**Introducción**

La aplicación web es sistema de punto de venta (POS) diseñado para gestionar ventas, inventario y clientes de manera eficiente. Permite a los usuarios registrar productos, procesar transacciones y generar reportes en tiempo real. Con una interfaz intuitiva y funciones clave como gestión de usuarios, historial de ventas y control de stock, esta aplicación facilitara la administración de un negocio y mejora la toma de decisiones.

Esta aplicación de punto de venta optimizara la gestión de ventas e inventario en un negocio, facilitando el control de productos, clientes y transacciones en un solo sistema. Es necesaria porque permite automatizar procesos, reducir errores manuales y mejorar la eficiencia operativa. Además, proporciona reportes en tiempo real, lo que ayuda en la toma de decisiones basadas en datos precisos. En un entorno comercial competitivo, contar con un sistema digital como esta mejora la organización y contribuye al crecimiento del negocio.

**Requerimientos Funcionales:**

**Front-end:**

* Panel para realizar ventas rápidas con escaneo de códigos de barras.
* Gestión de productos, categorías y precios dinámicos.
* Interfaz para aplicar descuentos y promociones.
* Integración de métodos de pago (efectivo, tarjeta, transferencia).

**Back-end:**

* Base de datos para productos, transacciones y clientes.
* Algoritmo para calcular descuentos automáticos y promociones.
* API para gestionar reportes de ventas en tiempo real.
* Análisis de ventas por hora, día y producto más vendido

**Requerimientos Generales:**

* Diseño Responsivo (adaptado a cada pantalla)
* Login y Registro de usuarios
* Aplicar integración de inteligencia artificial
* Manual técnico y de usuario (PDF)
* Documentación del progreso del desarrollo (inicio, final)
* CRUD de todas las entidades de la aplicación ej. (Productos, Usuarios, Empleados)
* Permisos de usuario
* UI/UX
* Diseño de reportes o informe de resultado en cada proyecto ej. Ver la ventas del último mes

**Tecnologías implementadas en el proyecto:**

**Front-end:**

* **Tecnologías base:**
* HTML
* CSS
* JavaScript
* **Librerías y herramientas:**
* TailwindCss
* FlowBite
* Google Fonts
* Font-Awesome
* Heroicons
* MerakiUI
* Wickedblocks
* DjangoTemplates
* UnDraw
* Django-Jazmín

**Back-end:**

* **Tecnologías base:**
* Django (Python)
* MySQL
* **Librerías y herramientas:**
* Hunter.io (API para verificar los correos electrónicos)
* Stripe (API para los pagos con tarjeta.)
* Chart.js (API para los gráficos.)
* Gmail (API para enviar emails.)
* Html5QrCode (API para manejar los códigos QRs)

**Plan de trabajo**

**Semana 1:**

En la primera semana del desarrollo, estuve analizando los requisitos mas profundamente, y tratando de determinar como seria el flujo de mi aplicación, además de las tecnologías y librerías que iba usar para UI/UX esta semana fue para pensar sobre la estructura de mi proyecto.

**Semana 2:**

En esta semana lo único que hice fue crea la estructura de carpetas de mi proyecto.

**Semana 3:**

En esta semana empecé creando mi base de datos y tabla de usuario, hice un login y algunas views.

También empecé y terminé algunas funcionalidades del dashboard como agregar usuarios, editar usuarios, enviar correos y captar errores.

**Semana 4:**

En esta semana agregué la funcionalidad de los productos y categorías y hice una parte del apartado de los empleados que donde se renderizan los productos.

**Semana 5:**

En esta semana implemente la mayor parte de la funcionalidad de escanear productos con QR y hice la parte de pagos de efectivo y transferencia solo front-end.

**Semana 6:**

En esta semana termine de implemente la funcionalidad de escanear productos con QR con todos los métodos de pagos y descuentos, también empezamos con el front-end del inventario y las tablas de esta app.

**Semana 7:**

En esta semana empezamos a implementar los gráficos de las ventas, y terminamos el inventario completamente.

**Semana 8:**

En esta semana tuve que arreglar los roles de mi app y luego de terminar implemente views de “forgot you password” y de editar perfil, aparte de eso también terminamos de implementar los gráficos de ventas e hicimos el historial de movimientos.

**Semana 9:**

En esta semana implementamos el apartado de devoluciones, agregamos notificaciones sobre stock e hicimos los gráficos sobre las devoluciones.

**Semana 10:**

En esta semana agregamos un Bot de WhatsApp para el soporte y reportes.

**Semana 11:**

En esta semana hicimos el manual de uso, el manual técnico y terminamos la documentación.

**Documentación de las APIs**

En este sistema, se han integrado diversas APIs para mejorar la funcionalidad y la experiencia del usuario. A continuación, se detallan las APIs utilizadas, su propósito y cómo se implementaron.

**1. Hunter.io (Verificación de Correos Electrónicos)**

**Descripción**: Hunter.io es una API que permite verificar direcciones de correo electrónico para determinar su validez.

**Propósito en el sistema:** Se utiliza para validar las direcciones de correo electrónico proporcionadas por los usuarios y reducir el riesgo de registros con correos inválidos.

**Como se usa:**

* Se envía una solicitud a la API con la dirección de correo a verificar.
* La API responde con información sobre la validez del correo.
* Se analiza la respuesta para decidir si se permite o no el registro con ese correo.
* Para usar esta API debemos ir al sitio oficial de hunter.io crearnos una cuenta y buscar la API key luego la ponemos en nuestro settings.py y listo.

**2. Stripe (Pagos con Tarjeta)**

**Descripción:** Stripe es una API que facilita el procesamiento de pagos con tarjeta de crédito y débito.

**Propósito en el sistema:** Se utiliza para gestionar los pagos de los usuarios de forma segura y eficiente.

**Cómo se usa:**

* Se integra Stripe en modo *sandbox* para pruebas ya que en república dominicana no está disponible sin una cuenta de banco.
* Para usar Stripe debemos ir al sitio web de Stripe y crearnos una cuenta luego que este creada vamos al sandbox <https://dashboard.stripe.com/test/dashboard> de aquí vamos a developers y API KEY tomamos la PUBLIC API y la SECRET API y la ponemos en nuestro settings.py y listo.
* Se generan claves de API para autenticar solicitudes.
* Se crean sesiones de pago y se manejan respuestas de éxito o fallo.

**Enlace al sandbox:** [Stripe Sandbox](https://dashboard.stripe.com/test/dashboard)

**3. Chart.js (Generación de Gráficos)**

**Descripción:** Chart.js es una API que permite la generación de gráficos interactivos en páginas web.

**Propósito en el sistema:** Se utiliza para visualizar datos de manera gráfica, facilitando la interpretación de información clave.

**Cómo se usa:**

* Se incluyen los enlaces de Chart.js en el proyecto.
* Se definen los datos y configuraciones de los gráficos.
* Se renderizan los gráficos en el front-end usando *canvas* de HTML5.

**4. Gmail API (Envío de Correos Electrónicos)**

**Descripción:** La API de Gmail permite enviar correos electrónicos de manera programática a través de una cuenta de Google.

**Propósito en el sistema:** Se utiliza para el envío automatizado de correos, como confirmaciones de registro y notificaciones importantes.

**Cómo se usa:**

* Se configura un proyecto en Google Cloud para obtener credenciales.
* Se autentica la aplicación para acceder a la cuenta de Gmail.
* Se envían correos electrónicos a través de solicitudes HTTP a la API.

**5. HTML5-QRCode (Escaneo de Códigos QR)**

**Descripción:** HTML5-QRCode es una librería basada en JavaScript que permite escanear códigos QR usando la cámara del dispositivo.

**Propósito en el sistema:** Se utiliza para facilitar la lectura de códigos QR en tiempo real sin necesidad de aplicaciones externas.

**Cómo se usa:**

* Se importa la librería HTML5-QRCode en el proyecto.
* Se configura el escáner con opciones como la resolución y el número de cuadros por segundo.
* Se captura el código QR desde la cámara y se procesa su contenido.

Estas APIs han sido integradas para mejorar la funcionalidad del sistema y proporcionar una mejor experiencia de usuario.