# TiendPlus - Sistema Contable

Ingeniería Informática Nocturna

Integrantes:

Angela Gabriela Cruz Anacona

Cristian Alexis Hurtado Chantre

Luis Eduardo Hurtado Hormiga

Docente: María Bastidas

Fecha: 02/05/2025

Índice

[Este índice se generará automáticamente al final]

# Introducción

Nuestro proyecto es una aplicación diseñada para la gestión de ventas de tiendas de barrio, permitiendo la gestión eficiente de ventas, proveedores e inventario. Incluye funcionalidades clave como registro de productos, administración de proveedores y generación de historial de ventas. Además, maneja diferentes tipos de pago, incluidas ventas fiadas.

# Objetivos

- Objetivo general:

Desarrollar una aplicación de software llamada *TiendPlus - Sistema Contable*, que permita gestionar de manera eficiente las operaciones comerciales de una tienda o supermercado de barrio, incluyendo control de inventario, ventas, fiados, proveedores y reportes, mediante una interfaz gráfica intuitiva y segura.

- Objetivos específicos:

* Implementar un sistema de autenticación de usuarios con distintos roles (administrador, proveedor, cajero).
* Desarrollar funcionalidades CRUD para la gestión de proveedores y productos.
* Permitir el registro de ventas normales y ventas fiadas
* Diseñar una base de datos relacional que almacene de forma organizada la información de proveedores, productos, usuarios, clientes y ventas.
* Integrar reportes de ventas, productos más vendidos y clientes con mayor deuda.
* Aplicar principios SOLID y buenas prácticas de programación orientada a objetos.
* Garantizar la seguridad de los datos mediante cifrado de contraseñas y control de accesos por rol.
* Crear una interfaz gráfica sencilla e intuitiva para usuarios con poca experiencia tecnológica.
* Asegurar la escalabilidad y mantenibilidad del sistema para futuras mejoras o integraciones.
* Probar el sistema mediante casos de prueba para verificar su correcto funcionamiento.

# Requisitos del sistema

## Requisitos funcionales

- Crear, ver, actualizar y eliminar proveedores.  
- Registrar productos con detalles como nombre, categoría, precio y cantidad.  
- Modificar y eliminar productos.  
- Consultar el inventario disponible.  
  
Gestión de clientes y fiados:  
- Registrar clientes con información básica.  
- Registrar préstamo a clientes.  
- Actualizar y consultar el historial de pagos de los clientes.  
- Visualizar lista de préstamos.  
  
Seguridad y Autenticación:  
- Registro e inicio de sesión.  
- Diferentes roles y permisos (administrador, cajero y proveedor).  
  
Reportes y Estadísticas:  
- Reportes de ventas, productos más vendidos y clientes con mayor deuda.

## Requisitos no funcionales

Usabilidad:  
- Interfaz sencilla e intuitiva para personas con poca experiencia en tecnología.  
  
Escalabilidad:  
- Posibilidad de añadir más funcionalidades en el futuro sin afectar el sistema actual.  
  
Seguridad:  
- Cifrado de contraseñas y datos sensibles.  
- Control de acceso a ciertas funcionalidades según el rol del usuario.  
  
Rendimiento:  
- Carga rápida de información y procesamiento eficiente de datos.  
  
Compatibilidad:  
- Compatible con navegadores populares como Chrome y Firefox.

# Diseño del sistema

Se desarrolla con los siguientes componentes en base a la programación orientada a objetos:  
  
Clases principales:  
- Usuario, Producto, Historial y Pago  
  
Interfaces:  
- InterfazUsuario, InterfazVenta, InterfazHistorial y InterfazTipoPago  
  
Herencia y Polimorfismo:  
- Clases como VentaFiada y VentaNormal heredan de Venta, lo que permite flexibilidad en la gestión de ventas.

# Implementación

- Tecnologías usadas

* Lenguaje de programación usado:

Java: Usado para todo el desarrollo del sistema, incluyendo la lógica del CRUD, conexión con base de datos y la interfaz gráfica.

* Entorno de Desarrollo:

VS Code: Utilizado como editor principal para escribir el código Java, con extensiones para compilar y ejecutar.

* Interfaz Gráfica:

Java Swing: Librería usada para construir la interfaz gráfica de usuario (ventanas, botones, tablas, formularios, etc.).

* Base de datos:

Mysql: Motor de base de datos donde se guardan los productos, proveedores, usuarios, roles, etc.

Mysql Workbrench: Herramienta gráfica utilizada para crear, visualizar y administrar la base de datos y sus tablas.

# Arquitectura

MVC(Modelo, Vista, Controlador)

* **Modelo**: clases como Producto, ProveedorContacto, y sus DAOs (ProductoDAO, ProveedorDAO) que gestionan los datos.
* **Vista**: clases como VentanaProductos, VentanaProveedores, que crean la interfaz de usuario.
* **Controlador**: la lógica que conecta la vista con el modelo, en parte manejada en los listeners de botones.

# Diagramas

Diagrama de clases:

# 

Diagrama de Procesos

# Principios SOLID utilizados

Patrón DAO (Data Access Object)

El patrón DAO se usa para separar la lógica de acceso a datos (base de datos) del resto de la aplicación permitiendo un código más limpio y mantenible.

SRP (Principio de Responsabilidad Única):  
- Cada clase tiene una única responsabilidad. Ejemplo: La clase Usuario maneja autenticación, mientras que la clase Venta solo gestiona las ventas.  
  
OCP (Principio de Abierto/Cerrado):  
- Venta se extiende con VentaFiada y VentaNormal sin modificar su estructura base.  
  
DIP (Principio de Inversión de Dependencias):  
- Pago define una interfaz para PagoDigital y PagoEfectivo, lo que permite agregar nuevos métodos de pago sin alterar el código existente.

Patrón Strategy (Estrategia): Para implementarlo, podrías tener una interfaz MetodoPago que defina un método procesarPago(), y luego varias clases concretas que implementen esta interfaz, como PagoConTarjeta y PagoConPayPal.

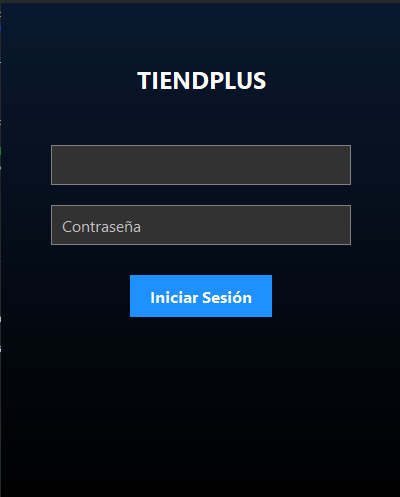
# Estandares de calidad

Estándares de Calidad que se Emplearan en el Proyecto:

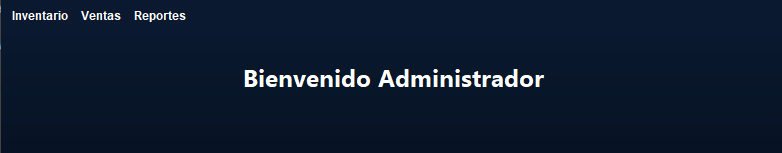
* Usabilidad: Diseño de una interfaz gráfica sencilla para que los usuarios puedan tener una interacción fácil con los diferentes módulos.
* Modificabilidad: Las clases están separadas por módulos, lo que nos permitirá añadir cuando se requiera nuevas funcionalidades fácilmente, como ejemplo: integraciones con pagos digitales.
* Rendimiento: Optimización a las consultas a la base de datos para que los reportes de ventas y de productos se generen rápidamente cuando el usuario lo requiera.
* Seguridad: Implementación de autenticación de usuario, encriptación de contraseñas y control de acceso mediante roles, asegurando que solo usuarios autorizados puedan realizar funciones en la aplicación.

# Imagines del Proyecto

* Login:



* Menu del Admin:





* Menu Rol(Proveedor)

