🗂 ESTRUCTURA COMPLETA DE ARCHIVOS

Sistema de Inventario - Documentación Técnica

🌳 ÁRBOL DE DIRECTORIOS

```
Sistema de Invetario/
│
├── 📄 .env                                  # Variables de entorno (SECRETO - NO
subir a Git)
├── 📄 .gitignore                            # Archivos ignorados por Git
├── 📄 package.json                          # Dependencias del BACKEND
├── 📄 package-lock.json                     # Lock de versiones backend
├── 📄 README.md                             # Documentación principal
│
├── 🔧 SCRIPTS DE INICIO
│   ├── iniciar.ps1                        # Inicia backend + frontend
automáticamente
│   ├── iniciar-simple.ps1                 # Versión simplificada
│   ├── cargar-todo.ps1                    # Carga datos de prueba completos
│   └── cargar-ventas.ps1                  # Carga solo ventas de ejemplo
│
├── 🗄 ARCHIVOS DE BASE DE DATOS
│   ├── ESTRUCTURA_BD_COMPLETA.sql         # Crea las 8 tablas con relaciones
│   ├── DATOS_PRUEBA.sql                   # Inserta datos de ejemplo
│   ├── DATOS_VENTAS_MOVIMIENTOS.sql       # Datos de ventas e inventario
│   └── AGREGAR_CAMPOS_RESET_PASSWORD.sql # Migración para recuperar contraseña
│
├── 💻 BACKEND (Node.js + Express)
│   ├── 📄 index.js                        # ⭐ SERVIDOR PRINCIPAL - 1784 líneas
│   ├── 📄 db.js                           # Conexión a MySQL con pool
│   ├── 📄 emailService.js                 # Servicio de envío de emails
(Nodemailer)
│   ├── 📄 crearAdmin.js                   # Script para crear usuario administrador
│   └── 📁 backend/                        # Módulos adicionales (vacío por ahora)
│
├── 🎨 FRONTEND (React + Vite)
│   └── 📁 frontend-react/
│       ├── 📄 package.json                # Dependencias del FRONTEND
│       ├── 📄 vite.config.js              # Configuración de Vite
│       ├── 📄 index.html                  # HTML base (punto de entrada)
│       │
│       ├── 📁 src/                        # CÓDIGO FUENTE REACT
│       │   ├── 📄 main.jsx                # Entry point - Monta React en #root
│       │   ├── 📄 App.jsx                 # Componente raíz - Rutas principales
│       │   ├── 📄 styles.css              # Estilos globales (3500+ líneas)
│       │   ├── 📄 lightMode.css           # Estilos del tema claro
│       │   ├── 📄 lottieIcons.css         # Animaciones de iconos
│       │   ├── 📄 animatedIcons.css       # Animaciones CSS personalizadas
```

```
│   │   ├── searchInput.css      # Estilos del buscador
│   │   │
│   │   ├── pages/               # PÁGINAS PRINCIPALES
│   │   │   ├── Auth.jsx         # Login + Registro (tab switch)
│   │   │   ├── Login.jsx        # Login standalone
│   │   │   ├── Signup.jsx       # Registro standalone
│   │   │   ├── Dashboard.jsx    # ⭐ PANEL PRINCIPAL - 3741 líneas
│   │   │   ├── ForgotPassword.jsx  # Recuperar contraseña
│   │   │   └── ResetPassword.jsx   # Restablecer contraseña con token
│   │   │
│   │   ├── components/          # COMPONENTES REUTILIZABLES (15 archivos)
│   │   │   ├── ProtectedRoute.jsx         # Protege rutas con JWT
│   │   │   ├── ToastProvider.jsx          # Sistema de notificaciones
│   │   │   ├── ThemeSwitch.jsx            # Switch tema claro/oscuro
│   │   │   ├── LottieIcon.jsx             # Iconos SVG animados
│   │   │   ├── AnimatedIcons.jsx          # Wrapper de iconos
│   │   │   ├── SearchInput.jsx            # Buscador con animación
│   │   │   │
│   │   │   ├── 📋 MODALES
│   │   │   ├── ProductModal.jsx           # Crear/Editar productos
│   │   │   ├── CategoryModal.jsx          # Crear/Editar categorías
│   │   │   ├── SupplierModal.jsx          # Crear/Editar proveedores
│   │   │   ├── SaleModal.jsx              # Registrar ventas
│   │   │   ├── MovementModal.jsx          # Movimientos de inventario
│   │   │   ├── ChangePasswordModal.jsx    # Cambiar contraseña
│   │   │   ├── EditProfileModal.jsx       # Editar perfil de usuario
│   │   │   ├── UserRoleModal.jsx          # Gestionar usuarios (Admin)
│   │   │   └── ConfirmDialog.jsx          # Confirmación de acciones
│   │   │
│   │   ├── utils/               # UTILIDADES
│   │   │   ├── api.js           # Cliente HTTP con JWT automático
│   │   │   ├── company.js       # Datos de la empresa
│   │   │   └── export.js        # Exportar PDF y Excel
│   │   │
│   │   └── assets/              # RECURSOS ESTÁTICOS
│   │       └── lottie/          # Animaciones JSON Lottie
│   │           ├── dashboard.json
│   │           ├── products.json
│   │           └── sales.json
│   │
│   └── public/                  # ARCHIVOS PÚBLICOS
│       └── assets/
│           └── logo.png         # Logo de la empresa
│
├── public/                      # FRONTEND ANTIGUO (HTML/CSS/JS vanilla)
│   ├── index.html               # Login antiguo
│   ├── dashboard.html           # Dashboard antiguo
│   ├── style.css                # Estilos antiguos
│   └── app.js                   # JavaScript antiguo
│
├── diagramas/                   # DIAGRAMAS UML (20 imágenes PNG)
│   ├── Actores.png
│   ├── Casos_de_Uso.png
│   ├── Diagrama_Completo_Casos_de_Uso.png
```

```
│   ├── Diagrama_de_Clases.png
│   ├── Secuencia_Login.png
│   ├── Secuencia_Registrar_Venta.png
│   ├── Secuencia_Gestionar_Producto.png
│   ├── Secuencia_Movimiento_Inventario.png
│   ├── Secuencia_Recuperar_Contraseña.png
│   ├── Secuencia_Gestionar_Usuario.png
│   ├── Secuencia_Gestionar_Categoria.png
│   ├── Secuencia_Dashboard.png
│   ├── Colaboracion_Login.png
│   ├── Colaboracion_Registrar_Venta.png
│   ├── Colaboracion_Crear_Producto.png
│   ├── Colaboracion_Entrada_Inventario.png
│   ├── Colaboracion_Recuperar_Contraseña.png
│   ├── Colaboracion_Crear_Usuario.png
│   ├── Colaboracion_Crear_Categoria.png
│   └── Colaboracion_Dashboard.png
│
└── 📁 DOCUMENTACION/                    # 📄 ESTA DOCUMENTACIÓN
    ├── 01-INTRODUCCION-GENERAL.md
    ├── 02-ESTRUCTURA-ARCHIVOS.md       # ← Estás aquí
    ├── 03-API-ENDPOINTS.md
    ├── 04-COMPONENTES-FRONTEND.md
    ├── 05-GRAFICOS-REPORTES.md
    └── 06-ANIMACIONES.md
```

📄 DESCRIPCIÓN DETALLADA DE ARCHIVOS

🔧 BACKEND - Archivos Principales

1. index.js (⭐ ARCHIVO MÁS IMPORTANTE - 1784 líneas)

Propósito: Servidor HTTP principal que maneja TODAS las peticiones API

Estructura interna:

```
// LÍNEAS 1-50: Imports y configuración
import express from 'express'
import cors from 'cors'
import bcrypt from 'bcrypt'
import jwt from 'jsonwebtoken'
// ... más imports

// LÍNEAS 51-100: Middlewares de seguridad
app.use(helmet())        // Headers de seguridad
app.use(cors())          // Control de acceso
app.use(express.json())  // Parsear JSON

// LÍNEAS 101-150: Rate Limiters
const loginLimiter = rateLimit({
```

```
  windowMs: 15 * 60 * 1000,
  max: 5
})

// LÍNEAS 151-200: Middleware de autenticación JWT
function verificarToken(req, res, next) {
  // Verifica que el token sea válido
}

// LÍNEAS 201-400: ENDPOINTS DE USUARIOS
app.post('/api/usuarios/login', loginLimiter, ...)
app.post('/api/usuarios/registro', ...)
app.get('/api/usuarios/me', verificarToken, ...)
app.patch('/api/usuarios/logout', ...)
// + 14 endpoints más de usuarios

// LÍNEAS 401-600: ENDPOINTS DE PRODUCTOS
app.get('/api/productos', verificarToken, ...)
app.post('/api/productos', verificarToken, ...)
app.put('/api/productos/:id', verificarToken, ...)
// + 5 endpoints más

// LÍNEAS 601-800: ENDPOINTS DE CATEGORÍAS
app.get('/api/categorias', verificarToken, ...)
// + 4 endpoints más

// LÍNEAS 801-1000: ENDPOINTS DE PROVEEDORES
app.get('/api/proveedores', verificarToken, ...)
// + 4 endpoints más

// LÍNEAS 1001-1300: ENDPOINTS DE VENTAS
app.get('/api/ventas', verificarToken, ...)
app.post('/api/ventas', verificarToken, ...)
// + 5 endpoints más (con transacciones)

// LÍNEAS 1301-1500: ENDPOINTS DE MOVIMIENTOS
app.get('/api/movimientos', verificarToken, ...)
app.post('/api/movimientos', verificarToken, ...)
// + 3 endpoints más

// LÍNEAS 1501-1700: ENDPOINTS DE DASHBOARD
app.get('/api/dashboard/estadisticas', ...)
app.get('/api/dashboard/ventas-recientes', ...)
app.get('/api/dashboard/productos-bajo-stock', ...)
// + 5 endpoints más

// LÍNEAS 1701-1784: Inicio del servidor
app.listen(PORT, () => {
  console.log(`Servidor corriendo en puerto ${PORT}`)
})
```

Características clave:

- ☑ **58 endpoints REST** totales
- ☑ Validación con express-validator
- ☑ Transacciones de base de datos
- ☑ Manejo de errores con try-catch
- ☑ Respuestas JSON estandarizadas

---

2. db.js (10 líneas)

Propósito: Configuración de conexión a MySQL

```
import { createPool } from 'mysql2/promise'

export const pool = createPool({
  host: 'localhost',
  user: 'root',
  password: '',
  port: 3306,
  database: 'db_rectificadoraderepuesto',
  charset: 'utf8mb4'
})

console.log('Conectado a la base de datos')
```

¿Por qué pool?

- Reutiliza conexiones en lugar de crear una nueva cada vez
- Mejora el rendimiento
- Maneja múltiples requests simultáneos

---

3. emailService.js (280 líneas)

Propósito: Servicio de envío de emails con Nodemailer

Estructura:

```
// LÍNEAS 1-30: Configuración de Nodemailer
const transporter = nodemailer.createTransport({
  host: 'smtp.gmail.com',
  port: 587,
  auth: {
    user: process.env.EMAIL_USER,
    pass: process.env.EMAIL_PASSWORD
  }
})

// LÍNEAS 31-250: Plantilla HTML del email
function getPasswordResetTemplate(nombre, resetLink) {
```

```
  return `<!DOCTYPE html>...` // Email profesional con estilos
}

// LÍNEAS 251-280: Función de envío
export async function enviarEmailRecuperacion(email, nombre, token) {
  const resetLink = `${process.env.FRONTEND_URL}/restablecer-contrasena/${token}`
  await transporter.sendMail({...})
}
```

Plantilla incluye:

- 🎨 Header con logo y gradiente
- 📝 Mensaje personalizado
- ⭕ Botón de acción
- ⏱️ Advertencia de expiración (1 hora)
- 📊 Footer con datos de la empresa

---

4. crearAdmin.js (50 líneas)

Propósito: Script para crear el primer usuario administrador

```
// Ejecutar: node crearAdmin.js
// Crea usuario con:
// - Email: admin@rectificadora.com
// - Contraseña: admin123 (hasheada con bcrypt)
// - idRol: 1 (Administrador)
```

🎨 FRONTEND - Archivos Principales

1. main.jsx (14 líneas) - Entry Point

```
import React from 'react'
import { createRoot } from 'react-dom/client'
import { BrowserRouter } from 'react-router-dom'
import App from './App.jsx'
import './styles.css'

createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
)
```

¿Qué hace?

1. Importa React y estilos
2. Envuelve App en BrowserRouter (para navegación)
3. Monta todo en el div con id="root" del index.html

---

2. App.jsx (68 líneas) - Componente Raíz

```
export default function App() {
  const [theme, setTheme] = useState('dark')

  return (
    <ToastProvider>
      <div className="app-gradient-root">
        {/* Fondo con gradiente dinámico */}
        <AnimatePresence mode="wait">
          <Routes>
            <Route path="/" element={<Auth />} />
            <Route path="/dashboard" element={
              <ProtectedRoute>
                <Dashboard />
              </ProtectedRoute>
            } />
            {/* ... más rutas */}
          </Routes>
        </AnimatePresence>
      </div>
    </ToastProvider>
  )
}
```

Funciones:

- 🖼️ Define rutas principales
- 🎨 Maneja tema global (claro/oscuro)
- 🔒 Protege rutas con ProtectedRoute
- 🎭 Anima transiciones entre páginas

---

3. Dashboard.jsx (⭐ 3741 líneas) - Panel Principal

Estructura gigante:

```
export default function Dashboard() {
  // LÍNEAS 1-100: Estados
  const [activeView, setActiveView] = useState('dashboard')
  const [products, setProducts] = useState([])
  const [categories, setCategories] = useState([])
```

```
  // ... 50+ estados más

  // LÍNEAS 101-300: Effects y funciones de carga
  useEffect(() => {
    fetchUserData()
    fetchProducts()
    fetchCategories()
    // ...
  }, [])

  // LÍNEAS 301-500: Funciones CRUD
  const handleCreateProduct = async () => {...}
  const handleUpdateProduct = async () => {...}
  const handleDeleteProduct = async () => {...}
  // ... más funciones

  // LÍNEAS 501-700: Sidebar y Header
  const renderSidebar = () => {...}
  const renderHeader = () => {...}

  // LÍNEAS 701-1500: Vista Dashboard (estadísticas)
  const renderDashboardView = () => {
    return (
      <>
        {/* Cards de resumen */}
        {/* Gráficos con Recharts */}
        {/* Tablas de datos recientes */}
      </>
    )
  }

  // LÍNEAS 1501-2000: Vista Productos
  const renderProductsView = () => {...}

  // LÍNEAS 2001-2300: Vista Categorías
  const renderCategoriesView = () => {...}

  // LÍNEAS 2301-2600: Vista Proveedores
  const renderSuppliersView = () => {...}

  // LÍNEAS 2601-3000: Vista Ventas
  const renderSalesView = () => {...}

  // LÍNEAS 3001-3300: Vista Inventario
  const renderInventoryView = () => {...}

  // LÍNEAS 3301-3600: Vista Reportes
  const renderReportsView = () => {...}

  // LÍNEAS 3601-3741: Render principal
  return (
    <div className="dashboard-container">
      {renderSidebar()}
      <main className="dashboard-main">
```

```
        {renderHeader()}
        {activeView === 'dashboard' && renderDashboardView()}
        {activeView === 'products' && renderProductsView()}
        {/* ... más vistas */}
      </main>

      {/* Modales */}
      <ProductModal open={showProductModal} />
      <CategoryModal open={showCategoryModal} />
      {/* ... más modales */}
    </div>
  )
}
```

¿Por qué tan largo?

- 📊 7 vistas diferentes en un solo archivo
- 🎨 Renderizado condicional de todos los componentes
- 📋 Lógica completa de CRUD para cada entidad
- 📈 Gráficos con Recharts
- 🔄 Gestión de estado local para todo

---

4. styles.css (3500+ líneas) - Estilos Globales

Organización:

```
/* LÍNEAS 1-200: Variables CSS */
:root {
  --color-primary: #f97316;
  --color-bg: #0f0f0f;
  --spacing-sm: 0.5rem;
  /* ... 100+ variables */
}

/* LÍNEAS 201-500: Reset y Base */
* { box-sizing: border-box; }
body { font-family: 'Inter', sans-serif; }

/* LÍNEAS 501-1000: Layout */
.dashboard-container { display: flex; }
.sidebar { width: 260px; }

/* LÍNEAS 1001-1500: Componentes */
.button { padding: 12px 24px; }
.card { border-radius: 12px; }
.modal { position: fixed; }

/* LÍNEAS 1501-2000: Animaciones */
@keyframes fadeIn {...}
@keyframes slideIn {...}
```

```
@keyframes pulse {...}

/* LÍNEAS 2001-2500: Responsivo */
@media (max-width: 768px) {...}

/* LÍNEAS 2501-3000: Tema Oscuro */
.theme-dark {...}

/* LÍNEAS 3001-3500: Tema Claro */
.light-mode {...}
```

BASE DE DATOS - Archivos SQL

1. ESTRUCTURA_BD_COMPLETA.sql (200 líneas)

```
-- Crear base de datos
CREATE DATABASE IF NOT EXISTS db_rectificadoraderepuesto;
USE db_rectificadoraderepuesto;

-- Tabla 1: Rol
CREATE TABLE Rol (...);

-- Tabla 2: Usuario
CREATE TABLE Usuario (
  idUsuario INT PRIMARY KEY AUTO_INCREMENT,
  nombre VARCHAR(100),
  email VARCHAR(150) UNIQUE,
  contraseña VARCHAR(255),
  idRol INT,
  FOREIGN KEY (idRol) REFERENCES Rol(idRol)
);

-- ... 6 tablas más
```

2. DATOS_PRUEBA.sql (500 líneas)

```
-- Insertar roles
INSERT INTO Rol VALUES (1, 'Administrador'), (2, 'Vendedor');

-- Insertar usuarios
INSERT INTO Usuario VALUES (1, 'Admin', 'admin@...', '$2b$10$...', 1);

-- Insertar 20 categorías
INSERT INTO Categoria VALUES (...);

-- Insertar 50 productos
INSERT INTO Producto VALUES (...);
```

```sql
-- Insertar 10 proveedores
INSERT INTO Proveedor VALUES (...);
```

🔑 ARCHIVOS CRÍTICOS (NO SUBIR A GIT)

.env (Variables de Entorno)

```
# Seguridad
JWT_SECRET=clave_super_secreta_de_64_caracteres_minimo

# Base de Datos
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=
DB_NAME=db_rectificadoraderepuesto

# Email
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
EMAIL_USER=tu_email@gmail.com
EMAIL_PASSWORD=tu_password_de_aplicacion

# URLs
FRONTEND_URL=http://localhost:5174
BACKEND_URL=http://localhost:3000

# Configuración
PORT=3000
NODE_ENV=development
BCRYPT_ROUNDS=10
RATE_LIMIT_WINDOW_MS=900000
RATE_LIMIT_MAX_REQUESTS=5
```

⚠ **IMPORTANTE:** Este archivo contiene datos sensibles y NO debe compartirse

---

📦 package.json - Dependencias

Backend (raíz)

```
{
  "type": "module",
  "dependencies": {
    "bcrypt": "^6.0.0",              // Encriptación
    "cors": "^2.8.5",                // CORS
    "dotenv": "^17.2.3",             // Variables .env
    "express": "^5.1.0",             // Framework web
```

```
    "express-rate-limit": "^8.1.0",   // Rate limiting
    "express-validator": "^7.2.1",    // Validaciones
    "helmet": "^8.1.0",               // Seguridad headers
    "jsonwebtoken": "^9.0.2",         // JWT
    "mysql2": "^3.15.2",              // MySQL driver
    "nodemailer": "^7.0.9"            // Emails
  }
}
```

Frontend (frontend-react/)

```
{
  "dependencies": {
    "framer-motion": "^12.23.24",     // Animaciones
    "jspdf": "^3.0.3",                // PDF
    "jspdf-autotable": "^5.0.2",      // Tablas PDF
    "ldrs": "^1.1.7",                 // Loaders
    "lottie-react": "^2.4.1",         // Animaciones Lottie
    "react": "^18.3.1",               // React
    "react-dom": "^18.3.1",           // React DOM
    "react-router-dom": "^6.26.2",    // Routing
    "recharts": "^3.3.0",             // Gráficos
    "styled-components": "^6.1.19",   // CSS-in-JS
    "xlsx": "^0.18.5"                 // Excel
  },
  "devDependencies": {
    "@vitejs/plugin-react": "^4.3.1", // Plugin Vite
    "vite": "^5.4.8"                  // Build tool
  }
}
```

📊 TAMAÑO DE ARCHIVOS

Archivo	Líneas	Tamaño	Propósito
index.js	1784	~80 KB	Backend completo
Dashboard.jsx	3741	~150 KB	Panel principal
styles.css	3500+	~120 KB	Estilos globales
DATOS_PRUEBA.sql	500	~25 KB	Datos de ejemplo

Total del proyecto: ~50 MB (sin node_modules) **Con node_modules:** ~800 MB

Siguiente documento: 03-API-ENDPOINTS.md