**Casos de estudio Calidad de Software.**

**Presentado por:**

**Juan Manuel Cordoba Florez.**

**Presentado a:**

**Miguel Tobar**

**Calidad de software.**

**Pasto.**

* **Caso de estudio 1.**

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

**Requisitos funcionales.**

**RQF001:** Los pacientes podrán registrarse en el sistema.

**RQF002:** Los pacientes podrán iniciar sesión.

**RQF003:** Los pacientes podrán reservar una cita médica.

**RQF004:** Los médicos organizaran su agenda.

**RQF005:** El administrador podrá controlar usuarios, citas y reportes del sistema.

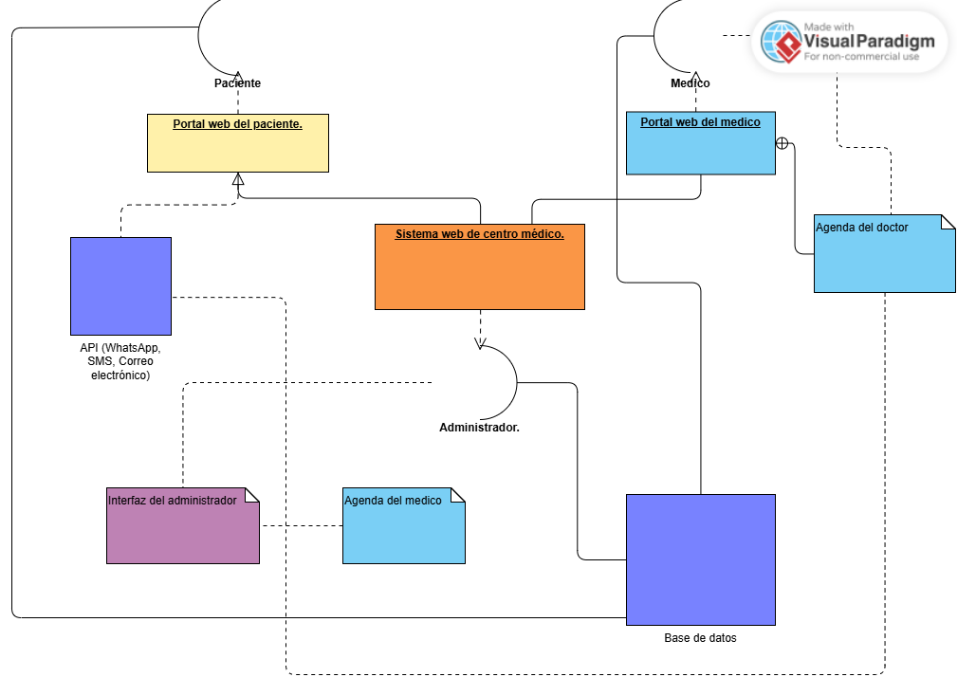
**Componentes.**

|  |  |
| --- | --- |
| Función | Componente |
| Paciente puede ver si su médico tiene espacio para atenderlo. | Agenda médica. |
| Recordar al paciente un día antes de la cita. | Notificaciones. |
| Permitir que el medico escoja los días de trabajo. | Médicos y agenda. |
| El administrador puede sacar un reporte del sistema y médicos más solicitados. | Administrador. |
| Verificar que el paciente no agende más de dos citas para un día. | Citas. |
| Guardar el historial de citas pasadas del paciente. | Pacientes. |

**Interfaces.**

1. Página web para que ingresen pacientes, médicos y el administrador.
2. Base de datos para guardar reservas, cancelar y ver agenda de médicos y pacientes.
3. Mensajes para recordar a los pacientes (SMS, WhatsApp, o correo electrónico)

Este caso de estudio sirve para modularizarlo porque hablamos de un ente de salud y como ingeniero pienso que eventualmente todo junto sería muy enredado y en caso de presentarse dificultades, implicaría gastos innecesarios que se podrían evitar si estuviese modularizado. Por ejemplo, si el módulo de citas falla, el módulo de reportes no se vería afectado y se podría solucionar el problema de manera más rápida y sin dañar las otras partes, lo que no solo supone una ayuda económica al centro médico sino también que ahorramos tiempo y dinero para nosotros como programadores.

****

* **Caso de estudio 2.**

**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.

**Requerimientos funcionales.**

**RQF001:** Los usuarios pueden registrarse, buscar restaurantes y pedir comida.

**RQF002:** El restaurante administra su menú y confirman pedidos.

**RQF003:** Los repartidores reciben pedidos y hacen entregas.

**RQF004:** Pagos online para confirmar pedidos.

**RQF005:** Notificaciones en tiempo real.

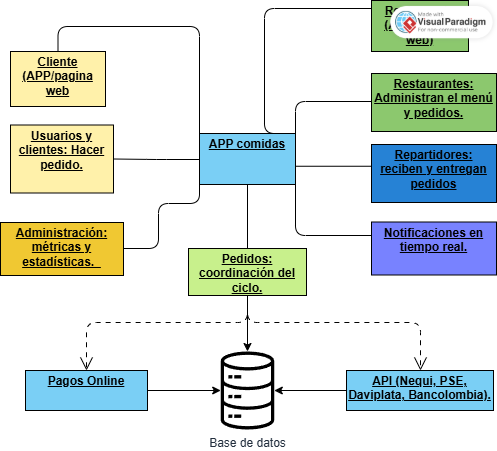
**Componentes.**

|  |  |
| --- | --- |
| Función. | Componente. |
| Registro de clientes, restaurantes y repartidores. | Usuarios y seguridad. |
| Hacer pedido. | Clientes. |
| Administrar menú y aceptar pedido. | Restaurantes. |
| Recibir pedido y ver ruta. | Repartidores. |
| Procesar cobro en línea. | Pagos. |
| Coordinar todo el ciclo del pedido. | Pedidos. |
| Notificar estado del pedido. | Notificaciones. |
| Ver métricas y estadísticas. | Administración. |

**Interfaces.**

1. APP de celular o página web para que clientes, restaurantes y repartidores la usen.
2. Base de datos que almacene pedidos, menús y entregas.
3. API para conectarse a servicio de pagos en línea (Nequi, PSE, Daviplata, Bancolombia)
4. Google maps para localizar al repartidor o buscar dirección.

Este caso de estudio es un muy buen ejemplo de modularizacion porque hay 3 tipos de actores, el repartidor, cliente y restaurantes. Cada uno cumple una función distinta, pero todos se conectan a pedidos. Entonces si algún modulo se cae se lo remedia, pero los pedidos aún siguen funcionando lo que garantiza un trabajo constante.



* **Caso de estudio 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.

**Requerimientos funcionales.**

**RQF001:** La biblioteca podrá manejar usuarios (Estudiantes, profesores, bibliotecarios)

**RQF002:** Los estudiantes podrán obtener libros.

**RQF003:** Los profesores podrán acceder a los libros.

**RQF004:** La biblioteca tendrá un catálogo de libros.

**RQF005:** La biblioteca podrá buscar libros.

**RQF006:** La biblioteca podrá hacer registro de libros.

**RQF007:** La biblioteca puede ver la edición de los libros.

**RQF008:** La biblioteca puede tramitar prestamos y devoluciones de libros.

**RQF009:** El sistema permitirá ver reportes (Quien debe libros, multas y estadísticas)

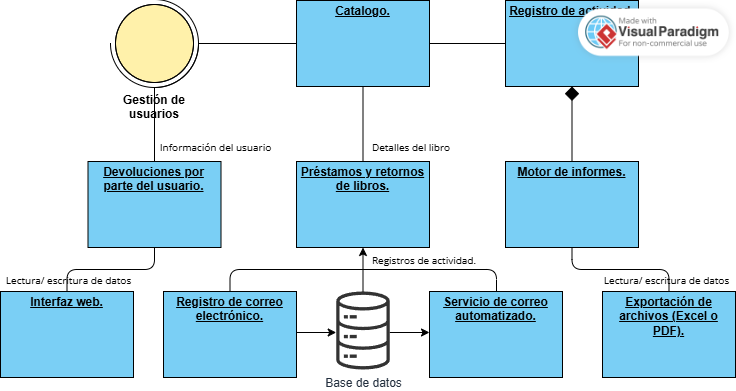
**Componentes.**

|  |  |
| --- | --- |
| Función. | Componente. |
| Avisar al estudiante cuando un libro reservado ya está disponible. | Reportes. |
| Permitir renovar un libro en línea sin ir a la biblioteca. | Préstamos y devoluciones. |
| Mostrar al bibliotecario los libros más reportados y usados. | Catalogo. |
| Controlar que estudiantes no superen el máximo de 5 libros prestados por día. | Usuarios. |
| Generar un ranking de los libros más leídos del mes. | Reportes. |

**Interfaces.**

1. Los usuarios entran desde una pagina web sencilla de la biblioteca.
2. Se implementa una base de datos para registrar prestamos, devoluciones y el catálogo de libros.
3. Los reportes pueden dar en Excel o PDF, dado que son formatos que pueden ser visibles para todos.
4. Se puede mandar un correo electrónico automatizado, donde se recuerde que es hora de devolver el libro, o si hay disponibilidad del libro.

Lo que se implemento fue separar cada parte en componentes que tengan una sola función clara. Así es más fácil de mantener y de mejorar. Este sistema estaba hecho en un solo bloque y eso lo hacía difícil de mantener. Entonces lo que se hizo fue dividirlo en usuarios, catalogo, prestamos reportes y base de datos. Así cada uno cumple una función específica, es más fácil de actualizar y no daña el sistema completo. La idea es mantener buenos estándares con eso me refiero a que sea mantenible y que pueda mejorarse en oportunidades futuras sin complicaciones.

****