Evaluación de Arquitecturas para la Solución de Gestión Médica.

1. Introducción:

Con el objetivo de mejorar la administración del centro médico, se propone evaluar dos enfoques arquitectónicos: el Modelo-Vista-Controlador (MVC) y la Arquitectura de Microservicios. Buscamos identificar la opción más adecuada que permita una gestión eficaz de la agenda médica, la recaudación de pagos, la emisión de comprobantes y la interacción con pacientes.

Este documento presenta una evaluación detallada de ambas arquitecturas, considerando atributos clave como escalabilidad, mantenibilidad y fiabilidad. Al analizar cada enfoque desde estos puntos, se busca determinar la mejor solución para optimizar la gestión médica, garantizando un sistema confiable y eficiente.

2. Arquitecturas Evaluadas

2.1. Modelo-Vista-Controlador (MVC):
Es un patrón de arquitectura de software que divide una aplicación en tres componentes principales: Modelo, Vista y Controlador.

Modelo: Representa los datos y la lógica de negocio de la aplicación.
Vista: Se encarga de la presentación de la información al usuario final.
Controlador: Actúa como intermediario entre el Modelo y la Vista, gestionando las interacciones del usuario y actualizando el Modelo según sea necesario.

La arquitectura MVC proporciona una separación clara de responsabilidades, lo que facilita el desarrollo, la mantenibilidad y la escalabilidad de la aplicación al organizar el código en capas distintas y modulares.

2.2 Arquitectura de Microservicios:
Es un patrón de arquitectura que se crea con componentes independientes que ejecutan cada proceso de la aplicación como un servicio. Estos servicios se comunican a través de una interfaz bien definida mediante API ligeras. Los servicios se crean para las capacidades empresariales y cada servicio desempeña una sola función. Debido a que se ejecutan de forma independiente, cada servicio se puede actualizar, implementar y escalar para satisfacer la demanda de funciones específicas de una aplicación.

3. Atributos de Calidad Seleccionados

- Escalabilidad
- Mantenibilidad
- Disponibilidad

4. Ponderación de Atributos de Calidad

- **Escalabilidad**: Ponderación alta= 0.27

  Crecimiento Horizontal (0.33):
  Seleccionamos el crecimiento horizontal ya que tiene la capacidad del sistema para escalar agregando más instancias de recursos de hardware o software. Se ha ponderado en 0.33 dentro del atributo de Escalabilidad debido a su importancia crítica para permitir que el sistema se expanda añadiendo más recursos según la demanda.

  Manejo Eficiencia de Datos (0.33):
  El manejo de los datos es esencial para permitir un buen rendimiento del sistema a medida que se incrementa la cantidad de información procesada y datos. Le agregamos una ponderación de 0.33, ya que refleja su importancia para asegurar que el sistema pueda manejar grandes volúmenes de datos de manera eficiente sin bajar la calidad del sistema.

  Adaptabilidad de Nuevas Funcionalidades (0.34):
  La adaptabilidad de nuevas funcionalidades ya que representa la capacidad del sistema para incorporar nuevas características o requisitos sin interrumpir el

sistema actual. Le dimos una ponderación de 0.34, ya que en si nuestro sistema es básico. se le podrían agregar nuevas funcionalidades y no afectar al sistema actual.

- **Mantenibilidad**: Ponderación media = 0.25

  Modularidad del Código (0.35):
  Le asignamos una ponderación de 0.35 porque es super importante que el sistema esté organizado en partes separadas y claras. Esto facilita hacer cambios sin afectar otras partes.

  Documentación Clara (0.30):
  Tiene una ponderación de 0.30 porque es importante tener instrucciones claras sobre cómo funciona el sistema. Ya que se podrían incorporar nuevos desarrolladores al desarrollo del sistema y para los desarrolladores actuales, les sea más fácil entender el código.

  Capacidad de Ser Modificado (0.34):
  Recibió una ponderación de 0.34 porque es esencial que el sistema pueda ser cambiado fácilmente y el sistema debe permitir cambios sin romper su funcionamiento básico.

- **Disponibilidad**: Ponderación alta = 0.4

  Monitoreo Continuo (0.33):
  Se asignó una ponderación de 0.33 porque es crucial supervisar constantemente el sistema para detectar problemas rápidamente.

Respaldo de Datos Regular (0.33):
Tiene la misma ponderación de 0.33 porque hacer copias de seguridad de los datos de manera regular es esencial para evitar perder información valiosa.

Plan de Continuidad del Negocio (0.34):
Recibió una ponderación ligeramente más alta de 0.34 debido a la importancia de tener un plan bien pensado para que los médicos puedan seguir trabajando si ocurre algún problema grave.

5. Evaluación de las Alternativas de Arquitectura

Escalabilidad

- Crecimiento Horizontal (0.33):

Le asignamos un valor ponderado de 2 en Microservicios y 1 en MVC. Este valor refleja la capacidad de los microservicios para expandirse horizontalmente (escalar fácilmente añadiendo más recursos) en comparación con el MVC, donde esa expansión es más limitada.

- Manejo Eficiencia de Datos (0.33):

Le asignamos un valor de 2 para Microservicios y 2 en MVC. Ambos patrones obtuvieron una calificación similar, lo que indica que ambos son igualmente eficientes en la gestión de grandes volúmenes de datos.

- Adaptabilidad de Nuevas Funcionalidades (0.34):

Se dio un valor de 2 en Microservicios y 3 en MVC. Los Microservicios se calificaron con un valor ligeramente menor que el MVC, indicando una capacidad levemente menor para adaptarse rápidamente a nuevas funcionalidades en comparación con el MVC.

Mantenibilidad

- Modularidad del Código (0.35):

Se asignó un valor de 2 en Microservicios y 3 en MVC. Indica que los Microservicios tienen una puntuación más baja en términos de modularidad en comparación con el MVC, lo que sugiere que el MVC es más modular y separa mejor las diferentes partes del código.

- Documentación Clara (0.30):

Ambos enfoques obtuvieron una calificación de 2, lo que significa que ambos sistemas ofrecen un nivel similar de documentación clara, facilitando entender cómo funcionan y se desarrollan.

- Capacidad de Ser Modificado (0.34):

Microservicios obtuvo un 2 y MVC un 3. Esto sugiere que el MVC es un poco más flexible en términos de permitir cambios sin afectar otras partes del sistema en comparación con los Microservicios.

Disponibilidad

- Monitoreo Continuo (0.33):

Tanto Microservicios como MVC recibieron una puntuación de 2, indicando que ambos sistemas tienen una capacidad similar para monitorear continuamente el sistema.

- Respaldo de Datos Regular (0.33):

Ambos enfoques obtuvieron una puntuación de 2, lo que sugiere que ambos proporcionan respaldo de datos regular de manera similar.

- Plan de Continuidad del Negocio (0.34):

Microservicios obtuvieron una calificación de 1 y MVC un 2. Esto indica que el MVC tiene un plan de continuidad del negocio más elaborado y detallado en comparación con los Microservicios

6. La Mejor Arquitectura

Nosotros elegimos el patrón de arquitectura MVC debido a diferentes razones:

- **Detalles del Caso y Requisitos**

Construcción de la API en Flask:
El Modelo-Vista-Controlador (MVC) proporciona una estructura clara y modular, permitiendo una fácil integración con nuestra API construida en Flask. La separación de capas en el MVC facilita el enlace de la lógica del negocio con la API para una gestión de datos más eficiente.

- **Desarrollo de la Página Web en Django**:

MVC se ajusta bien con el enfoque de Django. La separación de responsabilidades en MVC se ajusta a la estructura de Django, permitiendo una integración más sencilla de la lógica de negocio con la interfaz de usuario proporcionada por Django.

- **Base de Datos en Elephant SQL**:

MVC ofrece una separación clara de la capa de datos (Modelo), que facilita la integración con una base de datos como Elephant SQL. La capacidad de definir modelos de datos independientes en MVC se alinea con el uso de una base de datos externa.