Rediseño y Estructuración de Sistemas

Informe académico sobre modularización, componentización y análisis de sistemas.

Elaborado por: Carlos Santiago Correa Narvaez  
Carrera: Ingeniería de Software  
Materia: Diseño de Software  
Docente: Miguel Tovar  
Fecha: 17 de septiembre de 2025  
Ubicación: San Juan de Pasto  
Semestre: Cuarto

# Caso 1: Plataforma Digital para un Centro de Salud

## 1. Capacidades del Sistema

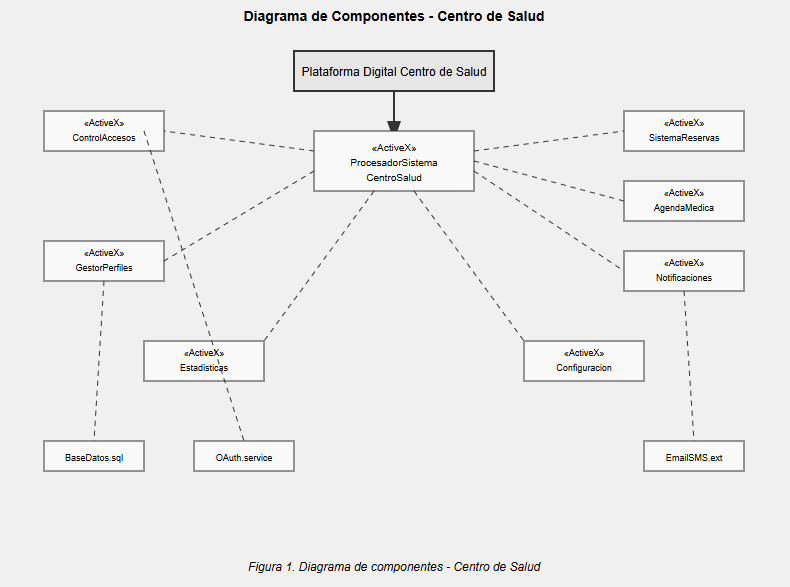
Los requerimientos funcionales se organizan en tres grupos de actores: pacientes, profesionales de la salud y personal administrativo.

### Para Usuarios (Pacientes)

RF01: Inscripción y acceso seguro.  
  
RF02: Localización de especialistas por disponibilidad o rama médica.  
  
RF03: Solicitud de consultas médicas.  
  
RF04: Consulta del historial de atenciones previas.  
  
RF05: Reprogramación o cancelación de turnos.  
  
RF06: Avisos y recordatorios automáticos.

### Para Profesionales de la Salud (Doctores)

RF07: Validación de identidad del médico.  
  
RF08: Configuración de la agenda de trabajo.  
  
RF09: Revisión de consultas agendadas.  
  
RF10: Confirmación o anulación de citas.



RF11: Administración del historial clínico simplificado.  
  
RF12: Bloqueo de horarios por inasistencias.

### Para Gestores Administrativos

RF13: Administración de cuentas de usuarios.  
  
RF14: Definición de especialidades.  
  
RF15: Reportes de funcionamiento del sistema.  
  
RF16: Organización de horarios institucionales.  
  
RF17: Monitoreo de la plataforma.

## 2. Fragmentación en Componentes

- Subsistema de Control de Accesos: inicio/cierre de sesión, validación de roles y permisos.  
  
- Subsistema de Perfiles: información de pacientes y doctores.  
  
- Subsistema de Reservas Médicas: solicitudes, validaciones y cambios de citas.  
  
- Subsistema de Agenda Médica: horarios, bloqueos y disponibilidad.  
  
- Subsistema de Notificaciones: envío de recordatorios y alertas.  
  
- Subsistema de Estadísticas: generación de informes y métricas.  
  
- Subsistema de Configuración General: parámetros institucionales y ajustes globales.

## 3. Interfaces y Vínculos

- API REST para comunicación frontend-backend.  
  
- Conector ORM/SQL para acceso a base de datos.  
  
- Servicios externos para mensajería (SMS, correo electrónico).  
  
- Integración opcional con sistemas de autenticación externos (OAuth/LDAP).  
  
- Registro de auditorías para seguridad y seguimiento.

## 4. Justificación de Diseño

El sistema fue diseñado bajo el principio de separación de responsabilidades, garantizando independencia entre módulos. Esto mejora la escalabilidad, seguridad y capacidad de evolución del sistema hacia nuevas tecnologías como telemedicina.

# Caso 2: Plataforma de Entregas a Domicilio

## 1. Funcionalidades del Sistema

Los requerimientos funcionales se dividen en clientes, restaurantes, repartidores y servicios generales del sistema.

### Clientes

RF01: Alta y acceso seguro de clientes.  
  
RF02: Filtrado de restaurantes por ubicación o categoría.  
  
RF03: Revisión de cartas y precios.  
  
RF04: Manejo del carrito de compras.  
  
RF05: Solicitud y pago de pedidos.  
  
RF06: Rastreo en tiempo real de la orden.  
  
RF07: Calificación de restaurantes y repartidores.  
  
RF08: Historial de pedidos realizados.

### Restaurantes

RF09: Registro de perfil del restaurante.  
  
RF10: Administración de menú y precios.  
  
RF11: Recepción y actualización de pedidos.  
  
RF12: Control de disponibilidad de productos.  
  
RF13: Confirmación de tiempos de preparación.  
  
RF14: Reportes y estadísticas de ventas.

### Repartidores

RF15: Registro e ingreso al sistema.  
  
RF16: Recepción de entregas disponibles.  
  
RF17: Rutas optimizadas mediante geolocalización.  
  
RF18: Confirmación de recogida y entrega.  
  
RF19: Administración de disponibilidad.  
  
RF20: Consulta de historial de entregas y ganancias.

### Servicios del Sistema

RF21: Procesamiento de pagos y comisiones.  
  
RF22: Servicio de geolocalización.  
  
RF23: Notificaciones en tiempo real.  
  
RF24: Cálculo de costos de envío.  
  
RF25: Centro de soporte y chat.

## 2. Arquitectura Basada en Microservicios

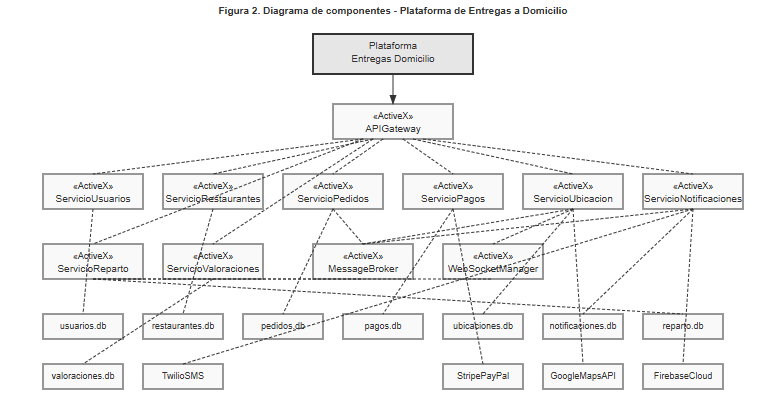
- Servicio de Usuarios: gestión de accesos y perfiles.  
  
- Servicio de Restaurantes: administración de menús y productos.  
  
- Servicio de Pedidos: lógica de órdenes y su estado.  
  
- Servicio de Pagos: transacciones y comisiones.  
  
- Servicio de Ubicación: rastreo en tiempo real y rutas.  
  
- Servicio de Notificaciones: avisos por push, correo y SMS.  
  
- Servicio de Reparto: asignación y seguimiento de entregas.  
  
- Servicio de Valoraciones: calificaciones y reseñas.

## 3. Interfaces y Conexiones

- API Gateway como punto de entrada.  
  
- APIs por microservicio (/users, /orders, /restaurants, /payments, etc.).  
  
- Message Brokers (Kafka/RabbitMQ) para comunicación asíncrona.  
  
- WebSockets para actualizaciones en tiempo real.  
  
- Integraciones externas: Stripe, PayPal, Google Maps, Firebase, Twilio.

## 4. Argumentos de Diseño

El uso de microservicios permite independencia tecnológica, escalabilidad y resiliencia. La comunicación basada en eventos asegura desacoplamiento, mientras que el API Gateway centraliza seguridad y enrutamiento.



# Caso 3: Sistema de Biblioteca Escolar

## 1. Problemas del Modelo Anterior

El sistema monolítico original presentaba alto acoplamiento, dificultad de mantenimiento, baja escalabilidad y escasa reutilización.

## 2. Requerimientos Funcionales

- Usuarios: registro, autenticación y permisos por tipo (estudiante/profesor).  
  
- Catálogo: administración de libros, filtros, clasificación y control de inventario.  
  
- Préstamos: solicitud, renovación, devolución y sanciones.  
  
- Reportes: estadísticas de uso, multas y popularidad de materiales.

## 3. Nueva Estructura Modular

- Módulo de Acceso: inicio de sesión y control de roles.  
  
- Módulo de Usuarios: datos personales e historial.  
  
- Módulo de Catálogo: búsqueda y clasificación de materiales.  
  
- Módulo de Préstamos: gestión de solicitudes, devoluciones y multas.  
  
- Módulo de Informes: reportes y métricas.  
  
- Módulo de Notificaciones: avisos a usuarios.  
  
- Módulo de Persistencia: base de datos con respaldo periódico.

## 4. Interfaces y Patrones de Integración

- Servicios internos (UserService, CatalogService, LoanService).  
  
- API REST para frontend-backend.  
  
- ORM/SQL para consultas de datos.  
  
- Patrones aplicados: Facade, Observer, Strategy.

## 5. Justificación del Diseño

La refactorización en módulos incrementa la flexibilidad, mantenibilidad y facilidad de pruebas. Cada subsistema responde al principio de responsabilidad única y permite evolución futura sin afectar el resto.

