

PROYECTO
FINAL

LA TIENDITA

INTEGRANTES

AMORES HERNÁNDEZ CARLOS DANIEL, A210367
CASTELLANOS PAVÓN BRAYAN DE JESÚS, A210416.
CIGARROA HERNÁNDEZ LUISA FERNANDA, A2104118.
CRUZ LEÓN JESÚS ADRIÁN, A210395.
ESTRADA DE LA CRUZ MARCO ANTONIO, A210179.



POR QUÉ SE ESCOGIÓ



Se eligió desarrollar “**La Tiendita**” como sistema de base de datos distribuida porque representa un negocio cercano, funcional y común en la vida cotidiana.

Las tiendas de abarrotes manejan productos, ventas, clientes e inventarios constantemente, lo que las hace ideales para aplicar conceptos clave del curso como replicación de datos, normalización, diseño relacional y sincronización entre entidades distribuidas.

Este proyecto busca digitalizar procesos que muchas veces siguen siendo manuales, ofreciendo:

- Control seguro de operaciones mediante autenticación y roles.
- Un sistema escalable para futuras sucursales.
- Una interfaz moderna y fácil de usar.

FINALIDAD DE LA BASE DE DATOS

La base de datos distribuida “**La Tiendita**” busca ofrecer una solución tecnológica robusta que gestione eficientemente una tienda de abarrotes, con capacidad de crecer, proteger los datos, asegurar su integridad y apoyar decisiones con información en tiempo real.

Su finalidad principal es:

- Centralizar la información crítica.
- Acceso controlado por roles.
- Trazabilidad y auditoría.
- Modelo distribuido para sucursales.
- Agilizar procesos diarios.



MODELO DE DATOS

El modelo de datos de “**La Tiendita**” fue diseñado para reflejar la estructura operativa de una tienda de abarrotes, siendo escalable, seguro y fácil de consultar y mantener, con entidades (tablas) que cumplen funciones específicas dentro del sistema.

Para mantener un sistema organizado y escalable, se definieron las siguientes tablas clave:

- Clientes.
- Ventas y Detalle de Ventas.
- Productos.
- Proveedores.
- Empleados.
- Inventarios.
- Compras: registra lo que el cliente compra.
- Bitácora.



NORMALIZACIÓN

Para garantizar la integridad de los datos y evitar redundancias, se aplicaron las tres primeras formas normales:



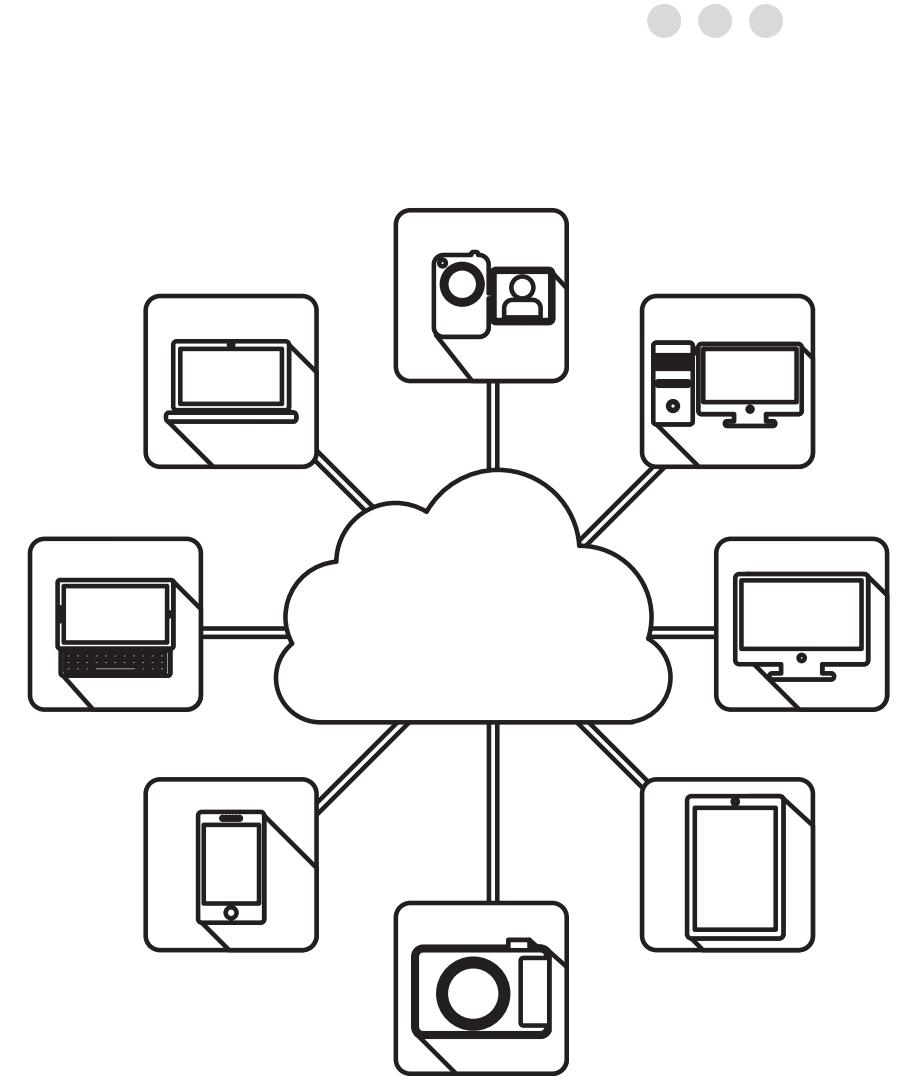
- **Primera Forma Normal (1FN):** Se eliminó cualquier grupo repetido de datos, cada campo almacena una sola pieza de información.
- **Segunda Forma Normal (2FN):** Se eliminaron dependencias parciales, cada campo no clave depende completamente de la clave primaria.
- **Tercera Forma Normal (3FN):** Se eliminaron dependencias transitivas. Ningún campo no clave depende de otro campo no clave.

ENTIDADES CENTRALIZADAS VS DISTRIBUIDAS

Se analizaron las operaciones para definir qué datos pueden estar centralizados (una sola instancia global) y cuáles distribuidos (cada sucursal o parte del sistema tiene su propia copia y luego sincroniza):

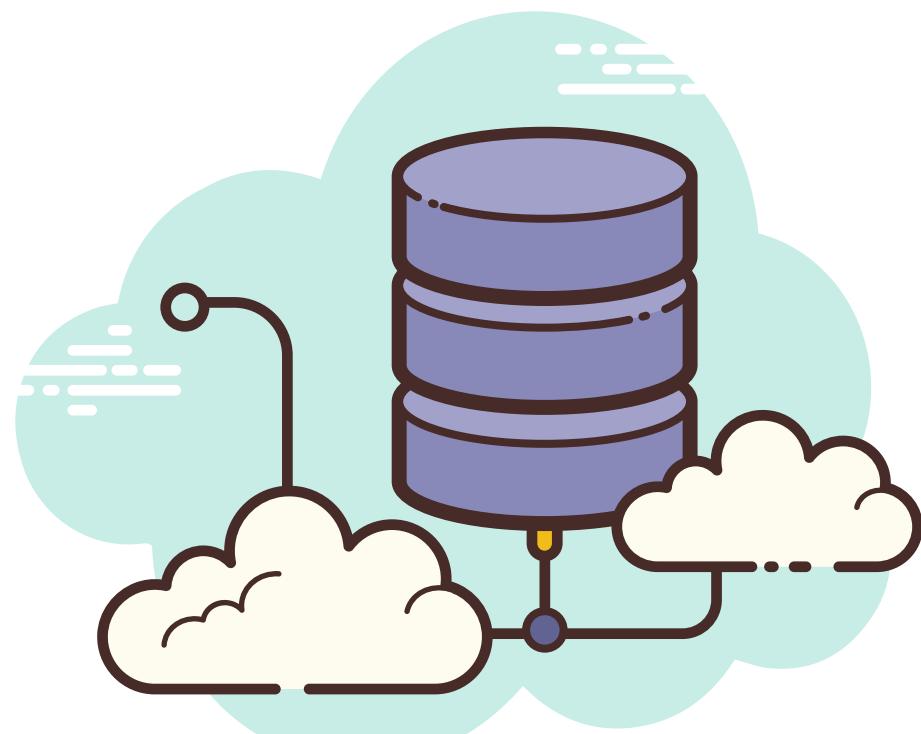
Centralizadas:

- **Proveedores:** Un catálogo global común.
- **Productos:** Lista base, aunque stock puede variar.
- **Bitácora:** Registros importantes deben estar centralizados para control.



Distribuidas:

- **Ventas:** Cada sucursal genera las suyas.
- **Clientes:** Cada punto puede registrar nuevos clientes.
- **Empleados:** Pueden pertenecer a diferentes ubicaciones.
- **Inventario:** Varía según la sucursal o almacén.



Esto permite que cada parte del sistema funcione localmente y se sincronice con la base global, ideal para arquitectura distribuida.



HERRAMIENTAS Y TECNOLOGÍAS UTILIZADAS

Estas herramientas se eligieron por su robustez, seguridad y amplia documentación, facilitando el desarrollo y mantenimiento del sistema.

Las herramientas seleccionadas son las siguientes:

- **Frontend:** React.js y TailwindCSS: React.js permite crear interfaces modernas y dinámicas, mientras que TailwindCSS facilita el diseño rápido y personalizable mediante clases utilitarias.
- **Backend Python + Flask:** Flask es un framework ligero ideal para desarrollar APIs RESTful, con fácil integración a SQL Server.
- **Base de Datos, SQL Server:** Proporciona un soporte robusto para integridad referencial, vistas, triggers, replicación y seguridad de los datos.
- **Seguridad:** JWT asegura la autenticación mediante tokens, bcrypt almacena las contraseñas de forma segura, y se implementan medidas de protección contra ataques como XSS y SQL Injection.



DISEÑO DEL SITIO Y UX/UI

El diseño del sistema "**La Tiendita**" fue concebido para brindar una experiencia de usuario moderna, intuitiva y visualmente agradable, tanto para el administrador como para el cajero. Se implementaron principios de diseño centrado en el usuario (UX) y se usaron herramientas visuales para facilitar la interacción, reducir la curva de aprendizaje y reflejar la identidad de una tienda de abarrotes.

El sistema tiene un enfoque amigable, limpio y profesional:

- Estilo visual tipo dashboard para administración eficiente.
- Login con efecto glassmorphism (vidrio borroso).
- Uso de colores con significado: **Amarillo pastel** (optimismo, energía), **Naranja claro** (calidez, confianza) y **Azul suave** (profesionalismo, limpieza).
- Todo es completamente responsive y se adapta a móviles.
- El panel cambia de vista dependiendo del tipo de usuario (admin o cajero).

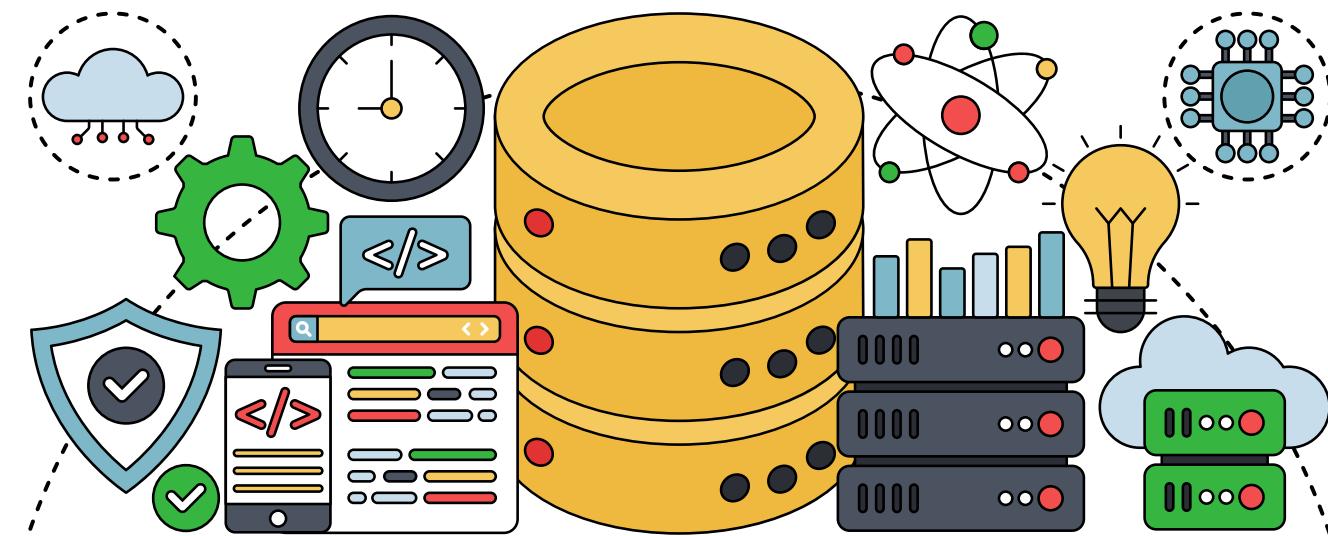


ARQUITECTURA DEL SISTEMA

El sistema se basa en una arquitectura **cliente-servidor**:

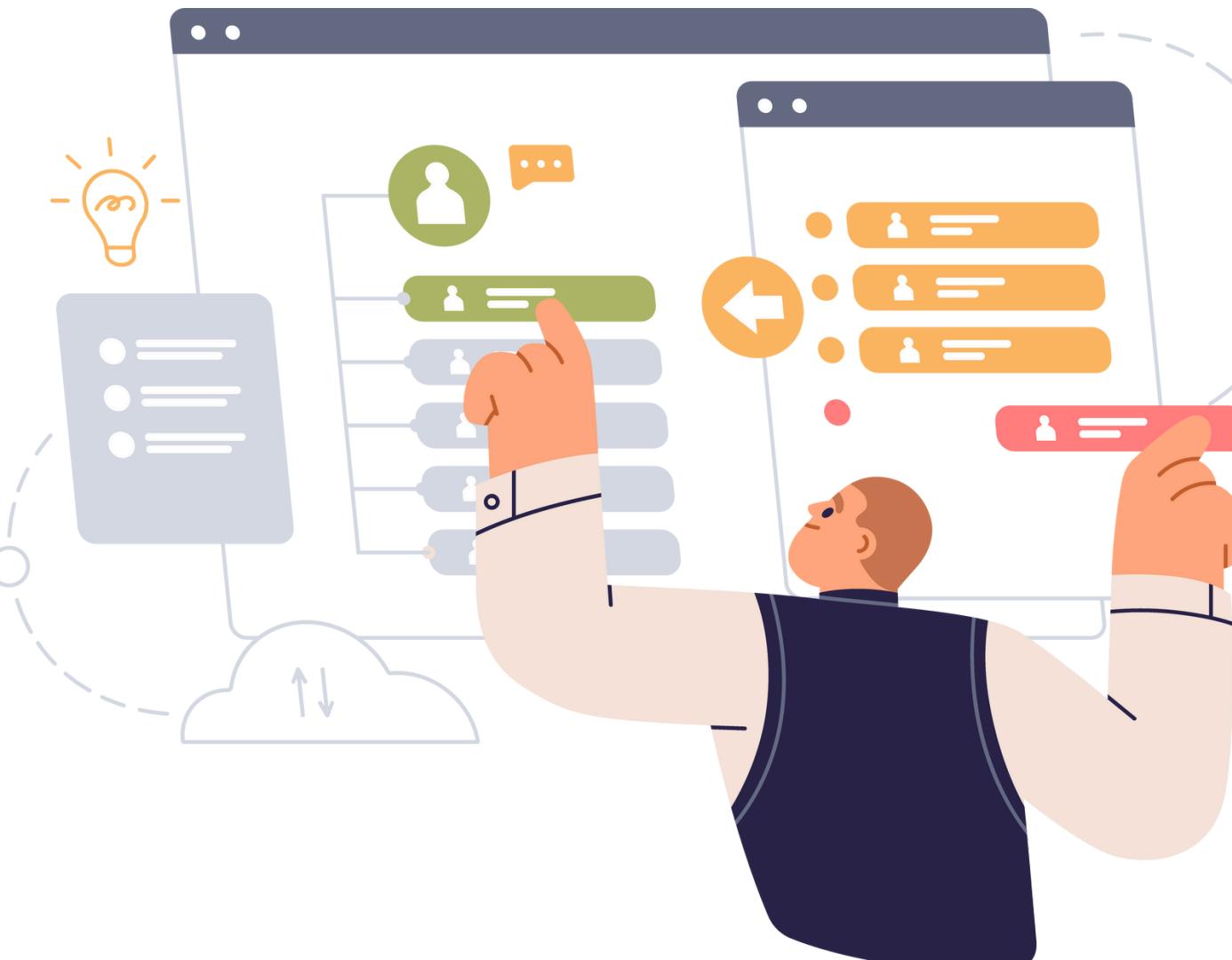
Frontend (React):

- Envía peticiones HTTP a la API con fetch o axios.
- Maneja rutas protegidas, autenticación, estados globales.



Backend (Flask):

- Expone rutas /login, /productos, /ventas, etc.
- Protege rutas con decoradores de token JWT.
- Realiza validaciones y comunicación con la base de datos.



Base de Datos (SQL Server):

- Estructura relacional, con claves y relaciones bien definidas.
- Se utilizan vistas, bitácoras, y se planea replicación para escalar.
- Optimización mediante índices, parámetros y consultas específicas para cada operación.

SEGURIDAD DEL SISTEMA

La seguridad es un componente fundamental en el diseño de “**La Tiendita**”, ya que se maneja información sensible como ventas, productos, usuarios y contraseñas. El sistema implementa múltiples capas de seguridad tanto en el frontend como en el backend para garantizar confidencialidad, integridad y disponibilidad de los datos.

El sistema implementa medidas de seguridad modernas:

🔒 Autenticación y Autorización:

- JWT en frontend/backend para acceso seguro.
- Los roles (admin, cajero) controlan qué rutas pueden usar.

🔑 Encriptación de contraseñas:

- bcrypt asegura que nunca se guarde una contraseña en texto plano.

🚫 Protección contra ataques:

- XSS: los campos están sanitizados.
- SQL Injection: se usan consultas parametrizadas.
- Bitácoras: registro de todas las acciones importantes del sistema.

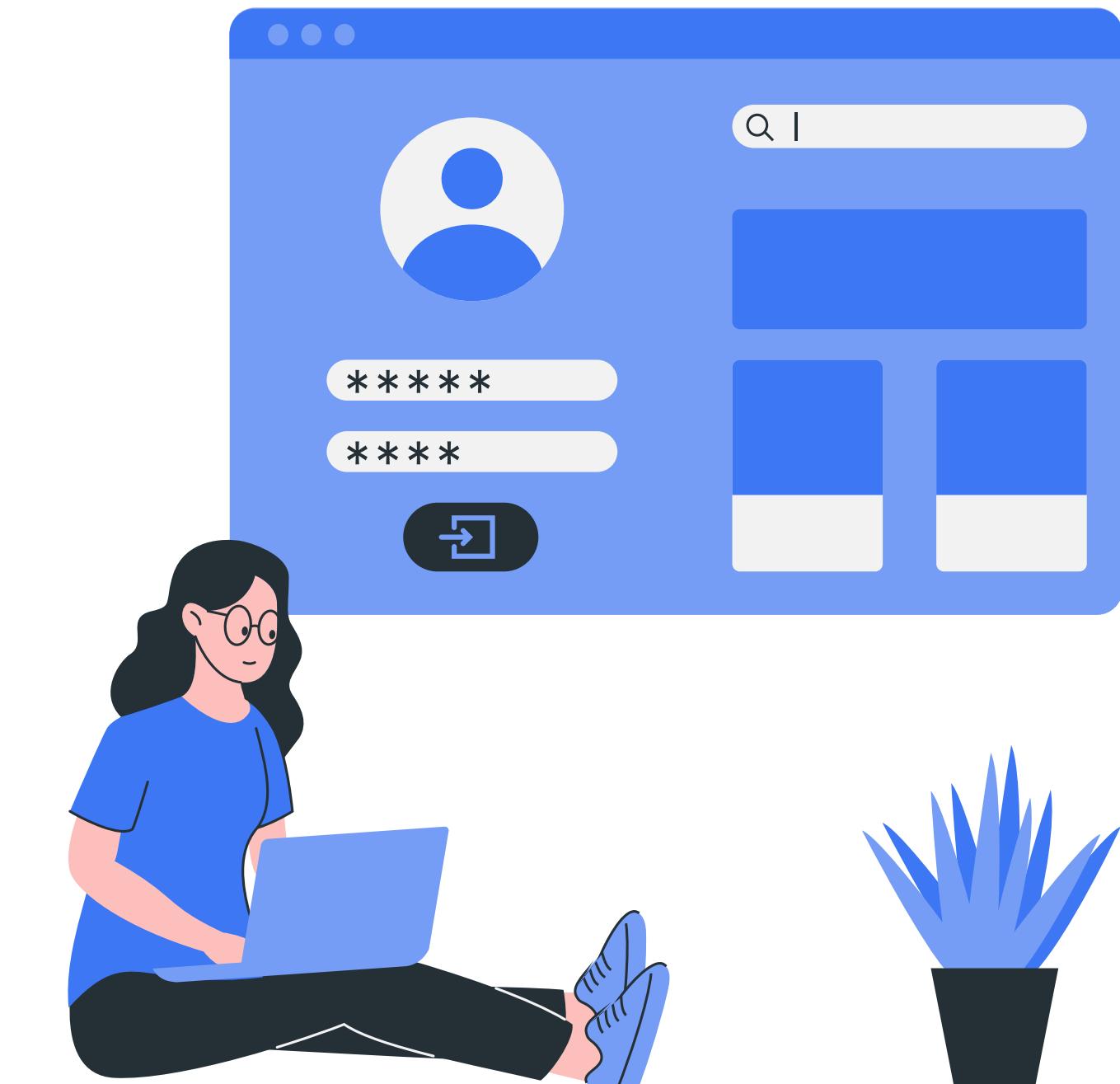
Todo el acceso a rutas sensibles requiere un token válido, evitando sesiones falsas o usuarios no autorizados.



MÓDULOS FUNCIONALES DEL SISTEMA

El sistema "**La Tiendita**" fue diseñado con una arquitectura modular que permite separar claramente cada funcionalidad según el rol del usuario. Esto mejora el mantenimiento, la escalabilidad y la experiencia de usuario. A continuación, se describen los módulos implementados hasta el momento:

- ✓ **1. Módulo de Inicio de Sesión (Login):** Valida credenciales y redirige según el rol. Usa bcrypt para contraseñas y JWT para autenticación segura.
- 🔧 **2. Panel de Administrador:** Muestra métricas clave (ventas, stock, clientes, etc.) en tarjetas visuales. Facilita decisiones rápidas desde un dashboard moderno.
- 📦 **3. Módulo de Productos:** Permite listar, agregar y gestionar productos. Se conecta a la base de datos y registra cambios en bitácora. Seguridad con validaciones y tokens.





MÓDULOS FUNCIONALES DEL SISTEMA

-  **4. Módulo de Cajero (planeado):** Permitirá registrar ventas escaneando productos, confirmar pagos y actualizar stock, todo de forma ágil.
-  **5. Módulo de Clientes (planeado):** Permitirá registrar datos de clientes frecuentes para promociones personalizadas y seguimiento de historial de compras.
-  **6. Módulo de Ventas (planeado):** Permitirá ver, filtrar y administrar ventas realizadas con detalles por producto, total y cajero.
-  **10. Modularidad y escalabilidad:** Cada módulo es independiente, fácil de mantener, con rutas protegidas por rol y capacidad de expansión futura.
-  **7. Inventario (planeado):** Controlará existencias, alertará sobre stock bajo y generará reportes por categoría o proveedor.
-  **8. Módulo de Compras (planeado):** Permitirá registrar compras a proveedores y controlar entradas al inventario con historial detallado.
-  **9. Bitácora (implementada):** Registra todas las acciones del sistema con fecha, usuario y tabla afectada, útil para auditoría y seguridad.

PROYECTO
FINAL

GRACIAS

