📚 Guía Completa del Sistema de Inventario

Rectificadora de Repuestos - Tarapoto

Versión: 1.0
Última actualización: Diciembre 2024

📖 Índice

¿Qué es este Sistema?

Este es un **Sistema de Inventario** diseñado para una rectificadora de repuestos. Permite:

- 📦 **Gestionar productos**: Agregar, editar, ver y eliminar repuestos
- 🏷️ **Organizar por categorías**: Motores, Frenos, Suspensión, etc.
- 🚚 **Controlar proveedores**: Quién te vende los productos
- 💰 **Registrar ventas**: Llevar un control de todas las ventas
- 📊 **Ver reportes**: Estadísticas de ventas, productos más vendidos
- 👥 **Gestionar usuarios**: Controlar quién puede acceder al sistema

¿Cómo funciona en términos simples?

Imagina que el sistema es como un **restaurante**:

1. **La cocina (Backend/Servidor)**: Es donde se prepara todo. Recibe los pedidos, busca los ingredientes en la despensa (base de datos), y prepara la respuesta.

2. **El menú y las mesas (Frontend/Interfaz)**: Es lo que el cliente ve y con lo que interactúa. Botones bonitos, formularios, tablas con información.

3. **La despensa (Base de Datos)**: Donde se guarda toda la información. Los productos, los clientes, las ventas, etc.

Lenguajes de Programación Utilizados

🟨 JavaScript (Principal)

¿Qué es? El lenguaje de programación más popular para crear sitios web.

¿Dónde se usa en este proyecto?

- Todo el servidor (backend)
- Toda la interfaz de usuario (frontend)
- Las funciones lógicas del sistema

Ejemplo simple:

```
// Esto suma dos números
const sumar = (a, b) => a + b;
sumar(5, 3); // Resultado: 8
```

◇ SQL (Structured Query Language)

¿Qué es? Un lenguaje especial para hablar con bases de datos.

¿Dónde se usa?

- Para guardar información
- Para buscar productos, ventas, usuarios
- Para actualizar datos

Ejemplo simple:

```sql
-- Buscar todos los productos que cuestan menos de 100 soles
SELECT nombre, precio FROM producto WHERE precio < 100;
```

CSS (Cascading Style Sheets)

¿Qué es? El lenguaje que hace que las páginas web se vean bonitas.

¿Dónde se usa?

- Colores, tamaños, posiciones
- Animaciones y efectos visuales
- Diseño responsivo (que se vea bien en celular y computadora)

Ejemplo simple:

```css
/* Esto hace que un botón sea azul y redondeado */
.boton {
  background-color: blue;
  border-radius: 10px;
  color: white;
}
```

HTML (HyperText Markup Language)

¿Qué es? El lenguaje que estructura el contenido de las páginas web.

¿Dónde se usa?

- Definir botones, formularios, tablas
- Organizar el contenido de cada página

Ejemplo simple:

```html
<!-- Esto crea un botón -->
<button>Guardar Producto</button>
```

JSX (JavaScript + XML)

¿Qué es? Una extensión de JavaScript que permite escribir HTML dentro de JavaScript.

¿Dónde se usa?

- En todos los componentes de React
- Para crear la interfaz de usuario

Estructura del Proyecto

```
Sistema de Inventario/
│
├── index.js              ← El cerebro del servidor (backend)
├── db.js                 ← Conexión a la base de datos
├── package.json          ← Lista de dependencias del proyecto
│
├── database/             ← Scripts de la base de datos
│   ├── ESTRUCTURA_POSTGRESQL_v2.sql  ← Crea las tablas
│   └── DATOS_PRUEBA.sql     ← Datos de ejemplo
│
├── frontend-react/       ← La interfaz visual (lo que el usuario ve)
│   ├── src/
│   │   ├── App.jsx       ← Componente principal
│   │   ├── pages/        ← Las diferentes pantallas
│   │   │   ├── Login.jsx    ← Pantalla de inicio de sesión
│   │   │   ├── Dashboard.jsx← Panel principal
│   │   │   └── ...
│   │   ├── components/   ← Piezas reutilizables
│   │   │   ├── ProductModal.jsx    ← Formulario de productos
│   │   │   ├── SaleModal.jsx       ← Formulario de ventas
│   │   │   └── ...
│   │   ├── utils/        ← Funciones de ayuda
│   │   └── styles/       ← Estilos CSS
│   │
│   └── index.html        ← Página principal
│
├── services/             ← Servicios especiales
│   ├── reportService.js     ← Generación de reportes
│   ├── cacheService.js      ← Velocidad con caché
│   └── cronService.js       ← Tareas automáticas
│
└── DOCUMENTACION/        ← Esta documentación
```

La Base de Datos

¿Qué es una Base de Datos?

Imagina un **archivero gigante** con muchos cajones organizados. Cada cajón tiene carpetas, y cada carpeta tiene información específica. Así funciona una base de datos.

Motor de Base de Datos: PostgreSQL

Este proyecto usa **PostgreSQL**, una base de datos muy potente y gratuita. Es como tener un archivero súper inteligente que puede:

- Ordenar archivos automáticamente
- Buscar información en segundos
- Mantener todo organizado y seguro

Las Tablas del Sistema

Una **tabla** es como una hoja de Excel: tiene columnas y filas.

1 Tabla: `rol`

Propósito: Define los tipos de usuarios en el sistema.

Campo	Tipo	Descripción
idrol	Número	Identificador único
nombrerol	Texto	Nombre del rol (ej: "Administrador")
descripcion	Texto	Descripción del rol

Roles disponibles:

- **Administrador**: Puede hacer TODO en el sistema
- **Vendedor**: Solo puede vender y ver productos

---

2 Tabla: `usuario`

Propósito: Guarda la información de las personas que usan el sistema.

Campo	Tipo	Descripción
idusuario	Número	Identificador único
nombre	Texto	Nombre de usuario para ingresar
nombrecompleto	Texto	Nombre completo de la persona
contraseña	Texto	Contraseña encriptada
email	Texto	Correo electrónico
telefono	Texto	Número de teléfono
fotoperfil	Texto	Enlace a la foto de perfil
estado	Número	1 = Activo, 0 = Inactivo
idrol	Número	Qué rol tiene (Administrador o Vendedor)

---

3 Tabla: `categoria`

Propósito: Organiza los productos en grupos.

Campo	Tipo	Descripción
idcategoria	Número	Identificador único
nombre	Texto	Nombre de la categoría
descripcion	Texto	Descripción de qué incluye
codigoprefix	Texto	Prefijo para códigos (ej: "MOT" para Motores)
estado	Número	1 = Activa, 0 = Inactiva

Categorías de ejemplo:

- 🔧 **MOT** - Motores
- 🔘 **FRE** - Frenos
- 🔩 **SUS** - Suspensión
- ⚡ **ELE** - Eléctricos

---

4 Tabla: `proveedor`

Propósito: Guarda información de quiénes te venden los productos.

Campo	Tipo	Descripción
idproveedor	Número	Identificador único
nombrecontacto	Texto	Nombre de la persona de contacto
email	Texto	Correo electrónico
telefono	Texto	Número de teléfono
direccion	Texto	Dirección física
ruc	Texto	Número de RUC
estado	Número	1 = Activo, 0 = Inactivo

---

5 Tabla: `producto`

Propósito: El corazón del inventario. Guarda todos los repuestos.

Campo	Tipo	Descripción
idproducto	Número	Identificador único
codigo	Texto	Código del producto (ej: MOT-001)
nombre	Texto	Nombre del producto
descripcion	Texto	Descripción detallada
imagen	Texto	Enlace a la imagen del producto
marca	Texto	Marca del producto
modelocompatible	Texto	Con qué modelos funciona
ubicacion	Texto	Dónde está en el almacén
preciocompra	Decimal	Cuánto te costó comprarlo
precioventa	Decimal	A cuánto lo vendes
stockactual	Número	Cuántas unidades tienes

stockminimo	Número	Cantidad mínima antes de alertar
idcategoria	Número	A qué categoría pertenece
idproveedor	Número	De qué proveedor viene

6️⃣ Tabla: `venta`

Propósito: Registra cada venta realizada.

Campo	Tipo	Descripción
idventa	Número	Identificador único
numeroventa	Texto	Número de venta (ej: V-2024-0001)
clientenombre	Texto	Nombre del cliente
clientedocumento	Texto	DNI del cliente
metodopago	Texto	Efectivo, tarjeta, Yape, etc.
montototal	Decimal	Total de la venta
fechahora	Fecha/Hora	Cuándo se hizo la venta
idusuario	Número	Quién hizo la venta

7️⃣ Tabla: `detalleventa`

Propósito: Guarda qué productos se vendieron en cada venta.

Campo	Tipo	Descripción
iddetalleventa	Número	Identificador único
idventa	Número	A qué venta pertenece
idproducto	Número	Qué producto se vendió
cantidad	Número	Cuántas unidades
precioventaunitario	Decimal	Precio de cada unidad
subtotal	Decimal	cantidad × precio

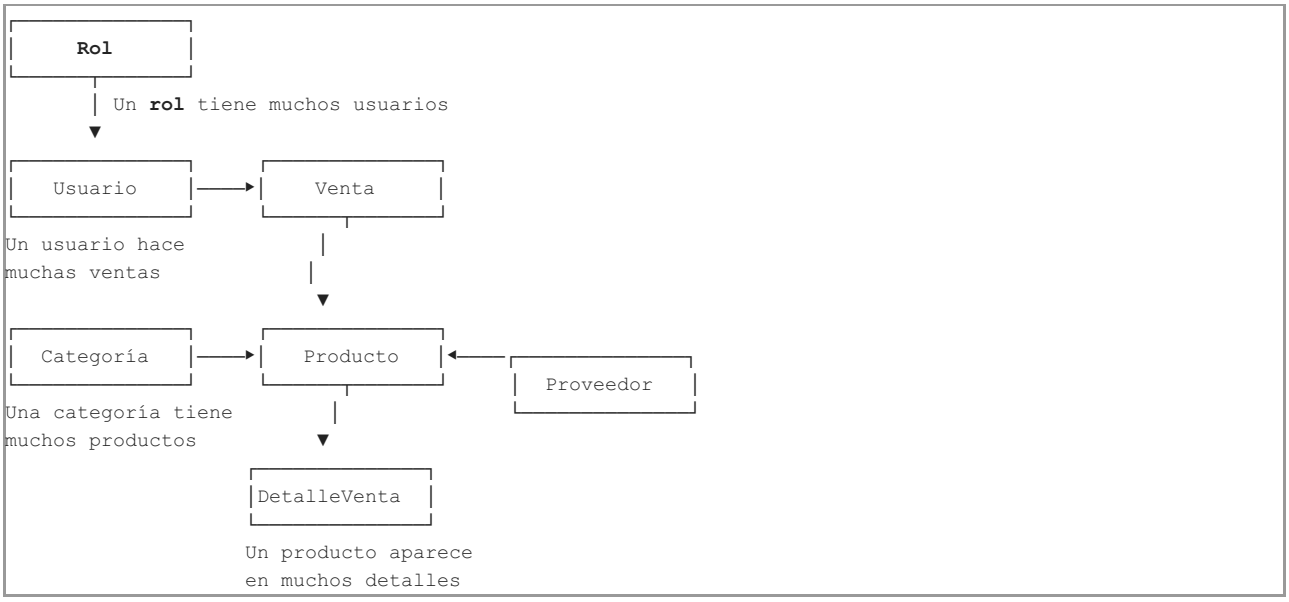
8️⃣ Tabla: `movimientoinventario`

Propósito: Registra entradas y salidas de productos.

Campo	Tipo	Descripción
idmovimientoinventario	Número	Identificador único
tipomovimiento	Texto	"entrada" o "salida"
cantidad	Número	Cuántas unidades
observaciones	Texto	Notas adicionales
idproducto	Número	Qué producto se movió
idusuario	Número	Quién registró el movimiento
fechahora	Fecha/Hora	Cuándo ocurrió

🔗 Relaciones Entre Tablas

Las tablas están **conectadas** entre sí:



El Backend (Servidor)

¿Qué es el Backend?

El backend es como la **cocina de un restaurante**: el cliente no la ve, pero es donde se prepara todo. Recibe pedidos, procesa información y devuelve respuestas.

Tecnologías Usadas

Tecnología	¿Para qué sirve?
Node.js	Ejecuta JavaScript en el servidor
Express.js	Crea rutas y maneja peticiones
PostgreSQL	Base de datos
JWT	Tokens de seguridad
bcrypt	Encripta contraseñas

Archivos Principales del Backend

`index.js` - El Cerebro del Sistema

Este archivo tiene **2,174 líneas** de código y contiene:

1. **Configuración inicial**: Carga las herramientas necesarias
2. **Middlewares de seguridad**: Protegen el sistema
3. **Endpoints/Rutas**: Los "menús" que el frontend puede pedir

```
// Ejemplo simplificado de cómo funciona una ruta
app.get("/api/productos", (req, res) => {
  // 1. Alguien pide la lista de productos
  // 2. Busco en la base de datos
  // 3. Devuelvo la lista
});
```

`db.js` - La Conexión a la Base de Datos

Este archivo hace la **conexión** con PostgreSQL.

¿Qué hace?

1. Lee la configuración (usuario, contraseña, servidor)
2. Crea una conexión permanente
3. Convierte los nombres de columnas a un formato compatible

```
// Conexión simplificada
const pool = new Pool({
  host: "servidor",
  user: "usuario",
  password: "contraseña",
  database: "inventario",
});
```

Las Rutas del API

Una **API** es como un menú de restaurante: lista las opciones disponibles.

Rutas de Autenticación

Método	Ruta	¿Qué hace?
POST	`/api/login`	Iniciar sesión
POST	`/api/registro`	Registrar nuevo usuario
POST	`/api/recuperar-password`	Solicitar recuperación de contraseña
POST	`/api/reset-password`	Cambiar contraseña

Rutas de Categorías

Método	Ruta	¿Qué hace?
GET	`/api/categorias`	Listar todas las categorías
GET	`/api/categorias/:id`	Ver una categoría específica
POST	`/api/categorias`	Crear nueva categoría
PUT	`/api/categorias/:id`	Editar categoría
DELETE	`/api/categorias/:id`	Eliminar categoría

Rutas de Proveedores

Método	Ruta	¿Qué hace?
GET	`/api/proveedores`	Listar todos los proveedores
GET	`/api/proveedores/:id`	Ver un proveedor específico
POST	`/api/proveedores`	Crear nuevo proveedor
PUT	`/api/proveedores/:id`	Editar proveedor
DELETE	`/api/proveedores/:id`	Eliminar proveedor

Rutas de Productos

Método	Ruta	¿Qué hace?
GET	`/api/productos`	Listar todos los productos
GET	`/api/productos/:id`	Ver un producto específico
GET	`/api/productos/buscar`	Buscar productos por criterios
POST	`/api/productos`	Crear nuevo producto
PUT	`/api/productos/:id`	Editar producto
DELETE	`/api/productos/:id`	Desactivar producto

Rutas de Ventas

Método	Ruta	¿Qué hace?
GET	`/api/ventas`	Listar todas las ventas
GET	`/api/ventas/:id`	Ver detalles de una venta
POST	`/api/ventas`	Registrar nueva venta
DELETE	`/api/ventas/:id`	Anular venta (devuelve stock)

Rutas de Inventario

Método	Ruta	¿Qué hace?
GET	`/api/movimientos`	Listar movimientos de inventario

POST	`/api/movimientos/entrada`	Registrar entrada de productos
POST	`/api/movimientos/salida`	Registrar salida de productos

Rutas de Usuarios

Método	Ruta	¿Qué hace?
GET	`/api/usuarios`	Listar usuarios (solo admin)
PUT	`/api/usuarios/:id`	Editar usuario
PUT	`/api/usuarios/:id/cambiar-rol`	Cambiar rol de usuario
DELETE	`/api/usuarios/:id`	Desactivar usuario

Rutas de Reportes

Método	Ruta	¿Qué hace?
GET	`/api/reportes/estadisticas`	Estadísticas generales
GET	`/api/reportes/ventas`	Reporte de ventas por fechas
GET	`/api/reportes/stock-bajo`	Productos con stock bajo
GET	`/api/reportes/top-productos`	Productos más vendidos

El Frontend (Interfaz)

¿Qué es el Frontend?

El frontend es **todo lo que el usuario ve y toca**: botones, formularios, tablas, menús, colores. Es la "cara" del sistema.

Tecnologías Usadas

Tecnología	¿Para qué sirve?
React	Librería para crear interfaces interactivas
Vite	Herramienta para desarrollo rápido
Framer Motion	Animaciones suaves
Chart.js	Gráficos y estadísticas

Estructura del Frontend

```
frontend-react/src/
├── pages/          ← Las pantallas principales
├── components/     ← Piezas reutilizables
├── utils/          ← Funciones de ayuda
├── styles/         ← Estilos CSS
└── App.jsx         ← Componente raíz
```

Las Páginas del Sistema

1. `Login.jsx` - Inicio de Sesión

¿Qué hace?

- Muestra campos para email y contraseña
- Valida que los datos sean correctos
- Redirige al Dashboard si las credenciales son válidas

Características:

- ☑ Opción "Recordarme"
- ☑ Enlace para recuperar contraseña
- ☑ Protección contra muchos intentos fallidos

---

2. `Dashboard.jsx` - Panel Principal

¿Qué hace?

- Es el centro de todo el sistema
- Contiene TODAS las vistas internas:
    - Vista de inicio (estadísticas)
    - Vista de productos
    - Vista de categorías
    - Vista de proveedores
    - Vista de ventas
    - Vista de inventario
    - Vista de reportes
    - Vista de usuarios (solo admin)
    - Vista de configuración

Es el archivo más grande del proyecto (3,839 líneas) porque contiene múltiples "sub-páginas".

---

3. `ForgotPassword.jsx` - Recuperar Contraseña

¿Qué hace?

- Permite solicitar un enlace de recuperación
- Envía un email con instrucciones

---

4. `ResetPassword.jsx` - Restablecer Contraseña

¿Qué hace?

- Permite crear una nueva contraseña
- Valida que las contraseñas coincidan

🧩 Los Componentes Reutilizables

Los componentes son **piezas de LEGO** que se pueden usar en diferentes partes del sistema.

`ProductModal.jsx` - Formulario de Productos

¿Para qué sirve? Agregar o editar productos.

Campos del formulario:

- Código del producto
- Nombre
- Descripción
- Categoría (selector)
- Proveedor (selector)
- Marca
- Modelo compatible
- Ubicación en almacén
- Precio de compra
- Precio de venta
- Stock actual
- Stock mínimo
- Imagen del producto

`SaleModal.jsx` - Formulario de Ventas

¿Para qué sirve? Registrar nuevas ventas.

Características:

- Buscar productos por nombre o código
- Agregar múltiples productos al carrito
- Calcular totales automáticamente
- Seleccionar método de pago
- Generar número de venta automático

`CategoryModal.jsx` - Formulario de Categorías

¿Para qué sirve? Agregar o editar categorías.

Campos:

- Nombre de la categoría
- Descripción
- Prefijo de código (ej: MOT, FRE)
- Estado (activo/inactivo)

`SupplierModal.jsx` - Formulario de Proveedores

¿Para qué sirve? Agregar o editar proveedores.

Campos:

- Nombre del contacto
- Email
- Teléfono
- Dirección
- RUC

`MovementModal.jsx` - Movimientos de Inventario

¿Para qué sirve? Registrar entradas o salidas de productos.

Campos:

- Tipo de movimiento (entrada/salida)
- Producto
- Cantidad
- Precio unitario (para entradas)
- Observaciones

`SearchInput.jsx` - Barra de Búsqueda

¿Para qué sirve? Buscar en cualquier lista.

Características:

- Búsqueda en tiempo real
- Ícono de búsqueda animado
- Botón para limpiar

`ThemeSwitch.jsx` - Cambiar Tema

¿Para qué sirve? Alternar entre modo claro y oscuro.

🔧 Utilidades (`utils/`)

`api.js` - Comunicación con el Servidor

¿Qué hace?

- Envía peticiones al backend
- Agrega automáticamente el token de seguridad
- Maneja errores de sesión expirada

```
// Ejemplo de uso
const productos = await api.get("/api/productos");
```

`export.js` - Exportar Reportes

¿Qué hace?

- Genera archivos PDF
- Genera archivos Excel

- Para descargar reportes

Funcionalidades del Sistema

🏠 Vista de Inicio (Dashboard)

¿Qué muestra?

- 📊 Tarjetas con estadísticas:
  - Total de productos
  - Total de categorías
  - Total de proveedores
  - Ventas del mes
- 📈 Gráfico de ventas recientes
- ⚠️ Alertas de stock bajo
- 🔝 Productos más vendidos

📦 Vista de Productos

Funcionalidades:

Función	Descripción
Listar	Ver todos los productos en tarjetas
Buscar	Filtrar por nombre, código o categoría
Agregar	Crear nuevo producto
Editar	Modificar información existente
Eliminar	Desactivar producto (no se borra, se oculta)
Ver detalle	Información completa del producto

🏷️ Vista de Categorías

Funcionalidades:

Función	Descripción
Listar	Ver todas las categorías
Agregar	Crear nueva categoría
Editar	Modificar nombre, descripción, prefijo
Eliminar	Desactivar categoría
Expandir	Ver productos de cada categoría

🚚 Vista de Proveedores

Funcionalidades:

Función	Descripción
Listar	Ver todos los proveedores
Agregar	Registrar nuevo proveedor
Editar	Actualizar información
Eliminar	Desactivar proveedor
Contactar	Ver email y teléfono

💰 Vista de Ventas

Funcionalidades:

Función	Descripción
Listar	Ver historial de ventas
Nueva venta	Registrar una venta
Ver detalle	Qué productos se vendieron
Buscar	Filtrar por fecha, cliente, etc.
Anular	Cancelar venta (devuelve stock)

Proceso de una venta:

1. Click en "Nueva Venta"
2. Buscar y agregar productos
3. Ingresar cantidad de cada uno
4. Escribir nombre del cliente
5. Seleccionar método de pago
6. Confirmar venta
7. El sistema automáticamente:
    - Descuenta el stock
    - Genera número de venta
    - Registra en el historial

Vista de Inventario

Funcionalidades:

Función	Descripción
Historial	Ver todos los movimientos
Entrada	Registrar llegada de productos
Salida	Registrar salida (no venta)
Filtrar	Por fecha, tipo, producto
Exportar	Descargar en Excel o PDF

Vista de Reportes

Reportes disponibles:

Reporte	Descripción
Ventas generales	Total vendido por período
Por categoría	Ventas agrupadas por categoría
Stock bajo	Productos que necesitan reposición
Top productos	Los más vendidos
Por vendedor	Ventas de cada usuario

Opciones de exportación:

- PDF (para imprimir)
- Excel (para analizar datos)

Vista de Usuarios (Solo Administradores)

Funcionalidades:

Función	Descripción
Listar	Ver todos los usuarios
Cambiar rol	Administrador ↔ Vendedor
Activar/Desactivar	Bloquear acceso
Ver actividad	Último inicio de sesión

Vista de Configuración

Opciones:

- Editar perfil (nombre, foto, teléfono)
- Cambiar contraseña
- Cambiar tema (claro/oscuro)
- Configurar notificaciones

Operaciones CRUD Explicadas

¿Qué es CRUD?

CRUD son las 4 operaciones básicas que se hacen con datos:

Letra	Significado	Acción	Ejemplo
C	Create	Crear	Agregar nuevo producto
R	Read	Leer	Ver lista de productos
U	Update	Actualizar	Editar precio de producto
D	Delete	Eliminar	Quitar producto del sistema

CRUD de Productos - Paso a Paso

CREATE - Crear Producto

En el Frontend (lo que tú ves):

1. Click en botón "Agregar Producto"
2. Se abre el formulario ProductModal
3. Llenas todos los campos
4. Click en "Guardar"

En el Backend (lo que pasa detrás):

```
// 1. Frontend envía los datos
POST /api/productos
{
  "codigo": "MOT-015",
  "nombre": "Pistón 200cc",
  "precio": 85.00,
  // ... más campos
}

// 2. Backend valida los datos
// 3. Guarda en la base de datos
INSERT INTO producto (codigo, nombre, precio...) VALUES (...)

// 4. Devuelve confirmación
{ "message": "Producto creado exitosamente", "id": 15 }
```

READ - Leer/Listar Productos

En el Frontend:

- Al entrar a la vista de productos, automáticamente carga la lista

En el Backend:

```
// 1. Frontend pide la lista
GET /api/productos

// 2. Backend busca en la base de datos
SELECT * FROM producto WHERE estado = 1

// 3. Devuelve los productos
{
  "productos": [
    { "id": 1, "nombre": "Pistón STD", "precio": 75.00 },
    { "id": 2, "nombre": "Anillos 150cc", "precio": 35.00 },
    // ...
  ]
}
```

UPDATE - Actualizar Producto

En el Frontend:

1. Click en el ícono de lápiz (editar)
2. Se abre el formulario con los datos actuales
3. Modificas lo que necesitas

4. Click en "Guardar"

En el Backend:

```
// 1. Frontend envía los cambios
PUT /api/productos/15
{
  "precio": 90.00  // precio nuevo
}

// 2. Backend actualiza en la base de datos
UPDATE producto SET precio = 90.00 WHERE idProducto = 15

// 3. Devuelve confirmación
{ "message": "Producto actualizado exitosamente" }
```

DELETE - Eliminar Producto

Importante: En este sistema, "eliminar" NO borra el producto. Solo lo DESACTIVA (para mantener el historial).

En el Frontend:

1. Click en el ícono de basura
2. Confirmas en el diálogo
3. El producto desaparece de la lista

En el Backend:

```
// 1. Frontend pide eliminación
DELETE /api/productos/15

// 2. Backend desactiva (no borra)
UPDATE producto SET estado = 0 WHERE idProducto = 15

// 3. Devuelve confirmación
{ "message": "Producto desactivado exitosamente" }
```

El mismo patrón CRUD aplica para:

- ☑ Categorías
- ☑ Proveedores
- ☑ Usuarios
- ☑ Ventas (con lógica adicional de stock)

Seguridad del Sistema

Autenticación con JWT

JWT (JSON Web Token) es como un pase de entrada a un concierto:

1. Muestras tu entrada (email + contraseña)
2. Te dan un brazalete (token)
3. Con el brazalete puedes entrar a todas las áreas

```
// El token se guarda en el navegador
localStorage.setItem('token', 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...')

// Se envía en cada petición
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

Encriptación de Contraseñas

Las contraseñas NUNCA se guardan tal cual. Se "encriptan" con **bcrypt**:

```
Contraseña original:    "admin123"
Contraseña encriptada:  "$2b$10$w9LpIw6WXgeISNY4cLJOcOUaYbZ5qV0D72g1L53ASlHaNWsFx5zSW"
```

Ni siquiera el administrador puede ver las contraseñas originales.

Rate Limiting

Protección contra ataques de "fuerza bruta" (intentar muchas contraseñas):

- **Límite de login:** 5 intentos en 15 minutos
- **Límite general:** 100 peticiones por 15 minutos

Si superas el límite, recibes el mensaje: "Demasiados intentos, espera 15 minutos"

Control de Roles

No todos pueden hacer todo:

Acción	Administrador	Vendedor
Ver productos	☑	☑
Crear productos	☑	❌
Editar productos	☑	❌
Hacer ventas	☑	☑
Ver usuarios	☑	❌
Ver reportes	☑	❌

Servicios Adicionales

`reportService.js` - Generador de Reportes

¿Qué hace?

- Genera reportes automáticos diarios
- Detecta productos con stock bajo
- Envía alertas por email

Funciones principales:

Función	Descripción
`generateDailyReport()`	Resumen de ventas del día anterior
`checkLowStock()`	Lista productos bajo mínimo
`sendDailyReportEmail()`	Envía reporte por correo
`sendLowStockAlert()`	Alerta de stock bajo

`cacheService.js` - Velocidad con Caché

¿Qué es caché?
Es como tener una nota adhesiva con respuestas rápidas. En vez de buscar siempre en la base de datos, guarda las respuestas frecuentes.

Usa Redis (una base de datos súper rápida para datos temporales).

Beneficios:

- Respuestas más rápidas
- Menos carga en la base de datos
- Ahorro de recursos

`cronService.js` - Tareas Automáticas

¿Qué hace?
Ejecuta tareas en horarios específicos, como un despertador.

Tareas programadas:

Tarea	Horario	Descripción
Reporte diario	6:00 AM	Genera resumen de ventas
Alerta de stock	8:00 AM	Verifica productos bajos

Glosario de Términos

Términos de Programación

Término	Significado Simple
API	Conjunto de "menús" que el servidor ofrece
Backend	La parte del sistema que no se ve (el servidor)
Frontend	La parte visual que el usuario ve
Base de datos	Donde se guarda toda la información
CRUD	Crear, Leer, Actualizar, Eliminar
Endpoint	Una URL específica del servidor
Token	Un "pase" digital que identifica al usuario
Caché	Memoria rápida para datos frecuentes
Componente	Pieza reutilizable de la interfaz

Términos del Sistema

Término	Significado
Stock	Cantidad disponible de un producto
Stock mínimo	Cantidad antes de necesitar reabastecer
Movimiento	Entrada o salida de productos
Proveedor	Empresa que te vende productos
Código de producto	Identificador único (ej: MOT-001)
Número de venta	Identificador de cada venta (ej: V-2024-0001)

Soporte

Si tienes dudas sobre el sistema, contacta al desarrollador o revisa la documentación técnica en la carpeta `DOCUMENTACION/`.
