**Trabajo hecho por Miguel Angel Mendoza Jaramillo**

**4to semestre de ingeniería de software**

**Diseño de software**

**Modularización y componetización.**

Elabore para cada uno de los siguientes enunciados:

1. Identifique las funcionalidades (requisitos funcionales)
2. Modularice el sistema que dará solución a las funcionalidades identificando componentesSSSS
3. Identifique las interfaces – conectores que permitirán comunicar los componentes
4. Agregue un análisis breve que justifique las decisiones de diseño para el modelo propuesto.
5. Socialice en clase sus propuestas de diseño.

**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.

1. Funcionalidades

* Registro e inicio de sesión de pacientes y médicos
* Reserva y cancelación de citas médicas
* Gestión de agenda de médicos
* Administración de usuarios y reportes
* Envío de notificaciones

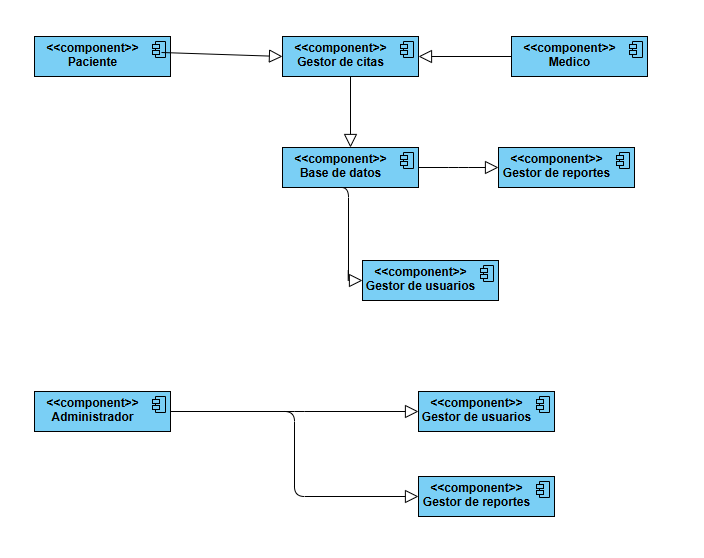
1. Modularización

* Gestor de Citas
* Gestor de Pacientes
* Gestor de Médicos
* Gestor de Usuarios y Autenticación
* Gestor de Reportes
* Servicio de Notificaciones
* Base de Datos

1. Interfaces y conectores

* API REST entre la interfaz web/móvil y los gestores
* Conectores de base de datos desde cada gestor hacia la BD
* Conector de notificaciones (correo, SMS)
* Autenticación basada en tokens

1. Análisis  
   El sistema se organiza en módulos independientes para reducir acoplamiento. El gestor de citas administra la agenda y se conecta a la base de datos, mientras que notificaciones es un servicio separado para poder cambiar el proveedor sin alterar la lógica del sistema.



**Ejercicio 2.** Un 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 – Aplicación tipo Rappi/Uber Eats

1. Funcionalidades

* Usuario: búsqueda de restaurantes, pedidos, pagos y seguimiento
* Restaurante: gestión de menú y actualización de pedidos
* Repartidor: aceptación y actualización de entregas
* Administración: reportes y soporte

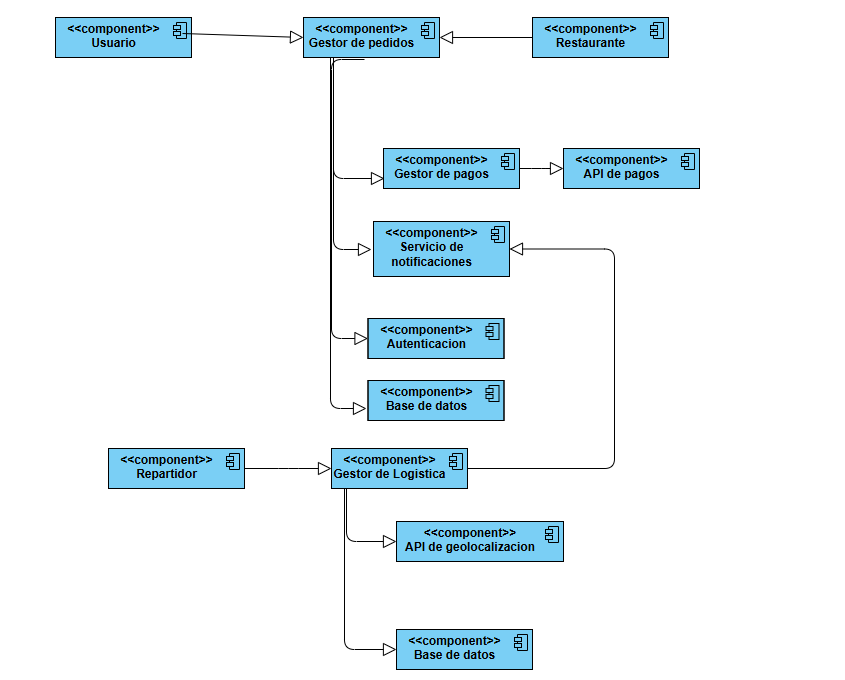
1. Modularización

* Gestor de Pedidos
* Gestor de Logística
* Gestor de Pagos
* Servicio de Notificaciones
* Autenticación
* API de Geolocalización
* API de Pagos
* Base de Datos

1. Interfaces y conectores

* Comunicación REST y WebSocket entre usuarios, restaurantes y gestores
* Gestor de Pagos hacia API de Pagos
* Gestor de Logística hacia API de Geolocalización
* Gestores hacia la base de datos
* Gestores hacia servicio de notificaciones
* Autenticación con tokens

1. Análisis  
   El diseño aísla los módulos de pagos y logística para poder escalar de manera independiente. Se usan WebSockets para las comunicaciones en tiempo real y APIs externas para pagos y geolocalización, encapsuladas para poder reemplazarlas en caso de ser necesario.



**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.

1. Funcionalidades

* Gestión de usuarios con roles
* Búsqueda y gestión del catálogo de libros
* Préstamos, devoluciones, renovaciones y multas
* Reportes estadísticos
* Notificaciones de recordatorio

1. Modularización

* Gestor de Usuarios
* Gestor de Catálogo
* Gestor de Préstamos
* Gestor de Reportes
* Servicio de Notificaciones
* Base de Datos
* Motor de Búsqueda opcional

1. Interfaces y conectores

* API REST entre usuarios y gestores
* Gestores hacia la base de datos
* Gestor de Préstamos hacia Usuarios y Catálogo para validaciones
* Gestor de Préstamos hacia Servicio de Notificaciones
* Catálogo hacia Motor de Búsqueda (opcional)

1. Análisis  
   Se refactoriza el sistema monolítico separando responsabilidades. El gestor de préstamos actúa como orquestador y consulta a usuarios y catálogo, mientras que reportes lee datos de la base. Esto reduce la deuda técnica y permite evolucionar módulos por separado.

