ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

INGENIERÍA EN SOFTWARE

APLICACIONES INFORMÁTICAS II

ESTUDIANTE: SCARLET CAYAPA – 7166

CURSO: OCTAVO

TEMA: REINGENIERIA INVERSA SISTEMA DE GESTIÓN DE INVENTARIO – CLASE DE RECUPERACIÓN

FECHA DE ENTREGA: 27/01/2026

DOCENTE: ING. JULIO SANTILLÁN

SEPTIEMBRE - FEBRERO 2026

ACTIVIDAD

1. Buscar un sistema de gestión de inventario.

Proyecto encontrado en GitHub en el siguiente enlace: https://github.com/Shahed-Chy-Suzan/Inventory-Management

2. Comprender el sistema, identificar:

- Arquitectura de la aplicación

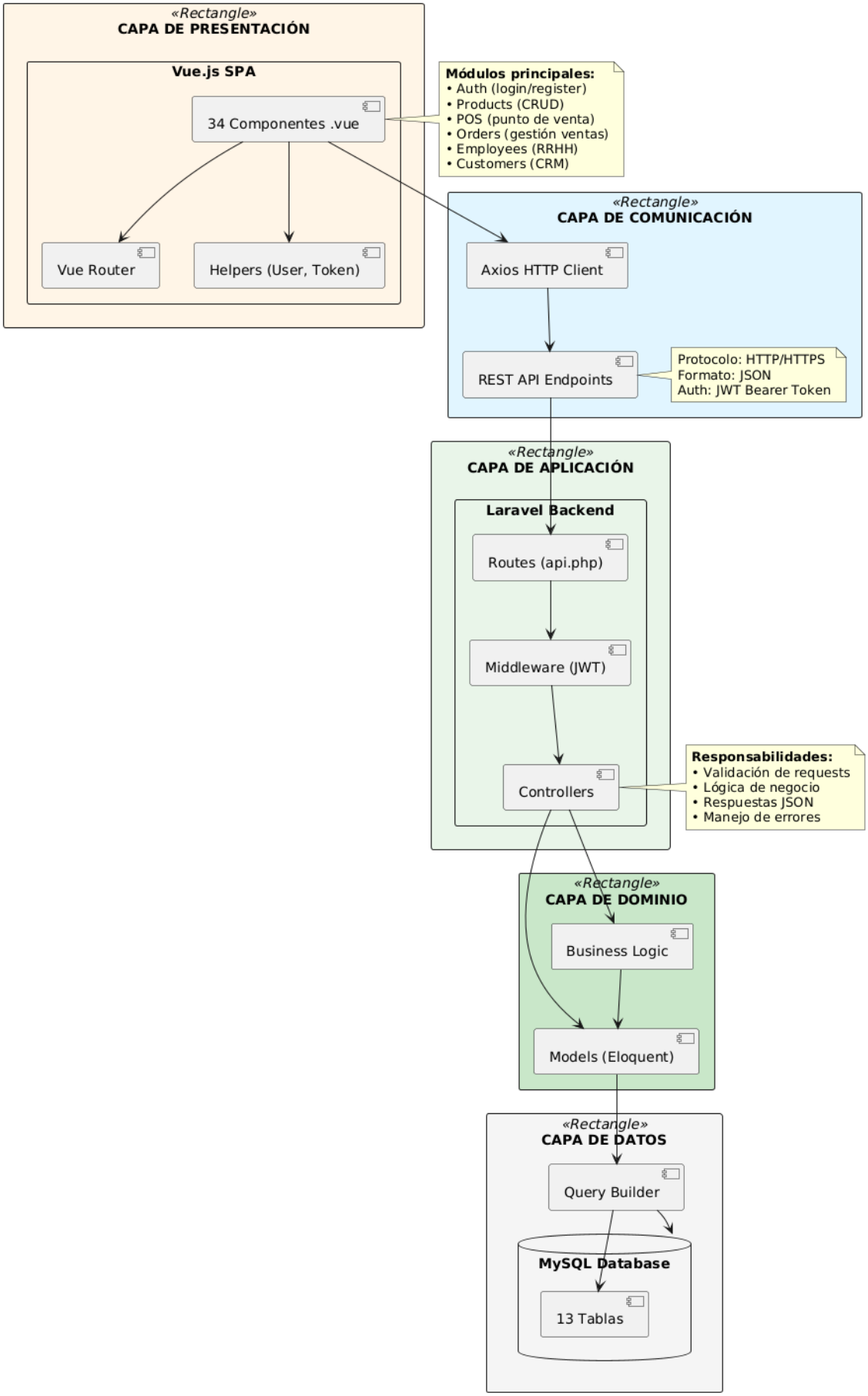
Arquitectura Monolítica MVC + SPA

Del diagrama completo:

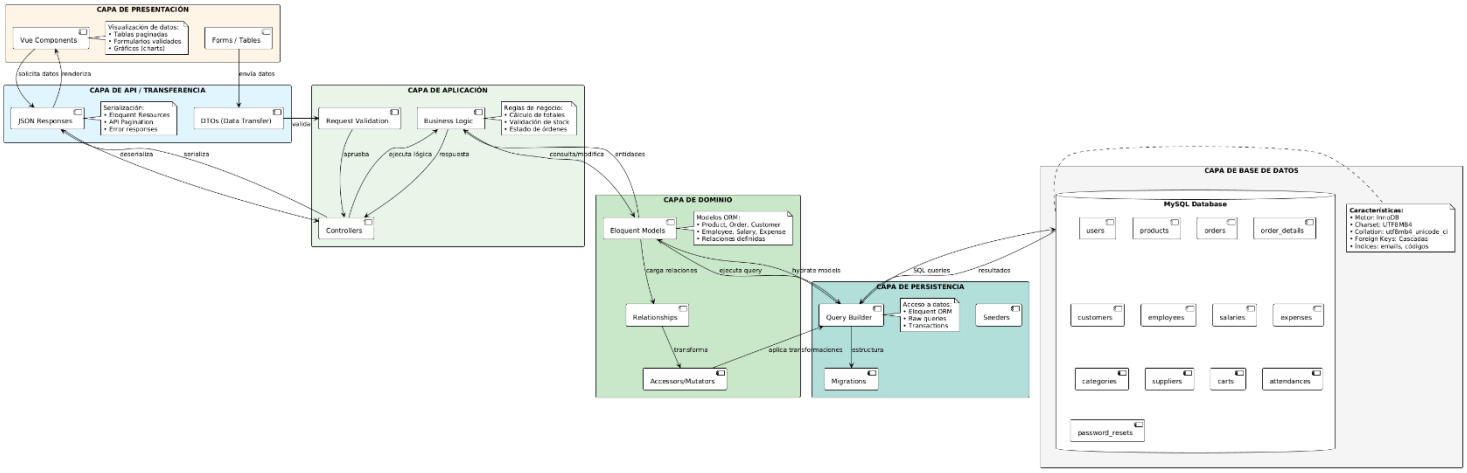
- Backend y Frontend están separados lógicamente, pero desplegados como un sistema unificado
- Laravel Backend contiene toda la lógica (Capa de Aplicación + Dominio + Datos)
- Vue.js Frontend consume la API del mismo backend

El sistema utiliza una combinación de dos patrones arquitectónicos:

Capa	**Patrón arquitectónico**	**Tecnología**
Backend	MVC (Model-View-Controller) + REST API	Laravel 5.8
Frontend	SPA Component-Based Architecture	Vue.js 2.x
Comunicación	RESTful API sobre HTTP	JSON + JWT


«Rectangle»
CAPA DE PRESENTACIÓN
Vue.js SPA
34 Componentes .vue
Vue Router
Helpers (User, Token)
Módulos principales:
• Auth (login/register)
• Products (CRUD)
• POS (punto de venta)
• Orders (gestión ventas)
• Employees (RRHH)
• Customers (CRM)
«Rectangle»
CAPA DE COMUNICACIÓN
Axios HTTP Client
REST API Endpoints
Protocolo: HTTP/HTTPS
Formato: JSON
Auth: JWT Bearer Token
«Rectangle»
CAPA DE APLICACIÓN
Laravel Backend
Routes (api.php)
Middleware (JWT)
Controllers
Responsabilidades:
• Validación de requests
• Lógica de negocio
• Respuestas JSON
• Manejo de errores
«Rectangle»
CAPA DE DOMINIO
Business Logic
Models (Eloquent)
«Rectangle»
CAPA DE DATOS
Query Builder
MySQL Database
13 Tablas

- Arquitectura de los datos



Capa de base de datos (Database Layer)

Esta es la capa más baja, donde residen los datos persistentes.

MySQL Database contiene múltiples tablas organizadas en grupos:

- Usuarios y Gestión: users, employees, password_resets
- Productos e Inventario: products, categories, suppliers
- Operaciones Comerciales: orders, order_details, customers, expenses, salaries
- Sistema de Asistencia: carts, attendance

Características técnicas: Primary keys, Foreign keys (relaciones entre tablas), índices para optimización, y tablas con nombres en plural siguiendo convenciones REST.

Capa de persistencia (Persistence Layer)

Esta capa maneja el acceso directo a la base de datos mediante:

- Query Builder: Construcción dinámica de consultas SQL
- Seeders: Datos iniciales para poblar la base de datos
- Migrations: Control de versiones del esquema de base de datos (aplica y revierte cambios en la estructura)

Función: Abstrae la comunicación con la BD, ejecuta consultas SQL y realiza transacciones, manteniendo separada la lógica de acceso a datos.

Capa de dominio (Domain Layer)

El corazón de la lógica de negocio, dividida en:

Eloquent Models: Representaciones ORM de las tablas

- Product, Order, Customer, User (modelos principales)

- Incluyen relaciones definidas entre entidades

Relationships: Define asociaciones entre modelos

- hasMany, belongsTo, hasOne, etc.
- Mapea las foreign keys a relaciones de objetos

Accessors/Mutators: Modifican cómo se accede y almacenan los datos

- Transformaciones automáticas de datos
- Formateo de valores al leer/escribir

Función: Contiene las reglas de negocio puras, independientes de frameworks o infraestructura.

Capa de aplicación (Application Layer)

Orquesta los casos de uso del sistema:

Business Logic:

- Reglas de negocio específicas
- Cálculo de totales
- Estado de órdenes

Request Validation: Valida entradas del usuario antes de procesarlas

DTOs (Data Transfer Objects): Objetos para transferir datos entre capas, serializando/deserializando información y encapsulando datos.

Controllers: Reciben requests HTTP, invocan lógica de negocio, y devuelven respuestas apropiadas.

Función: Coordina el flujo de la aplicación sin contener lógica de negocio compleja.

Capa de API / transferencia (API/Transfer Layer)

Maneja la comunicación externa:

JSON Responses:

- Estructura de respuestas API
- Códigos HTTP
- Error handling

DTOs (Data Transfer Objects): Transforman datos del dominio a formato JSON para APIs RESTful.

Función: Serializa/deserializa datos, gestiona formatos de intercambio, y proporciona interfaz de comunicación con el frontend.

Capa de presentación (Presentation Layer)

La interfaz con el usuario final:

Vue Components: Componentes reutilizables de interfaz, manipulación de datos frontend, y lógica de presentación.

Forms/Tables: Elementos de UI para visualización y captura de datos, con validaciones frontend.

Función: Renderiza la interfaz, captura datos del usuario, y envía datos al backend para procesamiento.

-Roles de usuarios

El sistema NO implementa un sistema de roles. Todos los usuarios registrados tienen acceso completo a todas las funcionalidades.

Autenticación Actual

Login: POST /api/auth/login - Entrada con email/password

Signup: POST /api/auth/signup - Registro libre

Logout: POST /api/auth/logout - Cierre de sesión

JWT Token: Almacenado en localStorage con nombre de usuario

- Procesos implementados y no implementados

Procesos Implementados

- **Autenticación:** Login/Registro/Logout JWT
- **Empleados:** CRUD completo con foto
- **Proveedores:** CRUD completo con foto
- **Categorías:** CRUD básico
- **Productos:** CRUD con imagen, categoría, proveedor
- **Stock:** Visualización y edición manual
- **Clientes:** CRUD completo con foto
- **Gastos:** CRUD básico
- **Salarios:** Pago a empleados, historial
- **POS:** Carrito, selección cliente, métodos pago
- **Órdenes:** Creación desde POS, historial
- **Dashboard:** Venta/Ingreso/Deuda/Gasto del día, Stock agotado

Procesos No Implementados

- **Roles/Permisos:** Sin diferenciación Admin/Empleado/Vendedor
- **Recuperación Password:** Página existe, pero sin backend funcional
- **Reportes:** Sin exportación PDF/Excel
- **Inventario:** Sin alertas automáticas de stock bajo
- **Compras:** Sin módulo de compras a proveedores

- **Devoluciones:** Sin gestión de devoluciones/reembolsos
- **Auditoría:** Sin log de acciones de usuarios
- **Pagos Pendientes:** No hay seguimiento de cobro de deudas
- **Multi-caja:** Un solo punto de venta
- **Multi-sucursal:** Una sola ubicación
- **Búsqueda:** Por fecha y mes en órdenes

- Oportunidades de mejora y mantenimiento

Críticas (Seguridad y Datos)

Problema	**Ubicación**	**Recomendación**
Rutas API sin protección JWT	Api.php	Envolver rutas CRUD con middleware('JWT')
Inyección SQL potencial	Uso de DB::raw() sin sanitizar	Usar Query Builder con bindings
Tipos de datos incorrectos	Precios como varchar en vez de decimal	Migrar a tipos numéricos
Validación frontend débil	Token.js	Corregir condición lógica
Hardcoded URLs	Tokem.js	Usar configuración de entorno

Funcionalidad

Problema	**Ubicación**	**Recomendación**
Sin sistema de roles	Modelo User	Implementar Spatie Laravel Permission
Carrito global (tabla POS)	pos Table	Sesión por usuario o carrito temporal con ID
Fechas como strings	Varias migraciones	Usar tipo DATE/TIMESTAMP
Sin paginación	Controladores API	Implementar ->paginate()

Mantenimiento

Problema	**Ubicación**	**Recomendación**
Laravel 5.8 obsoleto	composer.json	Actualizar a Laravel 10/11
Vue 2.x sin soporte	package.json	Migrar a Vue 3 con Composition API
Sin documentación API	Varias migraciones	Implementar Swagger/OpenAPI
Comentarios en bengalí	Varios archivos	Estandarizar idioma comentarios
Manejo de errores	Frontend .catch() vacíos	Implementar manejo global de errores