UNIVERSIDAD CENTRAL DEL ECUADOR

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

INGENIERÍA EN SISTEMAS DE INFORMACIÓN

SISTEMAS DE INFORMACIÓN EMPRESARIAL

MANUAL TÉCNICO

ING. NORALMA YANEZ

SEMESTRE: 10MO – P1

FRANCISCO CUMBAL

JEFFERSON JACOME

JONATHAN VARGAS

30/06/2025

2025-2025

MANUAL TÉCNICO

Aplicación Web: Sistema de Gestión – “Distribuidora Tapia”

Versión del prototipo: 0.9 | Fecha de elaboración: Julio 2025

1. Introducción

Este manual técnico proporciona toda la información necesaria para comprender, desplegar, mantener y escalar el prototipo funcional del sistema de gestión Distribuidora Tapia. La solución fue desarrollada con tecnologías modernas de desarrollo web, bajo arquitectura modular, con enfoque en velocidad de despliegue y simplicidad de uso. Actualmente opera sobre infraestructura serverless en Vercel, con datos simulados en memoria.

2. Objetivo del sistema

- Simular un sistema ERP web para pequeñas empresas de distribución.
- Implementar lógica de inventario y flujo de caja basada en el modelo Just in Time (JIT).
- Proporcionar una base escalable para futuras integraciones con bases de datos reales, autenticación y automatizaciones externas.

3. Tecnologías utilizadas

Capa	Tecnología	Detalle
Lenguaje principal	TypeScript	Uso en frontend (React) y backend (Next.js)
Framework web	Next.js v14	Renderizado híbrido, rutas dinámicas
Interfaz (UI)	Tailwind CSS + shadcn/ui	Diseño responsivo moderno
Estado global	Zustand	Slices personalizados para módulos
Validación de datos	Zod + React Hook Form	Seguridad en formularios
API Routes	Next.js API (mock in-memory)	CRUD local sin base de datos externa
Despliegue	Vercel	CI/CD automático con rama main
Testing	Vitest + React Testing Library	Pruebas de hooks, lógica y funciones

4. Estructura del proyecto

Estructura base del repositorio:

```
src/
├── app/                    # Next.js App Router
│   ├── api/                # API Routes
│   │   ├── products/       # CRUD de productos
│   │   ├── sales/          # Gestión de ventas
│   │   ├── purchases/      # Gestión de compras
│   │   ├── customers/      # Gestión de clientes
│   │   ├── suppliers/      # Gestión de proveedores
│   │   ├── inventory/      # Monitoreo de inventario
│   │   └── reports/        # Generación de reportes
│   ├── globals.css         # Estilos globales
│   ├── layout.tsx          # Layout principal
│   └── page.tsx            # Página principal
├── components/             # Componentes React
│   ├── ui/                 # Componentes de UI base
│   ├── Navigation.tsx      # Navegación principal
│   ├── ProductManagement.tsx
│   ├── SalesManagement.tsx
│   ├── PurchasesManagement.tsx
│   ├── InventoryManagement.tsx
│   ├── CustomerSupplierManagement.tsx
│   └── ReportsManagement.tsx
├── contexts/               # Contextos de React
│   └── ThemeContext.tsx    # Gestión de tema
├── lib/                    # Utilidades
│   ├── db.ts               # Configuración de base de datos
│   ├── init-db.ts          # Inicialización de tablas
│   └── utils.ts            # Funciones de utilidad
└── types/                  # Definiciones de TypeScript
    └── index.ts            # Tipos e interfaces
```

5. Modelado de datos (tipos principales)

- products - Información de productos
- customers - Datos de clientes
- suppliers - Datos de proveedores
- sales - Cabecera de ventas
- sale_items - Detalles de ventas
- purchases - Cabecera de compras

- purchase_items - Detalles de compras

6. API y endpoints simulados

Actualmente, el sistema opera con datos simulados, gestionados desde memoria. Las rutas disponibles son:

Método	Ruta	Función
GET	/api/products	Obtener listado de productos
POST	/api/products	Crear nuevo producto
PUT	/api/products/[id]	Editar producto
DELETE	/api/products/[id]	Eliminar producto
GET/POST/PUT/DELETE	/api/sales	CRUD para ventas
GET/POST/PUT/DELETE	/api/purchases	CRUD para compras
GET/POST/PUT/DELETE	/api/partners	CRUD para clientes/proveedores

7. Instalación local

Pasos para clonar y ejecutar el proyecto localmente:

1. Clonar el repositorio:

git clone https://github.com/calvin261/app_distruibuidora_tapia

2. Entrar al directorio:

cd app_distruibuidora_tapia

3. Instalar dependencias:

pnpm install

4. Ejecutar en desarrollo:

pnpm dev

Acceder vía http://localhost:3000

Nota: se recomienda Node.js ≥ 18 y PNPM ≥ 8

8. Variables de entorno

Actualmente no requiere base de datos ni autenticación externa, pero se preparó el archivo .env.local con la siguiente estructura:

NEXT_PUBLIC_APP_NAME=Distribuidora Tapia

(futuro) AUTH_SECRET=

(futuro) DATABASE_URL=

9. Scripts disponibles

Script	Función
pnpm dev	Ejecuta en modo desarrollo
pnpm build	Compila para producción
pnpm lint	Linter + prettier
pnpm test	Ejecuta pruebas unitarias
pnpm format	Formatea todo el código fuente

10. Seguridad y validaciones

- Validación en formularios: Zod (costos, campos requeridos).
- Control de stock negativo.
- Preventivo de inserción de scripts: sanitización en frontend.
- Middleware en rutas futuras: validación JWT (a implementar).

11. Despliegue y CI/CD

- El sistema se despliega automáticamente en Vercel:
- Cada push a la rama main activa un build.
- El sistema corre tests, lint y compila.
- Si pasa, se actualiza la versión en producción.
- También se generan URLs de previsualización para otras ramas.

Se puede hacer rollback manual desde el panel de control de Vercel.

12. Roadmap técnico

Mejora futura	Detalle
Persistencia real	Base de datos PostgreSQL + Prisma ORM
Login con roles	Auth0 / Firebase / Clerk
Exportación de reportes	PDF, CSV, emailing diario
Automatización alertas	WhatsApp Cloud API / correo
Gráficos en reportes	Recharts (ventas, rotación, tendencias)
E2E testing	Playwright / Cypress para pruebas funcionales
Dockerización	Contenedor para despliegue local u on-premise