**UNIVERSIDAD AUTÓNOMA GABRIEL RENÉ MORENO**

FACULTAD DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES

****

SISTEMA DE PUNTO DE VENTA (POS) CON INTELIGENCIA ARTIFICIAL PARA EL RECONOCIMIENTO DE VOZ Y RECOMENDACIONES DE PRODUCTOS A COMPRAR

**MATERIA:** SISTEMAS DE INFORMACION II

**DOCENTE:** ROLANDO ANTONIO MARTINEZ CANEDO

**GRUPO:** 14

**INTEGRANTES:**

BULACIA PAZ BRUNO LEANDRO 223041866

VARGAS OSINAGA RICHARD JUNIOR 219072736

**INDICE**

[1. PERFIL 4](#_Toc196733545)

[1.1. INTRODUCCION 4](#_Toc196733546)

[1.2. OBJETIVO GENERAL 5](#_Toc196733547)

[1.3. OBJETIVOS ESPECIFICOS 5](#_Toc196733548)

[1.4. DESCRIPCIÓN DEL PROBLEMA 6](#_Toc196733549)

[1.5. ALCANCE 7](#_Toc196733550)

[1.5.1. Módulo Usuario 8](#_Toc196733551)

[1.5.2. Módulo Autenticación y Autorización 8](#_Toc196733552)

[1.5.3. Módulo Inventario de Productos o catalogo 8](#_Toc196733553)

[1.5.4. Módulo Ventas 8](#_Toc196733554)

[1.5.5. Módulo Delivery 9](#_Toc196733555)

[1.5.6. Módulo Reportes 9](#_Toc196733556)

[2. MARCO TEORICO 10](#_Toc196733557)

[PARTE 1 – FUNDAMENTACION TEORICA 10](#_Toc196733558)

[1.1. E-COMMERCE 10](#_Toc196733559)

[1.1.1. Como usuario 12](#_Toc196733560)

[1.1.1.1. Amazon 12](#_Toc196733561)

[1.1.1.2. Alibaba 14](#_Toc196733562)

[1.1.1.3. Shopify 15](#_Toc196733563)

[1.2. PASARELAS DE PAGO 19](#_Toc196733564)

[1.2.1. FORMAS DE PAGO ONLINE 19](#_Toc196733565)

[1.2.1.1. Tarjetas de Débito y Crédito 19](#_Toc196733566)

[1.2.1.2. QR Codes 20](#_Toc196733567)

[1.2.1.3. Transferencias Bancarias 20](#_Toc196733568)

[1.2.2. LIBÉLULA 20](#_Toc196733569)

[1.2.3. STRIPE 21](#_Toc196733570)

[1.3. DELIVERY 22](#_Toc196733571)

[1.3.1. FUNCIONAMIENTO 22](#_Toc196733572)

[1.3.1.1. Proceso de Entrega: 22](#_Toc196733573)

[1.3.1.2. Tecnologías Utilizadas: 22](#_Toc196733574)

[1.3.2. CÁLCULOS DE PAGO 23](#_Toc196733575)

[1.4. PUDS 23](#_Toc196733576)

[1.4.1. Fases del Proceso Unificado: 24](#_Toc196733577)

[1.4.2. Características del Proceso Unificado: 24](#_Toc196733578)

[1.5. UML 25](#_Toc196733579)

[1.5.1. Diagramas de UML: 25](#_Toc196733580)

[1.5.2. Características de UML: 25](#_Toc196733581)

[CAPITULO 2 FLUJO DE TRABAJO: CAPTURA DE REQUISITOS 26](#_Toc196733582)

[2.1. Identificar actores y casos de uso 26](#_Toc196733583)

[2.1.1. Identificar actores 26](#_Toc196733584)

[2.1.2. Identificar casos de uso 27](#_Toc196733585)

[2.2. Priorizar casos de uso 28](#_Toc196733586)

[Ciclo #1 29](#_Toc196733587)

[Ciclo #2 30](#_Toc196733588)

[Ciclo #3 30](#_Toc196733589)

[2.3. Especificar casos de uso 31](#_Toc196733590)

[Ciclo #1 31](#_Toc196733591)

[CU1. Gestionar Usuario 31](#_Toc196733592)

[CU2. Gestionar Rol 32](#_Toc196733593)

[CU3. Gestionar Permisos 33](#_Toc196733594)

[CU4. Iniciar Sesión 34](#_Toc196733595)

[CU5. Cerrar Sesión 35](#_Toc196733596)

[CU6 Gestionar Productos 35](#_Toc196733597)

[CU7 Gestionar Categorías 37](#_Toc196733598)

[Ciclo #2 38](#_Toc196733599)

[CU9 Gestionar Ventas 38](#_Toc196733600)

[CU10. Gestionar Descuentos 39](#_Toc196733601)

[CU11. Gestionar Carrito de Compras 40](#_Toc196733602)

[CU17. Gestionar Historial de Compras 41](#_Toc196733603)

[CU18. Crear Recomendaciones IA 42](#_Toc196733604)

[CU19. Ejecutar Venta por Voz 43](#_Toc196733605)

[Ciclo #3 44](#_Toc196733606)

[CU8. Gestionar Notificaciones 44](#_Toc196733607)

[CU12. Gestionar Envío 45](#_Toc196733608)

[CU13. Gestionar Agencias de Reparto 46](#_Toc196733609)

[CU14. Gestionar Reportes 48](#_Toc196733610)

[CU15. Gestionar Compra de Inventario 49](#_Toc196733611)

[CU16. Gestionar Stock de Productos 50](#_Toc196733612)

[CU20. Gestionar Sucursales 51](#_Toc196733613)

[2.4. Estructurar el modelo de casos de uso 52](#_Toc196733614)

[Ciclo #1 52](#_Toc196733615)

[Ciclo #2 52](#_Toc196733616)

[Ciclo #3 52](#_Toc196733617)

[3. CAPITULO 3 FLUJO DE TRABAJO: ANALISIS 53](#_Toc196733618)

[3.1. Análisis de arquitectura 53](#_Toc196733619)

[3.1.1. Identificar paquetes 53](#_Toc196733620)

[3.1.2. Relacionar paquetes y casos de uso 54](#_Toc196733621)

[3.2. Análisis de casos de uso 58](#_Toc196733622)

[3.2.1. Diagramas de comunicación 58](#_Toc196733623)

[Ciclo #1 58](#_Toc196733624)

[Ciclo #2 58](#_Toc196733625)

[3.3. Análisis de clase 58](#_Toc196733626)

[3.4. Análisis de paquete 59](#_Toc196733627)

[4. CAPITULO 4 FLUJO DE TRABAJO: DISEÑO 60](#_Toc196733628)

[4.1. Diseño de arquitectura 60](#_Toc196733629)

[4.1.1. Diseño Físico (Diagrama de despliegue) 60](#_Toc196733630)

[4.1.2. Diseño Lógico (Diagrama Organizado en capas) 60](#_Toc196733631)

[4.2. Diseño de Datos 60](#_Toc196733632)

[4.2.1. Diseño de Datos Lógicos 60](#_Toc196733633)

[4.2.1.1. Diagrama de clase 60](#_Toc196733634)

[4.2.1.2. Mapeo 60](#_Toc196733635)

[4.2.1.3. Normalización 60](#_Toc196733636)

[4.2.2. Diseño de datos físicos 60](#_Toc196733637)

[4.2.2.1. Tabla de volumen 60](#_Toc196733638)

[4.2.2.2. Script 60](#_Toc196733639)

[4.2.2.3. Diagrama relacional 60](#_Toc196733640)

[4.2.2.4. Actualización de tuplas 60](#_Toc196733641)

[4.2.2.5. Consultas 60](#_Toc196733642)

[4.2.2.6. Procedimientos almacenados 60](#_Toc196733643)

[4.2.2.7. Triggers 60](#_Toc196733644)

[4.2.3. Diagrama de secuencia 60](#_Toc196733645)

[5. CAPITULO 5 FLUJO DE TRABAJO IMPLEMENTACION 61](#_Toc196733646)

[5.1. Elección de plataforma de desarrollo de software 61](#_Toc196733647)

[5.1.1. Lenguaje de programación 61](#_Toc196733648)

[5.1.2. Base de datos 61](#_Toc196733649)

[5.1.3. Sistemas operativos 62](#_Toc196733650)

[5.1.4. Otros 62](#_Toc196733651)

[5.1.5. Link de repositorio 63](#_Toc196733652)

[5.2. Implementación de la arquitectura del sistema 64](#_Toc196733653)

[5.3. Implementación de la arquitectura del subsistema 64](#_Toc196733654)

[6. CAPITULO 6 FLUJO DE TRABAJO: PRUEBAS 65](#_Toc196733655)

[6.1. Pruebas de autenticación 65](#_Toc196733656)

[6.2. Pruebas de gestión de productos 65](#_Toc196733657)

[BIBLIOGRAFIA 66](#_Toc196733658)

[ANEXOS 66](#_Toc196733659)

[1. Backend 66](#_Toc196733660)

[1.2 Modelos.py 66](#_Toc196733661)

[Usuarios: 66](#_Toc196733662)

[Productos: 68](#_Toc196733663)

[Pedidos: 70](#_Toc196733664)

[Direcciones: 73](#_Toc196733665)

[Sucursales: 73](#_Toc196733666)

[2. Frontend 74](#_Toc196733667)

[2.1 Pages: 74](#_Toc196733668)

[Home: 74](#_Toc196733669)

[Profile: 75](#_Toc196733670)

[Register: 84](#_Toc196733671)

[Login: 90](#_Toc196733672)

[Products: 95](#_Toc196733673)

# PERFIL

## **INTRODUCCION**

En el entorno comercial actual, la competitividad y la satisfacción del cliente dependen en gran medida de la capacidad de las empresas para ofrecer experiencias de compra ágiles, personalizadas e innovadoras. Los sistemas de punto de venta (POS) tradicionales, aunque útiles, se han quedado limitados a funciones básicas de registro y procesamiento de transacciones, sin aprovechar el potencial que ofrecen las nuevas tecnologías para transformar la interacción entre el negocio y sus clientes. Ante esta realidad, surge la necesidad de desarrollar soluciones más inteligentes y adaptativas que no solo simplifiquen los procesos de venta, sino que también enriquezcan la experiencia del usuario mediante herramientas avanzadas.

El proyecto **Smart Cart** nace como una propuesta para modernizar los sistemas POS, integrando inteligencia artificial y reconocimiento de voz con el fin de crear un asistente de compra inteligente. Su objetivo principal es facilitar la interacción entre los clientes y los vendedores, reduciendo tiempos de espera y ofreciendo recomendaciones basadas en los hábitos de consumo. A diferencia de los sistemas convencionales, este proyecto busca analizar el comportamiento histórico de los compradores para predecir sus necesidades y sugerir productos complementarios de manera automática. Por ejemplo, al detectar que un cliente está adquiriendo un dispositivo electrónico, el sistema podría proponer accesorios relacionados, mejorando así la satisfacción y aumentando el valor promedio de cada venta.

Otra innovación clave es la incorporación de reconocimiento de voz, que permitiría a los usuarios realizar acciones mediante comandos hablados, agilizando el proceso en situaciones donde la interacción manual no sea práctica. Esta funcionalidad no solo beneficiaría a los clientes, sino que también optimizaría el trabajo del personal, permitiéndoles enfocarse en brindar un servicio más personalizado en lugar de tareas repetitivas.

Además, se plantea que el sistema incluya herramientas para la gestión de inventario, alertando cuando los productos estén por agotarse, y módulos de reporting que faciliten el análisis de ventas y tendencias. Estas características estarían dirigidas a mejorar la toma de decisiones en el negocio, contribuyendo a una administración más eficiente y proactiva.

En definitiva, **Smart Cart** representa un paso hacia el futuro de los sistemas de punto de venta, donde la tecnología no solo automatiza procesos, sino que también humaniza la experiencia de compra al hacerla más intuitiva y adaptada a las necesidades individuales. Su desarrollo podría marcar un antes y un después en la forma en que las tiendas de electrónica y otros rubros interactúan con sus clientes, posicionándose como un referente de innovación en el sector retail. A lo largo de este proyecto, se explorarán los fundamentos, metodologías y desafíos involucrados en la creación de esta solución, con el fin de sentar las bases para su implementación futura.

## OBJETIVO GENERAL

Diseñar e implementar un sistema de comercio electrónico integral para una tienda de dispositivos electrónicos que permita expandir su operación a nivel nacional, mejorar la experiencia del cliente y aprovechar tecnologías modernas como inteligencia artificial y comandos de voz.

## OBJETIVOS ESPECIFICOS

* Desarrollar una plataforma web y móvil que permita la navegación del catálogo de productos sin necesidad de registro previo.
* Implementar un sistema de registro de usuarios y proceso de compra con diferentes métodos de pago y opciones de entrega.
* Incorporar un módulo de gestión de stock que sincronice en tiempo real la disponibilidad de productos en las distintas sucursales.
* Integrar un sistema de seguimiento de pedidos que permita al cliente conocer el estado de su compra en todo momento.
* Desarrollar un carrito de compras inteligente utilizando machine learning que sugiera productos en función del comportamiento de compra global.

Implementar una funcionalidad de comandos de voz para facilitar la interacción del usuario durante el proceso de compra.

## DESCRIPCIÓN DEL PROBLEMA

La tienda de dispositivos electrónicos cuenta actualmente con una infraestructura sólida basada en un sistema ERP (Enterprise Resource Planning), que permite una gestión adecuada de sus operaciones internas en su oficina central y sus tres sucursales ubicadas en distintas capitales del país. Esta infraestructura ha sido clave para su éxito a nivel local. Sin embargo, pese a este escenario favorable, la empresa ha identificado una serie de limitaciones estratégicas que limitan su crecimiento y capacidad de adaptación a las nuevas dinámicas del mercado.

Uno de los principales problemas detectados es la dependencia total de las sucursales físicas para la comercialización de sus productos. Esta condición limita su capacidad para alcanzar clientes en regiones donde no existe presencia física, desaprovechando una importante parte del mercado nacional. Además, este mismo modelo limita las ventas al horario comercial tradicional, lo que imposibilita brindar atención y servicio las 24 horas del día, factor crítico en un contexto donde los consumidores valoran la inmediatez y disponibilidad constante.

Otro aspecto crítico es la falta de un canal digital de ventas, el cual imposibilita mostrar el catálogo completo de productos a potenciales compradores de forma abierta y accesible. En un mercado donde las decisiones de compra se realizan, en gran medida, desde entornos digitales, la carencia de un sistema de comercio electrónico representa una desventaja competitiva frente a empresas que sí han adoptado estas tecnologías.

Además, el sistema actual no contempla funcionalidades modernas como la gestión automatizada del stock en línea ni la personalización de la experiencia de compra. Esto genera riesgo de vender productos sin disponibilidad y deja de lado herramientas inteligentes que podrían mejorar la conversión de ventas, como sistemas de recomendación basados en machine learning.

También se detecta la ausencia de una experiencia de compra optimizada, que contemple desde la elección del producto hasta el seguimiento del pedido. No se ofrece un sistema que permita a los usuarios escoger entre retiro en tienda o envío a domicilio, ni se brinda información en tiempo real sobre el estado del pedido, lo que reduce la transparencia y la confianza del consumidor.

A nivel tecnológico, el negocio no incorpora tendencias emergentes como la interacción por comandos de voz, que podrían brindar accesibilidad y comodidad adicional en el proceso de compra, especialmente en plataformas móviles.

En resumen, la empresa enfrenta un problema de estancamiento tecnológico: su sistema actual, aunque robusto para operaciones internas, no responde a las nuevas necesidades del consumidor digital ni a la tendencia del comercio electrónico. La falta de un canal de comercio electrónico moderno, accesible y escalable limita su crecimiento, reduce su competitividad y la aleja de estándares actuales en la industria de ventas minoristas.

Por ello, se plantea la necesidad de desarrollar una plataforma e-commerce integral, disponible en versión web y móvil, que permita extender su presencia comercial a nivel nacional, habilitar ventas 24/7, optimizar la experiencia del cliente mediante inteligencia artificial y automatización, e incorporar nuevas tecnologías como comandos de voz y reportes de gestión.

## ALCANCE

El sistema de e-commerce propuesto busca modernizar la tienda mediante la creación de una plataforma web y móvil, que permita llegar a clientes en cualquier parte del país, brindar atención las 24h y utilizar herramientas de última tecnología para mejorar la experiencia de compra.

El sistema incluirá los siguientes módulos:

### Módulo Usuario

* **Gestión de Usuarios**: Registro de datos de usuarios incluyendo número de identificación, nombre, teléfono, sexo, correo, contraseña y domicilio.
* **Administrar Usuarios del sistema:** Administración de usuarios del sistema para acceder al sistema de gestión, incluyendo número de identificación, nombre de usuario, contraseña, roles y permisos de acceso.

### Módulo Autenticación y Autorización

* **Gestión de Roles:** Definición y asignación de roles, se registrará el nombre.
* **Gestión de Permisos:** Asignación de privilegios o permisos a roles y usuarios, con descripción de cada permiso incluyendo nombre y descripción.

### Módulo Inventario de Productos o catalogo

*//añadir atributos cuando definamos la tabla de productos de la tienda*

* **Gestión de productos:** Registro de productos nombre, tipo, color, precio de compra, precio de venta, fotos, si está disponible.
* **Gestión de categorías:** Definición de categorías, con el nombre, descripción y una imagen.
* **Gestión de Sucursal:** Se administra las sucursales de la empresa, nombre, dirección, teléfono, correo, está suspendido.

### Módulo Ventas

* **Gestión de Ventas:** Se registrará las notas de ventas como el código, nombre cliente, nit, monto pagado, monto por cobrar, monto de cambio, método de pago, fecha, hora y detalles de venta como cantidad, precio y subtotal.
* **Gestión de Descuentos:** Se asignan descuentos a categorías de los productos disponibles a la venta, nombre, descripción, tipo (monto, porcentaje), monto, porcentaje, está activo.
* **Método de Pago:** Abarca la gestión de los diferentes métodos de pago disponibles para los usuarios en el sistema e-commerce.
* **Gestión de Carrito de Compras:** Permite a los usuarios agregar, modificar y eliminar productos de su carrito mientras navegan por el sitio web o la aplicación móvil.
* **Gestión Historial de Compras de Cliente:** Permite a los usuarios consultar y revisar sus compras pasadas en la plataforma.
* **Gestión de Recomendaciones con IA:** Permite a los usuarios obtener recomendaciones sobre que productos relacionados podría comprar.
* **Gestión de Compra o Venta por Voz:** Permite al usuario realizar una compra dirigiéndose al sistema mediante su propia voz.

### Módulo Delivery

* **Gestión de envío:** se encarga de coordinar y controlar todo el proceso de entrega de productos comprados, asegurando que lleguen de manera eficiente y oportuna al cliente.
* **Gestión de agencias de reparto:** Se enfoca en la administración de las agencias o empresas encargadas de la entrega de productos.

### Módulo Reportes

* **Gestión de Reportes:** Generación de informes de ventas, inventario y para proporcionar una visión del rendimiento de la tienda.

# MARCO TEORICO

## PARTE 1 – FUNDAMENTACION TEORICA

### E-COMMERCE

En la década de 1990, los comerciantes descubrieron una nueva manera de hacer negocios: Internet. En esencia, el comercio electrónico o ecommerce no es más que la compra y venta de bienes y servicios a través de Internet, cuando compras en línea. Sin embargo, el término se usa a menudo para describir todas las iniciativas de un vendedor a la hora de vender productos en línea directamente a los consumidores. Empieza cuando un cliente potencial descubre un producto y sigue con la compra, el uso y, a ser posible, con la lealtad del cliente.

Los expertos predicen que, para 2025, el mercado mundial de ecommerce estará valorado en más de 24 billones de dólares. Si has estado dándole vueltas a la idea de abrir un negocio, ahora es un buen momento.

Los datos alimentan las operaciones de ecommerce más satisfactorias, que aprovechan prácticas recomendadas como el marketing por correo electrónico, la segmentación del público y la automatización de marketing. Por ejemplo, el seguimiento de un cliente que ha puesto un artículo en el carrito de la compra en línea, pero no ha completado la transacción, puede aumentar significativamente las probabilidades de que finalice la venta.

Dicho esto, toda persona que desee abrir una tienda en línea y beneficiarse del auge del ecommerce debe conocer sus fundamentos.

Definición

Mucha gente define el ecommerce como la venta o compra de un producto físico en línea. No obstante, el ecommerce incluye también la venta y compra de bienes no físicos, como servicios o productos digitales. Básicamente, nos referimos a cuando una empresa vende en línea.

Algunos minoristas de ecommerce venden exclusivamente en línea. Por ejemplo, digamos que un empresario crea una empresa que vende productos de alta gama para mascotas. En los tiempos anteriores a Internet, tendría dos opciones: vendería su mercancía a través de su propia tienda, o la vendería a las tiendas nacionales de productos para mascotas. Hoy en día, los propietarios de negocios tienen una tercera opción: el ecommerce. Pueden vender sus productos a través de su propio sitio web, un sitio de terceros o en ambos.

Las líneas de las ecommerce y de la venta minorista tradicional a veces se solapan. Cuando un cliente compra con su smartphone, mientras está en una tienda física navegando por sus productos físicos, es difícil clasificar la experiencia como una u otra.

**Ventajas del e-commerce:**

Costes bajos de puesta en marcha y funcionamiento: Por lo general, iniciar un negocio de ecommerce es mucho más barato que abrir una tienda física porque no hay costes de alquiler ni de inmuebles, y puedes empezar con menos empleados.

Ofrece flexibilidad: El ecommerce es una solución más flexible para las empresas y sus clientes porque estos pueden comprar en cualquier momento del día y desde cualquier lugar.

Fácil de compaginar con el marketing basado en datos: Tu tienda de ecommerce puede proporcionarte un montón de datos de marketing a través de análisis e informes para ayudarte a tomar mejores decisiones para la promoción de tu negocio.

Amplia selección de productos disponibles: El ecommerce ofrece una mayor variedad de productos a los clientes en comparación con las tiendas físicas, que solo pueden mantener un inventario limitado en su espacio físico.

**Inconvenientes del e-commerce:**

Menor conexión con los clientes: Dado que no hay nadie con quien interactuar al realizar la compra, los clientes pueden sentirse menos conectados con tu negocio. Los clientes tampoco tendrán la oportunidad de tocar el producto antes de comprarlo para asegurarse de que es lo que realmente quieren.

Requiere más coordinación para la logística de los envíos: Los clientes de ecommerce pueden estar en cualquier parte del mundo, por lo que los empresarios deben asegurarse de que sus envíos y logística puedan llevar los productos a su destino de manera oportuna. Los clientes también tendrán que esperar para recibir sus productos.

Mínima atención al cliente: Muchas tiendas de ecommerce no cuentan con un representante de atención al cliente en todo momento para responder preguntas y resolver problemas.

### Como usuario

Desde el punto de vista del usuario, un sistema de e-commerce debe proporcionar una experiencia fluida, intuitiva y segura. Los usuarios esperan poder buscar productos fácilmente, obtener información clara, comprar sin complicaciones y recibir soporte ante cualquier problema.

Algunos ejemplos de grandes e-commerces exitosos pueden ser los siguientes:

#### Amazon

Amazon es la empresa líder en comercio electrónico a nivel mundial. Su éxito se basa en su enorme variedad de productos, un sistema de recomendación muy preciso, entregas rápidas y confiables, y una experiencia de usuario optimizada. Amazon también ha incorporado asistentes virtuales como Alexa para compras por voz. Amazon se destaca por su:

* Experiencia de Usuario: Interfaz intuitiva que facilita la búsqueda y compra de productos.
* Sistema de Recomendaciones: Basado en el historial de compras y navegación del usuario.
* Amazon Prime: Un servicio de suscripción que ofrece envíos rápidos y gratuitos, acceso a contenido de streaming, entre otros beneficios.
* Gestión de Inventarios: Uso de centros de distribución para almacenar y gestionar el inventario, permitiendo tiempos de envío rápidos.

**Interfaz de Usuario:**

Búsqueda y Navegación: Los usuarios pueden buscar productos mediante una barra de búsqueda o explorar categorías específicas. Amazon utiliza algoritmos para mostrar productos relevantes basados en el historial de navegación y compras del usuario.

Detalle del Producto: Cada producto tiene una página de detalles con descripciones, imágenes, opiniones de clientes y opciones de compra.

**Proceso de Compra:**

* Carrito de Compras: Los usuarios agregan productos al carrito, donde pueden revisar, modificar o eliminar artículos antes de proceder al checkout.
* Pago: Amazon ofrece múltiples métodos de pago, incluyendo tarjetas de crédito, débito, y Amazon Pay. El proceso de pago es seguro y puede incluir opciones de financiamiento.
* Confirmación y Envío: Tras la compra, los usuarios reciben una confirmación por correo electrónico y pueden rastrear el estado del envío en su cuenta de Amazon.

**Logística y Almacenamiento:**

* Centros de Distribución: Amazon opera numerosos centros de distribución para almacenar productos y cumplir con los pedidos. Utiliza tecnología avanzada y robots para gestionar el inventario y el procesamiento de pedidos.
* Envío: Ofrece opciones de envío estándar y exprés, con tiempos de entrega que varían según la ubicación y la opción seleccionada.

**Postventa:**

* Atención al Cliente: Amazon proporciona soporte al cliente para resolver problemas con pedidos, devoluciones y reembolsos.
* Devoluciones: Los usuarios pueden devolver productos dentro de un período determinado y recibir un reembolso o reemplazo.

#### Alibaba

Alibaba es una plataforma china que conecta a compradores con fabricantes y proveedores. A diferencia de Amazon, su enfoque es más B2B (business to business). Ofrece precios competitivos y opciones de personalización para grandes volúmenes de compra, convirtiéndose en una referencia mundial del comercio digital mayorista. Sus características incluyen:

* Plataforma B2B y B2C: Facilita tanto el comercio entre empresas como las ventas directas a consumidores.
* Alibaba Cloud: Ofrece servicios de computación en la nube para apoyar las operaciones de e-commerce.
* Sistema de Pago Alipay: Proporciona una solución segura para pagos en línea y transacciones financieras.

**Interfaz de Usuario:**

* Búsqueda y Navegación: Los usuarios pueden buscar productos a través de una barra de búsqueda o explorar categorías específicas. Alibaba conecta a compradores con proveedores mayoristas y fabricantes.
* Perfil del Vendedor: Cada proveedor tiene un perfil que muestra su información de contacto, productos ofrecidos y condiciones de venta.

**Proceso de Compra:**

* Consulta y Negociación: Los compradores pueden enviar consultas a proveedores para negociar precios y términos. Alibaba facilita la comunicación entre compradores y proveedores.
* Orden y Pago: Los compradores realizan pedidos a través de la plataforma y pueden utilizar Alibaba’s Trade Assurance para proteger las transacciones. Los pagos se realizan a través de la plataforma o mediante métodos acordados con el proveedor.

**Logística y Almacenamiento:**

* Servicios de Logística: Alibaba ofrece servicios de logística y gestión de envíos a través de socios y su red de servicios de entrega, facilitando el transporte de productos a nivel global.

**Postventa:**

* Resolución de Disputas: Alibaba proporciona un sistema de resolución de disputas para manejar conflictos entre compradores y proveedores.

#### Shopify

Shopify es una solución SaaS (Software as a Service) que permite a emprendedores y empresas crear tiendas en línea personalizadas sin necesidad de conocimientos técnicos avanzados. Es conocida por su facilidad de uso, integraciones con métodos de pago y herramientas de marketing. Sus características incluyen:

* Facilidad de Uso: Herramientas intuitivas para crear y gestionar una tienda en línea.
* Personalización: Temas y aplicaciones que permiten personalizar la apariencia y funcionalidad de la tienda.
* integraciones: Compatible con diversos servicios de pago y herramientas de marketing.
* Soporte y Escalabilidad: Ofrece soporte 24/7 y opciones para escalar a medida que el negocio crece.

**Interfaz de Usuario:**

* Creación de Tienda: Los usuarios pueden crear tiendas en línea utilizando plantillas prediseñadas y herramientas de personalización proporcionadas por Shopify.
* Gestión de Productos: Permite agregar productos, establecer precios, y gestionar inventarios desde un panel de administración.

**Proceso de Compra:**

* Carrito de Compras y Checkout: Los clientes pueden agregar productos al carrito y proceder al checkout, donde ingresan su información de envío y eligen un método de pago.
* Pago y Confirmación: Shopify admite varios métodos de pago, incluidos Shopify Payments, PayPal y tarjetas de crédito. Los clientes reciben confirmaciones de pedido y detalles de seguimiento.

**Logística y Almacenamiento:**

* Integración con Proveedores de Envío: Shopify se integra con diversos proveedores de envío para facilitar el procesamiento de pedidos y la generación de etiquetas.

**Postventa:**

* Soporte y Gestión: Shopify ofrece soporte técnico y herramientas para gestionar devoluciones y atención al cliente.

**1.1.2. COMO DESARROLLADOR:**

**1.1.2.1 Magento**

Magento es una plataforma de e-commerce de código abierto que permite a los desarrolladores crear tiendas en línea altamente personalizables. Fundada en 2008 y adquirida por Adobe, Magento ofrece:

* Flexibilidad y Personalización: Permite ajustes extensos en la apariencia y funcionalidad de la tienda.
* Escalabilidad: Adecuada para empresas de todos tamaños, desde pequeñas start ups hasta grandes corporaciones.
* Comunidad Activa: Gran cantidad de extensiones y soporte de la comunidad.

**Configuración de la Tienda:**

* Instalación: Los desarrolladores instalan Magento en un servidor web. Magento puede ser alojado en servidores locales o en la nube.
* Personalización: Magento permite la personalización extensiva mediante temas y módulos. Los desarrolladores pueden crear y ajustar funcionalidades según las necesidades del negocio.

**Gestión de Productos:**

* Catalogación: Los productos se añaden al catálogo, se agrupan en categorías y se establecen atributos como precios, descripciones y opciones.

**Proceso de Compra:**

* Carrito de Compras y Checkout: Magento gestiona el carrito de compras y el proceso de checkout, con soporte para múltiples métodos de pago y opciones de envío.
* Pagos y Envíos: Se integran soluciones de pago y proveedores de servicios de envío.

**Administración:**

* Panel de Administración: Proporciona herramientas para gestionar pedidos, clientes, inventario y reportes.

**1.1.2.2 PrestaShop**

PrestaShop es una solución de e-commerce de código abierto que facilita la creación de tiendas en línea con un enfoque en la facilidad de uso y la personalización. Lanzada en 2007, PrestaShop ofrece:

* Interfaz Intuitiva: Panel de administración fácil de usar para gestionar productos y pedidos.
* Extensiones y Módulos: Amplia gama de módulos y temas para personalizar la tienda.
* Multilingüe y Multimoneda: Soporte para múltiples idiomas y monedas.

**Configuración de la Tienda:**

* Instalación: PrestaShop se instala en un servidor web, y los usuarios pueden elegir entre una instalación en la nube o en sus propios servidores.
* Personalización: Ofrece una variedad de temas y módulos para adaptar la tienda a las necesidades específicas del negocio.

**Gestión de Productos:**

* Añadir Productos: Los usuarios pueden agregar productos al catálogo, configurar precios, descripciones y atributos, y gestionar el inventario.

**Proceso de Compra:**

* Checkout y Pagos: Incluye un proceso de checkout simplificado y soporta múltiples métodos de pago, como tarjetas de crédito y transferencias bancarias.
* Envío: Permite la configuración de opciones de envío y tarifas. Administración:
* Panel de Control: Ofrece herramientas para gestionar productos, pedidos, clientes y reportes de ventas.

**1.1.2.3 WooCommerce**

WooCommerce es un plugin para WordPress que convierte un sitio web en una tienda en línea. Lanzado en 2011, es popular por su integración con WordPress y su flexibilidad. Ofrece:

* Facilidad de Integración: Integración directa con sitios web de WordPress.
* Extensiones: Gran cantidad de plugins y temas disponibles.
* Escalabilidad: Adecuado para tiendas pequeñas y grandes.

**Configuración de la Tienda:**

* Instalación: WooCommerce es un plugin que se instala en un sitio web de WordPress. La configuración inicial incluye la selección de una plantilla y la configuración de opciones básicas.

**Gestión de Productos:**

* Añadir Productos: Los usuarios pueden añadir productos al catálogo, definir precios, descripciones y categorías, y gestionar inventario.

**Proceso de Compra:**

* Carrito y Checkout: WooCommerce gestiona el carrito de compras y el proceso de checkout, con opciones para múltiples métodos de pago y configuraciones de envío.
* Pagos y Envíos: Integración con diversos servicios de pago y opciones de envío.

**Administración:**

* Panel de Administración: Ofrece herramientas para gestionar productos, pedidos, clientes y reportes.

## PASARELAS DE PAGO

Las pasarelas de pago son servicios que permiten a los comerciantes aceptar pagos electrónicos de forma segura. Actúan como intermediarios entre el comerciante, el cliente y las instituciones financieras para garantizar que las transacciones sean procesadas de manera segura y eficiente. A continuación, se detalla cómo funcionan las distintas formas de pago online y se examinan las pasarelas

* + 1. FORMAS DE PAGO ONLINE

#### Tarjetas de Débito y Crédito

* Proceso: Cuando un cliente paga con una tarjeta de débito o crédito, la información de la tarjeta se transmite a través de una conexión segura a la pasarela de pago. La pasarela verifica la validez de la tarjeta y la disponibilidad de fondos mediante una solicitud a la red de tarjetas (Visa, MasterCard, etc.).
* Autorización: La entidad emisora de la tarjeta autoriza la transacción si los detalles son correctos y los fondos están disponibles. La respuesta se envía de vuelta a la pasarela, que luego comunica el resultado al comerciante.
* Confirmación: Una vez que la transacción es aprobada, el dinero se transfiere del banco del cliente al banco del comerciante.

#### QR Codes

* Proceso: Los clientes escanean un código QR proporcionado por el comerciante usando su aplicación de pago móvil. El código QR contiene la información de la transacción y el monto.
* Autorización: La aplicación del cliente envía una solicitud de pago a la pasarela de pago, que verifica la información y autoriza la transacción.
* Confirmación: La transacción es confirmada y el monto es transferido del cliente al comerciante.

#### Transferencias Bancarias

* Proceso: El cliente realiza una transferencia bancaria directa desde su cuenta al banco del comerciante. Esto se puede hacer a través de una aplicación de banca en línea o mediante un banco físico.
* Autorización: La transferencia se procesa a través del sistema bancario. Puede requerir una verificación adicional como un código de confirmación.
* Confirmación: La transferencia se completa y el dinero es transferido al comerciante una vez que se confirma el pago.

### LIBÉLULA

LIBÉLULA es una pasarela de pago popular en América Latina, especialmente en Bolivia. Su funcionamiento se basa en los siguientes aspectos:

**Proceso de Pago:**

* Integración: Los comerciantes integran LIBÉLULA en sus sitios web mediante un API que permite a los clientes realizar pagos en línea de forma segura.
* Autorización: Cuando un cliente realiza un pago, LIBÉLULA verifica la información del pago a través de su red y la red bancaria correspondiente.
* Confirmación: LIBÉLULA confirma la transacción y notifica al comerciante y al cliente sobre el estado del pago.

**Servicios Ofrecidos:**

* Pagos con Tarjeta: Acepta pagos con tarjetas de crédito y débito.
* Pagos en Efectivo: Permite a los clientes pagar en efectivo en puntos de venta asociados.
* Seguridad: Implementa medidas de seguridad como la encriptación de datos para proteger la información de los usuarios.

### STRIPE

Stripe es una pasarela de pago internacional ampliamente utilizada que ofrece soluciones para aceptar pagos en línea. A continuación, se describe su funcionamiento:

**Proceso de Pago:**

* Integración: Los comerciantes integran Stripe en sus sitios web y aplicaciones móviles utilizando las APIs de Stripe. Esto permite a los usuarios ingresar sus datos de pago de manera segura.
* Autorización: Stripe verifica la información del pago en tiempo real a través de las redes de tarjetas y otras instituciones financieras para autorizar la transacción.
* Confirmación: Una vez autorizada, Stripe procesa el pago y transfiere los fondos al banco del comerciante, descontando las tarifas correspondientes.

**Servicios Ofrecidos:**

* Métodos de Pago: Acepta tarjetas de crédito y débito, pagos ACH, transferencias bancarias y otros métodos locales.
* Suscripciones: Permite la gestión de suscripciones y pagos recurrentes. Seguridad: Implementa estándares de seguridad como PCI DSS para proteger la información de los clientes y prevenir fraudes.

## DELIVERY

El término delivery se refiere al proceso de entrega de productos desde un comerciante o almacén hasta el cliente final. En el contexto de e-commerce, el delivery es una parte crucial del servicio, ya que afecta la satisfacción del cliente y la eficiencia del proceso de compra. A continuación, se detalla cómo funcionan los servicios de delivery y cómo se calculan los pagos para una entrega.

### FUNCIONAMIENTO

#### Proceso de Entrega:

* Pedido: El cliente realiza un pedido a través de un sitio web o una aplicación móvil. Una vez confirmado el pedido, se genera una orden de entrega que contiene los detalles del producto, la dirección de envío y la información de contacto del cliente.
* Preparación del Pedido: El comerciante o el almacén prepara el pedido. Esto puede implicar la recolección de productos de inventario, el embalaje y la preparación para el envío.
* Asignación de Repartidor: El pedido es asignado a un repartidor o una empresa de logística. Esto puede hacerse internamente o a través de una empresa de delivery externa.
* Transporte: El repartidor recoge el pedido y lo transporta al destino del cliente. La ruta y el tiempo de entrega pueden optimizarse utilizando tecnologías de navegación y algoritmos de optimización de rutas.
* Entrega: El repartidor entrega el pedido al cliente en la dirección proporcionada. En algunos casos, el cliente puede tener la opción de rastrear el estado de su entrega en tiempo real.
* Confirmación y Cierre: Una vez realizada la entrega, se actualiza el estado del pedido y se puede solicitar una confirmación de recepción del cliente. El sistema de delivery cierra la orden y procesa cualquier pago pendiente.

#### Tecnologías Utilizadas:

* Sistemas de Gestión de Entregas (Delivery Management Systems): Estos sistemas ayudan a coordinar y optimizar el proceso de entrega, gestionando rutas, horarios y recursos de manera eficiente.
* Rastreo en Tiempo Real: Los sistemas de rastreo permiten a los clientes y comerciantes seguir el estado de la entrega en tiempo real, proporcionando actualizaciones sobre la ubicación del repartidor y el tiempo estimado de llegada.
* Algoritmos de Optimización de Rutas: Utilizan datos de tráfico y otros factores para determinar la ruta más eficiente para las entregas, reduciendo el tiempo y los costos asociados.

### CÁLCULOS DE PAGO

El cálculo del costo de entrega suele depender de varios factores, entre ellos:

* Distancia: El costo de entrega generalmente aumenta con la distancia entre el punto de origen y el destino. Se pueden utilizar tarifas por kilómetro o zonas geográficas para calcular este costo.
* Peso y Tamaño del Paquete: Paquetes más grandes y pesados suelen costar más en términos de entrega. Las empresas de delivery suelen calcular el costo en función del peso volumétrico y físico del paquete.
* Frecuencia de Entrega: En algunos casos, los servicios de delivery pueden ofrecer tarifas reducidas para entregas recurrentes o suscripciones, en comparación con entregas únicas.
* Tipo de Entrega: El costo puede variar según el tipo de entrega solicitado, como estándar, exprés, o mismo día. Las entregas más rápidas suelen tener un costo más alto.
* Zona de Entrega: Las tarifas pueden variar según la ubicación del cliente. Las entregas a áreas urbanas pueden tener un costo diferente en comparación con las entregas a áreas rurales o remotas.

## PUDS

El Proceso Unificado de Desarrollo de Software (PUDS), también conocido como Unified Process (UP), es una metodología para el desarrollo de software que se basa en un enfoque iterativo e incremental. El objetivo es proporcionar una guía estructurada y flexible para desarrollar software de alta calidad. A continuación, se detalla cómo funciona el PUDS y sus principales características:

### Fases del Proceso Unificado:

* **Inicio (Inception):** En esta fase, se define el alcance del proyecto y se identifican los requerimientos principales. Se establece el caso de negocio, los objetivos y la viabilidad del proyecto. Esta fase también implica la creación de un plan de proyecto inicial y la identificación de riesgos.
* **Elaboración (Elaboration):** En esta fase, se desarrolla un entendimiento más profundo de los requerimientos y se refina la arquitectura del sistema. Se crea un modelo detallado del sistema y se identifican los riesgos técnicos. La planificación de iteraciones y la definición de un prototipo de arquitectura son actividades clave en esta fase.
* **Construcción (Construction):** Durante la fase de construcción, se implementa el sistema basado en la arquitectura y los requerimientos definidos. Esta fase se lleva a cabo en iteraciones, donde cada iteración añade nuevas funcionalidades y refina el sistema. Se realizan pruebas continuas y se ajusta el desarrollo según los resultados de las pruebas.
* **Transición (Transition):** En esta fase, el sistema se pone en producción y se prepara para su uso por parte de los usuarios finales. Se realizan pruebas finales, se entrena a los usuarios y se realizan ajustes basados en los comentarios y problemas encontrados durante el uso inicial.

### Características del Proceso Unificado:

* Iterativo e Incremental: El PUDS se basa en ciclos repetitivos de desarrollo y entrega incremental de funcionalidades. Esto permite recibir retroalimentación temprana y ajustar el desarrollo según sea necesario.
* Orientado a Riesgos: Se enfoca en identificar y mitigar riesgos a lo largo del proceso de desarrollo, lo que ayuda a evitar problemas importantes en etapas posteriores.
* Basado en Componentes: Promueve el desarrollo de software en componentes modulares, lo que facilita la integración y reutilización de componentes.
* Documentación y Modelado: Utiliza modelos y documentación para definir y comunicar los requisitos, la arquitectura y el diseño del sistema.

## UML

UML (Unified Modeling Language) es un lenguaje de modelado estándar utilizado en el diseño y especificación de sistemas de software. Proporciona una serie de diagramas y herramientas para visualizar, especificar, construir y documentar los componentes de un sistema.

### Diagramas de UML:

* **Diagrama de Casos de Uso**: Representa las funcionalidades del sistema desde la perspectiva del usuario. Muestra cómo los usuarios (actores) interactúan con el sistema para lograr ciertos objetivos.
* **Diagrama de Clases:** Describe la estructura estática del sistema, mostrando las clases, atributos y relaciones entre ellas. Es útil para entender el diseño y las interacciones entre los componentes del sistema.
* **Diagrama de Secuencia:** Representa la interacción entre objetos en el sistema a lo largo del tiempo. Muestra cómo los mensajes se envían entre objetos para realizar una función específica.
* **Diagrama de Actividades:** Modela el flujo de trabajo o los procesos dentro del sistema. Es útil para representar los procedimientos y las decisiones dentro de un proceso.
* **Diagrama de Estados:** Describe los estados de un objeto a lo largo de su vida útil y las transiciones entre esos estados. Es útil para modelar el comportamiento dinámico del sistema.
* **Diagrama de Componentes:** Muestra la organización y las dependencias entre los componentes físicos del sistema, como bibliotecas y módulos.
* **Diagrama de Despliegue:** Representa la distribución física de los componentes del sistema en el hardware y las redes.

### Características de UML:

* Estándar y Amplio: UML es un estándar ampliamente aceptado y cubre una amplia gama de aspectos del diseño de software.
* Flexible y Extensible: UML se puede adaptar a las necesidades específicas del proyecto y se puede extender con estereotipos y anotaciones personalizadas.
* Documentación Clara: Facilita la documentación y la comunicación entre los miembros del equipo de desarrollo, proporcionando una representación visual del sistema.

# CAPITULO 2 FLUJO DE TRABAJO: CAPTURA DE REQUISITOS

## Identificar actores y casos de uso

### Identificar actores

|  |  |  |
| --- | --- | --- |
| **ID** | **ACTOR** | **DESCRIPCIÓN** |
| **A1** | Administrador Super Usuario (ASU) | Responsable de la gestión de módulos y permisos. |
| **A2** | Cliente (C) | Usuario que realiza compras en la plataforma. |
| **A3** | Agencia de Delivery (D) | Encargado de realizar los envíos al cliente. |

### Identificar casos de uso

|  |  |  |
| --- | --- | --- |
| **ID** | **NOMBRE** | **ENTORNO** |
| **CU1** | Gestionar Usuario | WEB, MÓVIL |
| **CU2** | Gestionar Rol | WEB |
| **CU3** | Gestionar Permisos | WEB |
| **CU4** | Iniciar Sesión | WEB, MÓVIL |
| **CU5** | Cerrar Sesión | WEB, MÓVIL |
| **CU6** | Gestionar Productos | WEB, MÓVIL(LISTAR) |
| **CU7** | Gestionar Categorías | WEB, MÓVIL(LISTAR) |
| **CU8** | Gestionar Notificaciones | MÓVIL(LISTAR) |
| **CU9** | Gestionar Ventas | WEB, MÓVIL |
| **CU10** | Gestionar Descuentos | WEB, MÓVIL(LISTAR) |
| **CU11** | Gestionar Carrito de Compras | WEB, MÓVIL |
| **CU12** | Gestionar Envío | WEB, MÓVIL |
| **CU13** | Gestionar Agencias de Reparto | WEB |
| **CU14** | Gestionar Reportes | WEB |
| **CU15** | Gestionar Compra de Inventario | WEB |
| **CU16** | Gestionar Ordenes de Compra | WEB |
| **CU17.** | Gestionar Historial de compras | WEB, MOVIL |
| **CU18.** | Crear Recomendaciones IA | WEB, MOVIL |
| **CU19.** | Ejecutar Venta por Voz | MOVIL |
| **CU20.** | Gestionar Sucursales | WEB, MOVIL(LISTAR) |

## Priorizar casos de uso

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NOMBRE** | **ACTOR** | **PRIORIDAD** |
| **CU1** | Gestionar Usuario | ASU, C | ALTA |
| **CU2** | Gestionar Rol | ASU | ALTA |
| **CU3** | Gestionar Permisos | ASU | ALTA |
| **CU4** | Iniciar Sesión | ASU,C,D | ALTA |
| **CU5** | Cerrar Sesión | ASU,C,D | ALTA |
| **CU6** | Gestionar Productos | ASU | MEDIA |
| **CU7** | Gestionar Categorías | ASU | BAJA |
| **CU8** | Gestionar Notificaciones | ASU | BAJA |
| **CU9** | Gestionar Ventas | ASU, C | ALTA |
| **CU10** | Gestionar Descuentos | ASU | BAJA |
| **CU11** | Gestionar Carrito de Compras | C | ALTA |
| **CU12** | Gestionar Envío | D, C | BAJA |
| **CU13** | Gestionar Agencias de Reparto | ASU | ALTA |
| **CU14** | Gestionar Reportes | ASU, D | MEDIA |
| **CU15** | Gestionar Compra de Inventario | ASU | ALTA |
| **CU16.** | Gestionar Ordenes de compra | ASU | ALTA |
| **CU17.** | Gestionar Historial de compras | C | ALTA |
| **CU18.** | Crear Recomendaciones IA | ASU | ALTA |
| **CU19.** | Ejecutar Venta por Voz | ASU | ALTA |
| **CU20.** | Gestionar Sucursales | ASU | BAJA |

### Ciclo #1

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NOMBRE** | **ACTOR** | **PRIORIDAD** |
| **CU1** | Gestionar Usuario | ASU, C | ALTA |
| **CU2** | Gestionar Rol | ASU | ALTA |
| **CU3** | Gestionar Permisos | ASU | ALTA |
| **CU4** | Iniciar Sesión | ASU,C,D | ALTA |
| **CU5** | Cerrar Sesión | ASU,C,D | ALTA |
| **CU6** | Gestionar Productos | ASU | MEDIA |
| **CU7** | Gestionar Categorías | ASU | BAJA |

### Ciclo #2

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NOMBRE** | **ACTOR** | **PRIORIDAD** |
| **CU9** | Gestionar Ventas | ASU,C | ALTA |
| **CU10** | Gestionar Descuentos | ASU | BAJA |
| **CU11** | Gestionar Carrito de Compras | C | ALTA |
| **CU17.** | Gestionar Historial de compras | C | ALTA |
| **CU18.** | Crear Recomendaciones IA | ASU, C | ALTA |
| **CU19.** | Crear Venta por Voz | C | ALTA |

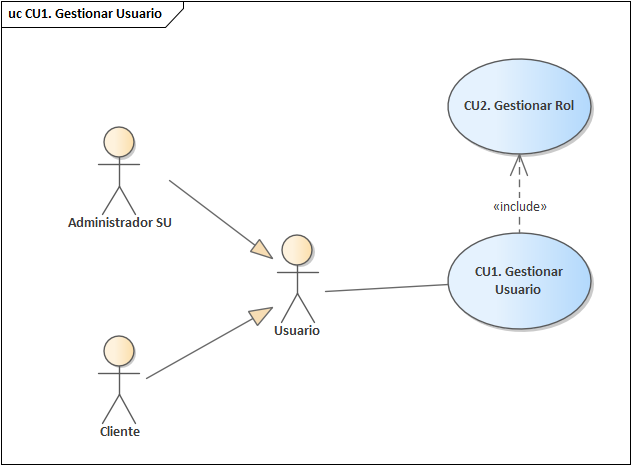
### Ciclo #3

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **NOMBRE** | **ACTOR** | **PRIORIDAD** |
| **CU8** | Gestionar Notificaciones | ASU | BAJA |
| **CU12** | Gestionar Envío | D,C | MEDIA |
| **CU13** | Gestionar Agencias de Reparto | ASU | MEDIA |
| **CU14** | Gestionar Reportes | ASU | BAJA |
| **CU15** | Gestionar Compra de Inventario | ASU | ALTA |
| **CU16** | Gestionar Stock de Productos | ASU | ALTA |
| **CU20.** | Gestionar Sucursales | ASU | BAJA |

## Especificar casos de uso

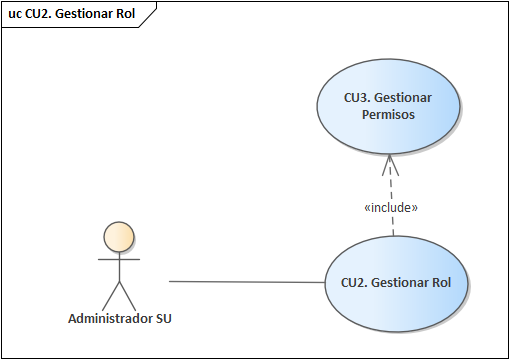
### Ciclo #1

#### CU1. Gestionar Usuario



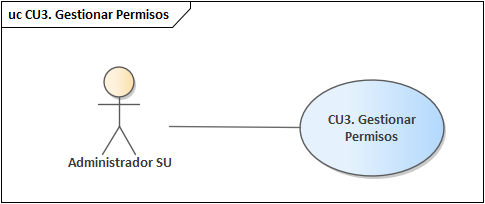
|  |  |
| --- | --- |
| **Nombre de CU** | CU1. Gestionar Usuario |
| **Propósito** | Registrar a los usuarios que puedan entrar al sistema |
| **Actores** | Administrador Super Usuario, Cliente |
| **Actor iniciador** | Administrador Super Usuario, Cliente |
| **Precondición** | CU2. Gestionar Rol |
| **Flujo principal** | Obtener usuarios   * Listar todos los usuarios * Visualizar un usuario * Buscar un usuario Registrar usuarios * Ingresar los datos personales del usuario * Asignar rol * Asignar a una sucursal (opcional) Modificar usuarios * Buscar usuario existente en el sistema * Obtener datos del usuario * Actualizar los datos Eliminar * Buscar usuario activo en el sistema   Se deshabilita el usuario |
| **Postcondición** | Ninguna. |
| **Excepción** | Usuario ya existente  Contraseña mínima requerida (8 caracteres)  Email inválido |

#### CU2. Gestionar Rol



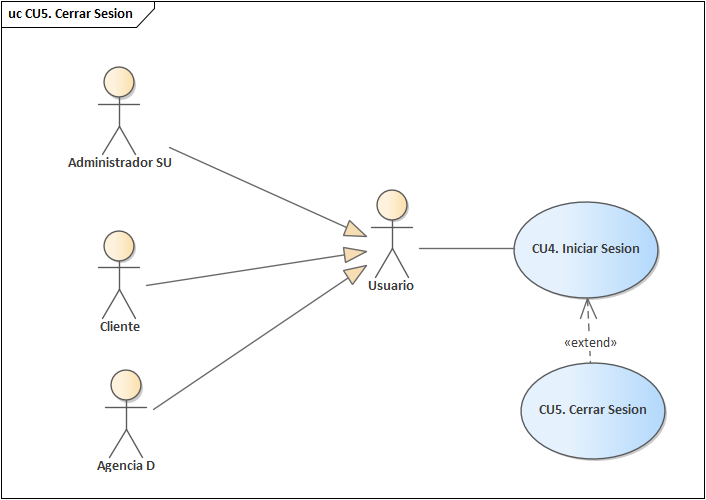
|  |  |
| --- | --- |
| **Nombre de CU** | CU2. Gestionar Rol |
| **Propósito** | Asignar roles a los usuarios que están registrados en el sistema |
| **Actores** | Administrador Super Usuario |
| **Actor iniciador** | Administrador Super Usuario |
| **Precondición** | CU3. Gestionar permisos |
| **Flujo principal** | Registrar Rol   * Ingresar los datos del rol al sistema * Asignar los privilegios correspondientes al rol Modificar Rol * Buscar rol existente en el sistema * Obtener datos del rol * Actualizar datos Eliminar Rol * Buscar rol existente en el sistema   Deshabilitar rol |
| **Postcondición** | Ninguna |
| **Excepción** | No se puede registrar si el rol ya existe  No se puede eliminar si el rol está asignado |

#### CU3. Gestionar Permisos



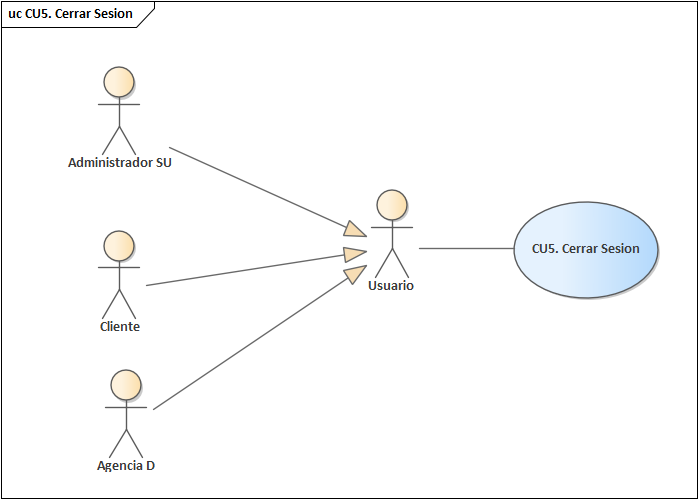
|  |  |
| --- | --- |
| **Nombre de CU** | CU3. Gestionar permisos |
| **Propósito** | Asignar permisos a los roles de los usuarios para tener acceso a diferentes partes del sistema |
| **Actores** | Administrador Super Usuario |
| **Actor iniciador** | Administrador Super Usuario |
| **Precondición** | Ninguna |
| **Flujo principal** | Registrar Permiso   * Ingresar los datos del permiso al sistema   Modificar Permiso   * Buscar permiso existente en el sistema * Obtener datos del permiso * Actualizar datos Eliminar Permiso * Buscar permiso existente en el sistema * Deshabilitar permiso |
| **Postcondición** | Ninguna |
| **Excepción** | No se puede eliminar si el permiso está asignado No se puede registrar si el permiso ya existe |

#### CU4. Iniciar Sesión



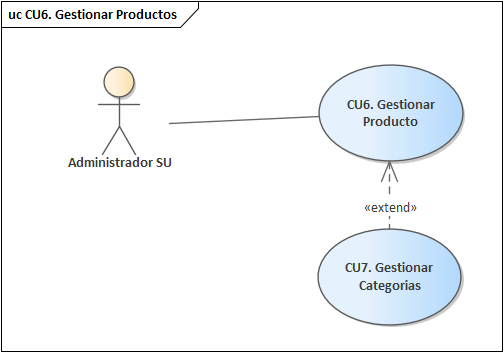
|  |  |
| --- | --- |
| **Nombre de CU** | C4. Iniciar Sesión |
| **Propósito** | Permitir el inicio de sesión a usuarios registrados en el sistema. |
| **Actores** | Administrador Super Usuario, Cliente, Agencia Delivery |
| **Actor iniciador** | Administrador Super Usuario, Cliente, Agencia Delivery |
| **Precondición** | CU1. Gestionar usuarios  CU2. Gestionar roles |
| **Flujo principal** | Ingreso de datos:   * Ingresar los datos requeridos para el login. Autenticar los datos: * Si los datos coinciden con un usuario registrado. |
| **Postcondición** | CU5. Cerrar Sesión |
| **Excepción** | Datos incorrectos. |

#### CU5. Cerrar Sesión



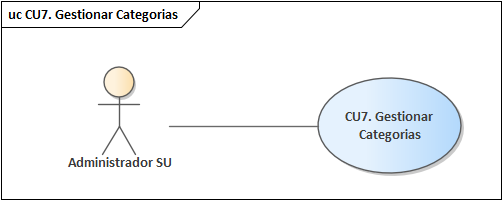
|  |  |
| --- | --- |
| **Nombre de CU** | C5. Cerrar Sesión |
| **Propósito** | Permitir cerrar una sesión abierta en el sistema. |
| **Actores** | Administrador Super Usuario, Cliente, Agencia Delivery |
| **Actor iniciador** | Administrador Super Usuario, Cliente, Agencia Delivery |
| **Precondición** | CU4. Iniciar Sesión |
| **Flujo principal** | Se solicita el cierre de sesión |
| **Postcondición** | Ninguna |
| **Excepción** | Ninguna |

#### CU6 Gestionar Productos



|  |  |
| --- | --- |
| **Nombre de CU** | CU6. Gestionar Productos |
| **Proposito** | Administrar los productos del sistema. |
| **Actores** | Administrador Super Usuario |
| **Actor iniciador** | Administrador Super Usuario |
| **Precondición** | CU7. Gestionar Categorias |
| **Flujo Principal** | Registrar Producto:   * El usuario selecciona la opción para agregar un nuevo producto. * Se ingresan los detalles del producto, como nombre, descripción, precio, etc. * El producto se registra en el sistema. Modificar Producto: * El usuario busca un producto existente en el sistema. * Se accede a la información actual del producto. * Se actualizan los detalles del producto según sea necesario. Eliminar Producto: * El usuario busca y selecciona un producto existente en el sistema.   El producto se elimina de la base de datos |
| **Postcondición** | Ninguna |
| **Excepción** | Producto ya existente. |

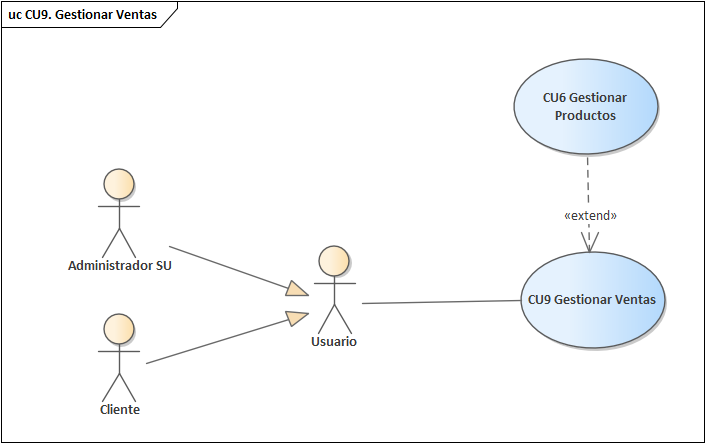
#### CU7 Gestionar Categorías



|  |  |
| --- | --- |
| **Nombre de CU** | CU7. Gestionar Categorías |
| **Proposito** | Administrar las categorias de productos en el sistema. |
| **Actores** | Gerente de sucursal |
| **Actor iniciador** | Gerente de sucursal |
| **Precondición** | Ninguna |
| **Flujo Principal** | Registrar Categoría:   * El usuario agrega una nueva categoría para organizar productos. * Se ingresan los detalles como nombre, descripción, atributos, etc. * La nueva categoría se registra en el sistema. Modificar Categoría: * El usuario busca una categoría existente en el sistema. * Se accede a los detalles actuales y se realiza la modificación según sea necesario.   Eliminar Categoría:   * El usuario selecciona una categoría existente para eliminar.   Se eliminan todos los productos asociados con la categoría seleccionada. |
| **Postcondición** | Ninguna |
| **Excepción** | No se puede registrar una categoría ya existente.  No se puede eliminar una categoría que ya tiene productos asignados. |

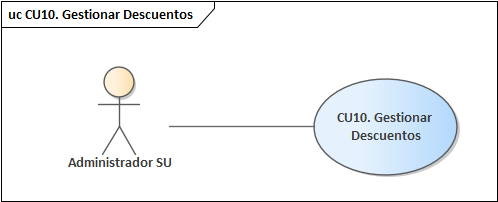
### Ciclo #2

#### CU9 Gestionar Ventas



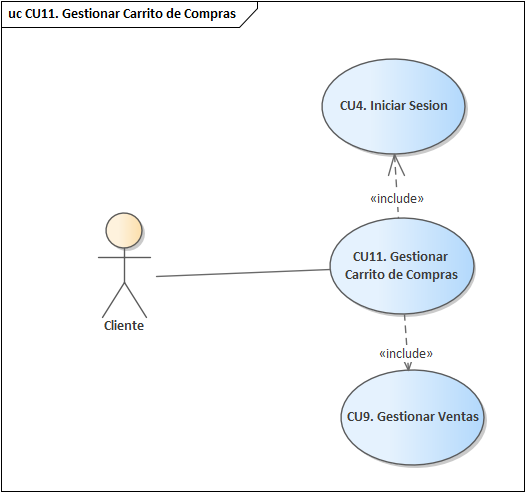
|  |  |
| --- | --- |
| **Nombre de CU** | CU9. Gestionar Ventas |
| **Propósito** | Gestionar las ventas de productos de su respectivo lote de sucursal |
| **Actores** | Administrador Super Usuario, Cliente |
| **Actor iniciador** | Administrador Super Usuario, Cliente |
| **Precondición** | CU6. Gestionar Producto |
| **Flujo principal** | Registrar venta de producto   * Ingresar los datos del producto a vender en el sistema Modificar venta de producto * Buscar una venta del producto existente en el sistema * Obtener datos de la venta del producto * Actualizar datos Eliminar venta de producto * Buscar venta del producto existente en el sistema * Eliminar venta de producto |
| **Postcondición** | Ninguna |
| **Excepción** | No se puede eliminar una venta de producto inexistente en el sistema |

#### CU10. Gestionar Descuentos



|  |  |
| --- | --- |
| **Nombre de CU** | CU10. Gestionar descuentos |
| **Propósito** | Registrar Descuentos en el sistema vinculados a una sucursal |
| **Actores** | Administrador Super Usuario |
| **Actor iniciador** | Administrador Super Usuario |
| **Precondición** | CU6. Gestionar Producto |
| **Flujo principal** | Registrar Descuento   * Ingresar los datos del descuento al sistema Modificar Descuento * Buscar descuento existente en el sistema * Obtener datos del descuento * Actualizar datos Desactivar/activar Descuento * Buscar descuento existente en el sistema * Desactivar/activar descuento |
| **Postcondición** | Ninguna |
| **Excepción** | Ninguna |

#### CU11. Gestionar Carrito de Compras



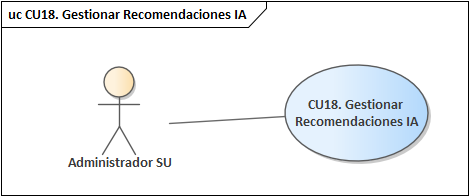
|  |  |
| --- | --- |
| **Nombre de CU** | CU11. Gestionar Carrito de Compras |
| **Propósito** | Permite al usuario gestionar su carrito de compras |
| **Actores** | Cliente |
| **Actor iniciador** | Cliente |
| **Precondición** | CU4. Iniciar Sesión  CU9. Gestionar Ventas |
| **Flujo principal** | Obtener productos del carrito   * Listar todos los productos del carrito * Visualizar un producto del carrito * Buscar un producto del carrito Añadir productos del carrito * Ingresar los datos del producto a adicionar Modificar productos del carrito * Buscar productos del carrito * Obtener datos de productos del carrito * Actualizar los datos Eliminar * Buscar usuario activo en el sistema * Se deshabilita el usuario |
| **Postcondición** | Ninguna |
| **Excepción** | No se puede eliminar un producto que no esté en el carrito. |

#### CU17. Gestionar Historial de Compras

\

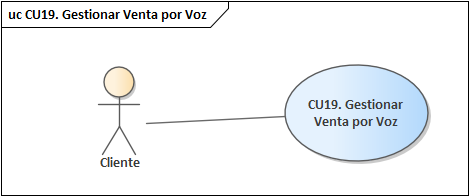
|  |  |
| --- | --- |
| **Nombre de CU** | CU17. Gestionar Historial de Compras |
| **Propósito** | Permite al usuario tener un historial de todas las compras realizadas en el sistema |
| **Actores** | Cliente |
| **Actor iniciador** | Cliente |
| **Precondición** | CU4. Iniciar Sesión  CU9. Gestionar Ventas |
| **Flujo principal** | Obtener historial de compras   * Listar todo el historial de compras * Visualizar un producto del historial de compras * Buscar un producto del historial de compras Añadir productos al historial de compras * Ingresar los datos del producto a comprar para añadir al historial de compras   Modificar productos del historial de compras   * Buscar productos del historial de compras * Obtener datos de productos del historial de compras * Actualizar los datos Eliminar * Buscar producto del historial de compras en el sistema * Se deshabilita el producto del historial de compras |
| **Postcondición** | Ninguna |
| **Excepción** | No se puede eliminar un producto del historial de compras que no esté en el historial de compras. |

#### CU18. Crear Recomendaciones IA



|  |  |
| --- | --- |
| **Nombre de CU** | CU18. Gestionar Recomendaciones IA |
| **Actores** | Administrador Super Usuario, Cliente |
| **Actor iniciador** | Administrador Super Usuario, Cliente |
| **Precondición** | Ninguna |
| **Flujo Principal** | El usuario selecciona un producto luego se ven las recomendaciones de otros productos relacionados con su categoría que podría comprar. |
| **Postcondición** | Ninguna |
| **Excepción** | . El producto es único en su categoría. |

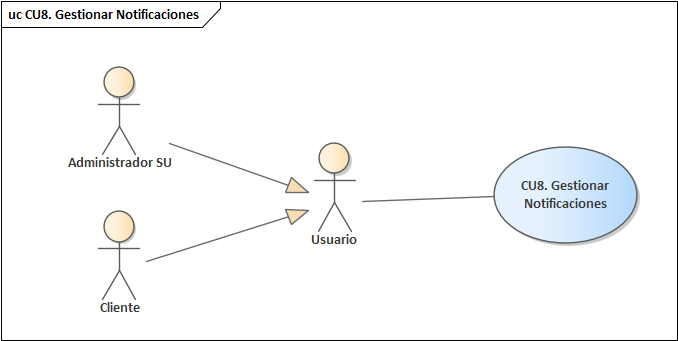
#### CU19. Ejecutar Venta por Voz



|  |  |
| --- | --- |
| **Nombre de CU** | CU19. Gestionar Venta por Voz |
| **Actores** | Cliente |
| **Actor iniciador** | Cliente |
| **Precondición** | Ninguna |
| **Flujo Principal** | El usuario selecciona la opción para comprar por voz.  Detalla que producto quiere comprar y la cantidad deseada de el mismo.  Se validan los datos de los productos en el carrito.  Se procede y registra la compra de todos los productos del carrito. |
| **Postcondición** | Ninguna |
| **Excepción** | El usuario indica dicta un producto que no existe. |

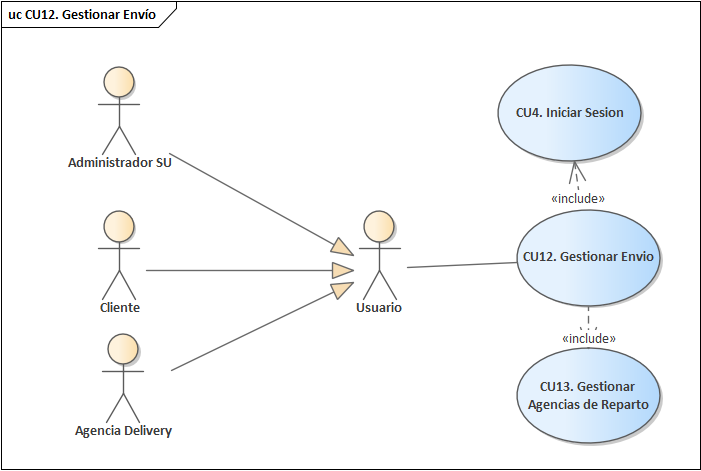
### Ciclo #3

#### CU8. Gestionar Notificaciones



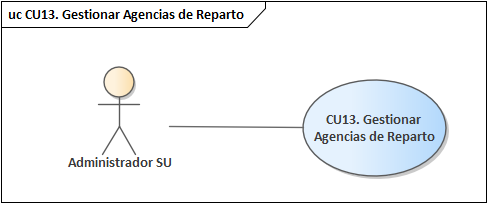
|  |  |
| --- | --- |
| **Nombre de CU** | CU8. Gestionar notificaciones |
| **Propósito** | Gestionar las notificaciones que saldrán en el sistema |
| **Actores** | Administrador Super Usuario, Cliente |
| **Actor iniciador** | Administrador Super Usuario, Cliente |
| **Precondición** | Ninguna |
| **Flujo principal** | Registrar Notificación   * Ingresar los datos de la notificación al sistema Modificar Notificación * Buscar una notificación existente en el sistema * Obtener datos de la notificación * Actualizar datos Eliminar notificación * Buscar notificación existente en el sistema * Eliminar notificación |
| **Postcondición** | Ninguna |
| **Excepción** | No se puede eliminar notificación inexistente en el sistema |

#### CU12. Gestionar Envío



|  |  |
| --- | --- |
| **Nombre de CU** | C12. Gestionar Envío |
| **Propósito** | Gestionar el envío del producto al domicilio del cliente. |
| **Actores** | Agencia de Delivery, Cliente |
| **Actor iniciador** | Agencia de Delivery, Cliente |
| **Precondición** | CU9. Gestionar Ventas  CU13. Gestionar Agencias de Reparto |
| **Flujo principal** | Obtener envío   * Listar envíos * Visualizar un envío * Buscar un envío Registrar envío * Ingresar los datos del envío * Asignar Agencia de Delivery * Asignar un cliente Modificar envío * Buscar envío existente en el sistema * Obtener datos del envío * Actualizar los datos Eliminar * Buscar envío activo en el sistema * Se deshabilita el envío |
| **Postcondición** | Ninguna |
| **Excepción** | No se puede deshabilitar un envío inexistente |

#### CU13. Gestionar Agencias de Reparto

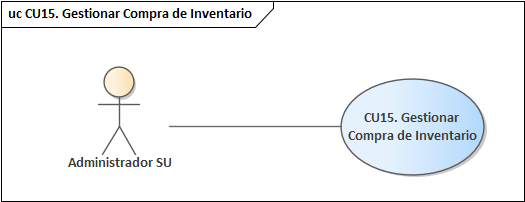


|  |  |
| --- | --- |
| **Nombre de CU** | C13. Gestionar Agencias de Reparto |
| **Propósito** | Registrar a las Agencias encargadas de realizar el reparto |
| **Actores** | Administrador Super Usuario |
| **Actor iniciador** | Administrador Super Usuario |
| **Precondición** | Ninguna |
| **Flujo principal** | Obtener Agencias de Reparto   * Listar todas las Agencias de Reparto * Visualizar una Agencia de Reparto * Buscar una Agencias de Reparto Registrar Agencias de Reparto * Ingresar los datos personales de la Agencia de Reparto * Asignar a una sucursal (opcional) Modificar Agencias de Reparto * Buscar Agencia de Reparto existente en el sistema * Obtener datos de la Agencia de Reparto * Actualizar los datos Eliminar * Buscar Agencia de Reparto activo en el sistema * Se deshabilita la Agencia de Reparto |
| **Postcondición** | Ninguna |
| **Excepción** | No se puede eliminar una Agencia de Reparto inexistente |

#### CU14. Gestionar Reportes

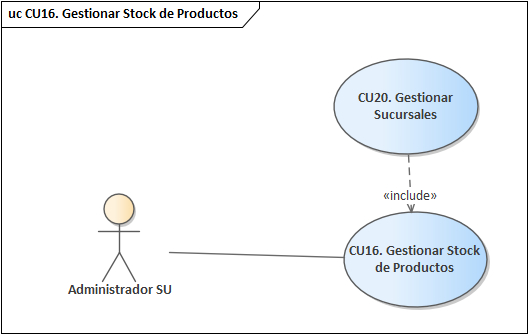
|  |  |
| --- | --- |
| **Nombre de CU** | CU14. Gestionar reportes |
| **Propósito** | Gestionar reportes del sistema |
| **Actores** | Administrador Super Usuario, Agencia de Delivery |
| **Actor iniciador** | Administrador Super Usuario, Agencia de Delivery |
| **Precondición** | Ninguna |
| **Flujo principal** | Registrar Reporte   * Ingresar los datos requeridos para generar reporte del sistema |
| **Postcondición** | Ninguna |
| **Excepción** | Ninguna |

#### CU15. Gestionar Compra de Inventario



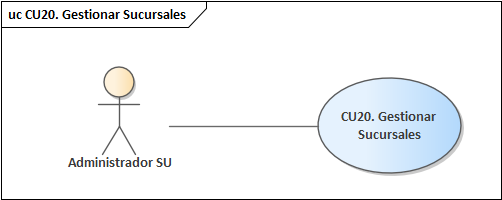
|  |  |
| --- | --- |
| **Nombre de CU** | CU15. Gestionar Compra de Inventario |
| **Propósito** | Gestionar Compras de la empresa |
| **Actores** | Administrador Super Usuario |
| **Actor iniciador** | Administrador Super Usuario |
| **Precondición** | Ninguna |
| **Flujo principal** | Obtener Compras   * Listar todas las Compras * Visualizar una Compra * Buscar una Compras Registrar Compras * Ingresar los datos de la Compra |
| **Postcondición** | Ninguna |
| **Excepción** | Ninguna |

#### CU16. Gestionar Stock de Productos



|  |  |
| --- | --- |
| **Nombre de CU** | CU16. Gestionar Stock de Productos |
| **Propósito** | Gestionar Stock o Inventario de los Productos Ofrecidos |
| **Actores** | Administrador Super Usuario |
| **Actor iniciador** | Administrador Super Usuario |
| **Precondición** | CU20. Gestionar Sucursales |
| **Flujo Principal** | Obtener lista de stocks disponibles por sucursal de cada producto.  Modificar stock disponible en la sucursal de cada producto. |
| **Postcondición** | Ninguna |
| **Excepción** | Ninguna |

#### CU20. Gestionar Sucursales



|  |  |
| --- | --- |
| **Nombre de CU** | CU20. Gestionar Sucursales |
| **Propósito** | Gestionar el registro de sucursales del e-commerce |
| **Actores** | Administrador Super Usuario |
| **Actor iniciador** | Administrador Super Usuario |
| **Precondición** | Ninguna |
| **Flujo Principal** | Registrar Sucursal   * Ingresar los datos de la sucursal al sistema Modificar Sucursal * Buscar sucursal existente en el sistema * Obtener datos de la sucursal * Actualizar datos Eliminar Sucursal * Buscar sucursal existente en el sistema   Deshabilitar sucursal |
| **Postcondición** | Ninguna |
| **Excepción** | Ninguna |

## Estructurar el modelo de casos de uso

### Ciclo #1

### Ciclo #2

### Ciclo #3

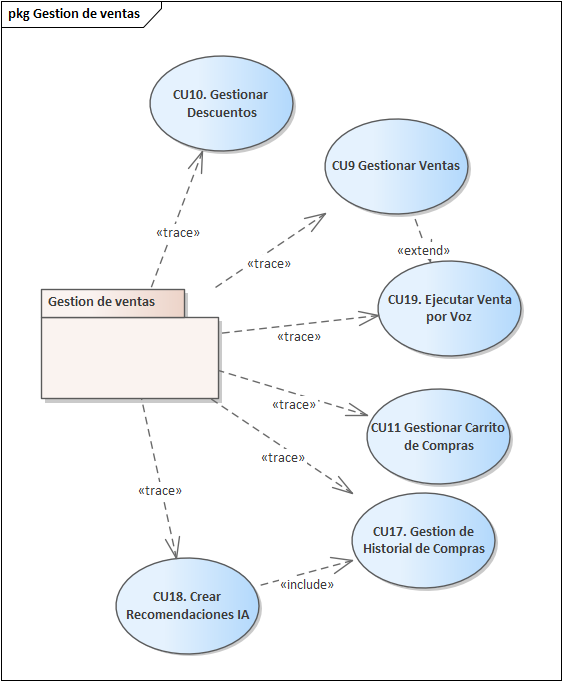
# CAPITULO 3 FLUJO DE TRABAJO: ANALISIS

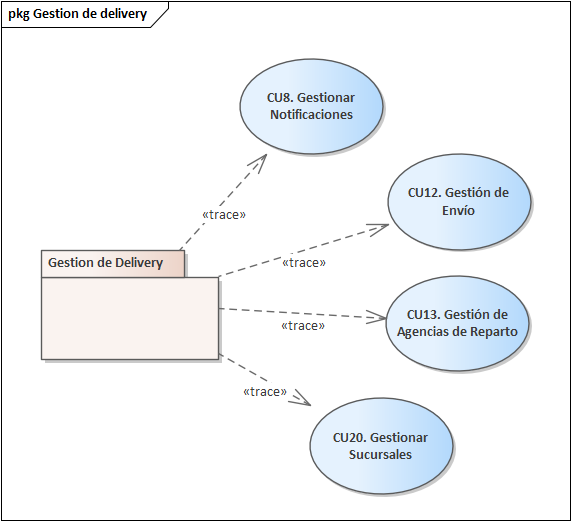
## Análisis de arquitectura

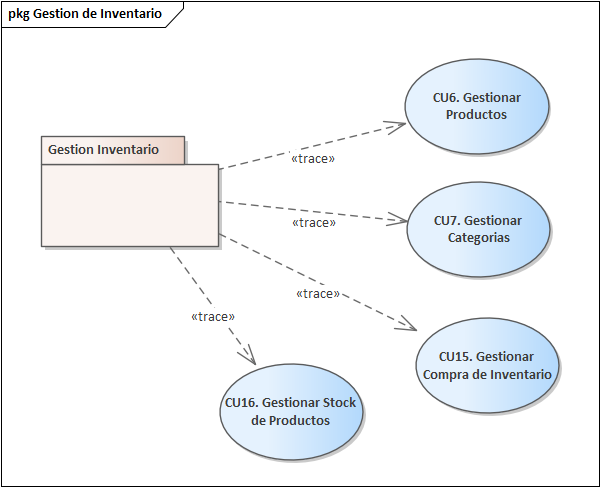
### Identificar paquetes

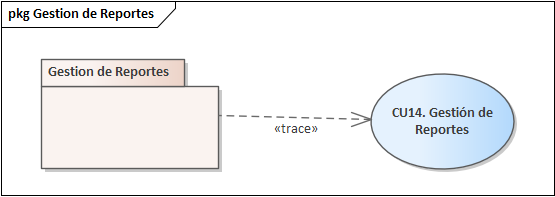
|  |  |
| --- | --- |
|  | Este paquete se encarga de la gestión de usuarios en el sistema. Incluye funcionalidades para crear, editar y eliminar cuentas de usuario, así como para gestionar sus roles y permisos de acceso y ocuparse de la autenticación y autorización de los usuarios en el sistema. |
|  | Este paquete se encarga de la gestión del inventario de productos en el sistema. Incluye funcionalidades para agregar, editar y eliminar productos, así como para realizar un seguimiento del stock disponible y gestionar las salidas de productos del almacén. |
|  | Este paquete se encarga de configurar y gestionar la estructura para las ventas. Incluye la gestión de descuentos y devoluciones. |
|  | Este paquete se encarga del envío de los productos, incluyendo la gestión del envío y la tarifa a aplicar. |
|  | Este paquete se encarga de la generación de reportes personalizados por el usuario. |

### Relacionar paquetes y casos de uso









## Análisis de casos de uso

### Diagramas de comunicación

#### Ciclo #1

#### Ciclo #2

## Análisis de clase

## Análisis de paquete

# CAPITULO 4 FLUJO DE TRABAJO: DISEÑO

## Diseño de arquitectura

### Diseño Físico (Diagrama de despliegue)

### Diseño Lógico (Diagrama Organizado en capas)

## Diseño de Datos

### Diseño de Datos Lógicos

#### Diagrama de clase

#### Mapeo

#### Normalización

### Diseño de datos físicos

#### Tabla de volumen

#### Script

#### Diagrama relacional

#### Actualización de tuplas

#### Consultas

#### Procedimientos almacenados

#### Triggers

### Diagrama de secuencia

# CAPITULO 5 FLUJO DE TRABAJO IMPLEMENTACION

## Elección de plataforma de desarrollo de software

### Lenguaje de programación

Para el desarrollo del **backend** de este proyecto, se optó por utilizar **Python** junto con el framework **Django**. Esta combinación proporciona un entorno robusto, flexible y con un ecosistema amplio que facilita el desarrollo de aplicaciones web escalables y seguras. Django sigue el principio "baterías incluidas", ofreciendo una gran cantidad de funcionalidades integradas que aceleran el desarrollo, como su potente ORM, sistema de autenticación y panel de administración automático.

Para gestionar la seguridad de la aplicación, se utilizó el sistema de autenticación integrado de Django, que proporciona una solución completa para el manejo de usuarios, grupos y permisos. Además, se implementó **Django REST framework (DRF)** para construir la API, aprovechando su soporte para autenticación basada en tokens, permisos personalizados y validación de datos.

En el **frontend**, se eligió **React**, una biblioteca muy popular para construir interfaces de usuario interactivas y dinámicas. Su capacidad de reutilización de componentes permite crear una experiencia de usuario fluida y mantenible en todas las plataformas. Además, se empleó **Axios** para gestionar las peticiones HTTP, facilitando la comunicación entre el frontend y el backend de manera eficiente y asíncrona.

Para asegurar el ciclo de vida del desarrollo del software, se eligió **GitHub** como plataforma central para el control de versiones mediante **Git**. Esto permite gestionar de manera eficiente las revisiones de código y facilita la integración continua (CI) con servicios de despliegue.

Este enfoque garantiza un desarrollo ágil, mantenible y seguro, aprovechando las fortalezas de Django para el backend y React para una interfaz de usuario moderna.

### Base de datos

Se decidió utilizar **PostgreSQL** como sistema de gestión de bases de datos. Esta elección se basa en su estabilidad, escalabilidad y capacidades avanzadas para manejar datos estructurados y semiestructurados. PostgreSQL es reconocido por su soporte de características como JSON nativo, lo que nos permite manejar datos complejos de manera eficiente.

**Django ORM**, una parte integral del framework, permite interactuar con PostgreSQL de manera sencilla y eficiente. El ORM de Django abstrae las operaciones básicas de la base de datos, reduciendo el código repetitivo y facilitando la implementación de consultas complejas mediante su potente sistema de **querysets**. Además, gracias a las **migraciones automáticas**, el esquema de la base de datos puede evolucionar de manera segura a medida que el proyecto crece.

Las razones clave para escoger PostgreSQL incluyen:

**-** **Escalabilidad:** La capacidad de gestionar grandes volúmenes de datos y conexiones simultáneas lo convierte en una excelente opción para sistemas en crecimiento.

**-** **Seguridad:** PostgreSQL cuenta con mecanismos avanzados de seguridad, incluyendo autenticación robusta y cifrado de datos, lo que asegura que nuestras aplicaciones cumplan con los estándares más estrictos.

**- Extensibilidad:** La naturaleza extensible de PostgreSQL permite la creación de funciones personalizadas y el uso de extensiones adicionales para análisis avanzado y búsqueda de texto completo, lo que proporciona flexibilidad en el desarrollo.

Django, junto con PostgreSQL, ofrece un stack tecnológico potente y eficiente, ideal para aplicaciones web modernas que requieren alto rendimiento, seguridad y facilidad de mantenimiento.

### Sistemas operativos

Este sistema de información está diseñado para ser accesible tanto en web como en dispositivos móviles, lo que garantiza su compatibilidad con múltiples plataformas.

Gracias a la arquitectura responsive implementada en el frontend con React, nuestra aplicación se adapta automáticamente a diferentes tamaños de pantalla, asegurando una experiencia de usuario óptima en computadoras de escritorio, tabletas y teléfonos inteligentes.

La aplicación es compatible con los sistemas operativos más utilizados en la actualidad, incluyendo Windows, macOS, Linux, Android e iOS, lo que permite su despliegue y acceso desde cualquier dispositivo. Esta compatibilidad asegura que los usuarios puedan interactuar con el sistema de manera eficiente, independientemente del entorno en el que se encuentren.

### Otros

GitHub ha sido seleccionado como la plataforma principal para gestionar el código fuente, dado su poderoso control de versiones con Git. GitHub facilita el seguimiento de problemas y una integración continua con servicios como GitHub Actions, lo que permite una entrega rápida y segura de nuevas versiones del software.

### Link de repositorio



**Repositorio del backend**



**Repositorio del frontend**



**Despliegue en AWS**

## Implementación de la arquitectura del sistema

## Implementación de la arquitectura del subsistema

# CAPITULO 6 FLUJO DE TRABAJO: PRUEBAS

## Pruebas de autenticación

Objetivo: Verificar que el sistema permita el inicio de sesión correcto y rechace las credenciales inválidas.

Entrada esperada:

Credenciales válidas: correo electrónico y contraseña correctos.

Credenciales inválidas: correo incorrecto, contraseña incorrecta, o ambos.

Proceso: Se enviarán peticiones POST al endpoint /login con distintas combinaciones de credenciales.

Salida esperada:

Para credenciales válidas: respuesta 200 OK con un token JWT.

Para credenciales inválidas: respuesta 401 Unauthorized con un mensaje de error.

## Pruebas de gestión de productos

Objetivo: Validar que los productos se creen, actualicen y eliminen correctamente mediante los endpoints correspondientes.

Entrada esperada:

Crear producto: datos válidos como nombre, descripción, precio, categoría, etc.

Actualizar producto: ID del producto existente y datos modificados.

Eliminar producto: ID del producto existente.

Proceso:

Enviar peticiones POST, PUT y DELETE a los endpoints /productos/productos/,

/productos/productos/{id}/, validando los distintos flujos.

Salida esperada:

Crear producto: respuesta 201 Created con los datos del producto creado.

Actualizar producto: respuesta 200 OK con los datos del producto actualizado.

Eliminar producto: respuesta 204 No Content si el producto fue eliminado correctamente.

# BIBLIOGRAFIA

# ANEXOS

## 1. Backend

## 1.2 Modelos.py

## Usuarios:

from django.contrib.auth.models import AbstractBaseUser, BaseUserManager, PermissionsMixin

from django.utils import timezone

from django.db import models

from direcciones.models import Direccion

*# Modelo: Rol*

class Rol(models.Model):

    nombre = models.CharField(max\_length=100)

    def \_\_str\_\_(self):

        return *self*.nombre

*# Modelo: Permiso*

class Permiso(models.Model):

    nombre = models.CharField(max\_length=100)

    descripcion = models.TextField()

    def \_\_str\_\_(self):

        return *self*.nombre

*# Modelo: PermisoRol*

class PermisoRol(models.Model):

    permiso = models.ForeignKey(Permiso, on\_delete=models.CASCADE)

    rol = models.ForeignKey(Rol, on\_delete=models.CASCADE)

    class Meta:

        unique\_together = ('permiso', 'rol')

    def \_\_str\_\_(self):

        return f"{*self*.permiso.nombre} - {*self*.rol.nombre}"

*# modelo: Usuario*

class UsuarioManager(BaseUserManager):

    def create\_user(self, nombre, apellidos, correo, password=None):

        if not nombre:

            raise ValueError("El usuario debe tener un nombre")

        if not correo:

            raise ValueError("El usuario debe tener un correo")

        if not apellidos:

            raise ValueError("El usuario debe tener apellidos registrados")

        user = *self*.model(

            nombre=nombre,

            apellidos=apellidos,

            correo=*self*.normalize\_email(correo),

        )

        user.set\_password(password)

        user.save(using=*self*.\_db)

        return user

    def create\_superuser(self, nombre, apellidos, correo, password):

        user = *self*.create\_user(nombre, apellidos, correo, password)

        user.is\_staff = True

        user.is\_superuser = True

        user.rol = Rol.objects.get(id=1)

        user.save(using=*self*.\_db)

        return user

    def get\_by\_natural\_key(self, correo):

        return *self*.get(correo=correo)

class Usuario(AbstractBaseUser, PermissionsMixin):

    nombre = models.CharField(max\_length=50)

    correo = models.EmailField(unique=True)

    apellidos = models.CharField(null=True, blank=True, max\_length=100)

    rol = models.ForeignKey(Rol, on\_delete=models.SET\_NULL, null=True, blank=True)

    direccion = models.OneToOneField(Direccion, on\_delete=models.SET\_NULL, null=True, blank=True)

    is\_active = models.BooleanField(default=True)

    is\_staff = models.BooleanField(default=False)

    objects = UsuarioManager()

    USERNAME\_FIELD = 'correo'

    REQUIRED\_FIELDS = ['nombre','apellidos']

    def \_\_str\_\_(self):

        return *self*.correo

*# Modelo: ActivityLogUsuario*

class ActivitylogUsuario(models.Model):

    id\_usuario = models.ForeignKey(Usuario, on\_delete=models.CASCADE)

    accion = models.CharField(max\_length=255)

    timestamp = models.DateTimeField(default=timezone.now)

    def \_\_str\_\_(self):

        return f"{*self*.id\_usuario.correo} - {*self*.accion} - {*self*.timestamp}"

## Productos:

from django.db import models

from django.db.models.signals import post\_save, post\_delete

from django.dispatch import receiver

*# Modelo: Proveedor*

class Proveedor(models.Model):

    nombre=models.CharField(max\_length=100, unique = True)

    contacto = models.CharField(max\_length=100, blank=True, null=True)

    telefono = models.CharField(max\_length=20, blank=True, null = True)

    correo = models.EmailField(blank=True, null=True)

    def \_\_str\_\_(self):

        return *self*.nombre

*# Modelo: Categoria*

class Categoria(models.Model):

    nombre = models.CharField(max\_length=100, unique=True)

    descripcion = models.TextField(max\_length=200, blank=True)

    def \_\_str\_\_(self):

        return *self*.nombre

*#Modelo: Producto*

class Producto(models.Model):

    nombre = models.CharField(max\_length=100)

    descripcion = models.TextField(max\_length=200)

    categorias = models.ManyToManyField(Categoria, through='Categoria\_Producto')

    esta\_activo = models.BooleanField(default=True)

    esta\_disponible = models.BooleanField(default=True)

    stock\_total = models.IntegerField(default=0)

    proveedor = models.ForeignKey(Proveedor, on\_delete=models.CASCADE, null=True, blank=True)

    def actualizar\_stock\_total(self):

        total = sum([s.stock for s in *self*.stock\_sucursal\_set.all()])

*self*.stock\_total = total

*self*.save()

    def \_\_str\_\_(self):

        return *self*.nombre

*#Modelo: Imagenes de un producto*

class Imagen\_Producto(models.Model):

    producto = models.ForeignKey(Producto, related\_name='imagenes', on\_delete=models.CASCADE)

    imagen = models.ImageField(upload\_to='productos/')

    descripcion = models.CharField(max\_length=255, blank=True)

    def \_\_str\_\_(self):

        return f"Imagen de {*self*.producto.nombre}"

*# Modelo: Detalle producto*

class Detalle\_Producto(models.Model):

    producto = models.OneToOneField(Producto, on\_delete=models.CASCADE, related\_name='detalle',null=True, blank=True)

    marca = models.CharField(max\_length=100, default='Generico')

    precio\_venta = models.DecimalField(max\_digits=10, decimal\_places=2, default=0.00)

    precio\_compra = models.DecimalField(max\_digits=10, decimal\_places=2, default=0.00)

    tiene\_descuento = models.BooleanField(default=False)

    porcentaje\_descuento = models.DecimalField(max\_digits=5, decimal\_places=2, default=0.00)

    @property

    def precio\_final(self):

        if *self*.tiene\_descuento and *self*.porcentaje\_descuento >0:

            descuento = (*self*.precio\_venta \* *self*.porcentaje\_descuento) / 100

            return round(*self*.precio\_venta - descuento, 2)

        return *self*.precio\_venta

    def \_\_str\_\_(self):

        if *self*.producto:

            return f"{*self*.producto.nombre} - {*self*.marca}"

        return f"Sin producto asignado - {*self*.marca}"

*# Modelo: Categoria\_Producto*

class Categoria\_Producto(models.Model):

    producto = models.ForeignKey(Producto, on\_delete=models.CASCADE)

    categoria = models.ForeignKey(Categoria, on\_delete=models.CASCADE)

    class Meta:

        unique\_together = ('producto', 'categoria')

        verbose\_name = 'Relación Producto-Categoría'

        verbose\_name\_plural = 'Relaciones Producto-Categoría'

    def \_\_str\_\_(self):

        return f"{*self*.producto.nombre} - {*self*.categoria.nombre}"

*# Modelo: Stock\_sucursal*

class Stock\_sucursal(models.Model):

    stock = models.PositiveIntegerField(default=0)

    producto = models.ForeignKey(Producto, on\_delete=models.CASCADE)

    sucursal = models.ForeignKey('sucursales.Sucursal',on\_delete=models.CASCADE)

    class Meta:

        unique\_together = ('producto', 'sucursal')

        verbose\_name = 'Stock en Sucursal'

        verbose\_name\_plural = 'Stock en Sucursales'

    def \_\_str\_\_(self):

        return f"{*self*.producto.nombre} - {*self*.sucursal.nombre} - {*self*.stock}"

## Pedidos:

from django.db import models

*#Modelo: Carrito*

class Cart(models.Model):

    usuario = models.ForeignKey('usuarios.Usuario', on\_delete=models.CASCADE)

    creado\_en = models.DateTimeField(auto\_now=True)

    ESTADOS = [

        ('activo', 'Activo'),

        ('confirmado', 'Confirmado'),

        ('cancelado', 'Cancelado'),

    ]

    estado = models.CharField(max\_length=20, choices=ESTADOS,default='activo')

    def \_\_str\_\_(self):

        return f"Carrito de {*self*.usuario}"

*#Modelo: Productos para el carrito*

class ItemCart(models.Model):

    cart = models.ForeignKey(Cart, related\_name='items', on\_delete=models.CASCADE)

    producto = models.ForeignKey('productos.Producto', on\_delete=models.CASCADE)

    cantidad = models.PositiveIntegerField(default=1)

    precio\_unitario = models.DecimalField(max\_digits=10, decimal\_places=2)

    def \_\_str\_\_(self):

        return f"{*self*.cantidad} x {*self*.producto}"

*#Modelo: Metodos de pago*

class Metodo\_Pago(models.Model):

    nombre = models.CharField(max\_length=50) *#tarjeta,qr,transferencia*

    proveedor = models.CharField(max\_length=100, blank=True, null=True) *#Stripe, banco bnb...*

    descripcion = models.TextField(blank=True, null=True)

    def \_\_str\_\_(self):

        return *self*.nombre

*#Modelo: Pago venta*

class Pago(models.Model):

    metodo = models.ForeignKey(Metodo\_Pago, on\_delete=models.SET\_NULL, null = True)

    monto = models.DecimalField(max\_digits=10, decimal\_places=2)

    estado = models.CharField(max\_length=20, choices = [

        ('pendiente', 'Pendiente'),

        ('completado', 'Completado'),

        ('fallido', 'Fallido'),

    ])

    fecha = models.DateTimeField(auto\_now\_add=True)

    referencia = models.CharField(max\_length=255, blank=True, null=True) *#codigo de qr, id de transferencia...*

    def \_\_str\_\_(self):

        return f"{*self*.metodo.nombre} - {*self*.estado}"

*#Modelo: Venta/Pedido*

class Venta(models.Model):

    usuario = models.ForeignKey('usuarios.Usuario', on\_delete=models.CASCADE)

    pago = models.OneToOneField(Pago, on\_delete=models.SET\_NULL, null=True, blank=True)

    fecha = models.DateTimeField(auto\_now=True)

    total = models.DecimalField(max\_digits=10, decimal\_places=2)

    estado = models.CharField(max\_length=50, choices=[

        ('pendiente', 'Pendiente'),

        ('procesando','Procesando'),

        ('enviado','Enviado'),

        ('entregado','Entregado'),

        ('cancelado','Cancelado')

    ])

    def \_\_str\_\_(self):

        return f"Venta #{*self*.id} - {*self*.usuario.correo}"

*#Modelo: Detalle de la venta*

class Detalle\_Venta(models.Model):

    venta = models.ForeignKey(Venta, related\_name='detalles', on\_delete=models.CASCADE)

    producto = models.ForeignKey('productos.Producto', on\_delete=models.CASCADE)

    cantidad = models. PositiveIntegerField()

    precio\_unitario = models.DecimalField(max\_digits=10, decimal\_places=2)

    def \_\_str\_\_(self):

        return f"{*self*.producto} x {*self*.cantidad}"

*#Modelo: Seguimiento de un pedido*

"""

class Seguimiento\_Pedido(models.Model):

    venta = models.OneToOneField(Venta, on\_delete=models.CASCADE, related\_name = 'seguimiento')

    direccion\_envio = models.CharField(max\_length = 255)

    ciudad = models.CharField(max\_length=100)

    fecha\_creacion = models.DateTimeField(auto\_now=True)

    fecha\_estimada = models.DateTimeField()

"""

## Direcciones:

from django.db import models

*# Modelo: Direccion*

class Direccion(models.Model):

    pais = models.CharField(max\_length=50)

    ciudad = models.CharField(max\_length=50)

    zona = models.CharField(max\_length=50, blank=True, null=True)

    calle = models.CharField(max\_length=100)

    numero = models.CharField(max\_length=10)

    referencia = models.TextField(blank=True, null=True)

    departamento = models.ForeignKey('sucursales.Departamento', on\_delete=models.CASCADE, null=True, blank=True)

    def \_\_str\_\_(self):

        return f"{*self*.calle} #{*self*.numero}, {*self*.zona or ''} - {*self*.ciudad}, {*self*.departamento}, {*self*.pais}"

## Sucursales:

from django.db import models

from direcciones.models import Direccion

*#Modelo: Departamento*

class Departamento(models.Model):

    nombre = models.CharField(max\_length=100, unique=True)

    def \_\_str\_\_(self):

        return *self*.nombre

*# Modelo: Sucursal*

class Sucursal(models.Model):

    nombre = models.CharField(max\_length=100)

    telefono = models.CharField(max\_length=20)

    direccion = models.OneToOneField(Direccion, on\_delete=models.CASCADE, null=True, blank=True)

    def \_\_str\_\_(self):

        return *self*.nombre

## 

## Frontend

## 2.1 Pages:

## Home:

import { ProductGrid } from "@/components/product-grid";

import { HeroSection } from "@/components/hero-section.tsx";

import { Categories } from "@/components/categories.tsx";

import { Outlet } from "react-router-dom";

import Navbar from "@/components/navbar";

export function Home() {

  return (

    <div *className*="flex flex-col gap-12 pb-12">

      <HeroSection />

      <div *className*="container mx-auto px-4">

        <Categories />

        <section *className*="mt-12">

          <div *className*="flex items-center justify-between mb-6">

            <h2 *className*="text-2xl font-bold text-white">Nuevos productos</h2>

            <a *href*="/products" *className*="text-slate-300 hover:text-white">

              Ver todos

            </a>

          </div>

          <ProductGrid />

        </section>

      </div>

    </div>

  );

}

export function StoreLayout() {

  return (

    <div *className*="flex flex-col min-h-screen bg-slate-900 text-slate-200">

      <Navbar />

      <main *className*="flex-1">

        <Outlet />

      </main>

    </div>

  );

}

## Profile:

"use client";

import { useState } from "react";

import { useForm } from "react-hook-form";

import { zodResolver } from "@hookform/resolvers/zod";

import \* as z from "zod";

import { Button } from "@/components/ui/button";

import {

  Card,

  CardContent,

  CardDescription,

  CardHeader,

  CardTitle,

} from "@/components/ui/card";

import {

  Form,

  FormControl,

  FormField,

  FormItem,

  FormLabel,

  FormMessage,

} from "@/components/ui/form";

import { Input } from "@/components/ui/input";

import { Tabs, TabsContent, TabsList, TabsTrigger } from "@/components/ui/tabs";

*// import { useToast } from "@/components/ui/use-toast"*

import { ProfileSidebar } from "@/components/profile-sidebar.tsx";

import { UserCircle, Mail } from "lucide-react";

import { useAuthStore } from "@/store/auth.ts";

import { updateUsuario, Usuarios } from "@/api/user";

import { updateDireccionRequest } from "@/api/direccion";

import { UpdateDireccionRequest } from "@/types/direccion";

import { updateUser } from "@/types/user";

const profileFormSchema = z.object({

  nombre: z.string().min(2, "El nombre deberia ser de al menos 2 caracteres."),

  apellidos: z.string().min(5, "Apellido minimo 5 caracteres."),

  correo: z.string().email("El correo no es valido."),

});

const addressFormSchema = z.object({

  pais: z.string().min(2, "Pais is required"),

  departamento: z.string().min(2, "Departamento is required"),

  ciudad: z.string().min(2, "Ciudad is required"),

  zona: z.string().min(2, "Zona is required"),

  calle: z.string().min(2, "Calle is required"),

  numero: z.string().min(1, "Numero is required"),

  referencia: z.string().optional(),

});

export function ProfilePage() {

*//   const { toast } = useToast()*

  const [isSubmitting, setIsSubmitting] = useState(false);

  const profile = useAuthStore((state) => state.profile);

  const direccion = useAuthStore((state) => state.directions);

  const profileForm = useForm<z.infer<typeof profileFormSchema>>({

    resolver: zodResolver(profileFormSchema),

    defaultValues: {

      nombre: profile?.nombre,

      apellidos: profile?.apellidos,

      correo: profile?.correo,

    },

  });

  const addressForm = useForm<z.infer<typeof addressFormSchema>>({

    resolver: zodResolver(addressFormSchema),

    defaultValues: {

      pais: direccion.pais,

      departamento: direccion.departamento.nombre,

      ciudad: direccion.ciudad,

      zona: direccion.zona,

      calle: direccion.calle,

      numero: direccion.numero,

      referencia: direccion.referencia,

    },

  });

  const onProfileSubmit = async (values: z.infer<typeof profileFormSchema>) => {

    setIsSubmitting(true);

    } catch (error) {

      console.error("Error updating profile:", error);

      setIsSubmitting(false);

    }

  };

  async function onAddressSubmit(values: z.infer<typeof addressFormSchema>) {

    setIsSubmitting(true);

    try {

      const updatedAddress: UpdateDireccionRequest = {

        id\_direccion: direccion.id,

        pais: values.pais || direccion.pais,

        departamento: values.departamento || direccion.departamento.nombre,

        ciudad: values.ciudad || direccion.ciudad,

        zona: values.zona || direccion.zona,

        calle: values.calle || direccion.calle,

        numero: values.numero || direccion.numero,

        referencia: values.referencia || direccion.referencia,

      };

      console.log(updatedAddress);

      const res = await updateDireccionRequest(updatedAddress);

      console.log(res.data);

      setIsSubmitting(false);

      useAuthStore.setState({ directions: res.data });

    } catch (error) {

      console.error("Error updating address:", error);

      setIsSubmitting(false);

    }

  }

  return (

    <div *className*="container mx-auto px-4 py-8">

      <h1 *className*="text-3xl font-bold text-white mb-8 font-">Mi Perfil</h1>

      <div *className*="grid grid-cols-1 md:grid-cols-4 gap-8">

        <div *className*="md:col-span-1">

          <ProfileSidebar />

        </div>

        <div *className*="md:col-span-3">

          <Tabs *defaultValue*="profile" *className*="w-full">

            <TabsList *className*="bg-slate-800 border-slate-700">

              <TabsTrigger

*value*="profile"

*className*="data-[state=active]:bg-slate-700 text-slate-200"

              >

                Perfil

              </TabsTrigger>

              <TabsTrigger

*value*="Direccion"

*className*="data-[state=active]:bg-slate-700 text-slate-200"

              >

                Direccion

              </TabsTrigger>

            </TabsList>

            <TabsContent *value*="profile" *className*="mt-6">

              <Card *className*="border-slate-700 bg-slate-800 shadow-xl">

                <CardHeader>

                  <CardTitle *className*="text-white text-2xl ">

                    Informacion Personal

                  </CardTitle>

                  <CardDescription *className*="text-slate-400">

                    Actualice sus datos personales.

                  </CardDescription>

                </CardHeader>

                <CardContent>

                  <Form {...profileForm}>

                    <form

*onSubmit*={profileForm.handleSubmit(onProfileSubmit)}

*className*="space-y-6"

                    >

                      <FormField

*control*={profileForm.control}

*name*="nombre"

*render*={({ field }) => (

                          <FormItem>

                            <FormLabel *className*="text-slate-200">

                              Nombre

                            </FormLabel>

                            <FormControl>

                              <div *className*="relative">

                                <div *className*="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none text-slate-500">

                                  <UserCircle *className*="h-4 w-4" />

                                </div>

                                <Input

*className*="pl-10 bg-slate-700 border-slate-600 text-white placeholder:text-slate-400 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                                  {...field}

                                />

                              </div>

                            </FormControl>

                            <FormMessage />

                          </FormItem>

                        )}

                      />

                      <FormField

*control*={profileForm.control}

*name*="apellidos"

*render*={({ field }) => (

                          <FormItem>

                            <FormLabel *className*="text-slate-200">

                              Apellidos

                            </FormLabel>

                            <FormControl>

                              <div *className*="relative">

                                <div *className*="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none text-slate-500">

                                  <UserCircle *className*="h-4 w-4" />

                                </div>

                                <Input

*className*="pl-10 bg-slate-700 border-slate-600 text-white placeholder:text-slate-400 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                                  {...field}

                                />

                              </div>

                            </FormControl>

                            <FormMessage />

                          </FormItem>

                        )}

                      />

                      <FormField

*control*={profileForm.control}

*name*="correo"

*render*={({ field }) => (

                          <FormItem>

                            <FormLabel *className*="text-slate-200">

                              Correo

                            </FormLabel>

                            <FormControl>

                              <div *className*="relative">

                                <div *className*="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none text-slate-500">

                                  <Mail *className*="h-4 w-4" />

                                </div>

                                <Input

*className*="pl-10 bg-slate-700 border-slate-600 text-white placeholder:text-slate-400 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                                  {...field}

                                />

                              </div>

                            </FormControl>

                            <FormMessage />

                          </FormItem>

                        )}

                      />

                      <Button

*type*="submit"

*className*="bg-slate-600 hover:bg-slate-500 text-white"

*disabled*={isSubmitting}

                      >

                        {isSubmitting ? "Guardando..." : "Guardar Cambios"}

                      </Button>

                    </form>

                  </Form>

                </CardContent>

              </Card>

            </TabsContent>

            <TabsContent *value*="Direccion" *className*="mt-6">

              <Card *className*="border-slate-700 bg-slate-800 shadow-xl">

                <CardHeader>

                  <CardTitle *className*="text-2xl text-white">

                    Informacion de la direccion.

                  </CardTitle>

                  <CardDescription *className*="text-slate-400">

                    Actualice sus detalles de envio.

                  </CardDescription>

                </CardHeader>

                <CardContent>

                  <Form {...addressForm}>

                    <form

*onSubmit*={addressForm.handleSubmit(onAddressSubmit)}

*className*="space-y-6"

                    >

                      <div *className*="grid grid-cols-1 md:grid-cols-2 gap-6">

                        <FormField

*control*={addressForm.control}

*name*="pais"

*render*={({ field }) => (

                            <FormItem>

                              <FormLabel *className*="text-slate-200">

                                Pais

                              </FormLabel>

                              <FormControl>

                                <Input

*className*="bg-slate-700 border-slate-600 text-white placeholder:text-slate-400 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                                  {...field}

                                />

                              </FormControl>

                              <FormMessage />

                            </FormItem>

                          )}

                        />

                        <FormField

*control*={addressForm.control}

*name*="departamento"

*render*={({ field }) => (

                            <FormItem>

                              <FormLabel *className*="text-slate-200">

                                Departamento

                              </FormLabel>

                              <FormControl>

                                <Input

*className*="bg-slate-700 border-slate-600 text-white placeholder:text-slate-400 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                                  {...field}

                                />

                              </FormControl>

                              <FormMessage />

                            </FormItem>

                          )}

                        />

                      </div>

                      <div *className*="grid grid-cols-1 md:grid-cols-2 gap-6">

                        <FormField

*control*={addressForm.control}

*name*="ciudad"

*render*={({ field }) => (

                            <FormItem>

                              <FormLabel *className*="text-slate-200">

                                Ciudad

                              </FormLabel>

                              <FormControl>

                                <Input

*className*="bg-slate-700 border-slate-600 text-white placeholder:text-slate-400 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                                  {...field}

                                />

                              </FormControl>

                              <FormMessage />

                            </FormItem>

                          )}

                        />

                        <FormField

*control*={addressForm.control}

*name*="zona"

*render*={({ field }) => (

                            <FormItem>

                              <FormLabel *className*="text-slate-200">

                                Zona

                              </FormLabel>

                              <FormControl>

                                <Input

*className*="bg-slate-700 border-slate-600 text-white placeholder:text-slate-400 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                                  {...field}

                                />

                              </FormControl>

                              <FormMessage />

                            </FormItem>

                          )}

                        />

                      </div>

                      <div *className*="grid grid-cols-1 md:grid-cols-3 gap-6">

                        <FormField

*control*={addressForm.control}

*name*="calle"

*render*={({ field }) => (

                            <FormItem>

                              <FormLabel *className*="text-slate-200">

                                Calle

                              </FormLabel>

                              <FormControl>

                                <Input

*className*="bg-slate-700 border-slate-600 text-white placeholder:text-slate-400 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                                  {...field}

                                />

                              </FormControl>

                              <FormMessage />

                            </FormItem>

                          )}

                        />

                        <FormField

*control*={addressForm.control}

*name*="numero"

*render*={({ field }) => (

                            <FormItem>

                              <FormLabel *className*="text-slate-200">

                                Numero

                              </FormLabel>

                              <FormControl>

                                <Input

*className*="bg-slate-700 border-slate-600 text-white placeholder:text-slate-400 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                                  {...field}

                                />

                              </FormControl>

                              <FormMessage />

                            </FormItem>

                          )}

                        />

                        <FormField

*control*={addressForm.control}

*name*="referencia"

*render*={({ field }) => (

                            <FormItem>

                              <FormLabel *className*="text-slate-200">

                                Referencia

                              </FormLabel>

                              <FormControl>

                                <Input

*className*="bg-slate-700 border-slate-600 text-white placeholder:text-slate-400 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                                  {...field}

                                />

                              </FormControl>

                              <FormMessage />

                            </FormItem>

                          )}

                        />

                      </div>

                      <Button

*type*="submit"

*className*="bg-slate-600 hover:bg-slate-500 text-white"

*disabled*={isSubmitting}

                      >

                        {isSubmitting ? "Guardando..." : "Guardar Direccion"}

                      </Button>

                    </form>

                  </Form>

                </CardContent>

              </Card>

            </TabsContent>

          </Tabs>

        </div>

      </div>

    </div>

  );

}

## Register:

"use client";

import { useState } from "react";

import { useForm } from "react-hook-form";

import { zodResolver } from "@hookform/resolvers/zod";

import \* as z from "zod";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import {

  Card,

  CardContent,

  CardDescription,

  CardHeader,

  CardTitle,

} from "@/components/ui/card";

import { Label } from "@/components/ui/label";

import {

  EyeIcon,

  EyeOffIcon,

  LockIcon,

  MailIcon,

  Package,

  UserIcon,

} from "lucide-react";

import { useNavigate, Link } from "react-router-dom";

import { registerRequest } from "@/api/auth.ts";

import { createUser, UserProfile } from "@/types/user.ts";

import { useAuthStore } from "@/store/auth.ts";

*// Esquema de validación con zod*

const registerSchema = z.object({

  nombre: z

    .string()

    .min(2, { message: "El nombre debe tener al menos 2 caracteres" }),

  apellidos: z

    .string()

    .min(2, { message: "Los apellidos deben tener al menos 2 caracteres" }),

  correo: z.string().email({ message: "Ingrese un correo electrónico válido" }),

  password: z

    .string()

    .min(8, { message: "La contraseña debe tener al menos 8 caracteres" })

    .regex(/[A-Z]/, {

      message: "La contraseña debe contener al menos una letra mayúscula",

    })

    .regex(/[0-9]/, {

      message: "La contraseña debe contener al menos un número",

    }),

});

type RegisterFormValues = z.infer<typeof registerSchema>;

export function RegisterPage() {

  const [showPassword, setShowPassword] = useState(false);

  const [isLoading, setIsLoading] = useState(false);

  const navigate = useNavigate();

  const setProfile = useAuthStore((state) => state.setProfile);

  const setToken = useAuthStore((state) => state.setToken);

  const {

    register,

    handleSubmit,

    formState: { errors },

  } = useForm<RegisterFormValues>({

    resolver: zodResolver(registerSchema),

    defaultValues: {

      nombre: "",

      apellidos: "",

      correo: "",

      password: "",

    },

  });

  const onSubmit = async (data: RegisterFormValues) => {

    setIsLoading(true);

    try {

      console.log("Datos del formulario:", data);

      const user: createUser = {

        nombre: data.nombre,

        apellidos: data.apellidos,

        correo: data.correo,

        password: data.password,

      };

      const response = await registerRequest(user);

      console.log("Respuesta del servidor:", response.data);

      setProfile({

        nombre: response.data.usuario.nombre,

        correo: response.data.usuario.correo,

        apellidos: response.data.usuario.apellidos,

        rol: response.data.usuario.rol,

      } as UserProfile);

      setToken(response.data.token);

      navigate("/inicio");

    } catch (error) {

      console.error("Error al registrar:", error);

    } finally {

      setIsLoading(false);

    }

  };

  const togglePasswordVisibility = () => {

    setShowPassword(!showPassword);

  };

  return (

    <div *className*="flex w-full min-h-screen items-center justify-center p-4 bg-slate-900">

      <div *className*="w-full max-w-md">

        <Card *className*="border-slate-700 bg-slate-800 shadow-xl">

          <CardHeader *className*="space-y-1">

            {*/\* Logo de ElectroTech \*/*}

            <div *className*="flex items-center justify-center gap-2 mb-2">

              <Package *className*="h-6 w-6 text-white" />

              <span *className*="font-bold text-xl text-white">ElectroTech</span>

            </div>

            <CardTitle *className*="text-3xl font-bold text-center text-white">

              Crear cuenta

            </CardTitle>

            <CardDescription *className*="text-slate-300 text-center">

              Ingrese sus datos para registrarse

            </CardDescription>

          </CardHeader>

          <CardContent>

            <form *onSubmit*={handleSubmit(onSubmit)} *className*="space-y-4">

              {*/\* Campo Nombre \*/*}

              <div *className*="space-y-2">

                <Label

*htmlFor*="nombre"

*className*="text-sm font-medium text-slate-200"

                >

                  Nombre

                </Label>

                <div *className*="relative">

                  <div *className*="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none text-slate-500">

                    <UserIcon *className*="h-4 w-4" />

                  </div>

                  <Input

*id*="nombre"

*type*="text"

*placeholder*="Ingrese su nombre"

*className*="pl-10 bg-slate-200 border-slate-600 text-zinc-800 placeholder:text-slate-500 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                    {...register("nombre")}

                  />

                </div>

                {errors.nombre && (

                  <p *className*="text-sm text-red-400 mt-1">

                    {errors.nombre.message}

                  </p>

                )}

              </div>

              {*/\* Campo Apellidos \*/*}

              <div *className*="space-y-2">

                <Label

*htmlFor*="apellidos"

*className*="text-sm font-medium text-slate-200"

                >

                  Apellidos

                </Label>

                <div *className*="relative">

                  <div *className*="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none text-slate-500">

                    <UserIcon *className*="h-4 w-4" />

                  </div>

                  <Input

*id*="apellidos"

*type*="text"

*placeholder*="Ingrese sus apellidos"

*className*="pl-10 bg-slate-200 border-slate-600 text-zinc-800 placeholder:text-slate-500 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                    {...register("apellidos")}

                  />

                </div>

                {errors.apellidos && (

                  <p *className*="text-sm text-red-400 mt-1">

                    {errors.apellidos.message}

                  </p>

                )}

              </div>

              {*/\* Campo Correo \*/*}

              <div *className*="space-y-2">

                <Label

*htmlFor*="correo"

*className*="text-sm font-medium text-slate-200"

                >

                  Correo electrónico

                </Label>

                <div *className*="relative">

                  <div *className*="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none text-slate-500">

                    <MailIcon *className*="h-4 w-4" />

                  </div>

                  <Input

*id*="correo"

*type*="email"

*placeholder*="ejemplo@correo.com"

*className*="pl-10 bg-slate-200 border-slate-600 text-zinc-800 placeholder:text-slate-500 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                    {...register("correo")}

                  />

                </div>

                {errors.correo && (

                  <p *className*="text-sm text-red-400 mt-1">

                    {errors.correo.message}

                  </p>

                )}

              </div>

              {*/\* Campo Contraseña \*/*}

              <div *className*="space-y-2">

                <Label

*htmlFor*="password"

*className*="text-sm font-medium text-slate-200"

                >

                  Contraseña

                </Label>

                <div *className*="relative">

                  <div *className*="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none text-slate-500">

                    <LockIcon *className*="h-4 w-4" />

                  </div>

                  <Input

*id*="password"

*type*={showPassword ? "text" : "password"}

*placeholder*="••••••••"

*className*="pl-10 pr-10 bg-slate-200 border-slate-600 text-zinc-800 placeholder:text-slate-500 focus-visible:ring-slate-500 focus-visible:border-slate-500"

                    {...register("password")}

                  />

                  <button

*type*="button"

*onClick*={togglePasswordVisibility}

*className*="absolute inset-y-0 right-0 flex items-center pr-3 text-slate-500 hover:text-slate-700"

                  >

                    {showPassword ? (

                      <EyeOffIcon *className*="h-4 w-4" />

                    ) : (

                      <EyeIcon *className*="h-4 w-4" />

                    )}

                    <span *className*="sr-only">

                      {showPassword

                        ? "Ocultar contraseña"

                        : "Mostrar contraseña"}

                    </span>

                  </button>

                </div>

                {errors.password && (

                  <p *className*="text-sm text-red-400 mt-1">

                    {errors.password.message}

                  </p>

                )}

                <p *className*="text-xs text-slate-400 mt-1">

                  La contraseña debe tener al menos 8 caracteres, una letra

                  mayúscula y un número.

                </p>

              </div>

              <div *className*="pt-2">

                <Button

*type*="submit"

*className*="w-full bg-slate-600 hover:bg-slate-500 text-white"

*disabled*={isLoading}

                >

                  {isLoading ? "Registrando..." : "Registrarse"}

                </Button>

              </div>

              <div *className*="text-center text-sm text-slate-300">

                ¿Ya tienes una cuenta?{" "}

                <Link

*to*="/login"

*className*="text-white hover:text-slate-200 font-medium"

                >

                  <span *className*="font-semibold">Iniciar sesión</span>

                </Link>

              </div>

            </form>

          </CardContent>

        </Card>

      </div>

    </div>

  );

}

## Login:

"use client";

import type React from "react";

import { useState } from "react";

import { loginRequest } from "@/api/auth";

import { useAuthStore } from "@/store/auth";

import type { UserLogin, UserProfile } from "@/types/user";

import { UserDirection } from "@/types/direccion";

import { Button } from "@/components/ui/button";

import { Input } from "@/components/ui/input";

import {

  Card,

  CardContent,

  CardDescription,

  CardHeader,

  CardTitle,

} from "@/components/ui/card";

import { Label } from "@/components/ui/label";

import { EyeIcon, EyeOffIcon, LockIcon, Package, UserIcon } from "lucide-react";

import { useNavigate, Link } from "react-router-dom";

export function LoginPage() {

  const [showPassword, setShowPassword] = useState(false);

  const [isLoading, setIsLoading] = useState(false);

  const setToken = useAuthStore((state) => state.setToken);

  const navigate = useNavigate();

  const handleSubmit = async (e: React.FormEvent<HTMLFormElement>) => {

    e.preventDefault();

    setIsLoading(true);

    try {

      const correo = (e.currentTarget.elements[0] as HTMLInputElement).value;

      const password = (e.currentTarget.elements[1] as HTMLInputElement).value;

      const user: UserLogin = {

        correo,

        password,

      };

      const resLogin = await loginRequest(user);

      console.log(resLogin.data);

      setToken(resLogin.data.token);

      const profile: UserProfile = {

        nombre: resLogin.data.user.nombre,

        correo: resLogin.data.user.correo,

        apellidos: resLogin.data.user.apellidos,

        rol: resLogin.data.user.rol,

        id: resLogin.data.user.id,

      };

      if (resLogin.data.user.direccion) {

        const direccion: UserDirection = {

          id: resLogin.data.user.direccion.id || "",

          departamento: {

            id: resLogin.data.user.direccion.departamento.id || "",

            nombre: resLogin.data.user.direccion.departamento.nombre || "",

          },

          pais: resLogin.data.user.direccion.pais || "",

          ciudad: resLogin.data.user.direccion.ciudad || "",

          zona: resLogin.data.user.direccion.zona || "",

          calle: resLogin.data.user.direccion.calle || "",

          numero: resLogin.data.user.direccion.numero || "",

          referencia: resLogin.data.user.direccion.referencia || "",

        };

        useAuthStore.setState({ directions: direccion });

      } else {

        const direccion: UserDirection = {

          id: 0,

          departamento: {

            id: 0,

            nombre: "",

          },

          pais: "",

          ciudad: "",

          zona: "",

          calle: "",

          numero: "",

          referencia: "",

        };

        useAuthStore.setState({ directions: direccion });

      }

      useAuthStore.setState({ profile });

      navigate("/inicio");

    } catch (error) {

      console.error("Login failed:", error);

    } finally {

      setIsLoading(false);

    }

  };

  const togglePasswordVisibility = () => {

    setShowPassword(!showPassword);

  };

  return (

    <div *className*="flex w-full h-screen min-h-[calc(100vh-150px)] items-center justify-center p-4 bg-slate-900">

      <div *className*="w-full max-w-md">

        <Card *className*="border-slate-700 bg-slate-800 shadow-xl">

          <CardHeader *className*="space-y-1">

            {*/\* Logo de ElectroTech \*/*}

            <div *className*="flex items-center justify-center gap-2 mb-2">

              <Package *className*="h-6 w-6 text-white" />

              <span *className*="font-bold text-xl text-white">ElectroTech</span>

            </div>

            <CardTitle *className*="text-3xl font-bold text-center text-white">

              Bienvenido

            </CardTitle>

            <CardDescription *className*="text-slate-300 text-center">

              Ingrese sus datos para acceder a su cuenta

            </CardDescription>

          </CardHeader>

          <CardContent>

            <form *onSubmit*={handleSubmit} *className*="space-y-4">

              <div *className*="space-y-2">

                <Label

*htmlFor*="correo"

*className*="text-sm font-medium text-slate-200"

                >

                  Usuario

                </Label>

                <div *className*="relative">

                  <div *className*="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none text-slate-500">

                    <UserIcon *className*="h-4 w-4" />

                  </div>

                  <Input

*id*="correo"

*type*="text"

*placeholder*="Correo"

*className*="pl-10 bg-slate-200 border-slate-600 text-zinc-800 placeholder:text-slate-500 focus-visible:ring-slate-500 focus-visible:border-slate-500"

*required*

                  />

                </div>

              </div>

              <div *className*="space-y-2">

                <Label

*htmlFor*="password"

*className*="text-sm font-medium text-slate-200"

                >

                  Contraseña

                </Label>

                <div *className*="relative">

                  <div *className*="absolute inset-y-0 left-0 flex items-center pl-3 pointer-events-none text-slate-500">

                    <LockIcon *className*="h-4 w-4" />

                  </div>

                  <Input

*id*="password"

*type*={showPassword ? "text" : "password"}

*placeholder*="••••••••"

*className*="pl-10 pr-10 bg-slate-200 border-slate-600 text-zinc-800 placeholder:text-slate-500 focus-visible:ring-slate-500 focus-visible:border-slate-500"

*required*

                  />

                  <button

*type*="button"

*onClick*={togglePasswordVisibility}

*className*="absolute inset-y-0 right-0 flex items-center pr-3 text-slate-400 hover:text-slate-200"

                  >

                    {showPassword ? (

                      <EyeOffIcon *className*="h-4 w-4" />

                    ) : (

                      <EyeIcon *className*="h-4 w-4" />

                    )}

                    <span *className*="sr-only">

                      {showPassword

                        ? "Ocultar contraseña"

                        : "Mostrar contraseña"}

                    </span>

                  </button>

                </div>

              </div>

              <Button

*type*="submit"

*className*="w-full bg-slate-600 hover:bg-slate-500 text-white"

*disabled*={isLoading}

              >

                {isLoading ? "Iniciando sesión..." : "Iniciar sesión"}

              </Button>

              <div *className*="text-center text-sm text-slate-300">

                ¿No tienes una cuenta?{" "}

                <Link

*to*="/registro"

*className*="text-white hover:text-slate-200 font-medium"

                >

                  <span *className*="font-semibold">Crear cuenta</span>

                </Link>

              </div>

            </form>

          </CardContent>

        </Card>

      </div>

    </div>

  );

}

## Products:

import { ProductGrid } from "@/components/product-grid";

import { ProductFilters } from "@/components/product-filters";

export function ProductsPage() {

  return (

    <div *className*="container mx-auto px-4 py-8">

      <h1 *className*="text-3xl font-bold text-white mb-8">All Products</h1>

      <div *className*="flex flex-col md:flex-row gap-8">

        <div *className*="w-full md:w-1/4">

          <ProductFilters />

        </div>

        <div *className*="w-full md:w-3/4">

          <ProductGrid />

        </div>

      </div>

    </div>

  );

}