Universidad Cooperativa de Colombia

Diseño de Software

Componentes

Valeria Estefanía Góngora Torres

Profesor: Miguel Tovar

17/09/25

Ejercicio 1. Un centro médico requiere un sistema web para que pacientes puedan reservar consultas, médicos gestionen su agenda y los administradores controlen el sistema.

Desarrollo:

**Ejercicio 1 = Centro Médico**

1. Funcionalidades = Requisitos funcionales

Pacientes: registrarse en el sistema, iniciar sesión, reservar sus consultas y ver su historial de citas.

Médicos: manejar su agenda, aceptar o rechazar citas y actualizar su disponibilidad.

Administradores: supervisar a todos los usuarios, controlar la plataforma y generar reportes.

1. Componentes del sistema

Autenticación y Roles: manejar el login y los permisos según el tipo de usuario (paciente, médico o admin).

Gestión de Pacientes: donde se guarda el perfil, historial y reservas de citas de cada paciente.

Gestión de Médicos: módulo que maneja las agendas, horarios y disponibilidad de los doctores.

Gestión de Consultas: el corazón del sistema, porque aquí se manejan las reservas de citas.

Administración del Sistema: módulo especial para los administradores, con control de usuarios y generación de reportes.

Base de Datos: donde se almacena toda la información de usuarios, citas y reportes.

1. Interfaces = Conectores

App Web = Backend: comunicación por medio de una API REST.

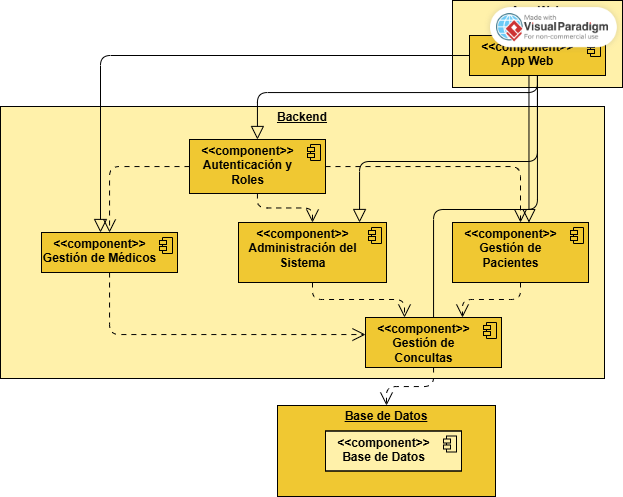
Gestión de Citas = Médicos/Pacientes: cuando alguien reserva o cancela, se envían notificaciones a las partes involucradas.

Backend = Base de Datos: consultas SQL para guardar o recuperar información.

Administradores = Sistema: acceso por un panel web con reportes y herramientas de control.

1. Análisis

El sistema se dividió en módulos porque cada rol (paciente, médico, administrador) tiene tareas diferentes.  
El módulo de consultas es el centro, ya que conecta directamente a pacientes y médicos.  
La administración del sistema se separó porque necesita control total, sin mezclarse con las operaciones normales.  
Con esta organización se logra un sistema más claro, que se puede mantener y escalar fácilmente en el futuro.



Ejercicio 2. Una startup quiere desarrollar una aplicación móvil y web tipo Rappi o Uber Eats, donde los usuarios pueden pedir comida, los restaurantes gestionan pedidos y los repartidores hacen las entregas.

Ejercicio 2 = Plataforma de Delivery (tipo Rappi/Uber Eats)

1. Funcionalidades = Requisitos funcionales

Usuarios: registrarse, iniciar sesión, explorar restaurantes y menús, hacer pedidos, pagar en línea y seguir el pedido en tiempo real.

Restaurantes: registrar restaurante y menú, recibir y aceptar pedidos, actualizar estado de preparación.

Repartidores: registrarse, iniciar sesión, ver pedidos asignados, marcar estados de entrega (recogido, en camino, entregado).

Administradores: monitorear usuarios, restaurantes, repartidores y generar reportes.

2. Componentes del sistema

Autenticación y Roles: manejar login y permisos según tipo de usuario (cliente, restaurante, repartidor o admin).

Gestión de Usuarios: perfiles, historial de pedidos, métodos de pago.

Gestión de Restaurantes: catálogo, menús, disponibilidad y pedidos recibidos.

Gestión de Pedidos: coordina creación, asignación y seguimiento de pedidos.

Gestión de Repartidores: manejo de repartidores, pedidos asignados y estados de entrega.

Pagos y Facturación: procesa pagos en línea y genera comprobantes.

Administración del Sistema: monitoreo y reportes para los administradores.

Base de Datos: almacena toda la información del sistema.

3. Interfaces = Conectores

App Móvil/Web = Backend: comunicación a través de una API REST o GraphQL.

Usuarios/Restaurantes/Repartidores = Gestión de Pedidos: cuando hay un pedido nuevo o cambia de estado, se envían notificaciones.

Pagos = Bancos/Plataformas externas: integración con pasarelas de pago para procesar transacciones.

Backend = Base de Datos: consultas y actualizaciones de información.

Administradores = Sistema: acceso a un panel web con métricas y reportes en tiempo real.

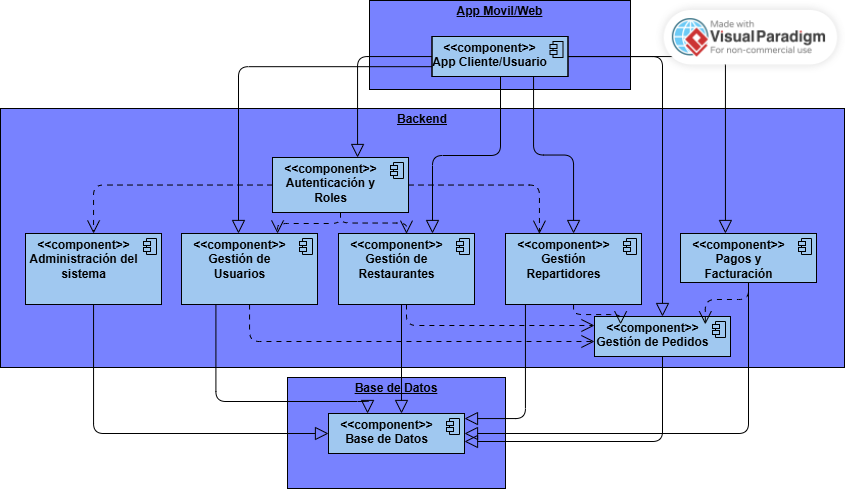
4. Análisis

El sistema se modularizó porque cada rol (usuario, restaurante, repartidor, administrador) tiene responsabilidades distintas.

El módulo de Gestión de Pedidos es el núcleo, ya que conecta a todos los actores (usuarios, restaurantes y repartidores).

Se separó Pagos y Facturación porque requiere seguridad especial y conexión con servicios externos.

Con esta organización se logra una arquitectura clara, segura y fácil de escalar cuando crezca el número de usuarios o se integren nuevas funciones.



Ejercicio 3. Un colegio tiene un sistema heredado de biblioteca que mezcla lógica de préstamo, usuarios, catálogo y reportes en un solo módulo monolítico. El sistema debe ser refactorizado y rediseñado usando principios modernos de diseño.

Ejercicio 3 = Sistema de Biblioteca

1. Funcionalidades = Requisitos funcionales

Usuarios: registrarse, iniciar sesión, consultar catálogo de libros, realizar préstamos y devoluciones, revisar historial.

Bibliotecarios: administrar catálogo (agregar, editar, eliminar libros), gestionar préstamos y devoluciones.

Administradores: generar reportes del sistema (usuarios activos, préstamos, libros más consultados), supervisar todo el sistema.

2. Componentes del sistema

Autenticación y Roles: controlar acceso y permisos de usuarios, bibliotecarios y administradores.

Gestión de Usuarios: perfiles, historial de préstamos, sanciones.

Gestión de Catálogo: administración de libros, autores, disponibilidad.

Gestión de Préstamos: lógica para préstamos, devoluciones y reservas de libros.

Reportes y Estadísticas: generación de reportes de uso y estadísticas de la biblioteca.

Administración del Sistema: control general y configuración del sistema.

Base de Datos: almacenamiento de información de usuarios, catálogo, préstamos y reportes.

3. Interfaces = Conectores

App Web = Backend: comunicación por API REST para acceder a funcionalidades desde navegador o app.

Gestión de Usuarios = Gestión de Préstamos: cada préstamo se registra en el perfil del usuario.

Gestión de Catálogo = Gestión de Préstamos: validar disponibilidad de libros antes de confirmar el préstamo.

Reportes = Todos los módulos: recolecta datos de usuarios, préstamos y catálogo.

Backend = Base de Datos: consultas y actualizaciones permanentes.

Administradores = Sistema: acceso por panel web para reportes y configuración.

4. Análisis

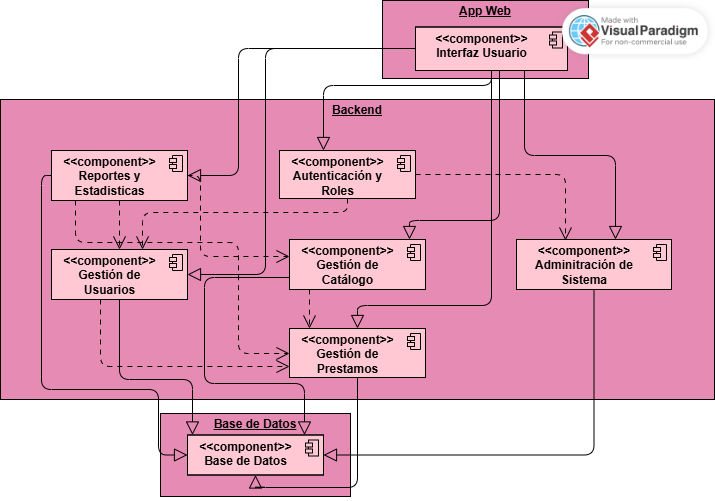
El sistema monolítico original dificultaba el mantenimiento porque toda la lógica estaba mezclada en un solo módulo.

La refactorización separa las responsabilidades en componentes independientes: usuarios, catálogo, préstamos y reportes.

Esto facilita la escalabilidad y el mantenimiento, porque cada módulo puede actualizarse sin afectar a los demás.

El módulo de Gestión de Préstamos es clave porque conecta a usuarios y catálogo, mientras que los reportes permiten un control global.

Con este rediseño se aplican principios modernos como separación de responsabilidades y modularización, lo que asegura un sistema más flexible y robusto.



Conclusiones

La modularización y la componentización permiten que los sistemas sean más organizados, fáciles de mantener y escalar. Al separar cada responsabilidad en componentes, se logra que los cambios en un módulo no afecten a todo el sistema.

En los tres ejercicios se evidenció que cada caso tiene un módulo central que conecta a los demás: consultas en el centro médico, pedidos en la app de delivery y préstamos en la biblioteca. Separar la administración, la autenticación y los reportes ayuda a tener un sistema más claro y flexible.

En conclusión, aplicar estos principios facilita el crecimiento de los sistemas y asegura que puedan adaptarse a las necesidades de los usuarios y de las organizaciones en el tiempo.