**Nombre: Steven Eraso Insuasti**

**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.

**Funcionalidades**

|  |  |  |
| --- | --- | --- |
| **Pacientes** | **Médicos** | **Administradores** |
| Registrarse e iniciar sesión | Iniciar sesión como médico. | Iniciar sesión como administrador. |
| Buscar médicos por especialidad y disponibilidad | Gestionar su agenda (crear, modificar o cancelar horarios) | Crear, editar y eliminar usuarios (pacientes y médicos) |
| Reservar, modificar o cancelar citas | Ver citas programadas | Supervisar y controlar el sistema. |
| Ver historial de consultas | Registrar observaciones medicas |  |

**Componentes**

|  |  |
| --- | --- |
| **Componente** | **Descripción** |
| Gestión de usuarios | Registro de pacientes, médicos y admin |
| Agenda medica | Horarios disponibles, gestión de citas, calendario médico |
| Reserva de citas | Búsqueda de disponibilidad, reserva, modificación y cancelación de citas |
| Gestión administrativa | Administración de usuarios y estadísticas del sistema |
| Historial medico | Visualización de citas pasadas y observaciones medicas |
| Notificaciones | Confirmaciones de cita, recordatorios por correo o SMS |

**Diagrama

El contenido generado por IA puede ser incorrecto.Diagrama de Componentes**

**Justificación**

El sistema tiene como componente central la “Gestión de Usuarios” ya que cualquier acción como reservar, revisar la agenda o acceder al historial médico, dependen de usuarios registrados y autenticados. También se separaron responsabilidades lo que favorece la modularidad y facilita futuras ampliaciones. Al tener servicios independientes se puede escalar solos módulos que lo requieran.

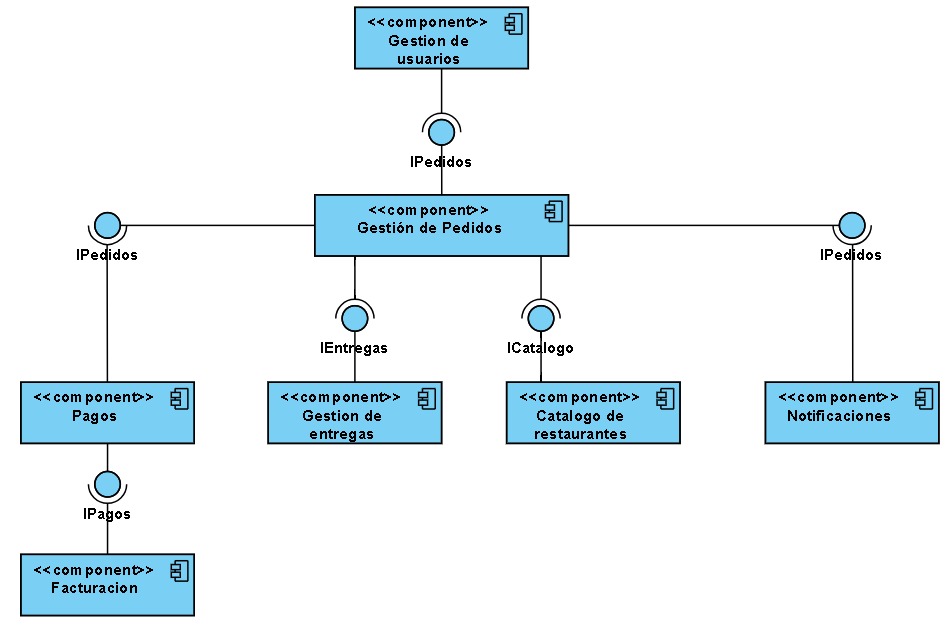
**Ejercicio 2:** Un startup quiere desarrollar una aplicación móvil y web tipo Rappi o Uber Eats, donde los usuarios pueden pedir comida, los restaurantes gestionan pedidos y los repartidores hacen las entregas.

**Funcionalidades**

|  |  |  |  |
| --- | --- | --- | --- |
| **Usuarios** | **Restaurante** | **Repartidores** | **Admin del sistema** |
| Registrarse e iniciar sesión | Iniciar sesión como restaurante | Registrarse e iniciar sesión como repartidor | Iniciar sesión como administrador |
| Navegar por restaurantes y menús. | Crear, editar y eliminar productos (menú) | Ver pedidos asignados | Gestionar usuarios, restaurantes y repartidores |
| Realizar pedidos y pagos | Recibir y aceptar pedidos | Ver ubicación del cliente | Supervisar el rendimiento general |
| Rastrear el estado del pedido | Actualizar estado del pedido | Marcar pedido como entregado | Resolver inconvenientes |

**Componentes**

|  |  |
| --- | --- |
| **Componente** | **Descripción** |
| Página web | Frontend |
| Gestión de usuarios | Registro e inicio de sesión según el rol (cliente, restaurante, repartidor, admin) |
| Catálogo de restaurantes | Menús, productos, generar una búsqueda por tipo de producto |
| Gestión de pedidos | Flujo completo del pedido (creación, estado, entrega) |
| Gestión de entregas | Asignación de repartidores y seguimiento de ubicación |
| Pagos | Procesamiento de pagos |
| Facturación | Generación de facturas |
| Notificaciones | Alertas de estado del pedido además de promociones o mensajes |

**Diagrama de Componentes**

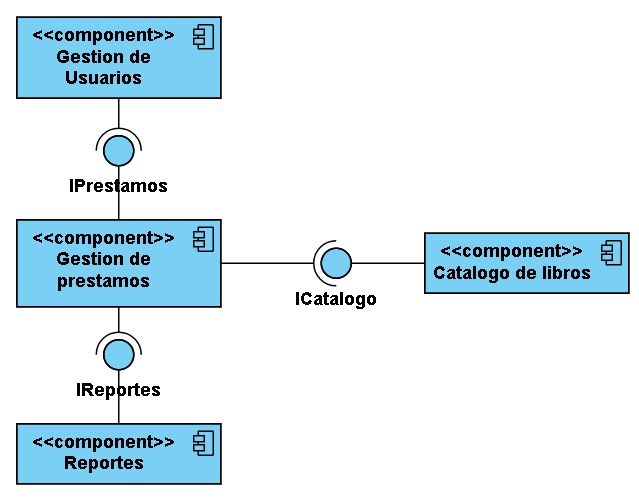
**Justificación**

Se hizo un sistema centralizado donde el módulo central es la “Gestión de pedidos” el cual coordina todo el sistema, esto se decidió ya que se reduce la complejidad y permite que otros módulos se mantengan desacoplados; además se hace separación de responsabilidades donde cada componente tiene una función bien definida, lo cual sigue el principio de alta cohesión y bajo acoplamiento, lo que facilita el mantenimiento y escabilidad.

**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.

|  |  |
| --- | --- |
| **Funcionalidad** | **Componente** |
| * Registro y administración de estudiantes y docentes * Asignaciones de permisos | Gestión de usuarios |
| * Consulta de títulos, autores y disponibilidad * Clasificaciones por genero literario y nivel educativo. | Catálogo de libros |
| * Solicitar, renovar y devolver libros * Control de fechas límites y sanciones | Gestión de prestamos |
| * Estadísticas de uso, cuáles son los libros mas prestados * Envió de datos para administración | Reportes |

**Diagrama de Componentes**



**Justificación**

Cada componente tiene una función clara, lo que facilita entender, desarrollar y probar cada módulo por separado, además que se vuelven componentes fáciles de escalar individualmente. También con esta estructura es sencillo añadir un API o una interfaz web.