Universidad Tecnológica de Bolívar

Software Architecture Document

E-Commerce

Autores:

Natalia Orozco Ropero – T00065870

Carlos David Rodríguez González - T00066078

César Fabricio Salas Ricaurte - T00065846

Josué de Jesús Torres Torres - T00066273

Version 1.0

ÍNDICE

Introducción

Propósito

El propósito de esta migración es mejorar la escalabilidad, flexibilidad y capacidad de integración de la plataforma, resolviendo los problemas actuales de fragilidad, dificultad de escalado y personalización limitada debido a la estructura monolítica. Se busca, además, facilitar la rápida incorporación de nuevos proveedores y servicios, optimizando el manejo de los datos de los clientes para personalizar mejor las ofertas y el contenido.

Alcance

El alcance de este documento abarca las facetas de la arquitectura del sistema, incluyendo la estructura del sistema, los componentes de software, y las interfaces del sistema. Este documento no cubrirá los detalles específicos de implementación o el código fuente, pero se centrará en la estructura y el diseño arquitectónico del sistema.

Definición del problema

Una empresa de comercio electrónico tiene una aplicación monolítica que gestiona los clientes web y móviles, el procesamiento de pagos, la gestión de pedidos, el inventario, etc. Pero se ha vuelto hinchada, frágil y difícil de escalar. Los datos de los clientes están fragmentados, lo que limita la capacidad de ofrecer experiencias personalizadas. También es difícil añadir nuevos servicios de terceros. La estructura monolítica dificulta enormemente la rápida implantación de nuevas funciones.

Por ejemplo, admitir un nuevo proveedor de pagos requiere cambios invasivos en todo el código base. La falta de datos aislados de los clientes también restringe la capacidad de la empresa de comercio electrónico para adaptar el contenido a sus necesidades. la capacidad de la empresa de comercio electrónico para adaptar el contenido y las ofertas en función de los intereses individuales. La empresa necesita una arquitectura más ágil alineada con las capacidades empresariales para acelerar el desarrollo de funciones y aprovechar los datos para la personalización y desarrollo de funciones y aprovechar los datos para la personalización. La solución debe adoptar un enfoque de microservicios, descomponiendo el monolito en servicios centrados en la capacidad empresarial. en servicios centrados en la capacidad empresarial. Estos servicios pueden exponer sus datos a través de API bien definidas gestionadas por una pasarela.

Una plataforma de datos en la nube puede agregar datos para análisis y aprendizaje automático. La atención debe centrarse en la transición incremental del monolito a microservicios libremente acoplados y orientados a dominios.

Objetivos

- Escalabilidad Mejorada: Al descomponer el monolito en microservicios independientes, la plataforma podrá escalar de manera más eficiente, permitiendo el crecimiento del negocio sin necesidad de realizar cambios invasivos en el código base.

- Agilidad en el Desarrollo: Los equipos podrán desarrollar, probar e implementar nuevas funcionalidades de manera independiente, reduciendo los tiempos de ciclo de desarrollo e impulsando la innovación.

- Integración con Servicios Externos: La implementación de API's facilitará la integración con terceros, como proveedores de pago y sistemas de gestión de inventario, acelerando la adopción de nuevas tecnologías o partners.

- Personalización Basada en Datos: El almacenamiento consolidado de datos y su análisis mediante herramientas de aprendizaje automático permitirá ofrecer experiencias personalizadas para los clientes, mejorando la satisfacción y reteniendo usuarios.

- Resiliencia: Al ser cada microservicio autónomo, cualquier fallo en un componente no afectará el funcionamiento general del sistema, mejorando su disponibilidad y confiabilidad.

Definiciones, Acronimos y Abreviaturas

- API (Application Programming Interface): Conjunto de definiciones y protocolos que permite la comunicación entre diferentes partes del software.
- SAD (Software Architecture Document): Documento que describe la arquitectura de software de un sistema.
- UI (User Interface): Interfaz de usuario, la parte del software con la que interactúan los usuarios.
- DB (Database): Base de datos, sistema de almacenamiento de datos.
- REST (Representational State Transfer): Estilo arquitectónico para diseñar servicios web.
- HTTPS (Hypertext Transfer Protocol Secure): Versión segura de HTTP.
- SQL (Structured Query Language): Lenguaje de programación estándar para gestionar bases de datos relacionales.
- CRUD (Create, Read, Update, Delete): Operaciones básicas para interactuar con bases de datos.
- JWT (JSON Web Token): Estándar para la creación de tokens de acceso que permiten la autenticación y el intercambio seguro de información.

- CI/CD (Continuous Integration/Continuous Deployment): Prácticas de desarrollo de software que permiten la integración y despliegue continuos de código.
- Azure DevOps: Plataforma de servicios de DevOps de Microsoft.
- MVC (Model-View-Controller): Patrón de arquitectura de software que separa la lógica de la aplicación en tres componentes interconectados.
- JSON (JavaScript Object Notation): Formato de intercambio de datos ligero y fácil de leer.
- IDE (Integrated Development Environment): Entorno de desarrollo integrado, una aplicación que proporciona herramientas comprensivas para el desarrollo de software.
- ORM (Object-Relational Mapping): Técnica de programación para convertir datos entre sistemas incompatibles usando objetos de programación orientada a objetos.

System Overview

E-Commerce company consiste en la transformación de una plataforma monolítica de comercio electrónico a una arquitectura basada en microservicios, mejorando la escalabilidad y personalización. Las principales características incluyen la descomposición de funcionalidades clave como la gestión de pedidos, pagos e inventario en microservicios independientes, la exposición de estos servicios a través de APIs unificadas gestionadas por una pasarela, la integración flexible de proveedores de pago y el almacenamiento de datos en una plataforma en la nube para análisis avanzados y personalización. Esta arquitectura permite un desarrollo más ágil y escalable, adaptado a las necesidades de los usuarios. Esta función general se desglosa en varias responsabilidades clave:

- Diseño Eficiente: El sistema de arquitectura debe proporcionar un diseño eficiente y escalable que permita cumplir con los requisitos funcionales y no funcionales del sistema, así como con las metas y objetivos del proyecto.
- Desarrollo y Despliegue Ágil: Facilitar un proceso de desarrollo ágil y eficiente, permitiendo la implementación continua de nuevas características y mejoras en la plataforma.
- Mantenimiento y Actualización: Proporcionar herramientas y procesos para el mantenimiento regular y la actualización del sistema, incluyendo la gestión de parches de seguridad y la corrección de errores.
- Integración de Servicios: Facilitar la integración de servicios internos y externos, permitiendo la interoperabilidad con otros sistemas y la expansión de la funcionalidad de la plataforma.
- Seguridad y Cumplimiento: Garantizar la seguridad de los datos del cliente y el cumplimiento de las regulaciones de privacidad y seguridad aplicables, mediante la implementación de medidas de protección adecuadas y la monitorización de posibles vulnerabilidades.

Metas Generales

- Mejora de la Experiencia del Cliente: Proporcionar una plataforma intuitiva y accesible que mejore la interacción entre los clientes y la empresa, facilitando un proceso de comunicación más fluido y eficiente.
- Eficiencia Operacional: Aumentar la eficiencia de los procesos internos mediante la automatización de tareas rutinarias y la reducción de la carga de trabajo manual en el personal de la empresa.

- Escalabilidad del Sistema: Desarrollar una arquitectura que permita escalar el sistema de manera eficiente para manejar un incremento en la demanda de usuarios y datos sin degradar el rendimiento.
- Seguridad y Privacidad de Datos: Garantizar que todos los datos de los clientes estén seguros y que el sistema cumpla con las normativas de privacidad y seguridad aplicables.

Objetivos Específicos:

- Integración de Servicios: Integrar diversas funcionalidades y servicios en una única plataforma, como gestión de cuentas, facturación y atención al cliente, para proporcionar un servicio integral a los usuarios.
- Soporte Multiplataforma: Asegurar que la Web App sea accesible y operativa en múltiples dispositivos y navegadores, proporcionando una experiencia de usuario consistente y de alta calidad.
- Desarrollo de Microservicios: Implementar una arquitectura de microservicios que permita el desarrollo, la prueba, y el despliegue independiente de cada servicio, mejorando así la mantenibilidad y la agilidad del proceso de desarrollo.
- Automatización de la Gestión de API: Utilizar un API Gateway para manejar y asegurar las interfaces de API, facilitando la integración con otros sistemas y servicios externos.

Requerimientos funcionales:

1. Gestión de Pedidos con Microservicios

- RF-1.1: Permitir la creación de nuevos pedidos ingresando los detalles del producto, cantidad, y método de pago.
- RF-1.2: Validar la disponibilidad del producto en el inventario antes de confirmar el pedido.
- RF-1.3: Actualizar el estado del pedido en cada una de sus etapas (pendiente, procesado, enviado, completado).
- RF-1.4: Notificar al cliente en cada actualización del estado de su pedido.

2. Procesamiento de Pagos Modular

- RF-2.1: Integrar diversos métodos de pago, permitiendo al usuario seleccionar su preferido.
- RF-2.2: Reintentar automáticamente las transacciones fallidas hasta tres veces antes de notificar al

usuario.
- RF-2.3: Asegurar la protección de los datos financieros del usuario mediante procesos de cifrado y
autenticación.

3. Gestión de Inventario Descentralizada

- RF-3.1: Actualizar en tiempo real la cantidad disponible en el inventario cuando se confirme un
pedido o se realice una devolución.

Page 6
- RF-3.2: Generar alertas automáticas cuando el inventario de un producto caiga por debajo de un
nivel mínimo definido.
- RF-3.3: Permitir la consulta del estado del inventario por parte de los usuarios y del equipo de
operaciones.

4. API Unificada para Clientes y Productos

- RF-4.1: Proporcionar acceso a los datos de clientes y productos a través de una API unificada,
permitiendo la consulta de historial de compras y preferencias del cliente.
- RF-4.2: Permitir la actualización de los datos de clientes y productos, validando la integridad de los
datos antes de guardarlos.
- RF-4.3: Ofrecer una interfaz unificada para la integración con servicios externos, facilitando la
personalización del contenido.

5. Almacenamiento de Datos en la Nube

- RF-5.1: Centralizar los datos de pedidos, clientes, pagos e inventario en un almacén de datos en la
nube.
- RF-5.2: Garantizar el acceso seguro y escalable a los datos almacenados en la nube, permitiendo su
consulta y modificación por parte de los microservicios.
- RF-5.3: Proporcionar redundancia en la nube para garantizar la integridad y disponibilidad de los
datos en todo momento.

6. Integración con Análisis y Aprendizaje Automático

- RF-6.1: Utilizar los datos almacenados en la nube para realizar análisis avanzados y generar
recomendaciones personalizadas basadas en el comportamiento de compra y las preferencias del
usuario.
- RF-6.2: Aplicar modelos de aprendizaje automático a los datos históricos para mejorar la
personalización y optimización de las ofertas.
- RF-6.3: Registrar y notificar errores en los procesos de análisis o generación de recomendaciones,
enviando alertas al equipo técnico para su resolución.

Requerimientos del sistema:

- RS-1. Escalabilidad: Escalar el sistema compeleto para soportar un aumento en la cantidad de usuarios y transacciones sin afectar el rendimiento.

- RS-2. Arquitectura de Microservicios: Implementar en microservicios independientes que gestionen áreas clave, como la gestión de pedidos, el procesamiento de pagos y el inventario.

- RS-3. API Unificadas: Exponer los datos de clientes y productos mediante API's bien definidas, que permitan a otras plataformas o servicios acceder a la información de manera segura y controlada.

- RS-4. Almacenamiento de Datos en la Nube: Almacenar los datos en una base de datos en la nube, permitiendo una mayor flexibilidad y disponibilidad de los datos.

- RS-5. Seguridad y Encriptación: Cumplir con altos estándares de seguridad, incluyendo la encriptación de datos sensibles y la autenticación de múltiples factores para el acceso de usuarios y administradores.

- RS-6. Tolerancia a Fallos: Mecanizar la redundancia y recuperación ante fallos para asegurar una alta disponibilidad, incluso en casos de fallos del servidor o de la red.

- RS-7. Interoperabilidad: Integrar fácilmente la integración y conexión de servicios de terceros (por ejemplo, servicios de envío, pasarelas de pago, y sistemas de inventario) mediante APIs.

Requerimientos del usuario:

- RU-1. Facilidad de Uso:
  - Los usuarios deben poder navegar por la aplicación sin dificultades, tanto en la versión web como móvil.
  - El sistema debe tener una interfaz amigable e intuitiva, que permita a los usuarios realizar compras, gestionar pedidos y hacer consultas fácilmente.

- RU-2. Acceso Multiplataforma: Los usuarios deben poder acceder al sistema desde cualquier dispositivo (móvil, tablet, computadora), manteniendo su experiencia de usuario coherente y fluida.

- RU-3. Gestión de Pedidos: Los usuarios deben poder visualizar, modificar y cancelar pedidos de forma sencilla, así como realizar el seguimiento de sus envíos.

- RU-4. Opciones de Pago: Los usuarios deben contar con opciones de pago y el sistema debe facilitar la integración con diferentes pasarelas de pago.

- RU-5. Seguridad de la Información: Los usuarios deben estar seguros de que sus datos personales y financieros están protegidos, y la plataforma debe cumplir con las normativas de protección de datos.

- RU-6. Gestión de PQRS: Los usuarios deben poder crear peticiones, quejas, reclamos (PQR) de manera rápida y recibir un seguimiento adecuado sobre el estado de su solicitud.

Contras técnicas

Políticas Corporativas y Normativas: El sistema debe cumplir con regulaciones específicas sobre protección de datos, como el GDPR, lo que limitará el manejo y almacenamiento de información personal de los clientes.

Limitaciones de Hardware: Las operaciones del sistema deberán optimizarse para funcionar en servidores con recursos limitados, lo que implica restricciones en el uso de memoria y requisitos de tiempo de respuesta para mantener un rendimiento aceptable.

Interfaces con Otras Aplicaciones: La integración con sistemas existentes, como plataformas de gestión de inventario y sistemas de atención al cliente,

requerirá que se respeten protocolos específicos, lo que podría limitar la flexibilidad en el diseño de las API's.

Tecnologías y Herramientas: Se utilizarán tecnologías y herramientas específicas, como contenedores Docker y Kubernetes para la orquestación, que pueden restringir ciertas decisiones de diseño y desarrollo en cuanto a la compatibilidad y la infraestructura.

Requisitos de Lenguaje y Protocolos de Comunicación: Se priorizará el uso de lenguajes de programación como JavaScript y Python, lo que puede limitar el uso de otras tecnologías. Además, se implementarán protocolos de comunicación como REST y gRPC para asegurar la interoperabilidad entre microservicios.

Consideraciones de Seguridad: La arquitectura deberá incorporar medidas de seguridad estrictas, como autenticación y autorización robustas, lo que puede restringir el diseño de las APIs y la interacción entre microservicios.

Convenciones de Diseño y Normas de Programación: Se seguirán convenciones de diseño específicas y normas de programación establecidas por la organización, lo que puede limitar la creatividad en el diseño del software y obligar a adherirse a patrones y prácticas predefinidos.

Compatibilidad e Integración:

Descripción: La migración de datos y funcionalidades desde sistemas heredados a una arquitectura de microservicios puede ser compleja debido a incompatibilidades técnicas.
Consecuencia: Esto puede resultar en dificultades para mantener la integridad de los datos y la funcionalidad continua durante la transición.
Solución Propuesta: Planificación cuidadosa y pruebas extensivas para asegurar una migración sin problemas.

Rendimiento y Escalabilidad:

Descripción: Asegurar que los microservicios manejen eficientemente el aumento del tráfico y la carga de trabajo puede ser desafiante.
Consecuencia: Problemas de rendimiento podrían afectar la experiencia del usuario y la eficiencia operativa.
Solución Propuesta: Implementar técnicas de optimización y escalado horizontal para mejorar el rendimiento y la escalabilidad.

Seguridad y Cumplimiento:

Autenticación y Autorización Segura: El sistema debe implementar autenticación multifactor (MFA) para todos los usuarios y anfitriones. Esto incluye el uso de contraseñas seguras combinadas con un segundo factor de autenticación, como un código enviado a un dispositivo móvil o una aplicación de autenticación. La autorización debe garantizar que los usuarios solo puedan acceder y modificar los datos y funcionalidades para los que tienen permiso, protegiendo así la integridad y confidencialidad de la información.

Fiabilidad del Sistema:

Descripción: La transición a una nueva arquitectura puede introducir puntos únicos de fallo.
Consecuencia: Fallos en componentes críticos pueden resultar en interrupciones del servicio.
Solución Propuesta: Diseñar el sistema con redundancias y mecanismos de recuperación ante desastres.

Mantenimiento y Actualizaciones:

Descripción: Mantener y actualizar múltiples microservicios puede aumentar la complejidad y el esfuerzo requerido.
Consecuencia: Esto puede resultar en tiempos de inactividad no planificados y mayores costos operativos.
Solución Propuesta: Adoptar prácticas de DevOps y CI/CD para facilitar el mantenimiento y las actualizaciones.
Contras Organizacionales

Gestión del Cambio:

Descripción: La transformación de una nueva arquitectura y tecnologías puede requerir un cambio significativo en la forma de trabajar del equipo.
Consecuencia: La resistencia al cambio puede ralentizar la implementación y la adopción de nuevas prácticas.
Solución Propuesta: Proporcionar capacitación adecuada y gestionar el cambio de manera efectiva para asegurar una transición suave.

Comunicación y Coordinación:

Descripción: Coordinar las actividades entre diferentes equipos y asegurarse de que todos estén alineados puede ser complicado.
Consecuencia: La falta de comunicación puede resultar en esfuerzos duplicados o conflictos en las tareas.
Solución Propuesta: Implementar herramientas de gestión de proyectos y comunicación para mejorar la coordinación y la colaboración.

Definición y Documentación de Procesos:

Descripción: La falta de documentación clara y definida puede llevar a malentendidos y errores en la implementación.
Consecuencia: Esto puede resultar en ineficiencias y errores costosos durante el desarrollo y la operación.
Solución Propuesta: Desarrollar y mantener una documentación detallada y actualizada de todos los procesos y procedimientos.

Priorización de Tareas:

Descripción: La falta de claridad en la priorización de tareas puede llevar a esfuerzos mal dirigidos y retrasos en la entrega.
Consecuencia: Esto puede afectar la calidad del proyecto y su capacidad para cumplir con los plazos establecidos.
Solución Propuesta: Utilizar metodologías ágiles para gestionar y priorizar las tareas de manera efectiva.

Consideraciones Adicionales:

Como estudiantes, enfrentamos desafíos adicionales debido a la falta de organización y tiempo limitado. Nuestras agendas y la necesidad de equilibrar este proyecto con otras responsabilidades académicas y personales complicaron la coordinación y la implementación efectiva del proyecto. A pesar de estos desafíos, continuamos trabajando para superar estos obstáculos y completar el proyecto de manera satisfactoria.

Stakeholders

Stakeholder	Descripción	Intereses	Influencia
Clientes	Usuarios finales del sistema que utilizarán la plataforma para gestionar sus cuentas y servicios.	Interfaz intuitiva, seguridad de datos, acceso fácil a servicios y productos, historial de pagos.	Alta, ya que su satisfacción es crucial para el éxito del sistema.
Empleados	Personal de Ecommerce que utilizará el sistema para gestionar cuentas y atender a los clientes.	Facilidad de uso, acceso rápido a la información del cliente, herramientas eficientes para la gestión.	Alta, ya que su productividad afecta la operación del negocio.
Equipo de Desarrollo	Programadores, diseñadores y testers responsables de crear y mantener el sistema.	Claridad en los requisitos, buen ambiente de trabajo, acceso a recursos necesarios, herramientas adecuadas.	Alta, ya que su trabajo determina la calidad del producto final.
Proveedores de Servicios en la Nube	Proveedores de infraestructura en la nube como Render, Railway, Azure.	Uso eficiente de recursos, cumplimiento de SLA, integración con servicios existentes.	Media, ya que proporcionan la infraestructura necesaria para el sistema.

Entorno de trabajo

Para el entorno de trabajo que se está usando en el proyecto, recurriremos a estos recursos de tecnologías avanzadas y eficientes para su desarrollo:

Visual Studio Code (VS Code)
Uso: Un editor de código fuente altamente configurable y extensible que soporta múltiples lenguajes de programación. Es ideal para desarrollar aplicaciones en Python y otras tecnologías utilizadas en este proyecto.

GitHub
Uso: Plataforma de hosting para control de versiones utilizando Git. Permite la colaboración en equipo, manejo de versiones del código, revisión de código, y almacenamiento de proyectos en repositorios.

FastAPI
Uso: Framework web moderno y de alto rendimiento para construir APIs con Python 3.6+ basado en estándar ASGI. Permite crear APIs de manera rápida y eficiente, con validación automática de datos y documentación interactiva.

Aiofiles
Uso: Biblioteca para el manejo de archivos de manera asíncrona en Python. Es útil para realizar operaciones de E/S sin bloquear el bucle de eventos, lo cual es ideal en aplicaciones web asíncronas.

Jinja2
Uso: Motor de plantillas para Python, utilizado para renderizar HTML de manera dinámica. Permite insertar datos en plantillas HTML de forma eficiente.

Uvicorn
Uso: Un servidor ASGI rápido y ligero que se usa para ejecutar aplicaciones FastAPI. Es ideal para manejar aplicaciones web asíncronas y de alto rendimiento.

Authlib
Uso: Biblioteca para el manejo de autenticación y autorización en aplicaciones web. Soporta varios protocolos de autenticación como OAuth y OpenID Connect.

HTTPX
Uso: Cliente HTTP para Python que soporta asincronía. Se usa para realizar solicitudes HTTP de manera eficiente dentro de aplicaciones FastAPI.

Psycopg2
Uso: Adaptador de base de datos PostgreSQL para Python. Permite interactuar con bases de datos PostgreSQL de manera eficiente y segura.

FastAPI_SSO
Uso: Extensión para FastAPI que facilita la implementación de Single Sign-On (SSO). Simplifica la integración de métodos de autenticación de terceros en aplicaciones FastAPI.

Python-Multipart
Uso: Biblioteca para manejar la carga de archivos en aplicaciones FastAPI. Permite procesar formularios HTML que contienen archivos adjuntos.

Tailwind
Uso: es un framework de css de código abierto para el diseño de paginas web. Ya que la principal característica de esta biblioteca es que a diferencia de otras como Bootstrap, no genera una serie de clases predefinidas para elementos como botones o tablas.

Python
Uso: Lenguaje de programación principal utilizado en este proyecto. Con su sintaxis clara y amplia gama de bibliotecas, es ideal para desarrollar aplicaciones web, manejar bases de datos y realizar análisis de datos.

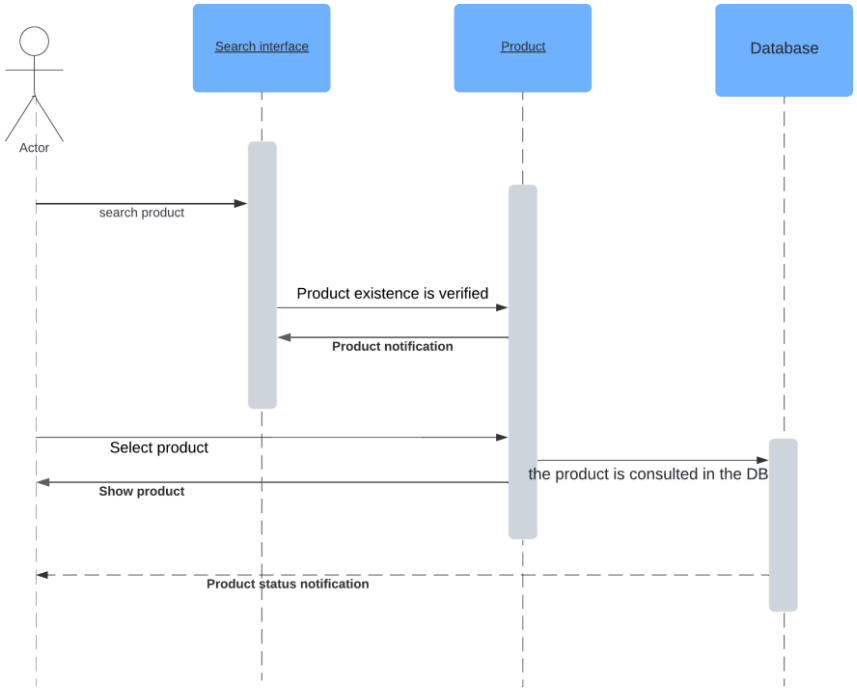
Estas herramientas y bibliotecas se combinan para proporcionar un entorno de desarrollo robusto y eficiente para la creación de aplicaciones web modernas y escalables. Más adelante se describirá los requisitos de hardware con el que se podrá usar la experiencia de usuario al máximo.

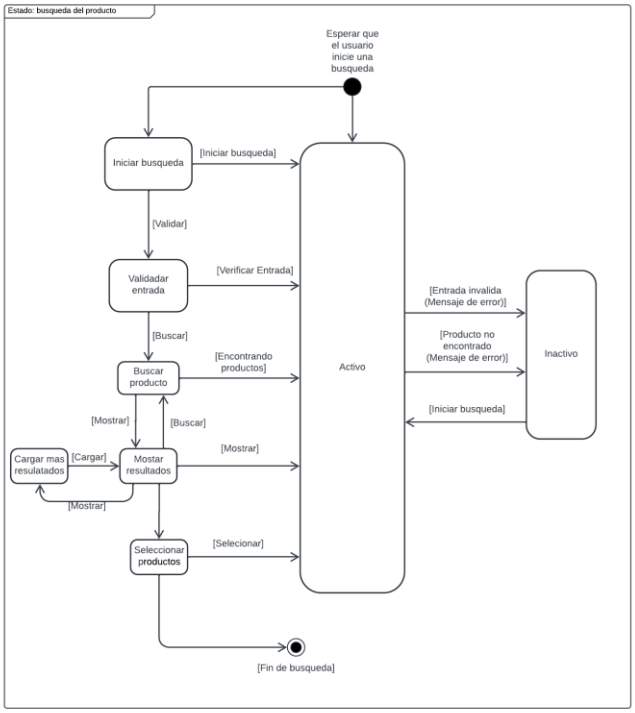
Descripción lógica del sistema (Anexos)

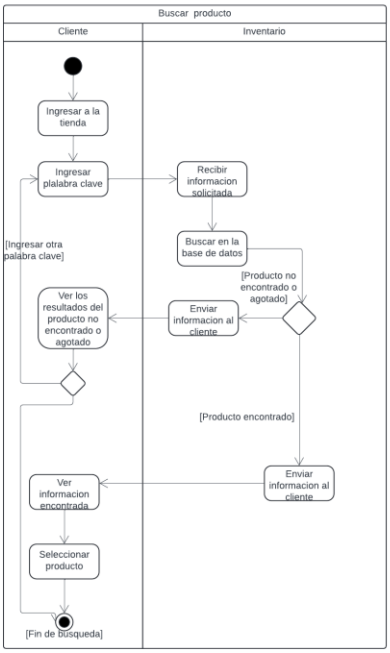
Se presentarán a continuación los casos de uso del funcionamiento/flujo que conlleva cada una de las diferentes funcionalidades que ofrece el programa:

Use Case ID:	UC-01		
Use Case Name:	Buscar productos		
Created By:	Natalia Orozco Carlos Rodríguez Josué Torres César Salas	**Last Updated By:**	Carlos Rodríguez César Salas Josué Torres Natalia Orozco
Date Created:	5/10/2024	**Date Last Updated:**	6/10/2024

Actors:	Cliente final, Sistema principal
Description:	El cliente decide buscar en la página principal de la tienda virtual, dirigiéndose directamente en la barra de búsqueda ingresando en ella que producto desea ver su información y teniendo la posibilidad de añadirlo al carrito de compra.
Trigger:	El cliente ingresa a la página principal de la tienda, identifica la barra de búsqueda y busca un producto.
Preconditions:	PRE-1: El usuario debe tener disponibilidad de un dispositivo tecnológico. PRE-2: Contar con acceso a internet.
Postconditions:	POST-1: El Sistema muestra una lista de productos igual o similares al que el cliente solicitó. POST-2: El sistema ofrece la opción de agregarlo a un carrito de compra.
Normal Flow:	1. El usuario debe tener un dispositivo tecnológico con acceso a internet. 2. El usuario debe estar conectado a internet. 3. El usuario ingresa a su navegador web de preferencia. 4. El usuario ingresa a la página principal. 5. El usuario ingresa palabras clave en la barra de búsqueda. 6. El sistema realiza una búsqueda de productos que coincidan con las palabras clave. 7. El sistema muestra los productos encontrados al cliente. 8. El usuario selecciona el producto de su interés. 9. Fin (si decide comprar es otro caso de uso).
Alternative Flows:	1. Si no llega el buscador a encontrar productos, el sistema muestra un mensaje informando que no hay resultados disponibles de momento o que el producto puede estar agotado por el momento.
Exceptions:	1. El sistema está fuera de línea. [Fatal/Informativo] 2. Mala conexión a internet. [Crítico] 3. Entradas no válidas (tipos de datos/caracteres incorrectos) [Advertencia] 4. Incompatibilidad [Advertencia/Fatal]
Priority:	Alta

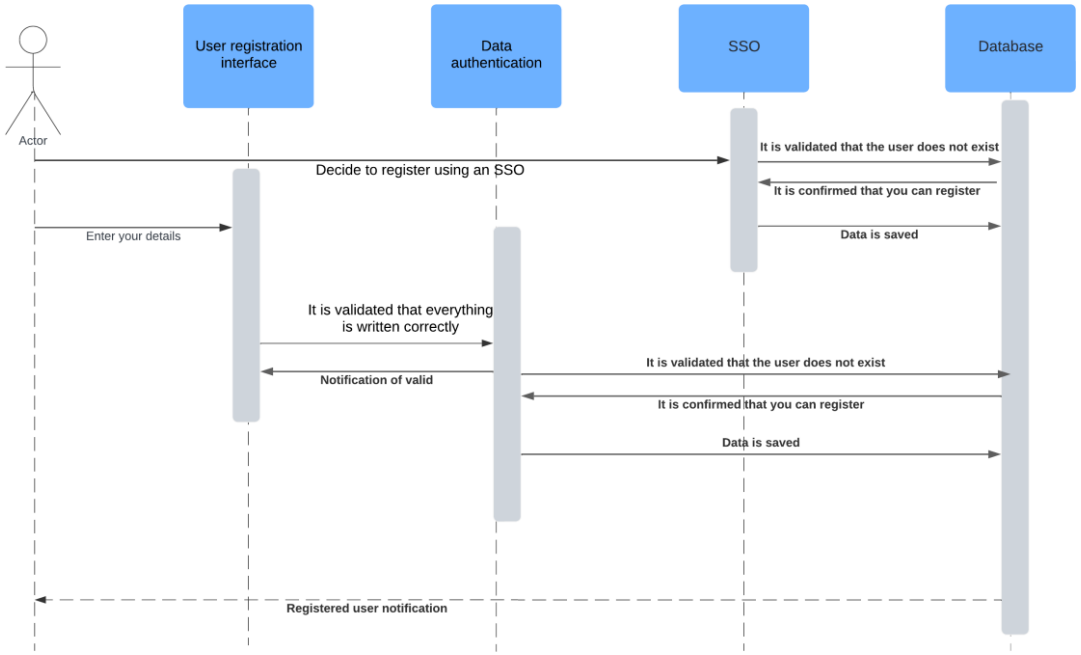

Search interface
Product
Database
Actor
search product
Product existence is verified
Product notification
Select product
the product is consulted in the DB
Show product
Product status notification

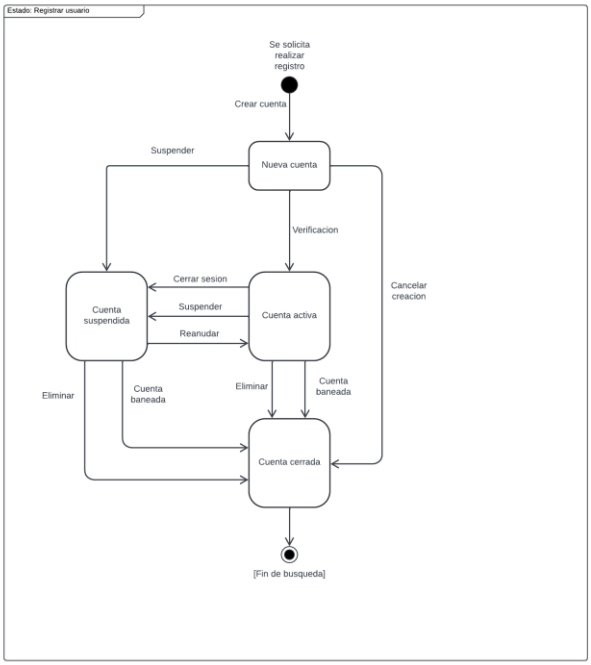

Estado: busqueda del producto
Esperar que el usuario inicie una busqueda
Iniciar busqueda
[Iniciar busqueda]
[Validar]
Validadar entrada
[Verificar Entrada]
[Buscar]
Buscar producto
[Encontrando productos]
[Mostrar]
[Buscar]
Cargar mas resulatados
[Cargar]
Mostar resultados
[Mostrar]
[Mostrar]
Seleccionar productos
[Selecionar]
Activo
[Entrada invalida (Mensaje de error)]
[Producto no encontrado (Mensaje de error)]
[Iniciar busqueda]
Inactivo
[Fin de busqueda]

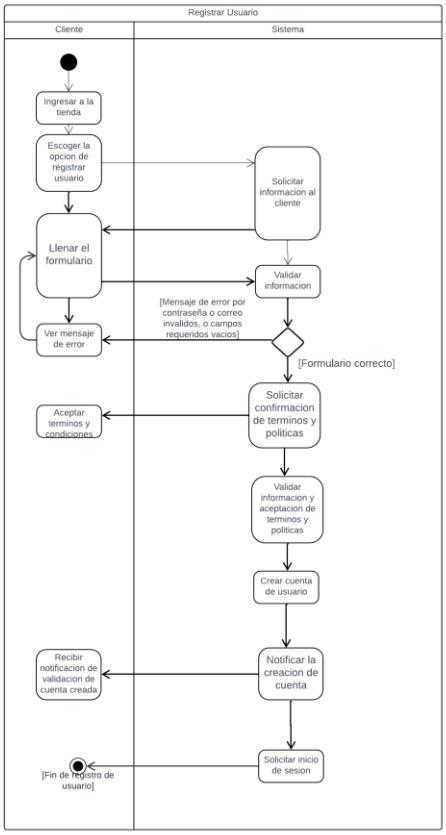

Buscar producto
Cliente
Inventario
Ingresar a la tienda
Ingresar plalabra clave
Recibir informacion solicitada
Buscar en la base de datos
[Ingresar otra palabra clave]
[Producto no encontrado o agotado]
Ver los resultados del producto no encontrado o agotado
Enviar informacion al cliente
[Producto encontrado]
Ver informacion encontrada
Enviar informacion al cliente
Seleccionar producto
[Fin de busqueda]

Use Case ID:	UC-02		
Use Case Name:	Registrar usuarios		
Created By:	Natalia Orozco Carlos Rodríguez Josué Torres César Salas	**Last Updated By:**	Carlos Rodríguez César Salas Josué Torres Natalia Orozco
Date Created:	8/10/2024	**Date Last Updated:**	9/10/2024

Actors:	Usuarios finales, Sistema (Registro)
Description:	Un nuevo usuario/cliente se registra por primera vez en la plataforma de compra E-Commerce para poder realizar compras, gestionar su información de usuario, métodos de pagos, dirección de envio, etc.
Trigger:	El cliente puede dirigirse directamente a la opción de registro de usuario; de otra forma, al encontrar un producto, al momento de realizar el pedido, el sistema solicitará un registro de usuario para poder realizar la compra.
Preconditions:	PRE-1: El usuario debe contar con una excelente conexión a internet. PRE-2: Tener posesión de un E-mail para su verificación. PRE-3: Deberá crear una contraseña segura para el registro/ingreso de la cuenta.
Postconditions:	POST-1: El sistema ofrecerá total libertad para administrar y cambiar sus datos personales. POST-2: El sistema podrá mostrar un historial de compras. POST-3: El sistema da la opción de elegir un método de pago para las transacciones. POST-4: El sistema permitirá solicitudes de PQR (caso de uso).
Normal Flow:	1. El usuario debe tener un dispositivo tecnológico con conexión a internet. 2. El usuario ingresa a la página principal. 3. El usuario elige la opción de registro directamente (o al momento de realizar la compra). 4. El sistema solicitará de manera obligatoria datos de registro (nombre, correo, contraseña, dirección residencial, teléfono, etc.). 5. El sistema valida los datos proporcionados. 6. El usuario tendrá que leer y confirmar los términos y condiciones (privacidad, tratamiento de datos, políticas, etc.). 7. El sistema creará la cuenta de usuario. 8. El sistema notificará la creación de cuenta. 9. El sistema pedirá inicio de sesión con la cuenta creada.
Alternative Flows:	1. Se advertirá si la contraseña es inválida o no cumple con los estándares mínimos 2. El sistema puede confirmar la inexistencia o invalidez de un correo electrónico 3. ...
Exceptions:	1. El sistema puede está fuera de línea [Fatal/informativo] 2. Mala conexión a internet [Crítico] 3. Correo inválido [Advertencia] 4. Contraseña inválida [Advertencia/Crítico]
Priority:	Alta

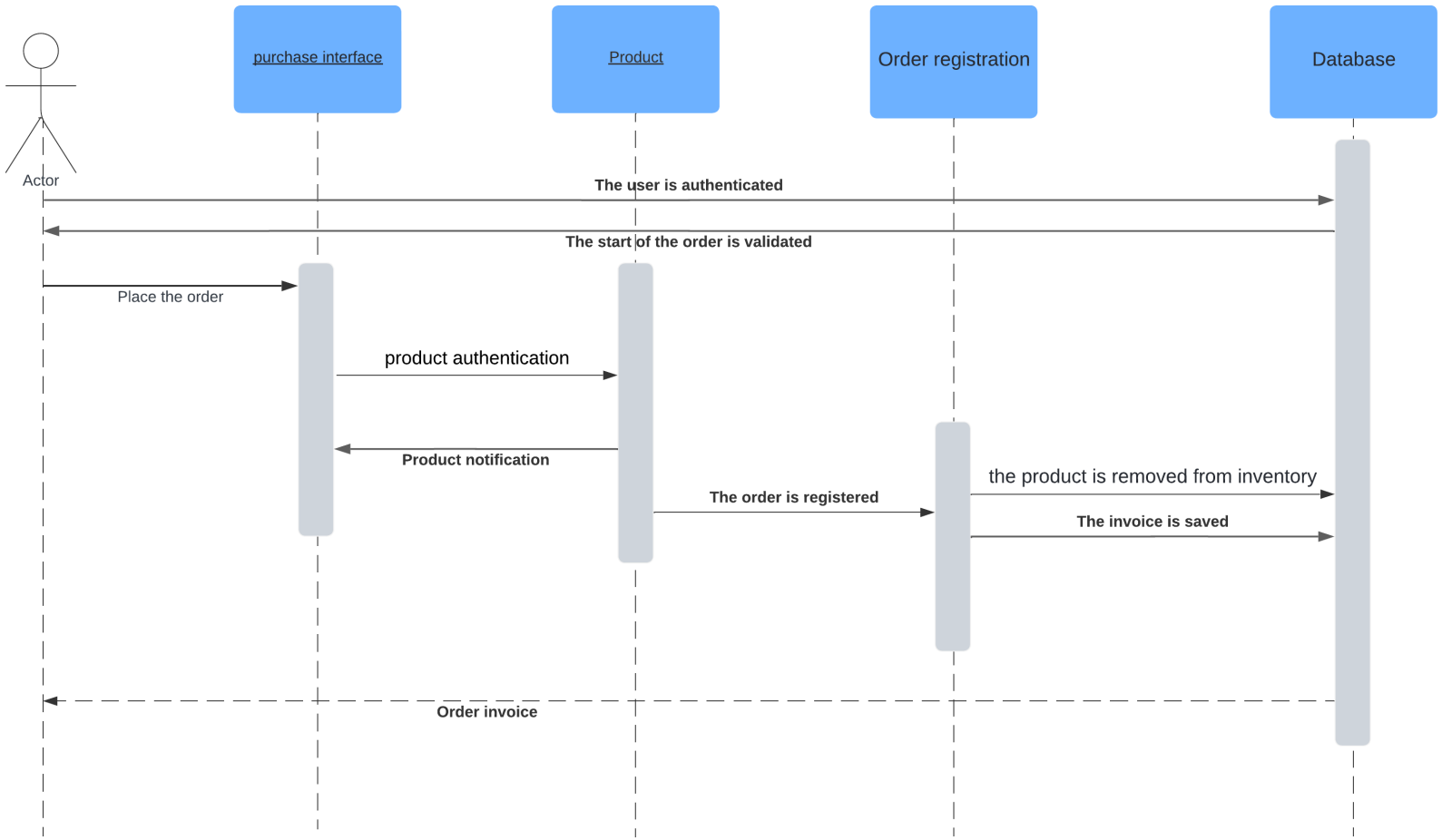

User registration interface
Data authentication
SSO
Database
Actor
Decide to register using an SSO
It is validated that the user does not exist
It is confirmed that you can register
Data is saved
Enter your details
It is validated that everything is written correctly
Notification of valid
It is validated that the user does not exist
It is confirmed that you can register
Data is saved
Registered user notification

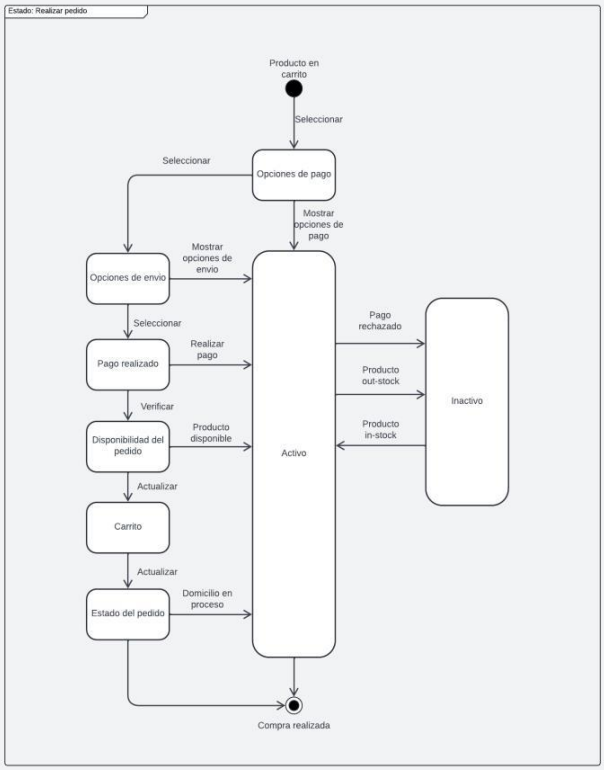

Estado: Registrar usuario
Se solicita realizar registro
Crear cuenta
Nueva cuenta
Suspender
Verificacion
Cancelar creacion
Cuenta suspendida
Cuenta activa
Cerrar sesion
Suspender
Reanudar
Eliminar
Cuenta baneada
Eliminar
Cuenta baneada
Cuenta cerrada
[Fin de busqueda]

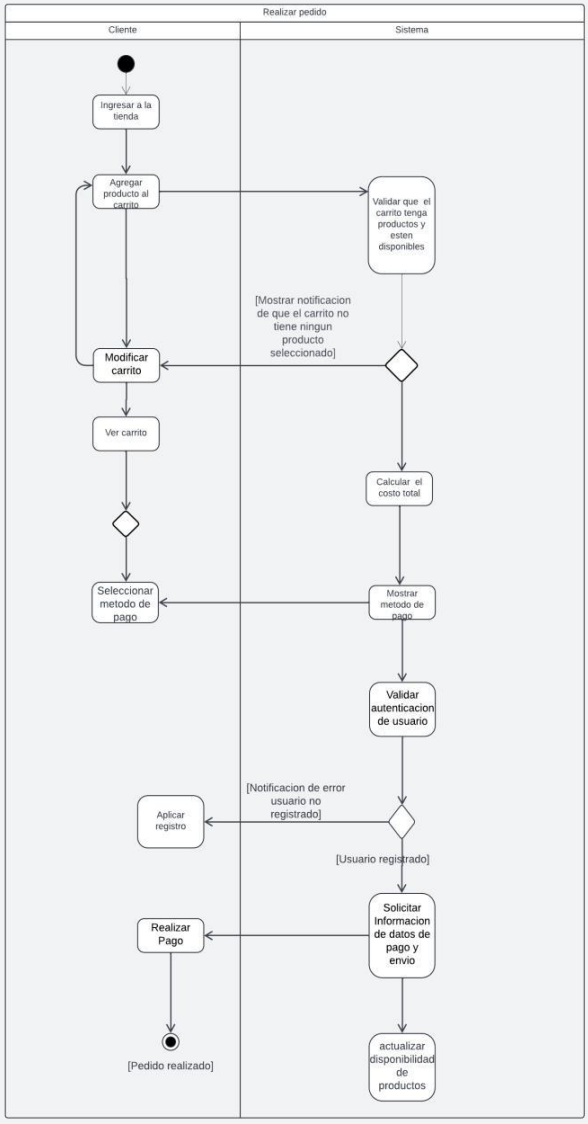

Registrar Usuario
Cliente
Sistema
Ingresar a la tienda
Escoger la opcion de registrar usuario
Solicitar informacion al cliente
Llenar el formulario
Validar informacion
[Mensaje de error por contraseña o correo invalidos, o campos requeridos vacios]
Ver mensaje de error
[Formulario correcto]
Solicitar confirmacion de terminos y politicas
Aceptar terminos y condiciones
Validar informacion y aceptacion de terminos y politicas
Crear cuenta de usuario
Recibir notificacion de validacion de cuenta creada
Notificar la creacion de cuenta
Solicitar inicio de sesion
[Fin de registro de usuario]

Use Case ID:	UC-03		
Use Case Name:	Realizar pedido		
Created By:	Natalia Orozco Carlos Rodríguez Josué Torres César Salas	**Last Updated By:**	Natalia Orozco Ropero César Salas Josué Torres Carlos Rodríguez
Date Created:	8/10/2024	**Date Last Updated:**	9/10/2024

Actors:	Cliente, sistema de carrito, sistema de pago, sistema de registro (si llega a ser necesario)
Description:	El usuario selecciona un producto de compra y procede a realizar un pedido a través de un carrito de compras.
Trigger:	En el momento en que el usuario agrega a su carrito de compras un producto, elige realizar la compra, convirtiéndose en un pedido nuevo.
Preconditions:	PRE-1: El usuario ha tenido que agregar productos al carrito de compras. PRE-2: El usuario deberá estar autenticado en el sistema. PRE-3: Hay métodos de pagos activos. PRE-4: Los productos del carrito están disponibles para la compra.
Postconditions:	POST-1: El pedido se ha creado y registrado dentro del sistema. POST-2: El inventario de los productos se ha actualizado. POST-3: El usuario recibirá una notificación de pedido a través de la plataforma. POST-4: Detalles del pago, transacción y código de pedido único han sido procesados (Facturación).
Normal Flow:	1. El cliente revisa los productos en el carrito y decide proceder al pago. 2. El sistema muestra las opciones de pago y envío. 3. El usuario selecciona un método de pago válido. 4. El sistema verifica la disponibilidad de los productos. 5. El sistema calcula el costo total del carrito. 6. El sistema actualiza el inventario y el estado del pedido.
Alternative Flows:	1. El pago puede llegar a ser rechazado. 2. El producto puede llegar a estar fuera de stock durante el proceso.
Exceptions:	1. Error en la pasarela de pagos [Fatal] 2. Direccion de envío incompleta [Advertencia]
Priority:	Alta

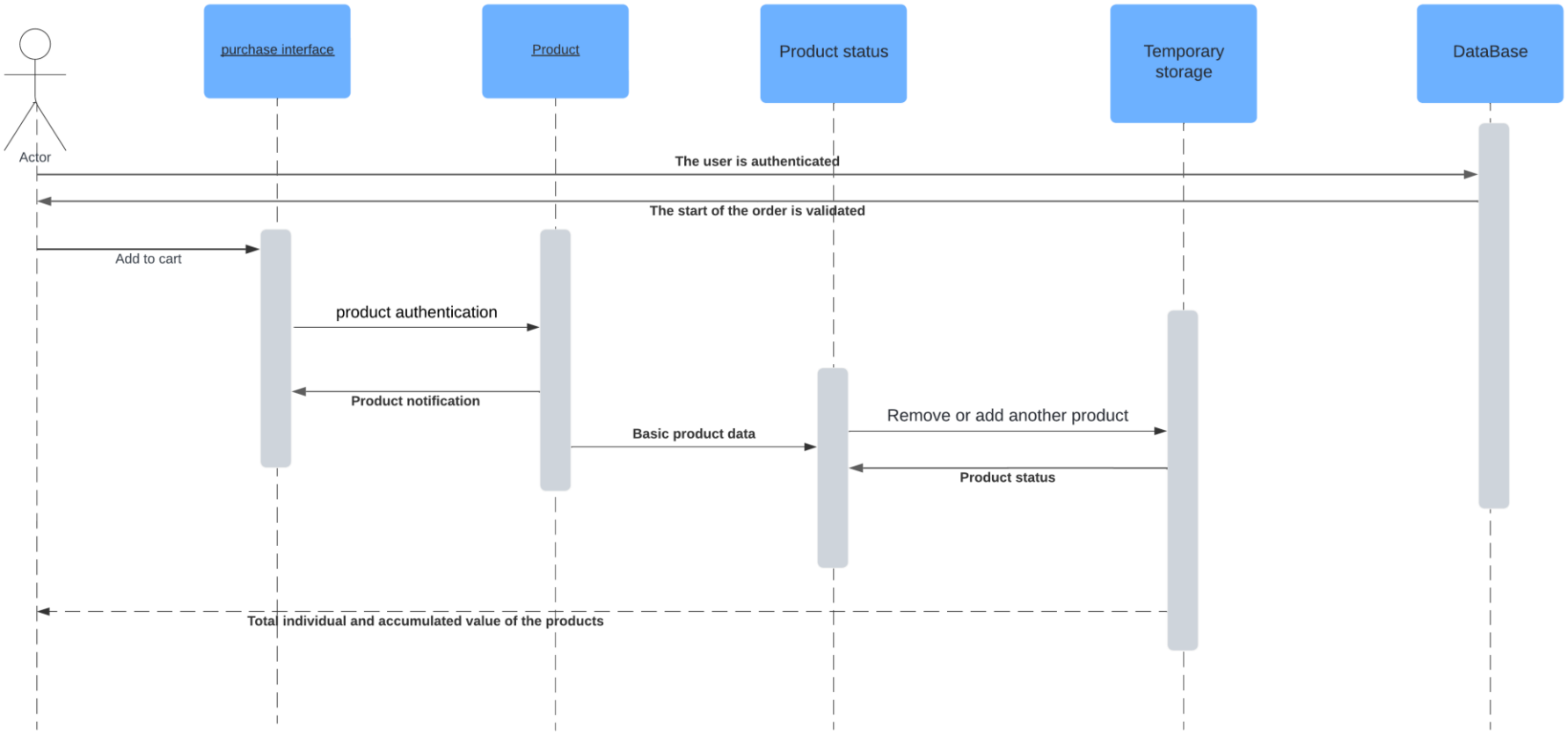

purchase interface
Product
Order registration
Database
Actor
The user is authenticated
The start of the order is validated
Place the order
product authentication
Product notification
The order is registered
the product is removed from inventory
The invoice is saved
Order invoice

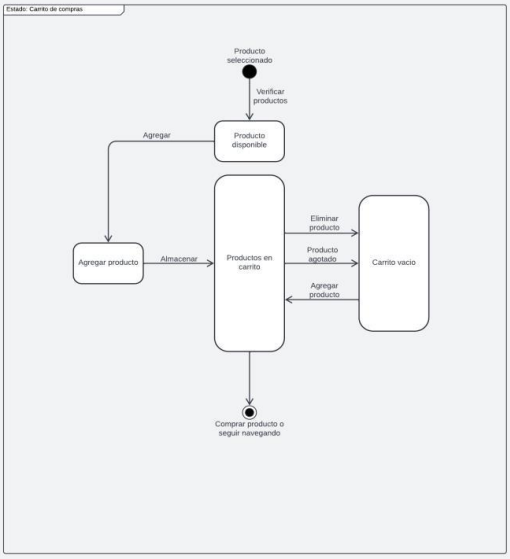

Estado: Realizar pedido
Producto en carrito
Seleccionar
Opciones de pago
Seleccionar
Mostrar opciones de pago
Opciones de envio
Mostrar opciones de envio
Seleccionar
Pago realizado
Realizar pago
Verificar
Disponibilidad del pedido
Producto disponible
Actualizar
Carrito
Actualizar
Estado del pedido
Domicilio en proceso
Activo
Pago rechazado
Producto out-stock
Producto in-stock
Inactivo
Compra realizada

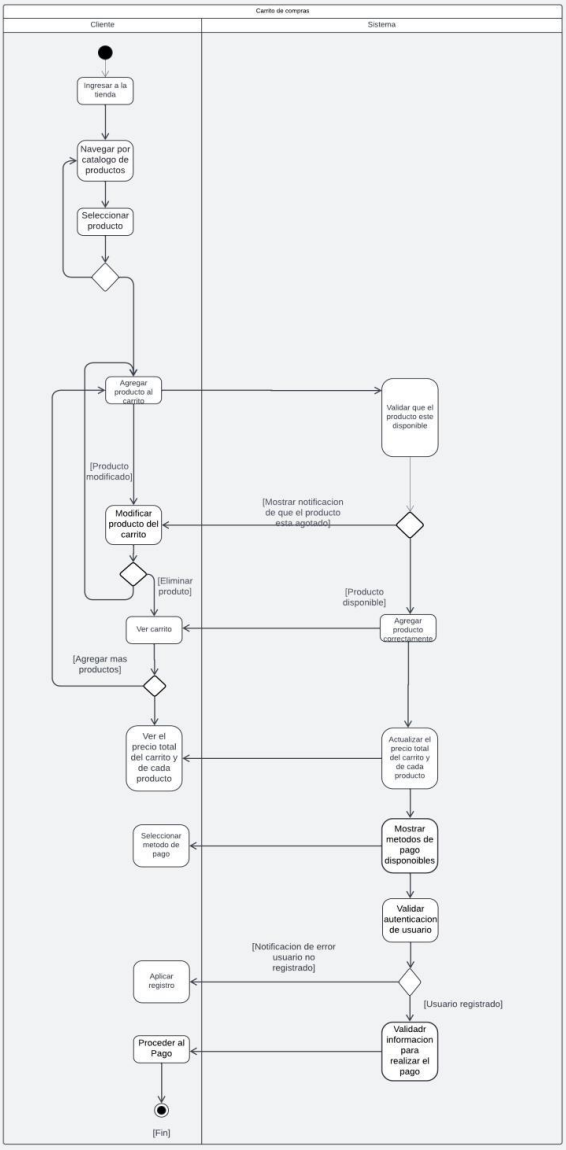

Realizar pedido
Cliente
Sistema
Ingresar a la tienda
Agregar producto al carrito
Validar que el carrito tenga productos y esten disponibles
[Mostrar notificacion de que el carrito no tiene ningun producto seleccionado]
Modificar carrito
Ver carrito
Calcular el costo total
Seleccionar metodo de pago
Mostrar metodo de pago
Validar autenticacion de usuario
[Notificacion de error usuario no registrado]
Aplicar registro
[Usuario registrado]
Solicitar Informacion de datos de pago y envio
Realizar Pago
[Pedido realizado]
actualizar disponibilidad de productos

Use Case ID:	UC-04		
Use Case Name:	Carrito de compras		
Created By:	Natalia Orozco Carlos Rodríguez Josué Torres César Salas	**Last Updated By:**	Josué Torres Natalia Orozco Carlos Rodriguez César Salas
Date Created:	8/10/2024	**Date Last Updated:**	9/10/2024

Actors:	Usuario, sistema de carrito
Description:	El carrito de compras permite a los usuarios del E-Commerce agregar productos de interés mientras navegan. Estos productos se almacenan temporalmente, permitiendo además a los usuarios revisar, modificar y eliminar artículos antes de proceder al pago.
Trigger:	Este inicia cuando un usuario selecciona un producto dentro del catálogo de la tienda y lo agrega al carrito. Permitiendo al usuario interactuar con el carrito hasta que decida proceder al pago o eliminar artículos.
Preconditions:	PRE-1: El usuario ha iniciado sesión o tiene una sesión activa dentro de la aplicación web. PRE-2: Un disponible catálogo de productos. PRE-3: Búsqueda de un producto. PRE-4: El sistema de inventario debe estar activo y disponible.
Postconditions:	POST-1: Los productos seleccionados se almacenan en el carrito y se asocian al usuario, mediante sesión. POST-2: Las actualizaciones de stock y el precio total se refleja en el carrito antes del pago.
Normal Flow:	1. El usuario navega por el catálogo de productos. 2. El usuario selecciona un producto y lo agrega al carrito de compras. 3. El sistema de manera interna verifica la disponibilidad del producto en el inventario. 4. El producto se agrega exitosamente al carrito. 5. El usuario puede continuar agregando productos, modificar las cantidades, e eliminar artículos. 6. El usuario accede a la vista del carrito, donde se actualizan los precios. 7. El usuario decide proceder al pago (Caso de uso) o seguir navegando.
Alternative Flows:	1. El usuario agrega un producto al carrito. 2. El sistema detecta un cambio de precio del producto. 3. El sistema notificar al usuario sobre el nuevo precio antes de proceder al pago.
Exceptions:	1. Error al agregar un producto [Critico/Advertencia] 2. Notificación de producto descontinuado o con indisponibilidad [Fatal/Informativo]
Priority:	Media - Alta

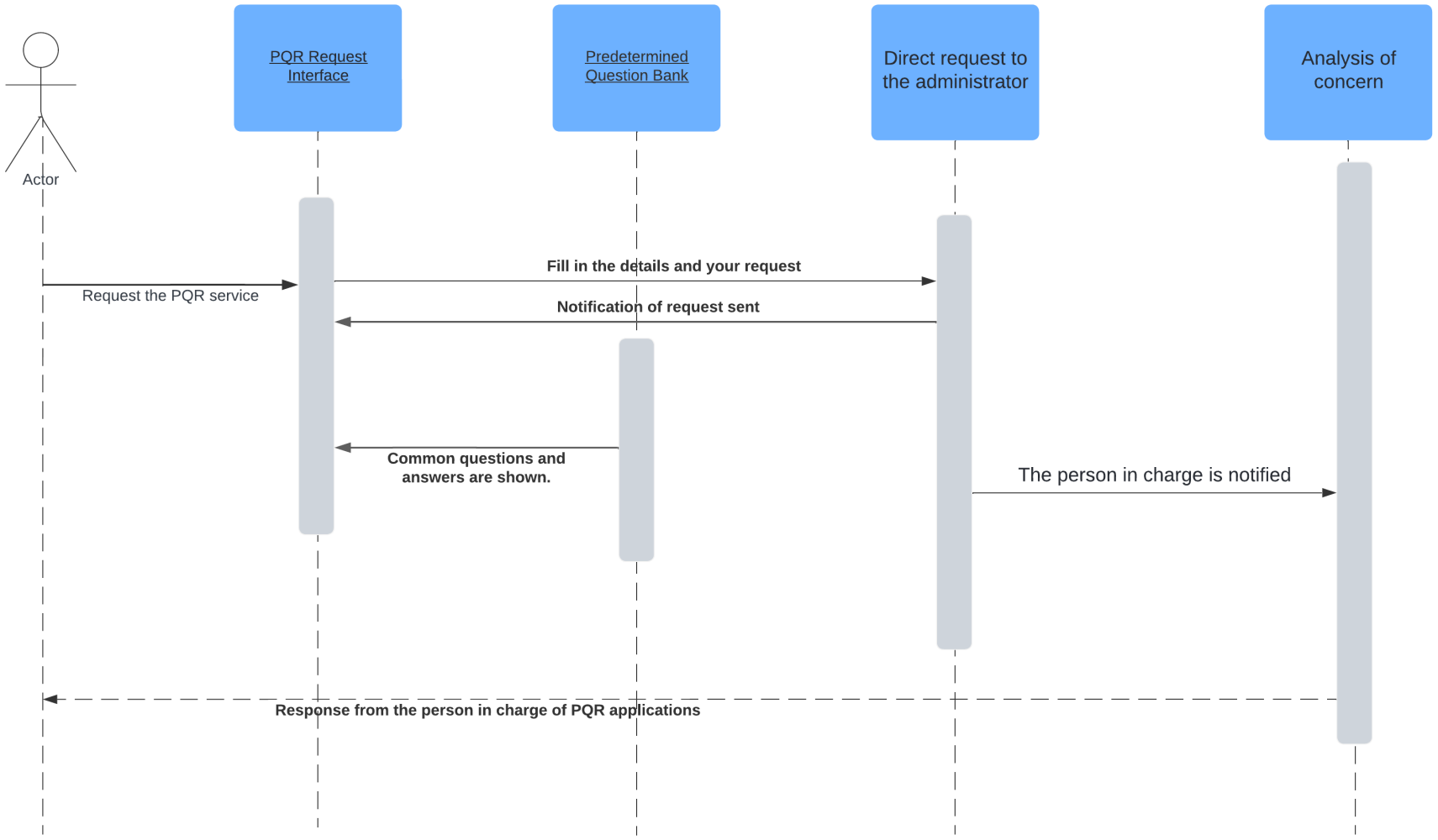

Actor
purchase interface
Product
Product status
Temporary storage
DataBase
The user is authenticated
The start of the order is validated
Add to cart
product authentication
Product notification
Basic product data
Remove or add another product
Product status
Total individual and accumulated value of the products

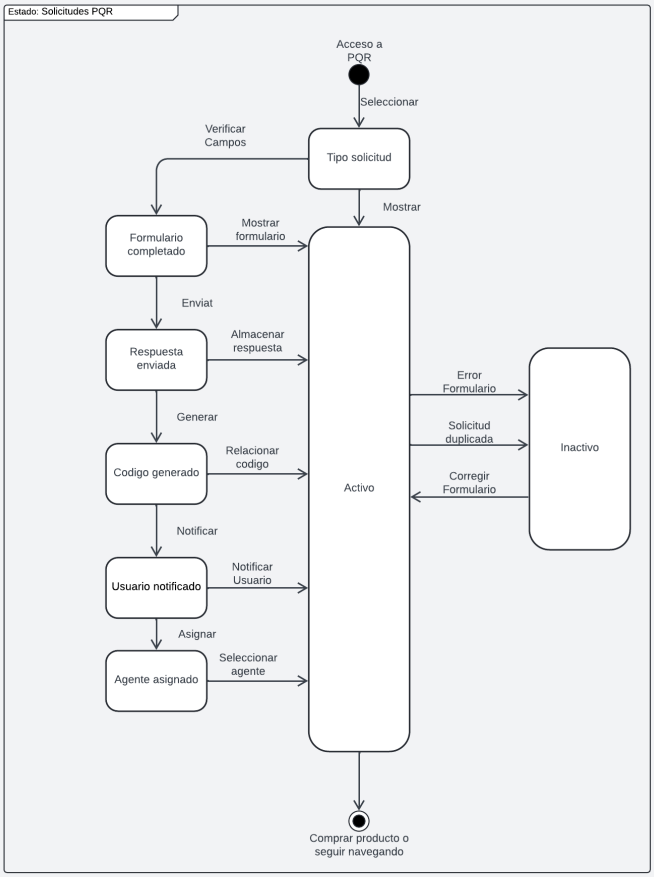

Estado: Carrito de compras
Producto seleccionado
Verificar productos
Agregar
Producto disponible
Agregar producto
Almacenar
Productos en carrito
Eliminar producto
Producto agotado
Agregar producto
Carrito vacio
Comprar producto o seguir navegando


Carrito de compras
Cliente
Sistema
Ingresar a la tienda
Navegar por catalogo de productos
Seleccionar producto
Agregar producto al carrito
Validar que el producto este disponible
[Producto modificado]
Modificar producto del carrito
[Mostrar notificacion de que el producto esta agotado]
[Eliminar produto]
[Producto disponible]
Ver carrito
Agregar producto correctamente
[Agregar mas productos]
Ver el precio total del carrito y de cada producto
Actualizar el precio total del carrito y de cada producto
Seleccionar metodo de pago
Mostrar metodos de pago disponoibles
Validar autenticacion de usuario
[Notificacion de error usuario no registrado]
Aplicar registro
[Usuario registrado]
Proceder al Pago
Validadr informacion para realizar el pago
[Fin]

Use Case ID:	UC-05		
Use Case Name:	Solicitudes PQR		
Created By:	Natalia Orozco Carlos Rodríguez Josué Torres César Salas	**Last Updated By:**	Natalia Orozco Josué Torres Carlos Rodríguez César Salas
Date Created:	8/10/2024	**Date Last Updated:**	9/10/2024

Actors:	Usuario, sistema de soporte, Administrador
Description:	PQR (peticiones, quejas y reclamos) por parte de los clientes o usuarios que tengan alguna inconformidad con respecto al servicio del E-Commerce.
Trigger:	Este caso de uso comienza cuando un usuario, decide redactar o registrar una petición, queja o reclamo, a través del sistema de sí misma.
Preconditions:	PRE-1: El usuario tiene que estar registrado. PRE-2: El sistema debe tener habilitado el módulo de apartado PQR. PRE-3: Dentro del módulo debe contar con un formulario para el registro de la solicitud.
Postconditions:	POST-1: La solicitud ha sido creada y registrada dentro del sistema, con un número de radicado único. POST-2: El sistema ha notificado a los administradores correspondientes, de la solicitud para su procesamiento. POST-3: El estado de la solicitud será notificada sobre su estado actual, en tiempo real, en proceso de respuesta.
Normal Flow:	1. El usuario accede al módulo de PQR desde la plataforma. 2. El sistema presenta al usuario un formulario para ingresar los detalles de la solicitud (tipo de solicitud: petición, queja o reclamo). 3. El usuario completa el formulario, incluyendo los detalles relevantes. 4. El sistema verifica que todos los campos obligatorios estén completos. 5. El usuario confirma el envío de la solicitud. 6. El sistema genera un número de radicado único para la solicitud y lo presenta al usuario. 7. El sistema envía una notificación de confirmación al usuario, informando que la solicitud ha sido registrada y está en proceso. 8. El sistema asigna automáticamente la solicitud al agente correspondiente para su resolución.
Alternative Flows:	1. Al rellenar de manera incompleta el formulario, el sistema notificará que falta apartados por completar. 2. Detección de una posible duplicación de solicitud, por parte de un mismo usuario, se advertirá que ya existe una en proceso.
Exceptions:	1. Error en la creación del número de radicado [Fatal] 2. Falla en notificación al usuario [Crítica]
Priority:	Baja-Media


Actor
PQR Request Interface
Predetermined Question Bank
Direct request to the administrator
Analysis of concern
Request the PQR service
Fill in the details and your request
Notification of request sent
Common questions and answers are shown.
The person in charge is notified
Response from the person in charge of PQR applications


Estado: Solicitudes PQR
Acceso a PQR
Seleccionar
Tipo solicitud
Verificar Campos
Mostrar
Formulario completado
Mostrar formulario
Enviat
Respuesta enviada
Almacenar respuesta
Generar
Codigo generado
Relacionar codigo
Notificar
Usuario notificado
Notificar Usuario
Asignar
Agente asignado
Seleccionar agente
Activo
Error Formulario
Solicitud duplicada
Corregir Formulario
Inactivo
Comprar producto o seguir navegando

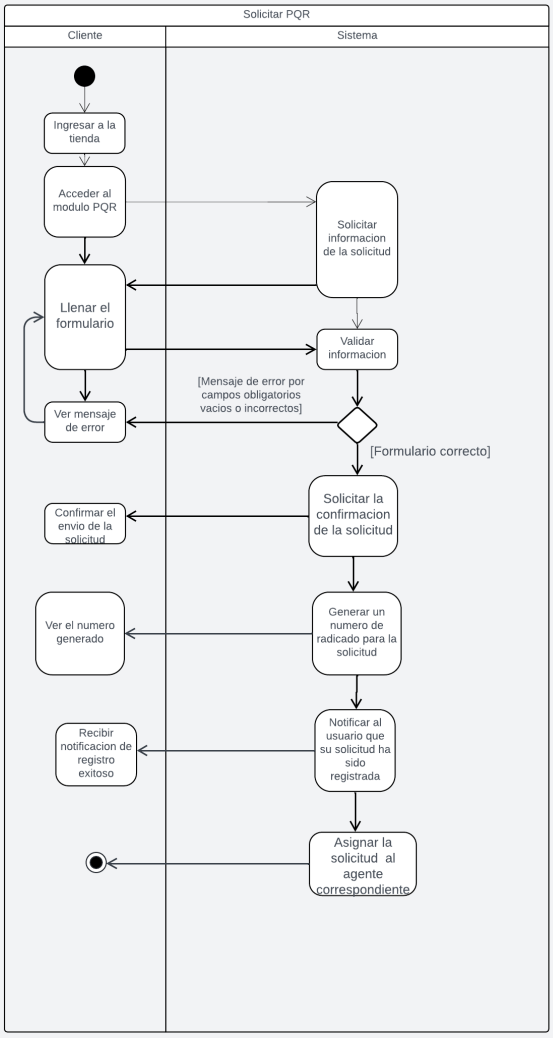
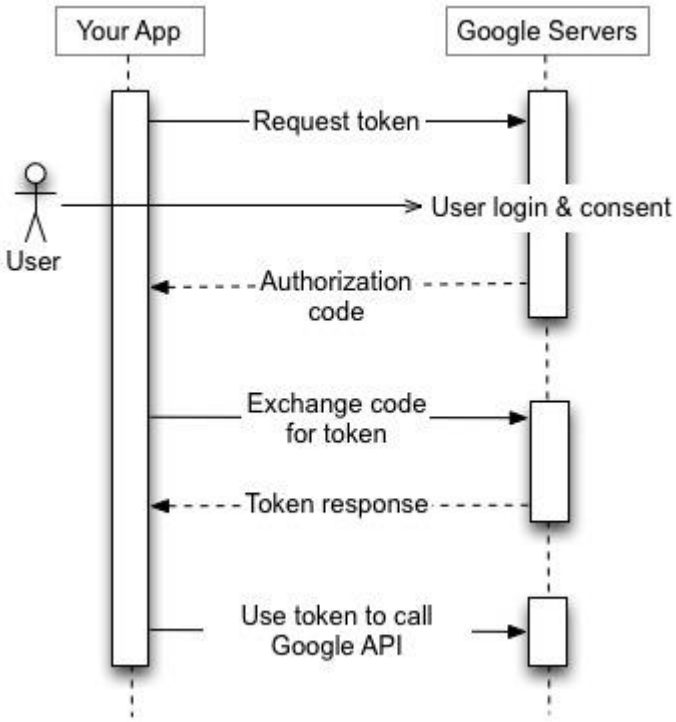

Solicitar PQR
Cliente
Sistema
Ingresar a la tienda
Acceder al modulo PQR
Solicitar informacion de la solicitud
Llenar el formulario
Validar informacion
[Mensaje de error por campos obligatorios vacios o incorrectos]
Ver mensaje de error
[Formulario correcto]
Solicitar la confirmacion de la solicitud
Confirmar el envio de la solicitud
Generar un numero de radicado para la solicitud
Ver el numero generado
Notificar al usuario que su solicitud ha sido registrada
Recibir notificacion de registro exitoso
Asignar la solicitud al agente correspondiente

Diagrama SSO Google



Aplicaciones de servidor web

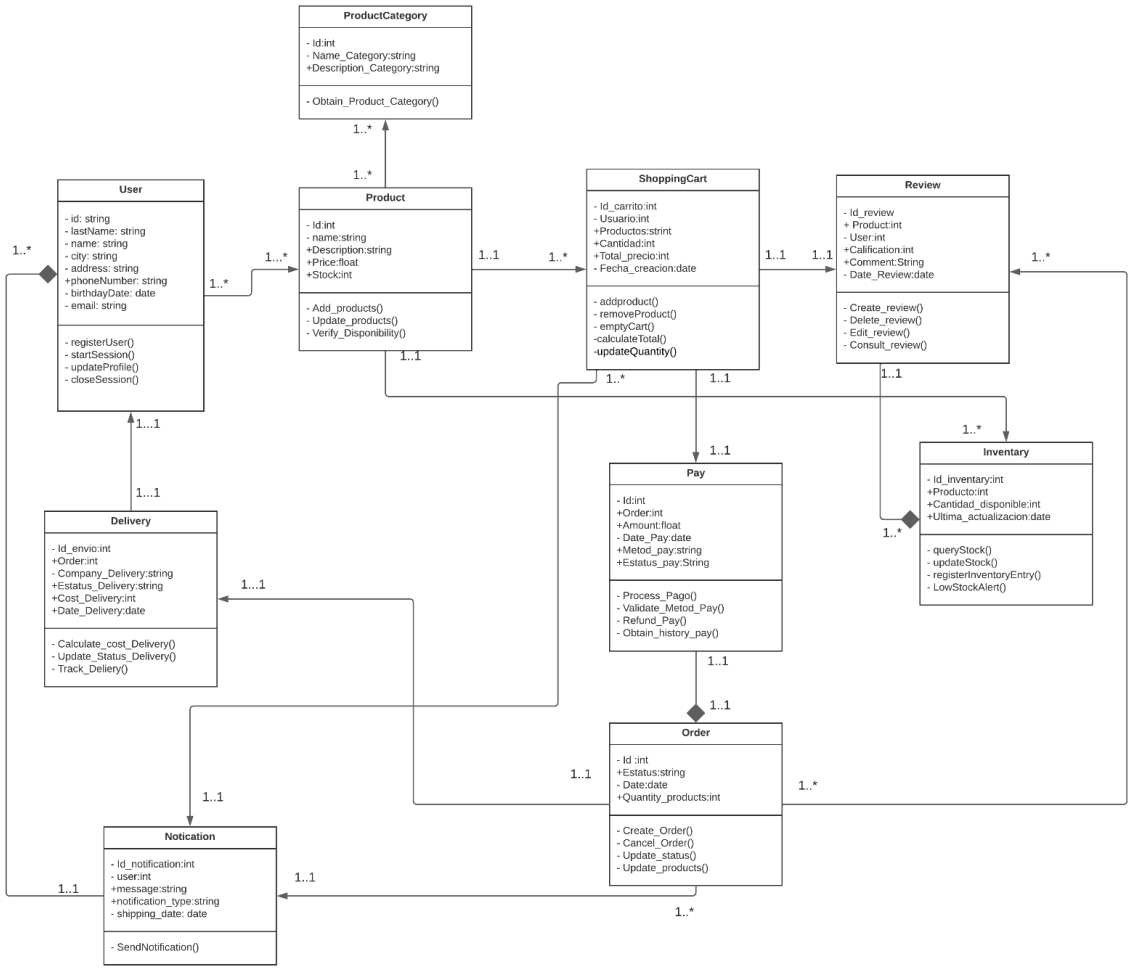
El extremo de Google OAuth 2.0 es compatible con aplicaciones de servidor web que usan lenguajes y frameworks como PHP, Java, Go, Python, Ruby y ASP.NET.

La secuencia de autorización comienza cuando tu aplicación redirecciona un navegador a una URL de Google. La URL incluye parámetros de consulta que indican el tipo de acceso que se solicita. Google se encarga de la autenticación del usuario, la selección de la sesión y el consentimiento del usuario. El resultado es un código de autorización que la aplicación puede intercambiar por un token de acceso y un token de actualización.

La aplicación debe almacenar el token de actualización para usarlo en el futuro y usar el token de acceso para acceder a una API de Google. Una vez que vence el token de acceso, la aplicación usa el token de actualización para obtener uno nuevo.

La aplicación envía una solicitud de token al servidor de autorización de Google, recibe un código de autorización, intercambia el código por un token y lo usa para llamar a un extremo de la API de Google.

Para la creación de la base de datos nos guiaremos un poco de como seria su visualización con un diagrama UML el modelo relacional por su fácil entendimiento:



Éstas son las bases más sólidas para la elaboración del proyecto ya que emplea la lógica que tendrá cada servicio y su implementación.

Interfaz del sistema

En este apartado se presentará como está desplegado las interfaces del sistema UI/UX

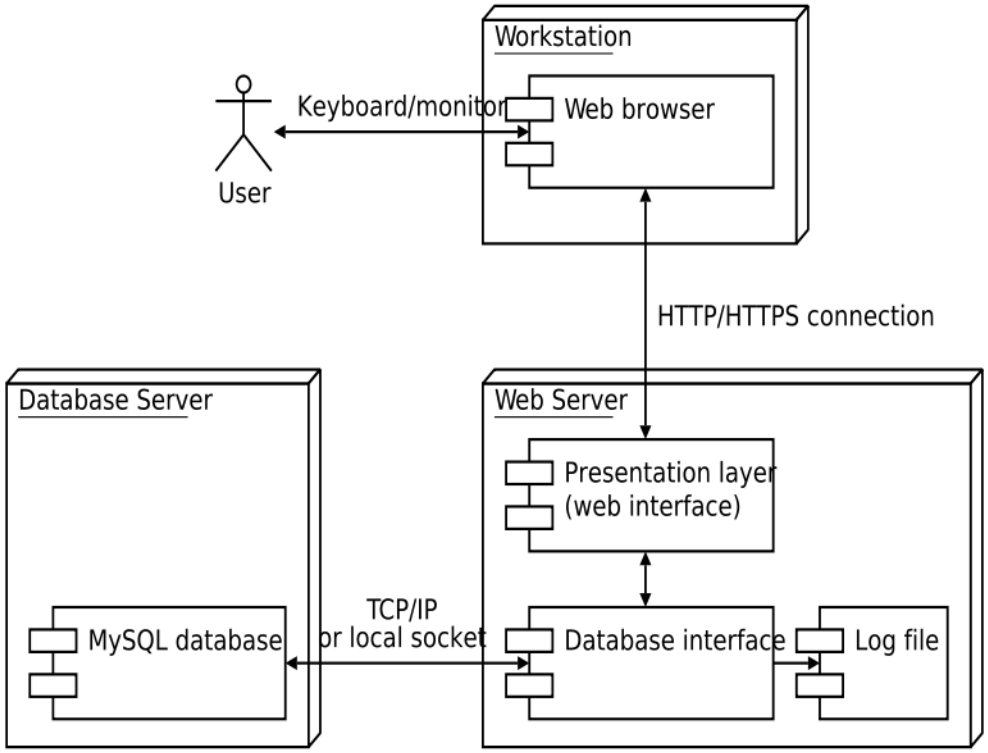
Página de inicio
Donde se podrá acceder a todos los microservicios que puede ofrecer el proyecto, desde iniciar sesión, registrarse como usuario nuevo, etc.

Página de cobertura

Pagina sobre la información, desarrolladores y objetivos, donde podrá acceder y saber sobre cada uno de los desarrolladores que lo conforman.

Interfaz de inicio de sesión, desde donde se podrá acceder a los perfiles de cada una de las cuentas ya sea con una cuenta independiente o con una de Google

Despliegue



El diagrama describe una arquitectura web de tres capas, utilizada en el desarrollo de aplicaciones web. Esta arquitectura se compone de tres componentes principales: la estación de trabajo del usuario (Workstation), el servidor web (Web Server) y el servidor de base de datos (Database Server). A continuación, se detalla cada componente y su relación con el proyecto descrito anteriormente.

Estación de Trabajo (Workstation)
- Web Browser: La estación de trabajo del usuario, típicamente una computadora o un dispositivo móvil, interactúa con la aplicación a través de un navegador web. Los usuarios, tanto clientes como empleados, acceden a la interfaz de usuario de la aplicación web mediante navegadores como Chrome, Firefox, Edge, o Safari.

En nuestro proyecto, los usuarios utilizarán navegadores web para acceder a las diferentes funcionalidades del sistema, como la gestión de cuentas, la consulta de productos y servicios, la realización de pagos y la recepción de notificaciones y alertas.

Servidor Web (Web Server)

- Presentation Layer (Web Interface): Esta capa se encarga de presentar la interfaz de usuario y manejar la lógica de presentación. Aquí se utilizan tecnologías como HTML, CSS, JavaScript, y frameworks como Bootstrap para construir una interfaz de usuario interactiva y amigable.

- Database Interface: La interfaz de la base de datos se encarga de la comunicación entre la capa de presentación y el servidor de base de datos. Esta capa maneja las consultas a la base de datos y la recuperación de datos para su presentación en la interfaz de usuario.

- Log File: Los archivos de registro (log files) almacenan eventos y transacciones importantes para el seguimiento y la auditoría del sistema.

- Relación con el Proyecto: En nuestro proyecto, el servidor web utiliza FastAPI para manejar las solicitudes HTTP y proporcionar las respuestas correspondientes. FastAPI se encarga de la lógica de la aplicación, gestionando las solicitudes de los usuarios y las interacciones con la base de datos PostgreSQL. El uso de tecnologías como Jinja2 para plantillas y Bootstrap para el diseño de la interfaz asegura que la presentación de datos sea eficiente y atractiva.

Servidor de Base de Datos (Database Server)
- MySQL Database: Aunque el diagrama menciona MySQL, en nuestro proyecto utilizamos PostgreSQL como sistema de gestión de bases de datos. Este servidor almacena toda la información relevante del sistema, incluyendo datos de usuarios, productos, cuentas, pagos, y eventos.

- Relación con el Proyecto: PostgreSQL en nuestro proyecto se encarga de almacenar y gestionar todos los datos críticos, garantizando integridad y seguridad. La base de datos está estructurada y normalizada según un modelo entidad-relación que facilita el acceso y la manipulación de datos de manera eficiente y segura. Utilizamos psycopg2 como el adaptador de base de datos para conectar FastAPI con PostgreSQL.

Integración y Flujo de Datos

Usuario Accede a la Aplicación: El usuario abre un navegador web y accede a la URL de la aplicación.

Interacción con la Capa de Presentación: El navegador web envía solicitudes HTTP al servidor web, que son manejadas por FastAPI. La capa de presentación genera las páginas web dinámicamente utilizando Jinja2 y los estilos proporcionados por Tailwind.

Consultas a la Base de Datos: Las solicitudes que requieren datos se envían desde la capa de presentación a la base de datos a través de la interfaz de base de datos. FastAPI utiliza psycopg2 para ejecutar consultas SQL en PostgreSQL y recuperar los datos necesarios.

Respuesta al Usuario: Los datos recuperados se procesan y se envían de vuelta al navegador del usuario en forma de páginas web HTML.

Manejo de Eventos y Notificaciones: El sistema también incluye la gestión de eventos y notificaciones, donde se detectan y notifican inconsistencias u otros eventos importantes a través de la interfaz de usuario.

Conclusión

Este diagrama de arquitectura de alto nivel ilustra cómo se integran y funcionan los diferentes componentes del sistema. Desde la interacción del usuario con la aplicación a través del navegador web, pasando por la lógica de negocio gestionada por FastAPI en el servidor web, hasta el almacenamiento y recuperación de datos en el servidor de base de datos PostgreSQL, cada componente juega un papel crucial en el funcionamiento del sistema. Esta arquitectura asegura que la aplicación sea escalable, segura, y eficiente, cumpliendo con los requisitos del proyecto.

Conclusión

En el desarrollo de nuestra aplicación web para la gestión de servicios E-Commerce, hemos adoptado un enfoque meticuloso y sistemático, asegurando que cada componente del sistema esté diseñado para ser escalable, seguro y eficiente. A lo largo del proyecto, se han seguido las mejores prácticas de arquitectura de software, empleando modelos de arquitectura limpia, eventos, comandos y microservicios. Esto nos ha permitido construir un sistema robusto que puede manejar las complejas necesidades de los usuarios y la empresa.

En conclusión, el desarrollo de esta aplicación web ha sido una experiencia de aprendizaje y crecimiento. Hemos logrado construir un sistema integral que no solo cumple con los requisitos funcionales y no funcionales, sino que también está preparado para adaptarse y crecer con las futuras necesidades de la empresa y sus clientes. La arquitectura elegida y las tecnologías implementadas garantizan que el sistema sea robusto, seguro y eficiente, proporcionando una base sólida para futuros desarrollos y mejoras.