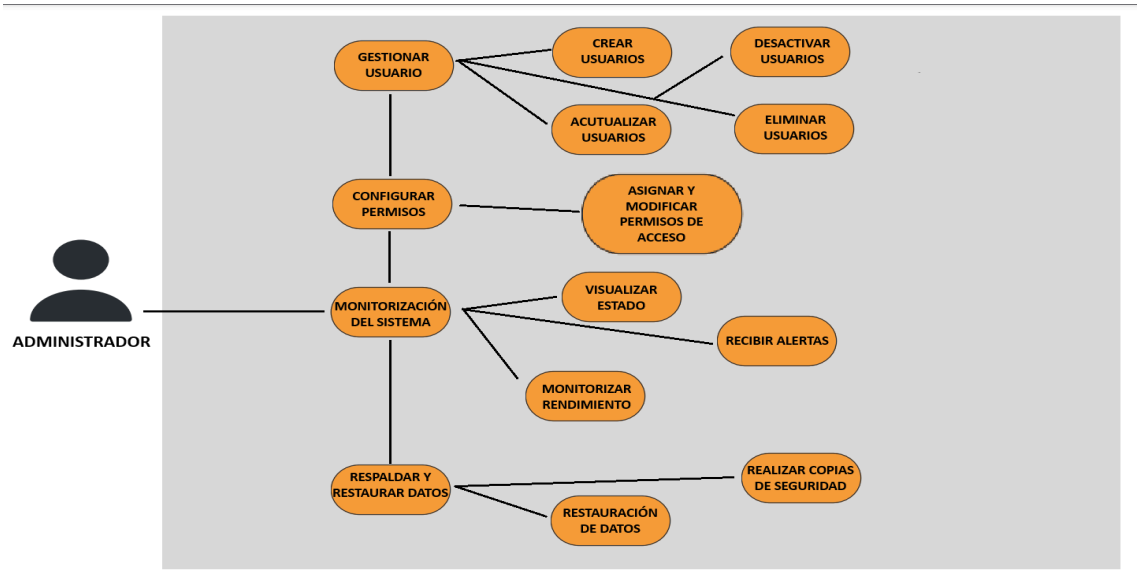
Herramientas a Utilizar

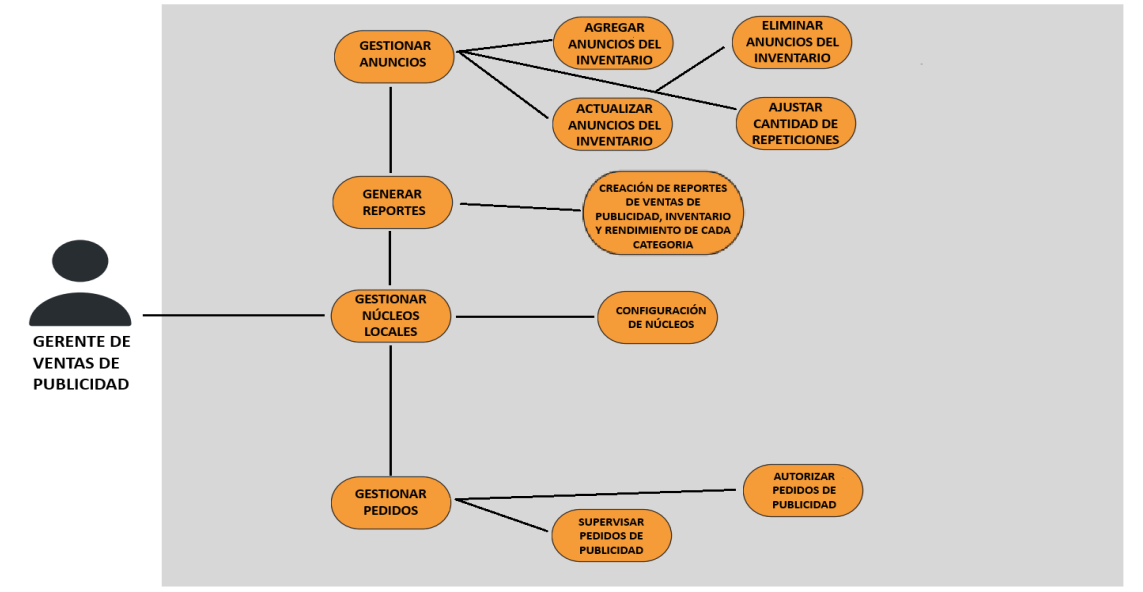
El Stack tecnológico que se emplea para el desarrollo del caso son:

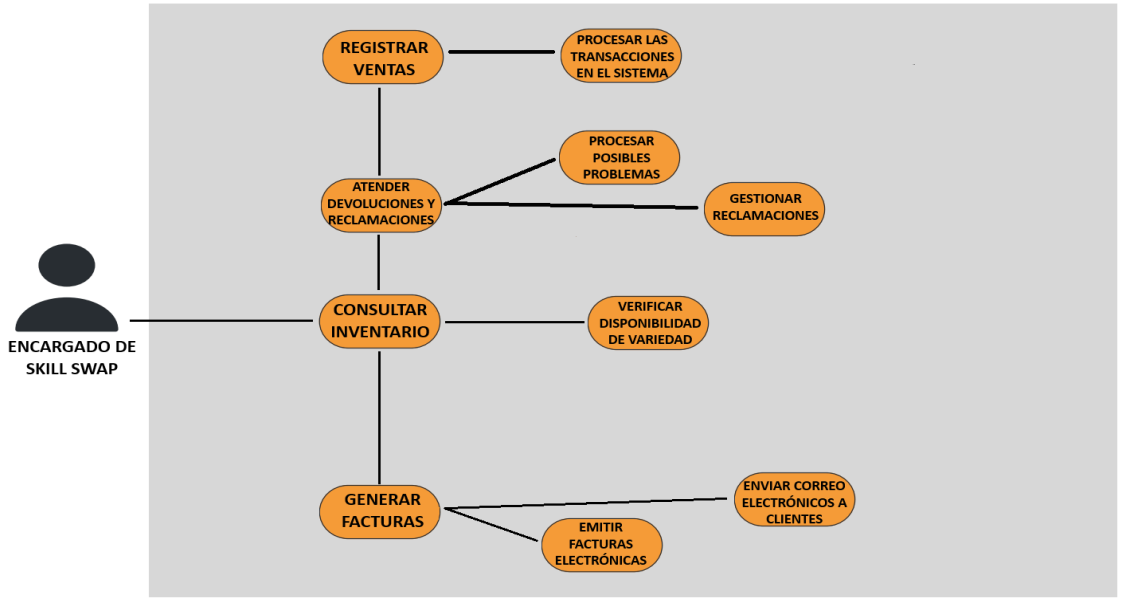
- Lenguajes: Java, Node.js o Python. Estos van a depender según el servicio en el que se emplee.
- Base de Datos: PostgreSQL, MongoDB, Elasticsearch, Redis.
- Comunicación entre servicios: Kafka
- Autenticación: OAuth 2.0 + JWT
- Almacenamiento de logs: MongoDB

Diagramas de casos de uso.

En este apartado se presenta el modelado de los casos de uso que efectuarán los usuarios con el sistema.






ENCARGADO DE SKILL SWAP
REGISTRAR VENTAS
PROCESAR LAS TRANSACCIONES EN EL SISTEMA
ATENDER DEVOLUCIONES Y RECLAMACIONES
PROCESAR POSIBLES PROBLEMAS
GESTIONAR RECLAMACIONES
CONSULTAR INVENTARIO
VERIFICAR DISPONIBILIDAD DE VARIEDAD
GENERAR FACTURAS
ENVIAR CORREO ELECTRÓNICOS A CLIENTES
EMITIR FACTURAS ELECTRÓNICAS

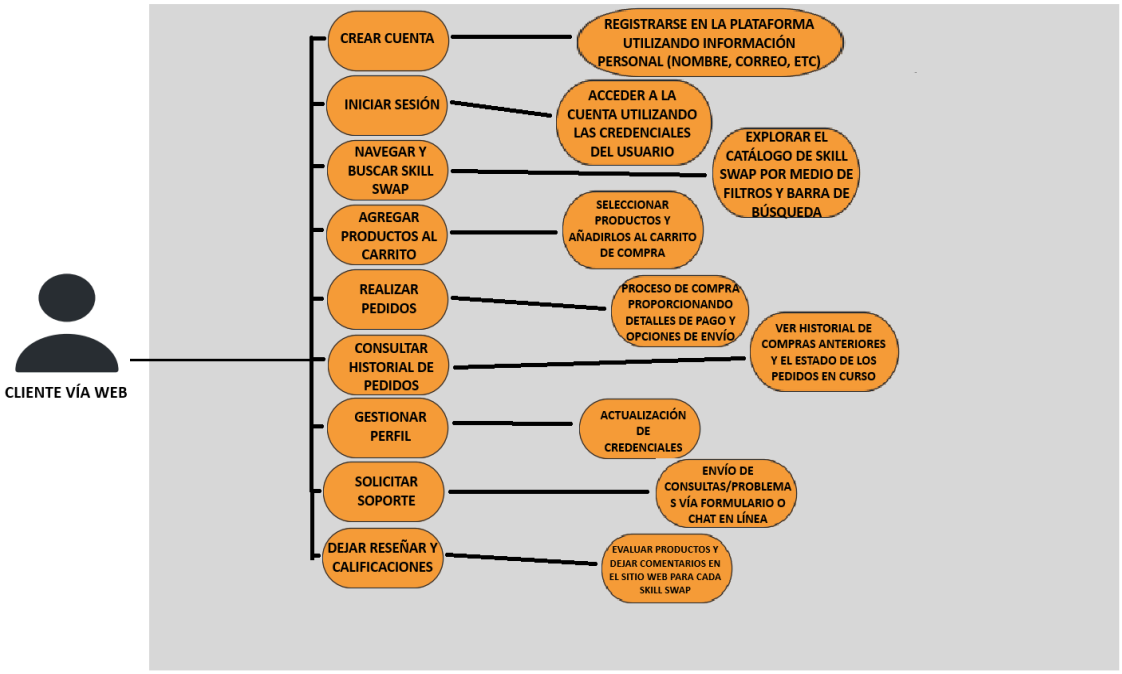

CLIENTE VÍA WEB
CREAR CUENTA
REGISTRARSE EN LA PLATAFORMA UTILIZANDO INFORMACIÓN PERSONAL (NOMBRE, CORREO, ETC)
INICIAR SESIÓN
ACCEDER A LA CUENTA UTILIZANDO LAS CREDENCIALES DEL USUARIO
NAVEGAR Y BUSCAR SKILL SWAP
EXPLORAR EL CATÁLOGO DE SKILL SWAP POR MEDIO DE FILTROS Y BARRA DE BÚSQUEDA
AGREGAR PRODUCTOS AL CARRITO
SELECCIONAR PRODUCTOS Y AÑADIRLOS AL CARRITO DE COMPRA
REALIZAR PEDIDOS
PROCESO DE COMPRA PROPORCIONANDO DETALLES DE PAGO Y OPCIONES DE ENVÍO
CONSULTAR HISTORIAL DE PEDIDOS
VER HISTORIAL DE COMPRAS ANTERIORES Y EL ESTADO DE LOS PEDIDOS EN CURSO
GESTIONAR PERFIL
ACTUALIZACIÓN DE CREDENCIALES
SOLICITAR SOPORTE
ENVÍO DE CONSULTAS/PROBLEMAS VÍA FORMULARIO O CHAT EN LÍNEA
DEJAR RESEÑAR Y CALIFICACIONES
EVALUAR PRODUCTOS Y DEJAR COMENTARIOS EN EL SITIO WEB PARA CADA SKILL SWAP

Diagrama simplificado de los casos de uso según usuario:

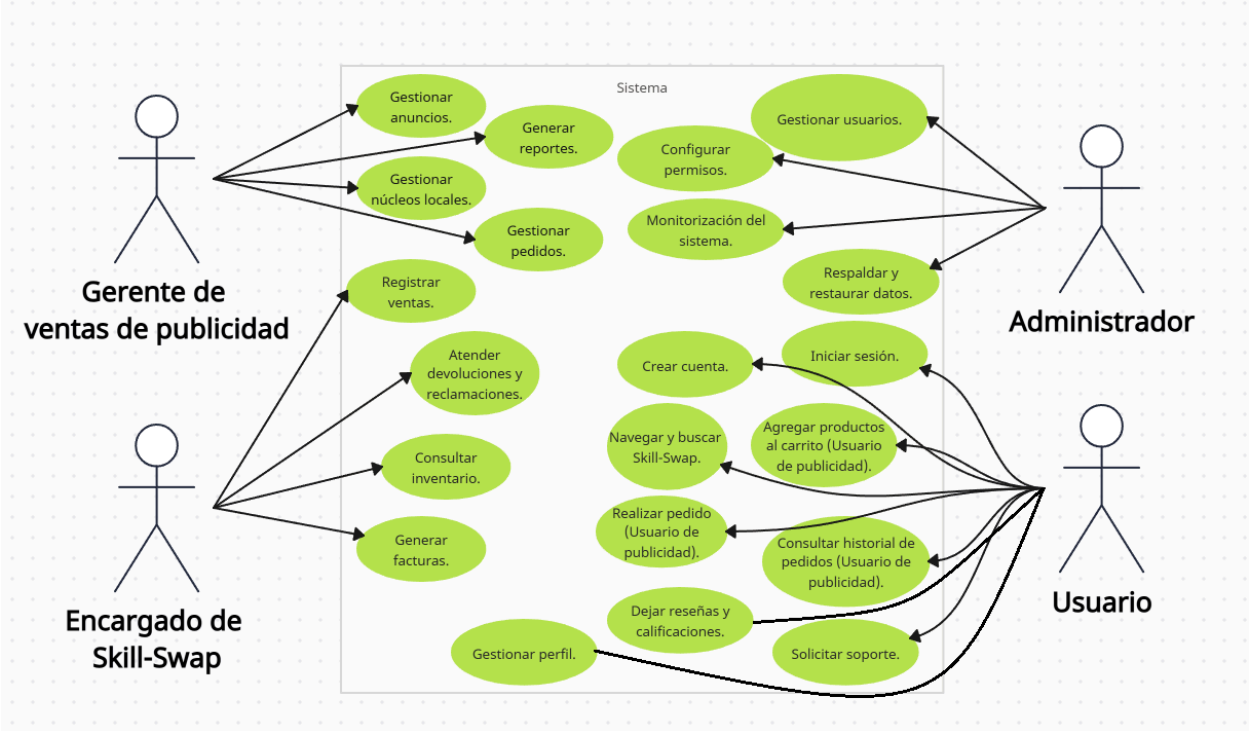
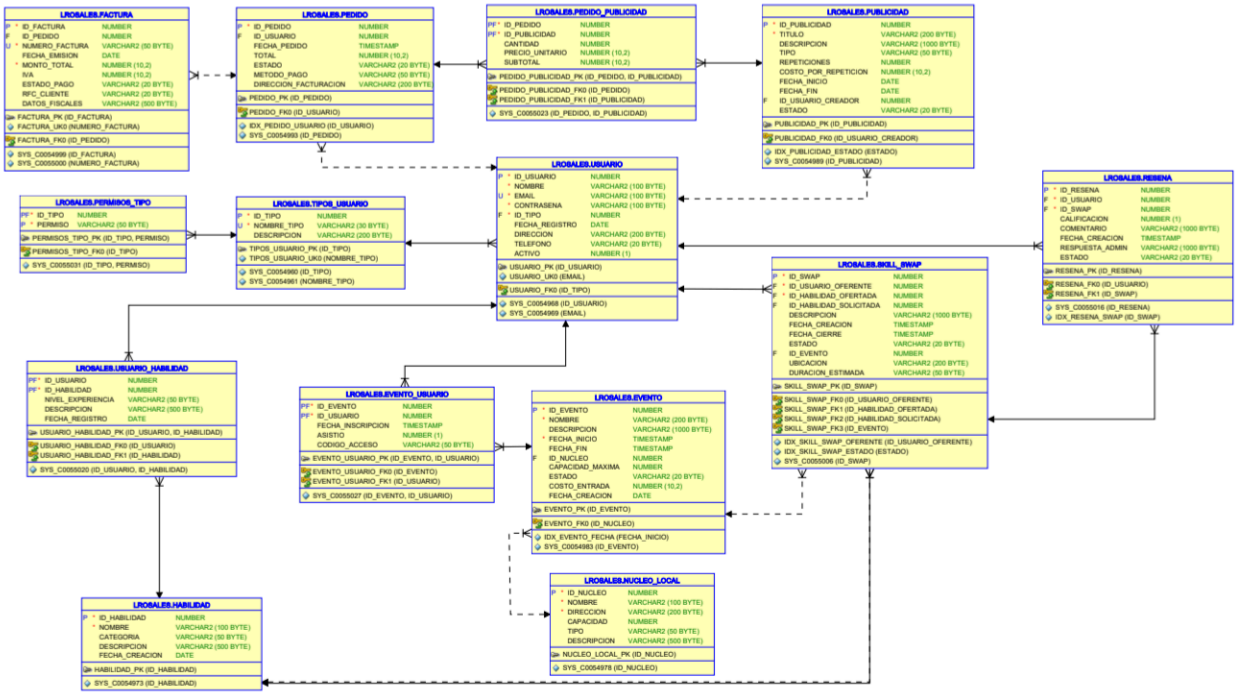


Diagrama MER

El diagrama de clases en este caso sirve para visualizar la estructura de las tablas de la base de datos (como USUARIOS, ANUNCIOS, EVENTOS) y cómo se relacionan entre sí, mostrando las columnas clave (IDs, nombres, fechas) y las conexiones (claves foráneas). Esto ayuda a entender cómo se almacenan y organizan los datos en el sistema, facilitando el diseño de los microservicios y asegurando que cada uno acceda a la información correcta de manera eficiente.



Diagramas de cada caso y Nueva Arquitectura

Esta arquitectura distribuye la funcionalidad en servicios independientes y especializados, permitiendo una mayor escalabilidad, mantenibilidad y resiliencia del sistema. Cada microservicio está enfocado en un conjunto específico de responsabilidades —como autenticación, gestión de anuncios, eventos, intercambios de habilidades y notificaciones— y se comunica de manera eficiente mediante APIs. Este diseño modular no solo mejora el rendimiento y disponibilidad del sistema, sino que también facilita su evolución y adaptación frente al crecimiento continuo de la empresa.

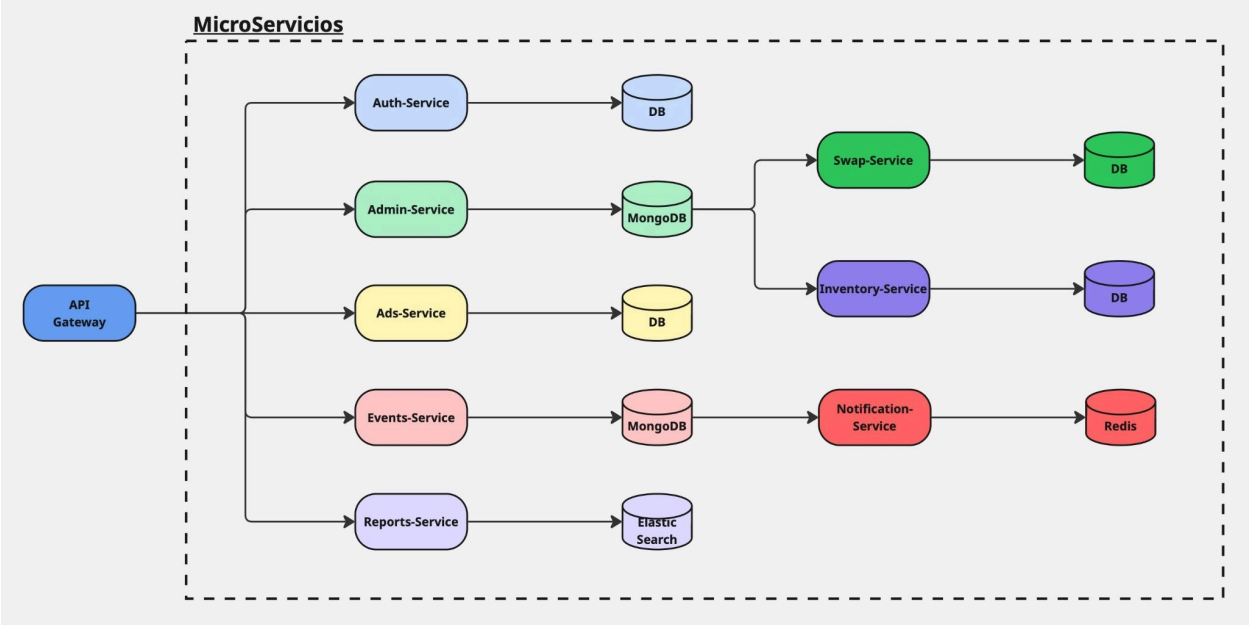
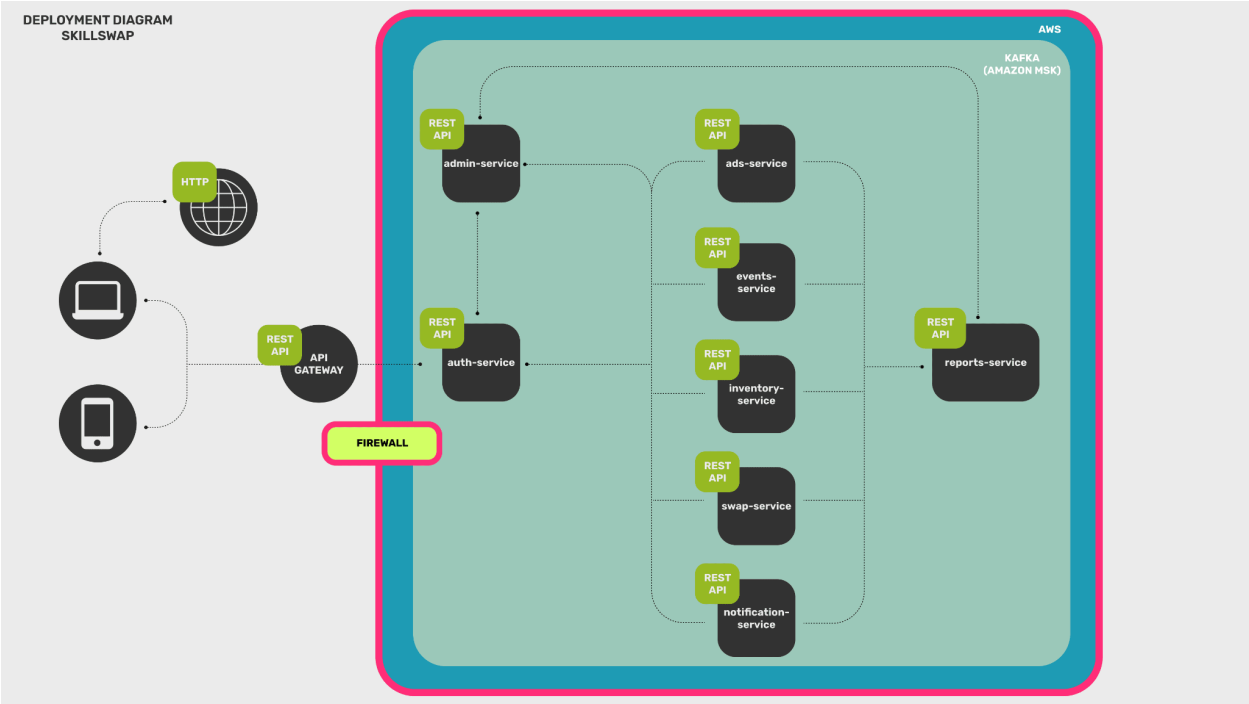


Diagrama de despliegue

El siguiente diagrama muestra la forma en que se desplegaría la solución de tipo web de Skill Swap. Esta se llevaría a cabo en un servicio de la nube como AWS, utilizando Apache Kafka (Integrado en AWS) para comunicarse entre los diferentes servicios. Todo esto por supuesto con un API Gateway para garantizar la seguridad de los datos y credenciales de empresa, además de un firewall interno de AWS para evitar posibles ataques o interacciones maliciosas.



Planificación de la Migración

Con el fin de resolver las limitaciones actuales del sistema monolítico de Skill Swap —principalmente ligadas a la escalabilidad, disponibilidad y mantenibilidad— se propone una migración progresiva hacia una arquitectura basada en microservicios. Este enfoque busca minimizar el impacto sobre los usuarios, garantizando una alta disponibilidad del sistema durante todo el proceso (con tiempos de inactividad menores a un minuto por servicio), mediante la implementación de técnicas y estrategias reconocidas en la industria del desarrollo de software.

1. Estrategia de Migración

La migración se realizará utilizando el enfoque Strangler Pattern, el cual permite reemplazar gradualmente funcionalidades del sistema legado, redirigiendo el tráfico hacia nuevos servicios desacoplados sin necesidad de una interrupción total. Se emplearán además técnicas de Canary Deployment y Blue-Green Deployment, que permiten desplegar versiones de prueba controladas antes de un cambio total, disminuyendo así el riesgo de fallas en la producción.

Asimismo, se utilizará un API Gateway como punto de entrada unificado para el sistema, lo que permitirá que tanto los servicios nuevos como los del sistema monolítico continúen operando de forma conjunta mientras se realiza la transición.

2. Cronograma de Migración

La migración está planificada para ejecutarse en ocho fases distribuidas en 16 semanas, permitiendo así un desarrollo, prueba e integración controlada de cada nuevo servicio. A continuación se detallan las etapas propuestas:

Fase	Semanas	Actividad	Descripción
1	1-2	Evaluación del Sistema Actual	Mapeo de funcionalidades, dependencias críticas y componentes críticos
2	3	Diseño de Arquitectura	Definir servicios, sus responsabilidades, APIs, almacenamiento y canales de comunicación (mensajería, REST, etc)
3	4-5	Migración del Auth-Service	Extraer el sistema de autenticación y roles a un microservicio propio. Usar JWT + OAuth2. Hacer pruebas A/B
4	6-7	Migración del Swap-Service e Inventory-Service	Extracción de la lógica de intercambios de habilidades y disponibilidad profesional
5	8-9	Migración del Ads-Service y Notification-Service	Separación de los módulos de anuncios y sistema de notificaciones en tiempo real
6	10-11	Implementación del Reports-Service	Creación del servicio de reportes y métricas con Elasticsearch
7	12-13	Implementación del Event-Service	Integración con plataformas externas como TicketMaster y Google Calendar
8	14-16	Pruebas finales y desactivación de la arquitectura monolítica	Validación completa, monitoreo, documentación y desconexión progresiva del sistema legado

3. Mecanismos de Aseguramiento y Mitigación

Durante la migración, se implementarán mecanismos de seguridad, monitoreo y recuperación ante fallos:

- Monitoreo activo mediante herramientas como Grafana y Prometheus para el estado de los microservicios.
- Logs distribuidos y trazabilidad con herramientas como Jaeger.
- Pruebas automatizadas de integración y regresión para cada módulo antes del despliegue.
- Plan de rollback automatizado ante fallos en producción.
- Comunicación constante entre equipos de desarrollo, QA y operaciones, facilitada con tableros colaborativos (Trello, Miro, etc.).

4. Consideraciones Técnicas

La migración se basará en el uso de contenedores Docker y su orquestación a través de Kubernetes, lo que permitirá una escalabilidad horizontal sencilla y despliegues rápidos. Asimismo, los servicios se comunicarán a través de REST APIs y mensajería asincrónica mediante Kafka, permitiendo una arquitectura desacoplada y resiliente.

Implementación

En esta nueva etapa del proyecto, se nos solicitó desarrollar tres microservicios independientes que conforman la arquitectura principal de SkillSwap, los que son: Usuarios, Inventario y Pedidos. Cada uno de estos microservicios fue construido con su propia base de datos y logica de negocio, seguimos el principio de separación de responsabilidades, lo que facilita la escalabilidad, el mantenimiento y la evolución independiente de cada componente.

El microservicio de **Usuarios** gestiona el registro, autenticación y administración de los datos personales de los usuarios de la plataforma, incluyendo su nombre, correo, dirección, teléfono y fecha de registro. El microservicio de **Eventos** permite la creación y gestión de eventos de intercambio de habilidades, donde los usuarios pueden ofrecer o asistir a actividades en función de sus intereses y habilidades. Finalmente, el microservicio de **Pedidos** actúa como un carrito de participación: los usuarios pueden inscribirse en eventos disponibles y generar un pedido que vincula a un usuario con un evento específico.

Para lograr una integración eficiente, se implementó comunicación entre microservicios utilizando APIs REST. El microservicio de **pedidos** consulta los datos del usuario y del evento en tiempo real, garantizando que los pedidos se creen únicamente si el usuario existe y el evento está disponible. Esta comunicación fluida permite mantener la consistencia de la información y asegurar que cada interacción sea válida.

Con este enfoque de implementación basado en microservicios se nos permitió construir una plataforma modular, donde los servicios funcionan de forma independiente pero coordinada, lo que no solo mejora el rendimiento y la organización interna del sistema, sino que también permite escalar funcionalidades específicas sin afectar al resto de la aplicación.

1. Base de Datos

La base de datos del sistema SkillSwap está diseñada bajo un enfoque de microservicios, donde cada microservicio gestiona su propio esquema de datos, así garantizamos la independencia, escalabilidad y un mejor mantenimiento. Cada microservicio dispone de tablas específicas que reflejan su dominio funcional: usuarios, eventos y pedidos.

Tabla Usuario

La tabla **Usuario** almacena toda la información personal y de contacto de las personas registradas en la plataforma. Esto incluye datos fundamentales como el nombre, email y teléfono, que son campos únicos para identificar y comunicarse con cada usuario. La contraseña se almacena con seguridad (idealmente cifrada) para proteger la privacidad y la autenticación. Además, se registra la fecha en que el usuario se inscribió en el sistema, y su dirección física, que puede ser usada para facturación o envíos. Esta tabla es esencial para gestionar la identidad y el acceso de los usuarios, y es referenciada desde otros dominios como pedidos para vincular actividades específicas a cada persona.

Tabla Evento

La tabla **Evento** registra los eventos disponibles en la plataforma para intercambio o participación. Cada evento está definido con atributos detallados como nombre único, descripción, fechas de inicio y fin, capacidad máxima y costo de entrada. La fecha de creación permite llevar un control histórico sobre cuándo fue agregado el evento al sistema. Esta tabla es fundamental para organizar las actividades y servicios que los usuarios pueden solicitar o en los que pueden participar, facilitando la gestión de la oferta y control de la disponibilidad.

Tabla Pedido

La tabla **Pedido** actúa como un puente entre los usuarios y los eventos, almacenando las solicitudes de inscripción o compra que los usuarios realizan. Cada pedido está vinculado mediante claves foráneas a un usuario y a un evento específico, lo que permite mantener la trazabilidad de quién hizo qué solicitud y para qué evento. Además, se registra la fecha del pedido, el total pagado, el método de pago utilizado y la dirección de facturación, datos necesarios para el procesamiento financiero y logístico. Esta tabla es crucial para la operación de la plataforma, ya que representa el flujo de intercambio y las transacciones entre los usuarios y los servicios ofrecidos.

Relaciones y Llaves Foráneas

Las relaciones implementadas aseguran la integridad referencial:

- Un **usuario** puede tener múltiples **pedidos**, pero cada pedido pertenece a un solo usuario.

- Un **evento** puede tener múltiples **pedidos**, permitiendo así que varios usuarios se inscriban o participen en el mismo evento.

Mediante el uso de claves foráneas (usuario_id en Pedido hacia Usuario.id y evento_id en Pedido hacía Evento.id_evento), el sistema garantiza que no existan pedidos huérfanos, es decir, pedidos que estén sin un usuario o algún evento válido asociado.

Ventajas de esta estructura

Esta estructura modular nos permite que cada microservicio opere con su propia base de datos, reduciendo acoplamientos y facilitando la escalabilidad. Por ejemplo, la tabla de usuarios puede crecer o cambiar sin afectar el esquema de eventos o pedidos. Asimismo, la integridad y consistencia se mantienen a través de las relaciones bien definidas y el control de las llaves foráneas, lo que evita inconsistencias y errores en el manejo de datos. Esto también facilita las consultas y operaciones de mantenimiento, al tiempo que mejora la seguridad al restringir el acceso a datos sensibles sólo al microservicio correspondiente.

2. Microservicios

El sistema SkillSwap está compuesto por tres microservicios independientes, cada uno responsable de un dominio específico dentro de la plataforma. Esta arquitectura permite que cada servicio sea desarrollado, desplegado y escalado de manera autónoma, mejorando la flexibilidad, mantenibilidad y rendimiento del sistema en conjunto.

Microservicio de Usuarios

Este microservicio gestiona toda la información y lógica relacionada con los usuarios registrados en la plataforma. Incluye funcionalidades para registrar nuevos usuarios, actualizar sus datos personales, autenticar accesos y gestionar la seguridad, como el cifrado de contraseñas. Además, expone una API REST que permite consultar usuarios, buscar por nombre o email, y eliminar cuentas si es necesario. Al operar con su propia base de datos, garantiza la privacidad y seguridad de la información personal, manteniendo el control completo sobre los datos sensibles y validaciones de acceso. Es la base para la identificación y autorización dentro del ecosistema SkillSwap.

Microservicio de Eventos

Este servicio se encarga de gestionar el catálogo de eventos disponibles en la plataforma. Cada evento posee atributos detallados como nombre, descripción, fechas de inicio y fin, capacidad máxima y costo de entrada. El microservicio ofrece operaciones para crear, modificar, eliminar y consultar eventos, facilitando la organización y control de la oferta de actividades. Al mantener un repositorio propio, puede controlar la disponibilidad y estado de cada evento, permitiendo que los usuarios exploren las opciones disponibles y participen en ellas. Además, el microservicio válida la unicidad del nombre del evento y asegura que los datos sean consistentes.

Microservicio de Pedidos

El microservicio de pedidos funciona como el intermediario que conecta a los usuarios con los eventos en los que desean participar. Su responsabilidad principal es gestionar las solicitudes de inscripción o compra que realizan los usuarios para los eventos. Esto incluye la creación de pedidos, actualización del estado, manejo de información financiera como total pagado y método de pago, y la gestión de la dirección de facturación. Este servicio expone APIs para listar pedidos por usuario o evento, eliminar pedidos y modificar estados según el proceso de confirmación o cancelación. Su independencia permite escalar según la demanda y mantener un control detallado sobre las transacciones realizadas en la plataforma.

Integración y Comunicación

Aunque cada microservicio tiene su propia base de datos y lógica de negocio, están diseñados para interactuar de manera coordinada mediante APIs REST bien definidas. Por ejemplo, cuando un usuario realiza un pedido para un evento, el microservicio de pedidos consulta y valida la existencia del usuario y del evento a través de las APIs de los otros servicios, asegurando que sólo se procesen pedidos válidos. Esta comunicación desacoplada permite mantener la autonomía de cada servicio mientras se garantiza una experiencia integrada para el usuario final.

Beneficios de la arquitectura de microservicios

Esta estructura nos facilita el desarrollo en paralelo por equipos especializados en cada dominio, reduce el impacto de errores o fallos aislándolos en un único servicio, y mejora la escalabilidad al poder ajustar recursos específicamente donde se requiera. Además, favorece el despliegue continuo y la actualización incremental, permitiendo introducir mejoras o correcciones en un microservicio sin afectar al resto del sistema.

Etica del proyecto

El enfoque ético es fundamental en Skill Swap porque garantiza una plataforma confiable, sostenible y beneficiosa para sus usuarios. Al priorizar la transparencia, la privacidad y la equidad, construimos confianza entre los profesionales que intercambian habilidades, evitando prácticas manipulativas como algoritmos adictivos o venta oculta de datos. Esto no solo cumple con regulaciones como el GDPR y evita sanciones legales, sino que también diferencia a Skill Swap de otras redes sociales que sacrifican el bienestar digital por engagement vacío. Al rechazar sesgos algorítmicos, notificaciones intrusivas y desinformación, la plataforma se posiciona como un espacio seguro para el aprendizaje genuino, donde los usuarios tienen control sobre su experiencia.

Además, la ética asegura el crecimiento sostenible de Skill Swap. En un mundo donde la desconfianza hacia las plataformas digitales aumenta, comprometernos con un diseño responsable atrae a usuarios y socios que valoran la transparencia. Evitar la explotación de datos y la polarización no es solo un deber moral, sino una ventaja competitiva que fortalece la reputación a largo plazo. Skill Swap no quiere ser una red social más, sino una herramienta que empodera a las personas a través de intercambios justos, seguros y significativos, donde la tecnología sirve a las personas, no al revés.

Por lo cual a continuación se definen cinco puntos importantes sobre nuestro enfoque ético:

1. No Manipulación

Skill Swap evitará diseñar algoritmos que fomenten engagement tóxico, priorizando en cambio recomendaciones basadas en utilidad real: el sistema de matching conectará usuarios por compatibilidad de habilidades verificadas (no por popularidad o polémica), eliminará notificaciones engañosas ("¡10 personas te esperan!") y usará métricas de calidad (ej: porcentaje de intercambios exitosos) en lugar de clics vacíos.

2. Privacidad Real

La plataforma recolectará solo datos estrictamente necesarios (ej: no exigirá ubicación exacta para intercambios online) y garantizará transparencia total: los anunciantes podrán segmentar por intereses declarados (no por rastreo oculto), los usuarios tendrán control sobre qué información comparten, y se prohibirá la venta de datos a terceros, almacenando todo con encriptación AES-256.

3. Transparencia

Cada recomendación incluirá un botón "¿Por qué veo esto?" que explicará en lenguaje simple los criterios del algoritmo (ej: "Te sugerimos a Carlos por tu búsqueda de mentoría en Python"), junto con un panel donde los usuarios puedan ajustar preferencias, borrar historial o desactivar tracking, asegurando que comprendan y controlen su experiencia en la plataforma.

4. Equidad

El sistema auditará periódicamente sus recomendaciones para eliminar sesgos: no asociará habilidades con género/edad (ej: evitar estereotipos como "diseño=femenino"), mostrará diversidad en resultados de búsqueda, y usará herramientas como IBM Fairness 360 para detectar discriminación algorítmica, garantizando oportunidades equitativas para todos los perfiles.

5. Derecho a Desconectar

Skill Swap limitará notificaciones intrusivas (máximo 1/hora configurable) e incluirá un "Modo Enfoque" que silenciará alertas durante intercambios activos; además, evitará diseños adictivos (ej: scroll infinito) y recordatorios constantes, respetando tiempos de descanso y concentración de los usuarios.

Ejemplo Práctico Integrado

Cuando un usuario busque "clases de programación", el algoritmo priorizará profesionales con habilidades verificadas (no perfiles polémicos), explicará la recomendación ("Basado en tu interés en Java"), ofrecerá opciones diversas (género/edad/ubicación), y permitirá pausar notificaciones durante la clase, todo sin rastrear ubicación precisa ni vender sus datos.

Riesgos y plan de mitigación

Riesgos Técnicos

- Integración fallida de monolito a microservicios
  - Riesgo

    Puede ocurrir que durante, la transición de arquitectura monolítica a una arquitectura event-driven, se generen errores de convivencia entre ambas arquitecturas ya que la transición será progresiva.
  - Mitigación

    Pruebas de integración tempranas. La labor de la API Gateway es primordial para diseñar rutas definidas y que puedan ser monitoreadas. Usar feature flags para activar/desactivar.

- Fallas en despliegues de servicios nuevos
  - Riesgo:

    El despliegue de un microservicio podría provocar interrupciones inesperadas
  - Mitigación:

    Usar CodeDeploy para despliegues seguros de microservicios.

- Problemas de compatibilidad de datos
  - Riesgo

    Usar diferentes bases de datos para diferentes microservicios puede causar inconsistencias en el servicio completo si no se sincroniza bien.
  - Mitigación

    La integración de Kafka para mantener la coherencia eventual. Analizar dependencias antes de extraer cada servicio.

- Complejidad de comunicación entre servicios
  - Riesgo

    El cambio notable entre la comunicación directa de la arquitectura monolítica a un sistema REST o de mensajería puede generar latencia o errores en la interacción entre servicios.
  - Mitigación

    Definir una estrategia para la comunicación REST para operaciones síncronas y Kafka para asincrónicas. Documentación detallada y almacenada.

Riesgos organizacionales

- Falta de alineación entre equipos

- Riesgo

  Al dividir el sistema en servicios, los equipos deben coordinarse en equipos y disponer de buena comunicación para aquellos cambios que son compartidos.
- Mitigación

  Establecer reuniones semanales de sincronización. Usar tableros colaborativos y herramientas de seguimiento con JIRA.

- Resistencia al cambio
  - Riesgo

    Parte del equipo puede resistirse a ciertas herramientas o paradigmas como por ejemplo AWS o Apache Kafka.
  - Mitigación

    Capacitar progresivamente al equipo. Dar apoyo en cada etapa y proceso de desarrollo para que se estandarice el uso de las herramientas y los paradigmas.

Riesgos de Seguridad

- Gestión débil de autenticación entre servicios
  - Riesgo

    Podrían existir vulneraciones entre servicios si no hay un correcto manejo de tokens y credenciales
  - Mitigación

    Centralizar la autorización con OAuth 2.0 y JW. Asegurar la comunicación interna con HTTPS y tokens firmados.

- Exposición de servicios internos por error
  - Riesgo

    Si un microservicio queda expuesto sin filtros por algún motivo o externo, puede ser atacado
  - Mitigación

    Controlar eficientemente la API Gateway a nivel de trafico completo. Implementar controles de acceso a nivel del Gateway y servicio para evitar la interacción no deseada.

Riesgos de rendimiento y disponibilidad

- Latencia elevada
  - Riesgo

Se puede perder eficiencia y aumentar la latencia al separar los servicios si no se gestiona la comunicación de manera óptima.

- Mitigación

  Usar caching, balanceadores y colas asincrónicas para evitar latencia. Medir constantemente los tiempos de repuesta de los servicios y abordarlos.

- Caídas parciales mal manejadas
  - Riesgo

    Si un microservicio falla y existen otros dependiendo de aquel, podría generar un efecto cascada.
  - Mitigación

    Asegurar la tolerancia a fallos. Usar rutas alternas o espacios aislados para evitar los fallos cascada.

Riesgos en la fase de pruebas

- Cobertura de pruebas insuficiente
  - Riesgo

    Las pruebas podrían no detectar ciertos errores e interacción entre servicios.
  - Mitigación

    Plan detallado de la fase de pruebas para no omitir características sin probar. Incluir pruebas unitarias, de integración, de contrato y de regresión automatizadas para asegurar la inclusión de cada aspecto del software en prueba.

- Ambientes de staging mal configurados
  - Riesgo

    Falta de realismo en los entornos de prueba.
  - Mitigación

    Asegurar que los entornos se hagan con réplicas de los entornos de producción para asegurar el reflejo fiel del software.

Conclusión

Como equipo pudimos concluir que el desarrollo del sistema SkillSwap utilizando una arquitectura basada en microservicios representa una solución eficiente y escalable para gestionar una plataforma de intercambio de habilidades y eventos. A partir de los conocimientos adquiridos en la materia que incluyen diseño de bases de datos relacionales, modelado de entidades, establecimiento de relaciones e integridad referencial, así como el desarrollo de microservicios en Spring Boot con APIs REST se logró implementar un sistema modular que garantiza la separación clara de responsabilidades.

Cada microservicio, Usuarios, Eventos y Pedidos cuenta con su propia base de datos, permitiendo una gestión autónoma de los datos y evitando cuellos de botella asociados a arquitecturas monolíticas. La comunicación entre servicios a través de APIs REST asegura la integración necesaria para ofrecer una experiencia coherente al usuario, mientras que la independencia facilita el mantenimiento, la escalabilidad y la evolución continua del sistema.

Además, el proyecto permitió aplicar buenas prácticas de desarrollo de software, como el uso de frameworks modernos (Spring Boot), la implementación de seguridad básica en el manejo de datos sensibles, y el diseño de una arquitectura resiliente y flexible. Esto prepara a los desarrolladores para enfrentar desafíos reales en sistemas distribuidos y aplicaciones empresariales, aportando valor tanto a nivel académico como profesional.

En resumen, el caso SkillSwap es un claro ejemplo de cómo la combinación de bases de datos bien diseñadas con una arquitectura de microservicios puede dar lugar a soluciones robustas, mantenibles y adaptadas a las necesidades dinámicas de las plataformas digitales actuales.