Proyecto Digitalización PetSocity

Integrantes:

Vania Vargas

Alexis Ramirez

Alan Astudillo

Felipe Navarro

Docente: Tomas Ignacio Opazo Toro

Transformación Digital de "Petsocity"

Índice

1. Caso

Petsocity es una empresa chilena emergente dedicada a la venta de productos para mascotas, ofreciendo una amplia gama de artículos de calidad para el bienestar, entretenimiento y cuidado de los animales. Su misión es facilitar la vida de los dueños de mascotas, brindando soluciones prácticas y confiables que aseguren el bienestar y la felicidad de sus compañeros peludos.

Inicialmente, la empresa comenzó con una sucursal en el Barrio Meiggs en Santiago, pero su éxito en ventas tanto al por menor como al por mayor ha llevado a la apertura de nuevas sucursales en Concepción y Viña del Mar. 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.

2. Definición del Problema

Petsocity enfrenta un crecimiento acelerado que ha superado las capacidades de su actual sistema monolítico, provocando fallos, caídas y lentitud en el servicio. Estas deficiencias impactan negativamente en la experiencia del usuario, limitando su expansión nacional.

3. Objetivo general

El objetivo de este caso de estudio es desarrollar una solución al problema que presenta la empresa, pasando por el análisis, diseño, e implementación de una solución tecnológica que permita superar las limitaciones de su sistema actual y soportar su continuo crecimiento. El proyecto se desarrollará en tres partes, a lo largo de las tres evaluaciones parciales del semestre, culminando en una presentación final donde deberá defenderse la solución propuesta.

4. Estrategia de Microservicios a Utilizar

Se optará por una arquitectura basada en microservicios desacoplados, donde cada componente funcional será desarrollado e implementado de forma independiente, comunicándose mediante API REST.

La estrategia se basa en:

- Descomposición por dominio de negocio (DDN): Separar servicios como "shopping cart", "user", "delivery", "product inventory", "payment", entre otros.
- Base de datos por servicio: Cada microservicio tendrá su propia base de datos para evitar el acoplamiento. Sin embargo, servicios relacionados como "wishlist" y "shopping cart" compartirán una base de datos para facilitar la integración de funcionalidades comunes.
- Despliegue independiente: Todos los microservicios están contenidos con Docker. Esta decisión permite:
    - Asegurar la portabilidad del entorno de ejecución.
    - Facilitar el despliegue continuo y la escalabilidad.
    - Aislar fallos de un servicio sin afectar a los demás.
    - Preparar la arquitectura para su futura orquestación con Kubernetes.

5. Herramientas Utilizadas
    - Lenguajes y Frameworks

Herramienta	Función	Aporte al desarrollo
Java + Spring Boot	Desarrollo de microservicios	Facilita la creación de APIs REST robustas y escalables
React	Desarrollo de interfaces de usuario (frontend)	Permite crear interfaces de usuario interactivas y dinámicas con una arquitectura basada en componentes, lo que facilita la reutilización y el mantenimiento del código.
nodejs	Desarrollo de backend (servidores)	Permite crear servidores web eficientes y escalables, usando JavaScript en el backend para mantener una pila unificada en todo el stack.

    - Base de Datos

Herramienta	Función	Aporte al desarrollo
MySQL	Almacenamiento de datos	Soporte a cada microservicio con su base de datos individual

    - Testing y Desarrollo

Herramienta	Función	Aporte al desarrollo
Postman	Testing de APIs	Verificación de endpoints y flujos de datos

Git / GitHub	Control de versiones	Trabajo colaborativo y seguimiento del código fuente
Parsec	Escritorio remoto colaborativo	Permite acceso remoto fluido a otros equipos, ideal para debugging o trabajo conjunto en entornos de desarrollo visuales

6. Herramientas de Trabajo Colaborativo

Herramienta	Función	Aporte al desarrollo
Jira	Gestión de tareas	Organización ágil por sprints, seguimiento de historias de usuario
Miro / Draw.io	Diagramación	Visualización de arquitectura, clases, casos de uso
Teams	Comunicación, charla, reuniones	Para reuniones, charlas y colaboraciones en equipo.

Estas herramientas fortalecieron la comunicación, permitieron un flujo de trabajo ágil y replicaron un entorno colaborativo profesional.

7. Evaluación Ética en la Arquitectura

Tema Ético	Implicancia	Solución Propuesta
Privacidad de Datos	Protección de la información del cliente	Cifrado AES-256 y autenticación de doble factor
Responsabilidad en el Despliegue	Fallas en microservicios pueden afectar servicios dependientes	Pruebas automatizadas y despliegue progresivo

Impacto en el Empleo	Automatización puede reducir necesidades de soporte	Reentrenamiento y reasignación a tareas de mayor valor
Trazabilidad	Necesidad de auditar cambios y accesos	Logs individuales por microservicio y dashboards de monitoreo

Estas consideraciones se integraron directamente en el diseño de la arquitectura, promoviendo una solución tecnológica responsable.

8. Análisis de requerimientos

Identificar y documentar todo los requisitos funcionales y no funcionales del sistema

- Requisitos funcionales:
  i. Administrador de la plataforma de la empresa:

- Gestionar usuarios: Crear, actualizar, desactivar y eliminar cuentas.
- Configurar permisos de acceso con roles personalizados.
- Monitorizar el estado del sistema con alertas en tiempo real.
- Realizar copias de seguridad y automatizar su programación.

  ii. Gerente de sucursal:

- Gestionar inventario con niveles de stock mínimo y máximo.
- Generar reportes avanzados: agregar filtros personalizables para ventas, inventario y desempeño del personal.
- Configurar detalles de sucursal como políticas de devolución específicas.
- Gestionar pedidos con estados detallados (pendiente, en proceso, completado).

  iii. Acciones de los clientes vía web:

- Crear cuentas y acceder mediante autenticación multifactor.
- Buscar productos con recomendaciones basadas en historial de compras.
- Realizar pedidos con opciones de envío flexibles y visualización de tiempo estimado de llegada.
- Consultar historial de compras con detalles financieros.
- Gestionar perfil de usuario con preferencias de notificaciones.
- Solicitar soporte mediante chat en vivo o tickets.
- Dejar reseñas con puntuaciones verificadas.

- Requisitos No Funcionales

- Disponibilidad y rendimiento: Escalabilidad diseñada para soportar un crecimiento del 200% en usuarios en un periodo de 12 meses.
- Seguridad: Cumplir con estándares como ISO 27001 y cifrar datos sensibles con AES-256.
- Usabilidad: Realizar pruebas de accesibilidad (cumplir con WCAG 2.1) y encuestas a usuarios para optimizar interfaces.
- Compatibilidad: Asegurar accesibilidad desde navegadores modernos y dispositivos móviles con optimización responsive.
- Mantenimiento: Implementar actualizaciones automáticas y monitorear el sistema con dashboards en tiempo real.
- Eficiencia: Garantizar tiempos de respuesta de menos de 2 segundos para operaciones clave.

- Levantamiento de Requerimientos a través de Entrevistas con Usuarios Finales

Con el objetivo de comprender en profundidad las necesidades, expectativas y puntos de mejora desde la perspectiva del cliente, se realizaron entrevistas a usuarios reales de la plataforma actual. A continuación, se resumen las principales respuestas obtenidas, agrupadas por tema:

i. Opiniones del Cliente

- Motivación de uso y percepción general
  "Me motiva poder comprar desde la comodidad de mi casa. Lo que más valoro es la variedad de productos, pero la página a veces es lenta y eso podría mejorar."

- Experiencia de navegación y búsqueda
  "La navegación no siempre es intuitiva, especialmente cuando busco productos específicos. Un filtro más detallado y búsquedas inteligentes serían una gran mejora."

- Opciones de pago y envío
  "Las opciones de pago son limitadas. Me gustaría poder usar billeteras digitales como PayPal o transferencias bancarias. También sería útil tener un seguimiento más claro de los envíos."

- Historial de pedidos y personalización del perfil
  "El historial es útil, pero sería aún mejor si incluyera un botón para repetir compras fácilmente. También me gustaría gestionar mis preferencias de notificación directamente en mi perfil."

ii. Requerimientos Identificados (Historias de Usuario)

- Navegación eficiente
  Como cliente, quiero navegar la web de forma rápida y encontrar productos fácilmente con filtros útiles, para ahorrar tiempo en mis compras.
- Diversidad de medios de pago
  Como cliente, quiero tener más opciones de pago como billeteras digitales o transferencias, para pagar como me sea más conveniente.
- Gestión de historial y compras repetidas
  Como cliente, quiero ver mi historial de pedidos actualizado y tener la opción de repetir compras anteriores, para facilitar mis compras más recurrentes.
- Lista de productos favoritos
  Como cliente, quiero guardar productos en una lista de favoritos, para encontrarlos más rápido en futuras visitas.

9. Análisis del Sistema Actual

Sistema actual:

- Basado en una arquitectura monolítica.
- Estructura centralizada, dificultando la escalabilidad.
- Lentitud en el acceso simultáneo a productos o al agendar citas.
- Caídas del sistema por alta carga.
- Imposibilidad de actualizar o desplegar módulos sin afectar a todo el sistema.
- Problemas frecuentes: lentitud, caídas, dificultad para incorporar nuevas funcionalidades.

Causas de fallos:

- Carga excesiva de usuarios.
- Todos los módulos están acoplados; un fallo impacta en todo el sistema.
- Mantenimiento costoso y dependiente de una única base de código.

10. Diseño de la nueva arquitectura

Arquitectura microservicio

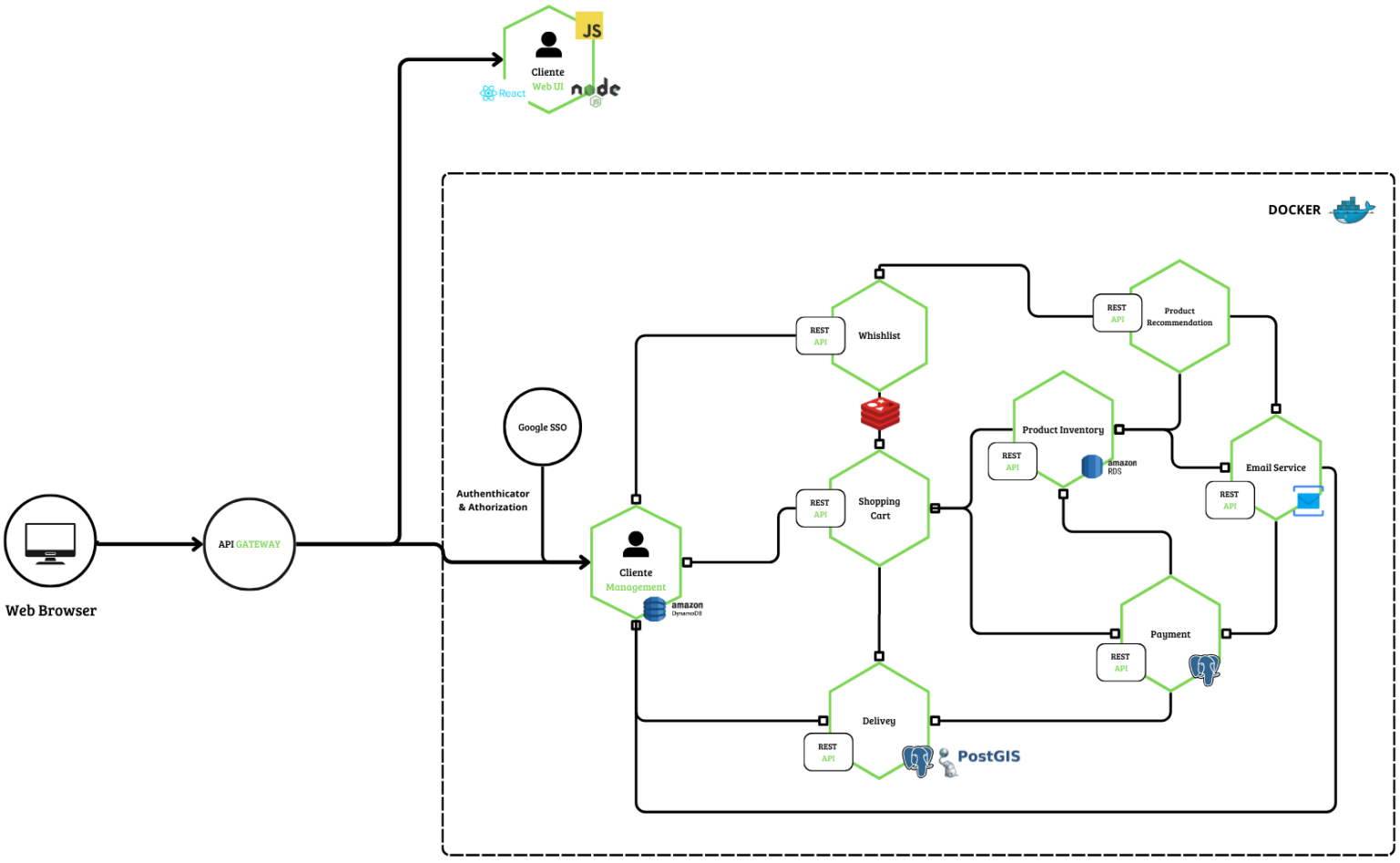


Diagrama de Clases y proyección a futuro

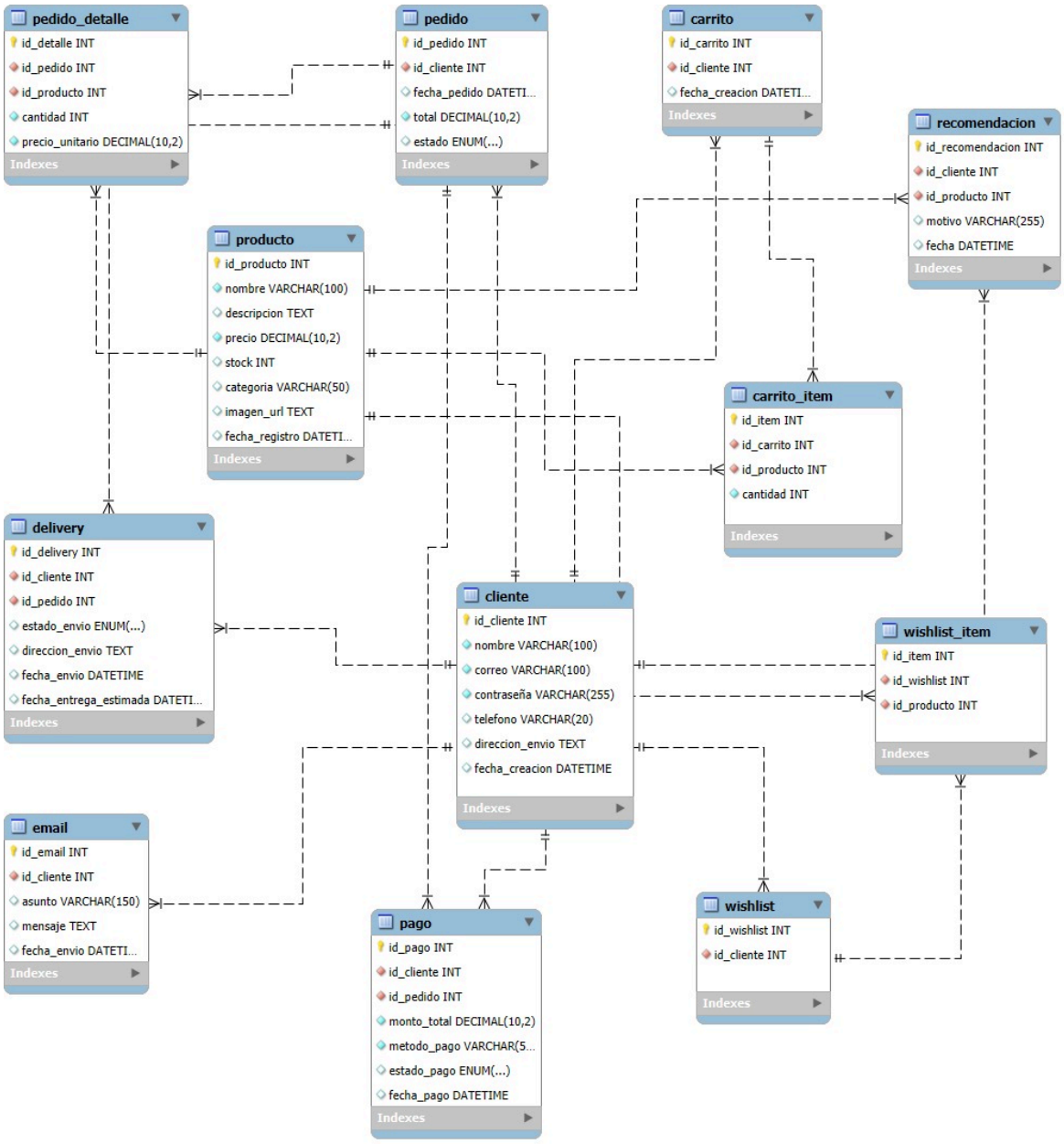
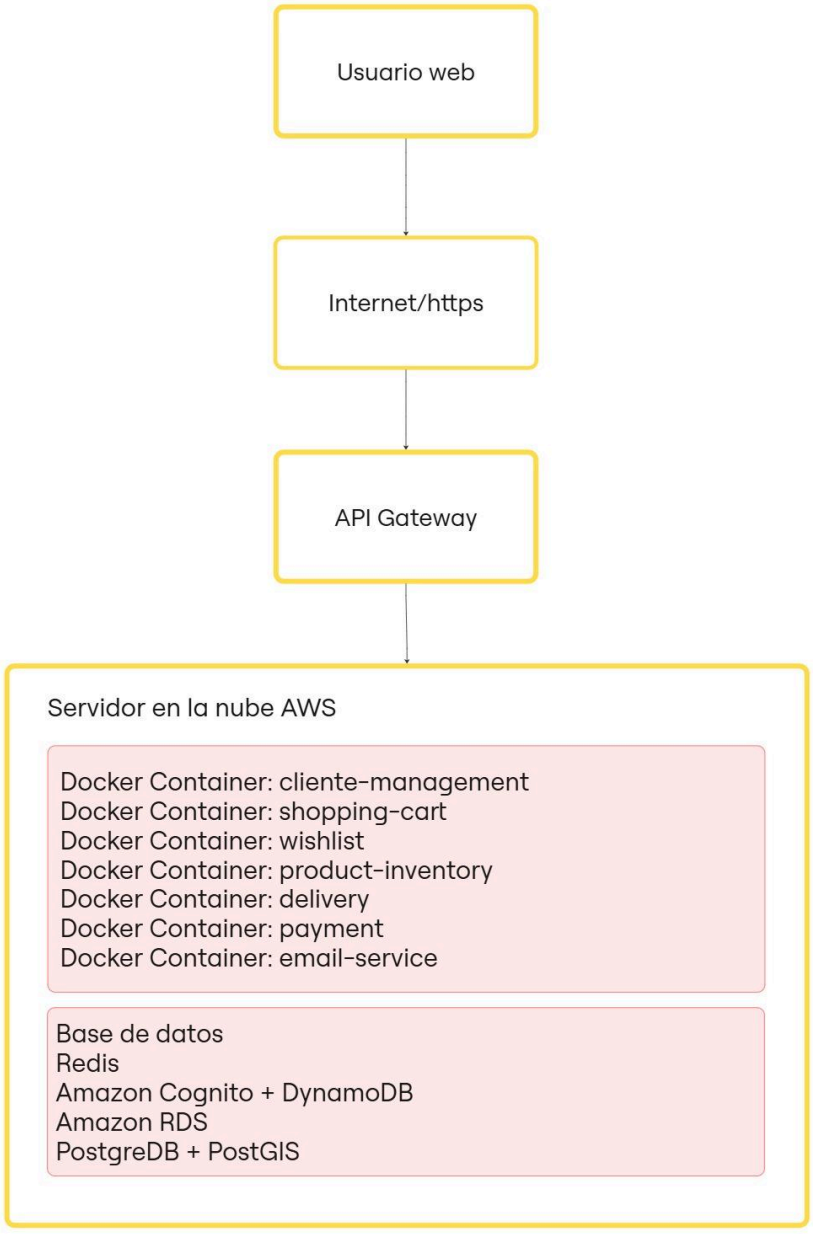


Diagrama de Despliegue



11. Planificación de la migración.
    - Etapas de Migración

- Identificación del sistema actual
  - Analizar y mapear los componentes del sistema monolítico.
- Desarrollo inicial y pruebas piloto
  - Crear el primer microservicio (Login/Usuarios).
  - Realizar pruebas controladas.
- Migración progresiva de funcionalidades
  - Reemplazar gradualmente los módulos del monolito con microservicios equivalentes.

- Validación con usuarios reales
  - Incluir usuarios finales en pruebas para evaluar funcionamiento real.

- Apagado gradual del monolito
  - Redirigir rutas antiguas a los nuevos servicios vía API Gateway.
  - Verificar estabilidad, realizar pruebas de carga y seguridad.
  - Eliminar módulos ya reemplazados.
  - Mantener el monolito como backup temporal.

12. Arquitectura Propuesta de Microservicios
    - Servicio de Autenticación y Usuarios
      i. Registro, login, recuperación de cuenta, edición de perfil.

    - Servicio de Productos y Stock
      i. Catálogo, búsquedas, detalles del producto, control de inventario.

    - Servicio de Carrito de Compras
      i. Agregar/quitar productos, resumen de compra.

    - Servicio de Pagos
      i. Checkout, validación, facturación.

    - Servicio de Pedidos y Envíos
      i. Registro de pedidos, historial, dirección de entrega, seguimiento.

    - Servicio de Lista de Deseos

i. Guardado y gestión de productos favoritos.

- Servicio de Notificaciones
  i. Emails, promociones, confirmaciones automáticas.

- Servicio de Recomendaciones
  i. Sugerencias personalizadas, historial de compras, descuentos automáticos.

13. Herramientas e Infraestructura a Utilizar
    - Frameworks

- Node.js: Para servicios de alto rendimiento como catálogo y usuarios.
- Spring (Java): Ideal para servicios críticos como pagos o autenticación.
- React: Desarrollo rápido y dinámico del frontend.

- Colaboración y Control de Versiones

- GitHub / GitLab: CI/CD, testing automático.
- Parsec: Soporte remoto entre desarrolladores.
- Jira / Teams / Miro: Gestión ágil de tareas y colaboración visual.

- Bases de Datos

- Amazon RDS / PostgreSQL + PostGIS: Datos estructurados y geolocalización.
- Redis: Cache y manejo de sesiones.
- Amazon Cognito + DynamoDB: Gestión de usuarios y autenticación escalable.

14. Riesgos y Mitigación

Riesgo	Mitigación
Incompatibilidad entre servicios	Definición de contratos API y versionamiento
Falta de experiencia del equipo	Capacitación previa y asesoría externa
Interrupciones del servicio	Migración por etapas con entorno de staging para pruebas

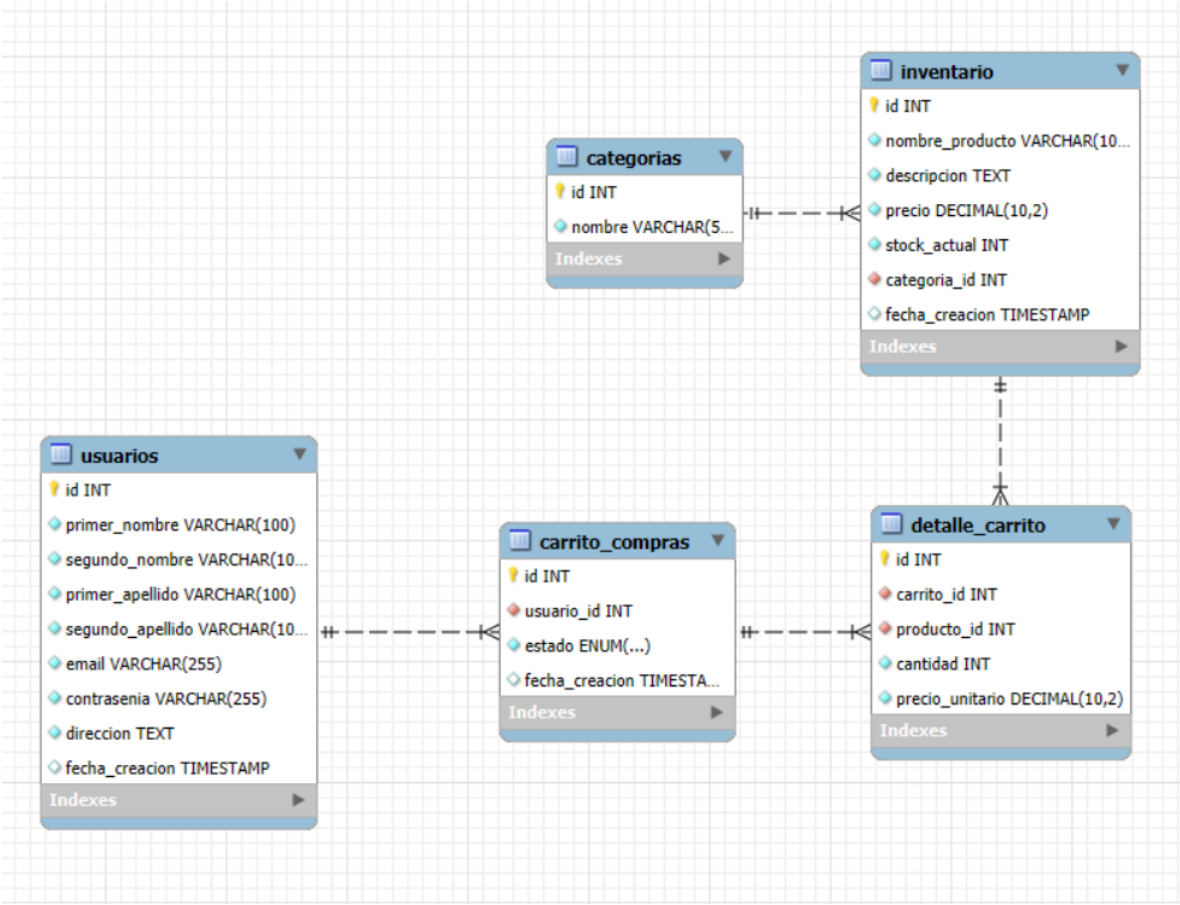
15. Apagado Progresivo del Monolito

- Verificar funcionalidades y estabilidad de cada microservicio.
- Redirigir rutas antiguas al nuevo sistema mediante API Gateway.
- Ejecutar pruebas de escalabilidad, seguridad y experiencia real.
- Desactivar y eliminar módulos del monolito ya reemplazados.
- Mantener respaldo completo del sistema monolítico.

16. Resultados Esperados

- Catálogo con búsquedas rápidas, eficientes y en tiempo real.
- Carrito y pagos sin interrupciones.
- Reducción del tiempo de respuesta en momentos de alto tráfico.
- Nuevas funcionalidades: recomendaciones personalizadas, historial de compras y promociones automáticas.

17. Base de datos realizada para proyecto PetSocity



Descripción de las tablas:

Usuarios

La tabla 'usuarios' almacena la información básica de los clientes como nombres, correo, contraseña, dirección y fecha de creación. Es una tabla principal que se relaciona con la tabla 'carrito_compras', ya que un usuario puede tener varios carritos.

Categorías

La tabla 'categorias' contiene los diferentes tipos de productos existentes. Cada categoría tiene un identificador único y un nombre.

Se relaciona con la tabla 'inventario', ya que un producto pertenece a una sola categoría. Es una relación de uno a muchos (1:N).

Inventario

La tabla “inventario” guarda los productos disponibles: nombre, descripción, precio, stock, categoría y fecha de creación.

Está relacionada con:

'categorías': cada producto pertenece a una única categoría.

'detalle_carrito': cada producto puede ser agregado muchas veces a diferentes carritos.

Carrito compras

Esta tabla representa los carritos asociados a un usuario. Puede estar en estado ACTIVO o FINALIZADO.

Está relacionada con:

'usuarios': un usuario puede tener varios carritos.

'detalle_carrito': un carrito puede contener múltiples productos.

Detalle carrito

Esta tabla intermedia relaciona productos con carritos, especificando la cantidad y el precio al momento de agregar el producto.

Relaciona:

'carrito_compras': cada detalle pertenece a un carrito.

'inventario': cada detalle se refiere a un producto específico.

18. Microservicios

Microservicio: Carrito

Nombre del microservicio: usCarrito

Propósito: Gestionar toda la lógica asociada a los carritos de compra que los usuarios utilizan para almacenar productos antes de realizar una compra.

Responsabilidades:

- Crear carritos de compra nuevos.
- Consultar carritos existentes por ID o por usuario.
- Filtrar carritos según su estado (por ejemplo: ACTIVO, FINALIZADO).
- Asociar el carrito con el identificador del usuario que lo está utilizando.

Entidad principal: Carrito

- id (Long): Identificador del carrito.
- usuarioId (Long): Identificador del usuario asociado al carrito.
- estado (Enum): Estado del carrito (ACTIVO, FINALIZADO, etc.).
- fechaCreacion (LocalDateTime): Fecha en que se creó el carrito.

Características técnicas:

- Utiliza Spring Boot y Spring Data JPA.
- Se comunica indirectamente con el microservicio de usuarios a través del campo usuarioId.
- No mantiene relaciones directas con otras entidades externas.

Microservicio: Inventario

Nombre del microservicio: usInventario

Propósito: Administrar los productos disponibles en la tienda, incluyendo sus detalles, categorías, precios y stock actual.

Responsabilidades:

- Crear y modificar productos.
- Consultar el inventario por producto o categoría.
- Gestionar el stock disponible.
- Categorizar productos para facilitar la navegación y búsqueda.

Entidad principal: Producto

- id (Long): Identificador del producto.
- nombre (String): Nombre del producto.
- descripcion (String): Descripción del producto.
- precio (Decimal): Precio unitario.
- stock (Integer): Cantidad disponible en inventario.
- categoriaId (Long): Identificador de la categoría asociada.

Características técnicas:

- Implementado con Spring Boot.
- Utiliza JPA para persistencia de datos.

- Diseñado para integrarse con sistemas de visualización de catálogo y carritos de compra.

Microservicio: Usuario

Nombre del microservicio: usUsuario

Propósito: Gestionar los datos de los usuarios registrados en la plataforma.

Responsabilidades:

- Registrar nuevos usuarios.
- Consultar información personal y de contacto.
- Mantener actualizados los datos de los usuarios.

Entidad principal: Usuario

- id (Long): Identificador único del usuario.
- primerNombre, apellido (String): Datos personales del usuario.
- email (String): Correo electrónico.
- direccion (String): Dirección de residencia.
- fechaRegistro (LocalDateTime): Fecha de inscripción.

Características técnicas:

- Servicio autónomo, desacoplado de los demás.
- Provee identificadores (id) que otros microservicios utilizan como referencia.