**Actividad 2. Modularización y componetización**

Estudiante:

Nicol Juliet Muñoz Ome

Docente:

Miguel Tovar

Curso:

Diseño de Software

Universidad Cooperativa de Colombia

Septiembre de 2025

Pasto, Colombia

# Ejercicio 1

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

1. Identifique las funcionalidades (requisitos funcionales)

**Pacientes**

RQ001: El sistema debe permitir el registro y autenticación de pacientes.

RQ002: El sistema debe permitir a los pacientes buscar médicos por especialidad y disponibilidad.

RQ003: El sistema debe permitir reservar una consulta en línea.

RQ004: El sistema debe permitir cancelar o reprogramar una cita.

RQ005: El sistema debe mostrar el historial de citas del paciente.

**Médicos**

RQ006: El sistema debe permitir el inicio de sesión de médicos.

RQ007: El sistema debe permitir que los médicos gestionen su agenda (definir horarios disponibles).

RQ008: El sistema debe permitir a los médicos aceptar o rechazar citas.

RQ009: El sistema debe mostrar el historial de consultas de cada paciente al médico.

**Administradores**

RQ010: El sistema debe permitir el registro y gestión de usuarios (pacientes y médicos).

RQ011: El sistema debe permitir la generación de reportes sobre el uso del sistema (consultas, disponibilidad).

RQ012: El sistema debe permitir la configuración de parámetros generales (especialidades, horarios, roles).

**Transversales**

RQ013: El sistema debe contar con autenticación y autorización según rol.

RQ014: El sistema debe enviar notificaciones de confirmación y recordatorios de citas.

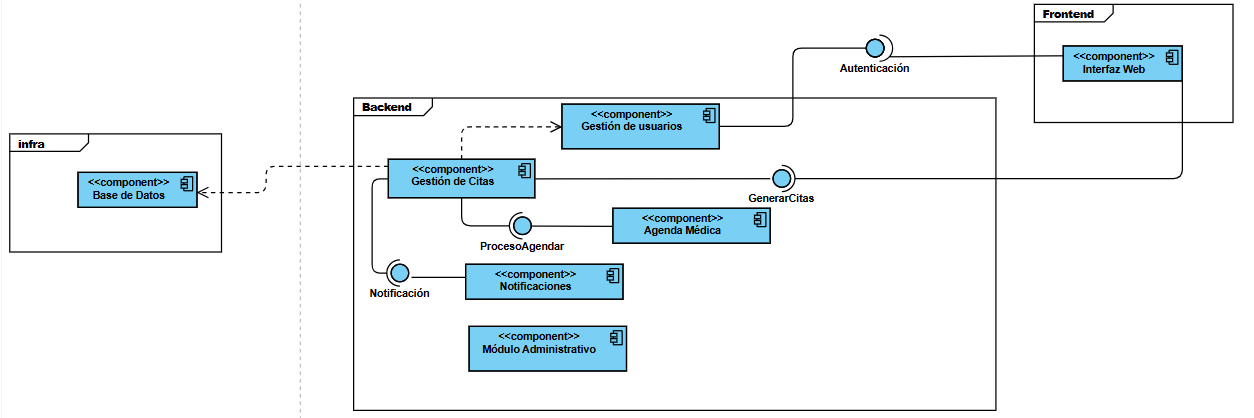
RQ015: El sistema debe garantizar el almacenamiento seguro de la información en una base de datos.

1. Modularice el sistema que dará solución a las funcionalidades identificando componentes

* **Gestión de Usuarios**: administración de registros, login, roles y perfiles.
* **Gestión de Citas**: permite reservar, cancelar y consultar historial de citas.
* **Agenda Médica**: define y gestiona la disponibilidad de los médicos.
* **Módulo Administrativo**: genera reportes, controla usuarios y configura el sistema.
* **Notificaciones**: envía confirmaciones y recordatorios a pacientes y médicos.
* **Base de Datos**: almacén central de toda la información del sistema.
* **Interfaz Web**: portal de acceso para pacientes, médicos y administradores.

1. Identifique las interfaces – conectores que permitirán comunicar los componentes

* **Interfaz Web ↔ Gestión de Usuarios**: autenticación, login, registro.
* **Interfaz Web ↔ Gestión de Citas**: consulta de disponibilidad y reservas.
* **Interfaz Web ↔ Agenda Médica**: acceso a horarios de médicos.
* **Interfaz Web ↔ Módulo Administrativo**: gestión de usuarios y reportes.
* **Gestión de Usuarios ↔ Base de Datos**: almacenamiento de credenciales y perfiles.
* **Gestión de Citas ↔ Base de Datos**: persistencia de citas y estados.
* **Agenda Médica ↔ Base de Datos**: almacenamiento de disponibilidad de médicos.
* **Módulo Administrativo ↔ Base de Datos**: consultas para reportes y control general.
* **Notificaciones ↔ Gestión de Citas y Usuarios**: envío de confirmaciones y recordatorios.



1. Agregue un análisis breve que justifique las decisiones de diseño para el modelo propuesto.

El sistema fue diseñado utilizando un enfoque modular para facilitar el mantenimiento, la escalabilidad y la claridad en la comunicación entre componentes. Se dividió en **Frontend**, **Backend** e **Infraestructura** con el fin de separar las responsabilidades y evitar acoplamientos innecesarios.

* **Frontend**: contiene la interfaz web, que se conecta a los servicios del backend mediante interfaces claramente definidas. Esto asegura que los pacientes, médicos y administradores accedan de manera unificada al sistema sin depender directamente de la lógica interna.
* **Backend**: se organizó en módulos especializados:
  + Gestión de Usuarios para autenticación y control de acceso.
  + Gestión de Citas como núcleo del sistema, encargado de procesar y administrar reservas.
  + Agenda Médica para el control de disponibilidad de los doctores.
  + Notificaciones para el envío de recordatorios y confirmaciones.
  + Módulo Administrativo para el control general y generación de reportes.  
    Esta modularización facilita que cada componente tenga funciones específicas y pueda evolucionar sin afectar al resto.
* **Infraestructura**: la Base de Datos se mantiene como un componente independiente que provee acceso de datos a los demás módulos, garantizando integridad y consistencia de la información.

El uso de **interfaces provistas y requeridas** permite definir claramente qué servicios ofrece cada componente y cuáles necesita consumir, generando un diseño flexible y fácil de comprender. Además, mantenemos la coherencia con los requisitos funcionales identificados, asegurando que cada necesidad del sistema esté respaldada por un módulo concreto.

# Ejercicio 2

**Enunciado**: 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.

1. Identifique las funcionalidades (requisitos funcionales)

**RQ001:** El sistema debe permitir el registro e inicio de sesión de usuarios, restaurantes y repartidores.

**RQ002:** El sistema debe permitir a los usuarios buscar y explorar restaurantes disponibles según su ubicación.

**RQ003:** El sistema debe mostrar los menús y precios de cada restaurante.

**RQ004:** El sistema debe permitir a los usuarios agregar productos a un carrito de compras.

**RQ005:** El sistema debe permitir confirmar pedidos mostrando el resumen y el costo total.

**RQ006:** El sistema debe procesar pagos en línea mediante diferentes métodos (tarjeta, billetera digital).

**RQ007:** El sistema debe notificar en tiempo real al restaurante cuando recibe un nuevo pedido.

**RQ008:** El sistema debe permitir a los restaurantes aceptar, rechazar y actualizar el estado de los pedidos.

**RQ009:** El sistema debe asignar un repartidor disponible para realizar la entrega.

**RQ010:** El sistema debe permitir a los repartidores aceptar o rechazar entregas.

**RQ011:** El sistema debe mostrar al usuario el seguimiento en tiempo real del pedido y la ubicación del repartidor.

**RQ012:** El sistema debe enviar notificaciones al usuario sobre el estado de su pedido (aceptado, en camino, entregado).

**RQ013:** El sistema debe permitir calificar restaurantes y repartidores después de cada pedido.

**RQ014:** El sistema debe contar con un módulo administrativo para gestionar usuarios, restaurantes y repartidores.

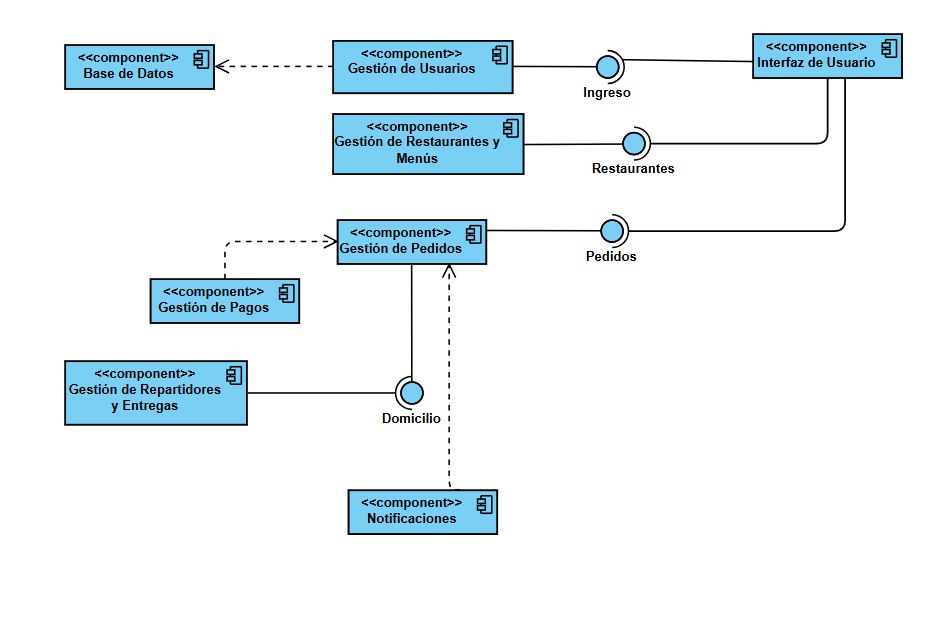
**RQ015:** El sistema debe generar reportes de pedidos, ventas y tiempos de entrega para los administradores.

1. Modularice el sistema que dará solución a las funcionalidades identificando componentes

* **Módulo de Usuarios**
  + Permite el registro e inicio de sesión de usuarios, restaurantes y repartidores.
  + Maneja los perfiles y roles de cada actor del sistema.
* **Módulo de Restaurantes**
  + Administra la información de los restaurantes.
  + Permite gestionar menús, precios y disponibilidad.
* **Módulo de Pedidos**
  + Se encarga de recibir los pedidos de los usuarios.
  + Controla el estado del pedido: confirmado, en preparación, listo, entregado.
* **Módulo de Pagos**
  + Procesa los pagos en línea de los pedidos.
  + Genera comprobantes y valida transacciones.
* **Módulo de Repartidores**
  + Asigna pedidos a repartidores disponibles.
  + Permite actualizar el estado de las entregas.
* **Módulo de Notificaciones**
  + Informa en tiempo real a usuarios, restaurantes y repartidores sobre pedidos y cambios de estado.
* **Módulo de Calificaciones**
  + Permite que los usuarios califiquen restaurantes y repartidores después de cada pedido.
* **Módulo Administrativo**
  + Facilita la gestión de cuentas y reportes para la administración del sistema.
* **Base de Datos**
  + Guarda toda la información del sistema: usuarios, menús, pedidos, pagos, entregas y calificaciones.

1. Identifique las interfaces – conectores que permitirán comunicar los componentes

* **Interfaz App/Web**
* **Gestión de Usuarios**
* **Gestión de Restaurantes**
* **Gestión de Pedidos**
* **Gestión de Repartidores**
* **Gestión de Pagos**
* **Notificaciones**
* **Base de Datos**

****

1. Agregue un análisis breve que justifique las decisiones de diseño para el modelo propuesto.

El modelo de componentes se diseñó buscando que el sistema sea claro y modular, dividiendo las responsabilidades en bloques separados. Se utilizó una Interfaz de Usuario que centraliza la interacción de los clientes con la aplicación móvil o web. Los componentes principales, como Gestión de Usuarios, Restaurantes y Menús, Gestión de Pedidos, Pagos y Repartidores, permiten organizar las funciones de manera independiente y facilitan el mantenimiento o posibles mejoras en el futuro.

El componente de Gestión de Pedidos se definió como el núcleo del sistema, ya que se comunica tanto con restaurantes (para enviar órdenes), como con repartidores (para asignar entregas) y con pagos (para procesar las transacciones). Además, el uso de Notificaciones asegura que los usuarios reciban información en tiempo real sobre el estado de sus pedidos.

Finalmente, la Base de Datos se estableció como soporte común para todos los módulos, centralizando la información de usuarios, restaurantes, pedidos y pagos. Esta separación de módulos ayuda a mantener la claridad en el diseño y refleja de forma sencilla cómo interactúan los distintos actores en la aplicación.

# Ejercicio 3

**Enunciado**: 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. Identifique las funcionalidades (requisitos funcionales)

**RQ001**: Registrar y gestionar usuarios de la biblioteca.

**RQ002**: Consultar el catálogo de libros.

**RQ003**: Realizar préstamos de libros.

**RQ004**: Registrar devoluciones.

**RQ005**: Consultar historial de préstamos por usuario.

**RQ006**: Generar reportes de uso (libros más prestados, usuarios activos).

**RQ007**: Administrar inventario de libros (altas, bajas, actualizaciones).

**RQ008**: Notificar retrasos en devoluciones.

**RQ009**: Permitir reservas de libros (apartados antes de préstamo).

**RQ010**: Gestionar multas por retraso en devoluciones.

**RQ011**: Generar credenciales o carnés digitales para los usuarios.

**RQ012**: Filtrar y buscar libros por categoría, autor o disponibilidad.

**RQ013**: Emitir reportes de multas y pagos.

**RQ014**: Controlar el acceso de administradores y bibliotecarios con distintos permisos.

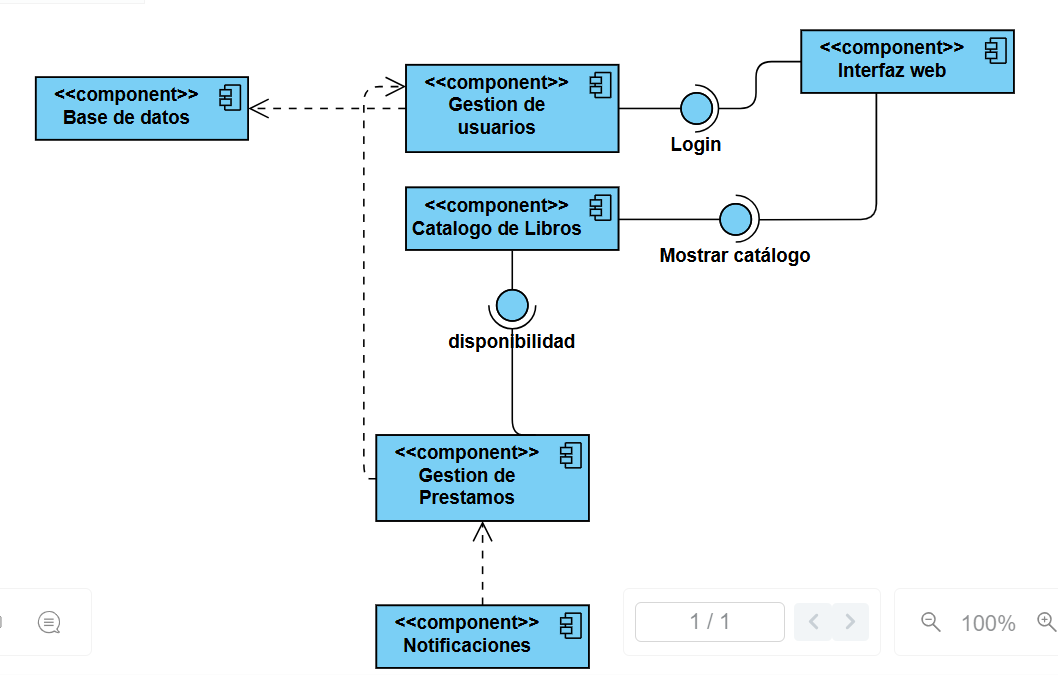
**RQ015**: Exportar reportes en formatos comunes (PDF, Excel).

1. Modularice el sistema que dará solución a las funcionalidades identificando componentes

* **Gestión de Usuarios**: manejo de registros y perfiles de usuarios.
* **Catálogo de Libros**: búsqueda y consulta de material.
* **Gestión de Préstamos**: registro de préstamos y devoluciones.
* **Inventario**: actualización del stock de libros.
* **Reportes**: generación de informes estadísticos.
* **Notificaciones**: avisos por correo o mensajes de retraso.
* **Base de Datos**: almacenamiento centralizado.
* **Interfaz Web**: acceso para usuarios y administradores.

1. Identifique las interfaces – conectores que permitirán comunicar los componentes

* La **Interfaz Web** se conecta con Usuarios, Catálogo y Préstamos.
* **Gestión de Préstamos** se comunica con Catálogo (para verificar disponibilidad) y Usuarios (para validar permisos).
* **Préstamos** también conecta con Notificaciones (avisar retrasos)
* **Todos los módulos** consultan y actualizan la **Base de Datos**.



1. Agregue un análisis breve que justifique las decisiones de diseño para el modelo propuesto.

El rediseño separa el sistema monolítico en módulos independientes, siguiendo principios de modularidad y responsabilidad única. Esto permite que cada componente (usuarios, catálogo, préstamos, reportes) pueda mantenerse o escalarse sin afectar a los demás. Además, se facilita la integración de nuevas funciones como notificaciones o estadísticas, y se centraliza la información en una base de datos común. La Interfaz Web se mantiene como el punto de acceso unificado, lo que simplifica la interacción de usuarios y administradores con el sistema.