Documento de diseño de software

1. Introducción

1.1. Propósito

El propósito de este documento consiste en especificar el diseño previo a la implementación del sistema de gestión y aforo por COVID-19, mediante la representación de la arquitectura, objetivos, análisis de la base de datos a usar, vista de casos de uso, vista lógica, vista de procesos y vista del despliegue del proyecto final.

1.2. Alcance

En cuanto al alcance del proyecto, se procurará usar una arquitectura la cual tenga un buen soporte y admisión de cambios en los requisitos. Así mismo, se procura que divida los distintos elementos referentes al proyecto para una fácil implementación que pueda ser dividida entre los integrantes del grupo de trabajo, facilitando también su escalabilidad debido a la metodología en uso.

2. Representación de la arquitectura.

Para escoger la arquitectura de software que servirá de base para desarrollar el sistema de gestión y aforo por COVID-19, se utilizó un conjunto de atributos de calidad importantes para el sistema para calificar algunas arquitecturas de software en una escala del 1 al 5, los cuales son:

- Rendimiento
- Escalabilidad
- Disponibilidad
- Modificabilidad

Con base en este criterio, se realizó un breve análisis a 4 tipos de arquitecturas de software candidatas a ser usadas para la implementación del proyecto, las cuales fueron

2.1. Patrón Cliente-servidor: Este patrón consiste en dos partes, un servidor y múltiples clientes. El servidor es la parte del desarrollo donde se construyen los servicios y se realiza toda la lógica del negocio, y por su parte la serie de clientes creados son quienes realizan peticiones al servidor para obtener información.

2.2. Arquitectura basada en microservicios: Funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma, proporcionando una funcionalidad de negocio completa. Los microservicios se comunican entre sí a través de APIs y cuentan con sistemas de almacenamiento propios, lo que evita la sobrecarga y caída de la aplicación.

2.3. Arquitectura monolítica: En este patrón el software se estructura de forma que todos los aspectos funcionales quedan acoplados y sujetos en un mismo programa. En este tipo de sistema, toda la información está alojada en un servidor, por lo que no hay separación entre módulos y las diferentes partes de un programa están muy acopladas, lo que genera una baja escalabilidad en el sistema.

2.4. Arquitectura por capas: La arquitectura basada en capas se enfoca en la distribución de roles y responsabilidades de forma jerárquica proveyendo una forma muy efectiva de separación de responsabilidades. Los componentes de cada capa se comunican con otros componentes en otras capas a través de interfaces definidas.

Teniendo en cuenta las características mencionadas anteriormente para cada arquitectura, los resultados de cada una fueron los siguientes:

Arquitectura	**Rendimiento**	**Escalabilidad**	**Disponibilidad**	**Modificabilidad**	**Total**
Cliente-servidor	2	4	2	3	**11**
Microservicios	3	4	2	3	**12**
Monolítica	5	3	2	1	**11**
Por capas	4	3	3	4	**14**

Dados los puntajes obtenidos, se optará por usar la “arquitectura por capas”, puesto que se pueden manejar y desarrollar las distintas capas de manera independiente, por lo que mejora su manejo a nivel individual y, así mismo, su susceptibilidad a cambios. Se requiere una capa de servidor en donde se procesarán las distintas solicitudes, ya sea para la aplicación de filtros a las tablas de datos solicitadas o a validaciones tales como el inicio de sesión de cualquier tipo de usuario.
El modelo gráfico de la arquitectura se encuentra adjunto en el portafolio.

3. *Base de datos*

Similar al apartado de la arquitectura de software, para escoger la base de datos que servirá para implementar la capa de datos dentro de la arquitectura, se utilizó un conjunto de atributos de calidad importantes para el sistema para calificar algunos tipos de base de datos en una escala del 1 al 5, los cuales son:

- Rendimiento
- Escalabilidad
- Disponibilidad

Con base en este criterio, se realizó un breve análisis a 4 tipos de bases de datos candidatas a ser usadas para la implementación del proyecto, las cuales fueron

3.1. ***Relacional:*** Las bases de datos relacionales se utilizan para hacer seguimiento de los inventarios, procesar transacciones de comercio electrónico, administrar grandes cantidades de información de clientes de misión crítica y mucho más. Se puede considerar una base de datos relacional para cualquier necesidad de información en la que los puntos de datos se relacionen entre sí y se deban administrar de una manera segura, consistente y basada en reglas.

3.2. ***Clave-valor:*** Las bases de datos clave-valor son altamente divisibles y permiten escalado horizontal a escalas que otros tipos de bases de datos no pueden alcanzar. Los casos de uso como juegos, tecnología publicitaria e IoT se prestan particularmente bien con el modelo de datos clave-valor.

3.3. ***Gráficos:*** El propósito de una base de datos de gráficos es facilitar la creación y la ejecución de aplicaciones que funcionan con conjuntos de datos altamente conectados. Los casos de uso típicos para una base de datos de gráficos incluyen redes sociales, motores de recomendaciones, detección de fraude y gráficos de conocimiento.

3.4. ***Documentos:*** En el código de aplicación, los datos se representan a menudo como un objeto o un documento de tipo JSON porque es un modelo de datos eficiente e intuitivo para los desarrolladores. Las bases de datos de documentos facilitan a los desarrolladores el almacenamiento y la consulta de datos en una base de datos mediante el uso del mismo formato de modelo de documento que emplean en el código de aplicación. La naturaleza flexible, semiestructurada y jerárquica de los documentos y las bases de datos de documentos permite que evolucionen según las necesidades de las aplicaciones. El modelo de documentos funciona bien con catálogos, perfiles de usuario y sistemas de administración de contenido en los que cada documento es único y evoluciona con el tiempo.

Base de datos	**Rendimiento**	**Escalabilidad**	**Disponibilidad**	**Total**
Relacional	3	5	1	**7**
Clave-valor	2	5	4	**11**
Gráficos	4	4	3	**11**
Documentos	5	5	4	**14**

Dados los puntajes obtenidos, se optará por usar una base de datos basada en documentos, puesto que una base de datos de documentos es una excelente opción para aplicaciones de administración de contenido como el registro de los datos de

cada uno de los tipos de usuario del sistema de gestión y aforo por COVID-19, ya que se puede escalar fácilmente y estos datos evolucionan o se transforman con el tiempo, tarea que el modelo de documentos facilita. Como sistema gestor de base de datos, se utilizará MongoDB.

La representación gráfica de la implementación de la lógica del sistema en MongoDB se encuentra en el portafolio del proyecto (Vista de base de datos.png).

4. Vista de casos de uso

Los casos de uso documentados sirven para describir de forma práctica el comportamiento del sistema.
Los casos de uso fueron divididos en función de cuatro tipos de usuario:

- Administrador
- Comercio
- Civil
- Entidad sanitaria

En términos generales, los casos de uso del administrador consisten en la administración de la información de los comercios, civiles y entidades sanitarias, la gestión de sus contraseñas y las distintas solicitudes que pueden generar en el sistema.

Los casos de uso de Comercio describen la revisión de registros y el registro de los civiles al momento de las visitas.

Los civiles tienen la capacidad de modificar los datos con los cuales fueron registrados en el sistema, recuperar su contraseña, y revisar los locales que han visitado.

Por otro lado, las entidades sanitarias pueden informar al sistema de usuarios infectados con COVID-19, así como de los usuarios pendientes de los resultados de sus exámenes y filtrar el listado de personas afiliadas a ellas.

La documentación los casos de uso individuales se encuentran adjuntos en el portafolio del proyecto.

5. Vista lógica

El diagrama de clases describe la estructura interna de cada tipo de usuario del sistema y de los registros asociados a los datos que debe manejar.
Usamos una superclase para derivar los cuatro tipos de usuario del sistema de ella. Adicionalmente, usamos clases específicas para las visitas, los historiales de pruebas y las modificaciones de la información de los usuarios que deben gestionar los administradores. Existen tres clases para determinar la ubicación por departamento, municipio y barrio. Por último, tenemos una clase interfaz para validar el registro de los nuevos usuarios en el sistema y el inicio de sesión.

Los diagramas de secuencia representan el flujo de los datos y las instrucciones dentro del sistema a través de la interfaz, el gestor de usuarios, el gestor de servicios y la base de datos.

Los diagramas de clases y el diagrama de secuencia del sistema se encuentran adjuntos en el portafolio del proyecto.

6. Vista de procesos

Mediante los diagramas de actividades se representa la lógica bajo la que se construye el sistema y los pasos realizados en la mayoría de los casos de uso de los usuarios. Por ejemplo, el flujo que sigue el inicio de sesión de un civil en caso de que la contraseña sea correcta o incorrecta.

El diagrama de actividades se encuentra adjunto en el portafolio del proyecto.

7. Vista de despliegue

El diagrama de despliegue representa la distribución física que tiene el sistema. El software se encuentra implementado en el sistema operativo Python y está desplegado en un servidor de Amazon a través de *Amazon Web Service*. Los usuarios pueden acceder a la página web a través del enlace *betaplay.shop.* La base de datos, como se indicó, está montada en un servidor de MongoDB en dos instancias para garantizar la disponibilidad del sistema.

El diagrama de despliegue se encuentra adjunto en el portafolio del proyecto.

8. Referencias

[1]"Ventajas y desventajas de tener tu propio servidor local - 3 mentes", 3 mentes, 2020. [Online]. Available: https://www.3mentes.com/index.php/ventajas-desventajas-propio-servidor-local/. [Accessed: 11- Nov- 2020].

[2]"Arquitectura de software", Es.wikipedia.org, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Arquitectura_de_software. [Accessed: 11- Nov- 2020].

[3]"Arquitectura de microservicios", Es.wikipedia.org, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Arquitectura_de_microservicios. [Accessed: 11- Nov- 2020].

[4]"Modelo–vista–controlador", Es.wikipedia.org, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador. [Accessed: 11- Nov- 2020].