Modularización y componetización.

PRESENTADO POR:

CARLOS ANDRES TAQUEZ MAYA

DISEÑO DE SOFTWARE

PRESENTADO A:

MIGUEL ANGEL TOVAR

UNIVERSIDAD COOPERATIVA DE COLOMBIA

INGENIERIA DE SOFTWARE

CAMPUS PASTO

NOVIEMBRE, 2025

**Ejercicio 1. - SISTEMA PARA CENTRO MÉDICO**

1. Funcionalidades (Requisitos funcionales)

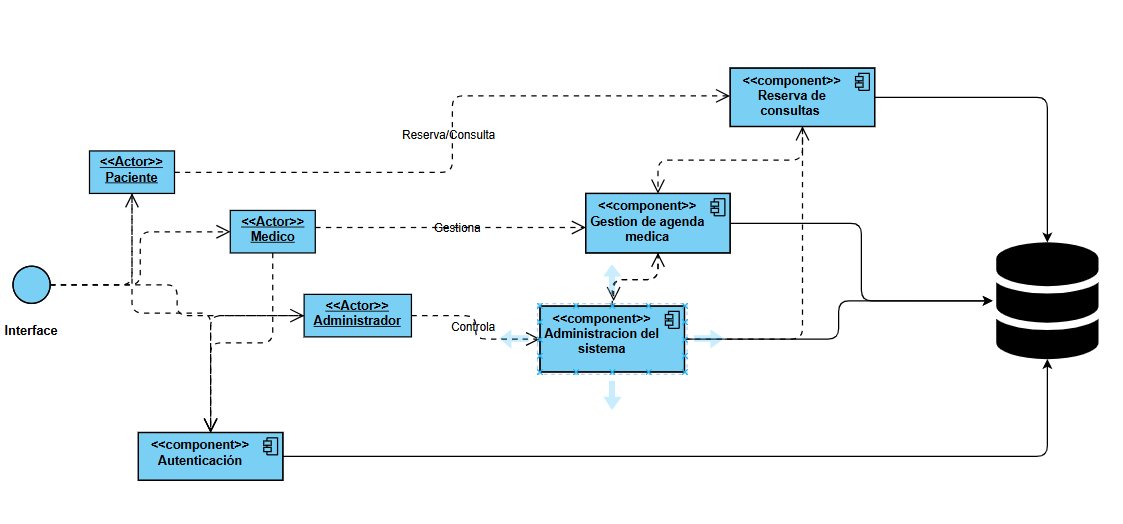
* Registro e inicio de sesión de usuarios (pacientes, médicos, administradores).
* Reserva de consultas por parte de pacientes.
* Consulta y gestión de agenda por parte de médicos.
* Gestión de usuarios, roles y accesos por parte de administradores.

2. **Modularización – Componentes del sistema**

|  |  |
| --- | --- |
| Componente | Función principal |
| Reserva de Consultas | Permite a los pacientes agendar, cancelar y consultar citas. |
| Gestión de Agenda Médica | Los médicos definen horarios, aceptan o rechazan citas. |
| Administración del Sistema | Alta/baja de usuarios, roles, accesos. |
| Autenticación y Seguridad | Login, registro, manejo de sesiones y roles. |
| Base de Datos – HospitalDB | Almacena pacientes, médicos, citas, usuarios, agendas. |

3. Interfaces / Conectores

* Reserva de Consultas → Gestión de Agenda Médica: para consultar disponibilidad.
* Todos los componentes → Autenticación: para validar sesiones y roles.
* Todos los componentes → HospitalDB: para acceder a la información.
* Administración del Sistema → Autenticación: para asignar roles.
* Reserva de Consultas ←→ Base de Datos: guardar y consultar citas.

****

4. Análisis y justificación del diseño

El sistema se ha dividido en componentes siguiendo el principio de responsabilidad única, permitiendo escalar o mantener módulos sin afectar al resto.

Separar la autenticación como un módulo central mejora la seguridad y reutilización, mientras que conectar todos los componentes a una base de datos centralizada (HospitalDB) garantiza integridad y consistencia de los datos.

Además, la estructura modular facilita la migración futura a una arquitectura de microservicios.

**Ejercicio 2. - APP ESTILO RAPPI / UBER EATS**

1. Funcionalidades (Requisitos funcionales)

Usuarios pueden explorar restaurantes, hacer pedidos y pagar.

Restaurantes gestionan su menú, aceptan o rechazan pedidos.

Repartidores aceptan entregas, siguen rutas y marcan entregas como completadas.

Sistema de login, registro y roles.

Notificaciones en tiempo real.

2. Modularización – Componentes del sistema

|  |  |
| --- | --- |
| Componente | Función principal |
| Pedidos | Gestión de pedidos (crear, asignar, estado). |
| Gestión de Restaurantes | Menú, pedidos recibidos, horarios. |
| Gestión de Repartidores | Toma de pedidos, seguimiento de entregas. |
| Base de Datos – AppPedidosDB | Almacena usuarios, pedidos, menús, entregas, etc. |

3. Interfaces / Conectores

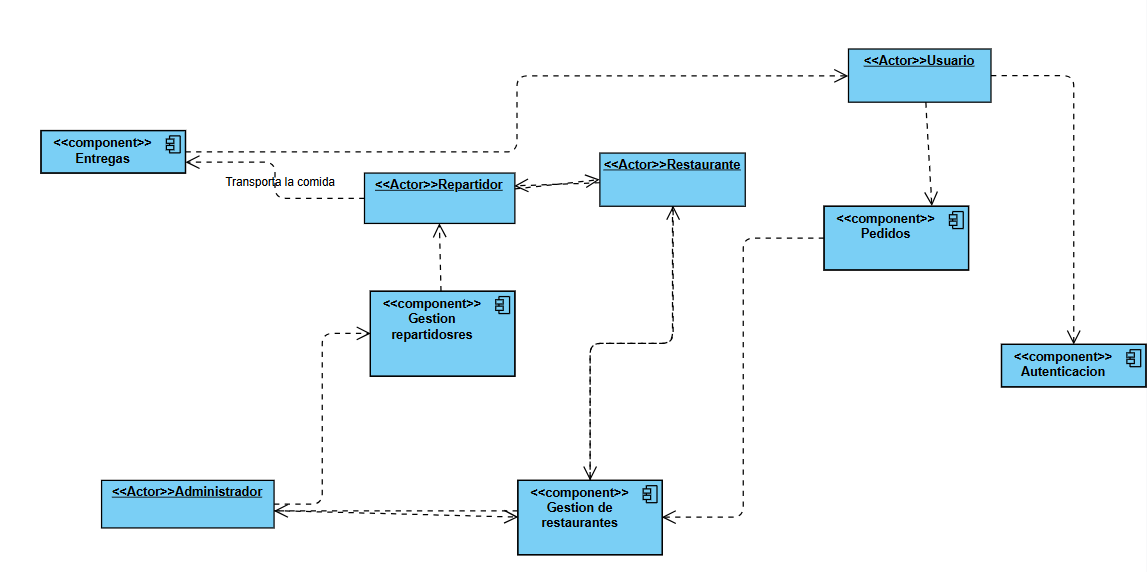
* Pedidos → Gestión de Restaurantes: para enviar nuevos pedidos.
* Pedidos → Gestión de Repartidores: para asignar entregas.
* Pedidos → Notificaciones: enviar estados a los usuarios.
* Todos → Autenticación: validar sesión y permisos.
* Todos → AppPedidosDB: consultar/guardar información.

4. Análisis y justificación del diseño

La separación clara por tipo de usuario (cliente, restaurante, repartidor) permite gestionar lógicas específicas para cada uno.

El componente de notificaciones desacoplado mejora la capacidad del sistema para crecer (push, correo, SMS).

Esta arquitectura es ideal para escalar partes específicas del sistema (por ejemplo, solo repartidores si hay muchos pedidos).

****

**Ejercicio 3 – REFACTORIZACIÓN DEL SISTEMA DE BIBLIOTECA**

1. Funcionalidades (Requisitos funcionales)

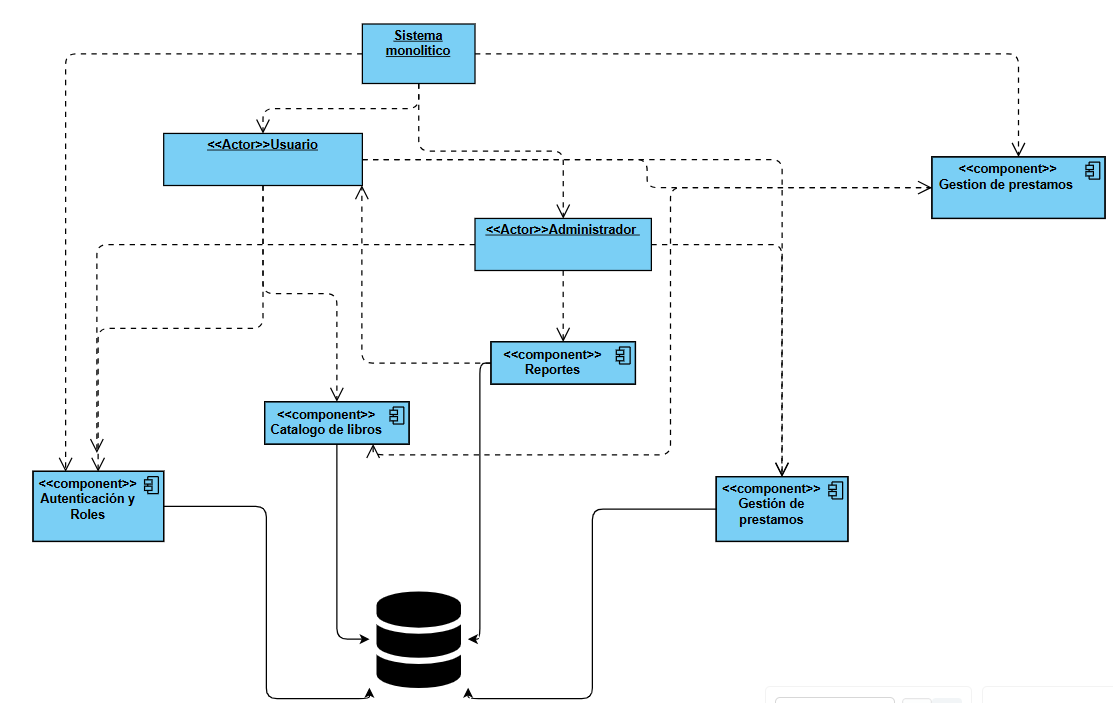
* Registro de usuarios (alumnos, docentes).
* Préstamos y devoluciones de libros.
* Consulta de catálogo.
* Generación de reportes (libros más prestados, usuarios activos, etc).

2. **Modularización – Componentes del sistema**

|  |  |
| --- | --- |
| Componente | Función principal |
| Gestión de Préstamos | Lógica de préstamo, devoluciones y multas. |
| Gestión de Usuarios | Registro y control de usuarios. |
| Catálogo de Libros | Registro, edición y búsqueda de libros. |
| Reportes | Generación de informes y estadísticas. |
| Autenticación y Roles | Login, roles y control de acceso. |
| Base de Datos – BibliotecaDB | Almacena usuarios, préstamos, libros, reportes, etc. |

3. Interfaces / Conectores

* Gestión de Préstamos → Catálogo de Libros: verificar disponibilidad.
* Gestión de Préstamos → Gestión de Usuarios: validar usuario.
* Todos → Autenticación: validar sesión.
* Reportes → Todos: recolectar datos para informes.
* Todos → BibliotecaDB: almacenamiento y consulta.



4. Análisis y justificación del diseño

Se parte de un sistema monolítico y se divide en módulos independientes, de acuerdo con el principio de separación de responsabilidades.

Esto permite escalar, actualizar o incluso migrar partes del sistema a la nube o a microservicios sin romper toda la lógica.

También mejora la mantenibilidad y permite asignar equipos de desarrollo separados a cada componente.