DOCUMENTO DE DISEÑO DE SOFTWARE

Pontificia Universidad Javeriana Cali

Facultad de Ingenieria y Ciencias

Realizado por : 1. Camilo Gutierrez (8935404)
2. Edixon Salas (8934749)
3. Francisco Suarez (8939317)

Enviado a : Gustavo Salazar
Profesor

November 11, 2020

Contents

1 Introducción

1.1 Propósito

El propósito del presente documento es definir la arquitectura y modelo de bases de datos destinados a ser usados en el sistema, además del ponderamiento de cada una de las opciones y un entendimiento de los motivos por los cuales estos fueron elegidos. También, Busca explicar el funcionamiento del sistema de tal manera de que se tenga un entendimiento de su funcionalidad.

1.2 Alcance del proyecto

Se realiza un conjunto de actividades con las cuales se buscan definir criterios válidos para la toma de decisiones acerca de cuál es la arquitectura y la base de datos que nos conviene usar para el sistema "Sistema Web para la Gestión de Accesos y Aforo por COVID-19", teniendo en cuenta los requisitos no funcionales ya establecidos, 3 tipos de arquitecturas (Microservicios, n-capas y arquitectura orientada a servicios), 5 atributos de calidad para la arquitectura(Rendimiento, Escalabilidad, Disponibilidad, Modificabilidad y Desplegabilidad), 4 modelos de bases de datos(Relacional, Llave-valor, orientada a columnas y orientada a documentos) y 3 criterios de selección para el modelo de base de datos(Rendimiento, Escalabilidad Disponibilidad), para así, obtener la mejor combinación entre arquitectura y modelo de base de datos para cumplir con los requisitos no funcionales del proyecto. Por otro lado, haciendo uso de un conjunto de herramientas que nos brinda UML y otros diagramas, se mostrará el funcionamiento de sistema

1.3 Definiciones, Acrónimos y Abreviaciones

- **Arquitectura de software:** La arquitectura de software es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software[16].
- **Microservicios:** Los microservicios son tanto un estilo de arquitectura como un modo de programar software[15].
- **Promedio ponderado:** El promedio ponderado es una forma un poco más compleja de calcular la media y es usada para realizar comparaciones[18].
- **Requerimientos no funcionales:** son la representación de las características generales y restricciones de la aplicación o sistema que se esté desarrollando[17].

1.4 Referencias

[1] Lose, 2020. Microservices Beyond The Hype: What You Gain And What You Lose. [online] Insights.sei.cmu.edu. Available at: ¡https://insights.sei.cmu.edu/sei_-blog/2015/11/microservices-beyond-the-hype-what-you-gain-and-what-you-lose.html¿ [Accessed 11 November 2020].

[2]"Software Architecture Patterns", O'Reilly Online Learning, 2020. [Online]. Available: https://www.oreilly.com/library/view/software-architecture-patterns/

9781491971437/ch01.html. [Accessed: 11- Nov- 2020].

[3]2020. [Online]. Available: https://www.redhat.com/en/topics/cloud-native-apps/what-is-an-application-architecture. [Accessed: 11- Nov- 2020].

[4]Kumar, N., 2020. Enterprise Benefits On Service Oriented Architecture - SOA. [online] Java Code Geeks. Available at: ¡https://www.javacodegeeks.com/2013/03/enterprise-benefits-on-service-oriented-architecture-soa.html¿ [Accessed 11 November 2020].

[5]N. Staff, "5 Sure Signs It's Time to Give Up Your Relational Database - Neo4j Graph Database Platform", Neo4j Graph Database Platform, 2020. [Online]. Available: https://neo4j.com/blog/five-signs-to-give-up-relational-database/. [Accessed: 11- Nov- 2020].

[6]M. Allen, "Relational Databases Are Not Designed For Scale — MarkLogic", MarkLogic, 2020. [Online]. Available: https://www.marklogic.com/blog/relational-databases-scale/. [Accessed: 11- Nov- 2020].

[7]"relational-databases", Ibm.com, 2020. [Online]. Available: https://www.ibm.com/cloud/learn/relational-databases. [Accessed: 11- Nov- 2020].

[8]"3 Benefits of Key-Value Stores for Unstructured Data", Medium, 2020. [Online]. Available: https://medium.com/@stellustechnologies/3-benefits-of-key-value-stores-for-unstructured-data-9267e7ab42af. [Accessed: 11- Nov- 2020].

[9]GroundAI. 2020. Cyclone: High Availability For Persistent Key Value Stores. [online] Available at: ¡https://www.groundai.com/project/cyclone-high-availability-for-persistent-key-value-stores/1¿ [Accessed 11 November 2020].

[10]Database.guide. 2020. What Is A Column Store Database? — Database.Guide. [online] Available at: ¡https://database.guide/what-is-a-column-store-database/¿ [Accessed 11 November 2020].

[11]C. System, "[PDF] Column-Oriented Databases to Gain High Performance for Data Warehouse System - Free Download PDF", Silo.tips, 2020. [Online]. Available: https://silo.tips/download/column-oriented-databases-to-gain-high-performance-for-data-warehouse-system. [Accessed: 11- Nov- 2020].

[12]"Fundamentals of Document Databases - DATAVERSITY", DATAVERSITY, 2020. [Online]. Available: https://www.dataversity.net/fundamentals-of-document-databases/. [Accessed: 11- Nov- 2020].

[13]"Bases de datos documentales — Qué son, marcas y usos", GraphEverywhere, 2020. [Online]. Available: https://www.grapheverywhere.com/bases-de-datos-documentales/. [Accessed: 11- Nov- 2020].

[14]"Software Architecture Patterns", O'Reilly Online Learning, 2020. [Online]. Available: https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html. [Accessed: 11- Nov- 2020].

[15]2020. [Online]. Available: https://www.redhat.com/es/topics/microservices. [Accessed: 11- Nov- 2020].

[16]"Arquitectura de software - EcuRed", Ecured.cu, 2020. [Online]. Available: https://www.ecured.cu/Arquitectura_de_software. [Accessed: 11- Nov- 2020].

[17]"Requerimientos no funcionales: Ejemplos", Pmoinformatica.com, 2020. [Online]. Available: http://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html. [Accessed: 11- Nov- 2020].

[18]"Document", Cca.org.mx, 2020. [Online]. Available: http://www.cca.org.mx/cca/cursos/estadisti ponderado.htm. [Accessed: 11- Nov- 2020].

2 Representación Arquitectónica

2.1 Arquitectura del sistema

En esta sección, se muestran las arquitecturas de software candidatas y a través de ciertos criterios, se realizó la selección de una de ellas para la implementación del Sistema Web para la Gestión de Accesos y Aforo por COVID-19. La selección se hizo a través de un promedio ponderado, en el cual se le dio a cada atributo de calidad un puntaje entre 1 y 14, siendo 14 la cantidad total de puntos que se debe repartir entre los diferentes atributos de calidad, es decir, la suma del puntaje de todos los atributos de calidad debe dar 14. Posteriormente, se le da un puntaje, entre 0 y 5, a cada arquitectura candidata con respecto a los atributos de calidad. Es seleccionada la arquitectura que mayor puntaje tenga. A continuación, se listan las arquitecturas candidatas:

- Microservicios.
- n-capas.
- Arquitectura orientada a servicios.

A continuación, se listan los atributos de calidad a considerar y el puntaje de cada uno entre paréntesis, son los siguientes:

- Rendimiento (3).
- Escalabilidad (4).
- Disponibilidad (3).
- Modificabilidad (2).
- Desplegabilidad (2).

Para cada arquitectura candidata, se hizo una investigación sobre su aporte a los atributos de calidad seleccionados. A continuación se listan las arquitecturas candidatas y sus características con respecto a los atributos de calidad. Además, se le da el puntaje que los desarrolladores consideraron apropiados.

ARQUITECTURA MICROSERVICIOS:

Rendimiento: Es probable que los servicios necesiten comunicarse a través de la red, mientras que los servicios dentro del monolito pueden beneficiarse de llamados locales. Además, si el microservicio usa el descubrimiento dinámico, la búsqueda del registro es una sobrecarga de rendimiento. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 2.[1]

Escalabilidad: Cada microservicio puede ser escalado de manera individual e independiente. Las características de implementación hacen que los microservicios se adapten perfectamente a la elasticidad de la nube. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 5.[1]

Disponibilidad: La implementación de una nueva versión de un microservicio requiere poco tiempo de inactividad, mientras que la implementación de una nueva versión de un servicio en una arquitectura monolítica requiere un reinicio de todo el sistema. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 5. [1]

Modificabilidad: Flexibilidad para utilizar nuevos frameworks, librerías, fuentes de datos y otros recursos. Además, los microservicios son componentes modulares de acoplamiento flexible a los que solo se puede acceder a través de sus contratos y, por lo tanto, son menos propensos a convertirse en una gran bola de barro. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 5.[1]

Desplegabilidad: Con esta arquitectura se tiene más facilidad para implementar nuevas versiones de un servicio debido a ciclos de compilación+prueba+implementación más cortos. Además, es flexible para emplear configuraciones de seguridad, replicación, persistencia y monitoreo específicas del servicio. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 5.[1]

ARQUITECTURA N-CAPAS:

Rendimiento: Aunque algunas implementaciones de esta arquitectura tienen buen rendimiento, el patrón no se presta para desarrollar aplicaciones de alto rendimiento, debido a la ineficiencia que se tiene al tener que moverse a través de múltiples capas de la arquitectura para completar un request de negocio. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 2[14].

Escalabilidad: Ya que existen tendencias a implementar esta arquitectura de manera monolítica y con alto acoplamiento, aplicaciones construidas con esta arquitectura son, por lo general, difíciles de escalar. Se puede hacer un escalamiento al dividir las capas en despliegues separados o replicando la aplicación completa en múltiples nodos, pero la granularidad es demasiado amplia y escalar puede resultar muy costoso. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 2[14].

Disponibilidad: Hacer cambios a un componente puede significar un nuevo despliegue de toda la aplicación (o una gran parte de esta), resultando en despliegues que necesitan se planeados, programados y ejecutados fuera de horario de trabajo o en fines de semana, disminuyendo la disponibilidad del sistema. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 3[14].

Modificabilidad: Realizar cambios en esta arquitectura puede ser un trabajo engorroso y consumir mucho tiempo, debido a la naturaleza monolítica de muchas implementaciones de esta arquitectura, así como a sus componentes altamente acoplados. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 3[14].

Desplegabilidad: Dependiendo de cómo se implemente esta arquitectura, el despliegue puede ser un problema, y más particularmente para aplicaciones grandes. Y de esta manera, este patrón no se presta fácilmente para realizar un pipeline de continous delivery. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 3[14].

ARQUITECTURA ORIENTADA A SERVICIOS:

Rendimiento: Esta arquitectura implica computación distribuida. Los servicios y los usuarios se encuentran ubicados en máquinas diferentes. La necesidad de comunicación sobre la red hace que se aumente el tiempo de respuesta. Además, la red usualmente no garantiza una latencia determinada, haciendo que esta arquitectura sea mala para aplicaciones y sistemas en tiempo real, en las cuales el tiempo es un requisito estricto. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 2[2].

Escalabilidad: Esta arquitectura posibilita la ejecución de los servicios en diferentes lenguajes de programación, servicios y plataformas, lo cual aumenta la escalabilidad de manera considerable. Además, utiliza un protocolo de comunicación estandarizado para que las empresas puedan disminuir la interacción entre los clientes y los servicios, lo cual permite ampliar las aplicaciones con menos presiones e inconvenientes. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 5[3].

Disponibilidad: Como ya se sabe, la redundancia es un aspecto importante a tener en cuenta, cuando que nuestro sistema tenga una alta disponibilidad. En esta arquitectura, se puede conseguir esta redundancia introduciendo elementos redundantes a través de clústering. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 4[4].

Modificabilidad: Los servicios son autónomos, modulares y se puede acceder a ellos a través de interfaces cohesivas. Estas características contribuyen a implementaciones con bajo acoplamiento donde hay pocas dependencias. A raíz de esto, el costo de modificar la implementación de los servicios se reduce y la modificabilidad general del sistema

aumenta. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 5[2].

Desplegabilidad: Se proporciona transparencia de la ubicación. Los usuarios no necesariamente conocen la ubicación del servicio hasta que lo buscan en el registro. Por lo tanto, un servicio desplegado se puede mover de un lugar a otro sin afectar a los usuarios. Esta característica permite la implementación de servicios en múltiples ubicaciones, que pueden combinarse con una estrategia de balance de cargas para mejorar el throughput total y la disponibilidad del sistema. Por lo tanto, con respecto a este atributo de calidad, se le dará un puntaje de 5[2].

CONCLUSIÓN:

Al realizar un promedio ponderado, como el que se explicó anteriormente, con los datos de los puntos asignados a cada arquitectura de software, llegamos a la conclusión de que la opción más acertada es la arquitectura de microservicios. Este promedio ponderado se puede ver en el archivo llamado "Comparación de arquitecturas".

2.2 Base de datos del sistema

En esta sección se muestran los modelos de bases de datos candidatas y, a través de ciertos criterios, se realizó la selección de una de ellas para la implementación del Sistema Web para la Gestión de Accesos y Aforo por COVID-19. Para escoger la base de datos adecuada, se realizó la misma actividad que para la selección de la arquitectura del sistema. Sin embargo, para esta actividad se tomaron en cuenta 3 atributos de calidad y el promedio ponderado ya no se hará sobre 14 puntos, sino sobre 10, la calificación para cada modelo con respecto al atributo de calidad, se sigue haciendo de 0 a 5. A continuación, se listan los modelos de bases de datos candidatas:

- Relacional.
- Llave-valor.
- Orientada a columnas.
- Orientada a documentos.

A continuación se listan los atributos de calidad a considerar junto con su puntaje entre paréntesis:

- Rendimiento (3).
- Escalabilidad (4).
- Disponibilidad (3).

Para cada modelo de base de datos, se realizó una investigación sobre su aporte a los atributos de calidad seleccionados. A continuación se listan los modelos candidatos y sus características con respecto a los atributos de calidad. Se les da un puntaje, dependiendo de estas características.

MODELO RELACIONAL:

Rendimiento: Los modelos de base de datos relacionales tienen problemas de rendimiento, debido al rápido crecimiento no solo del volumen y la velocidad de los datos, sino también en su variedad, complejidad e interconectividad. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 3[5].

Escalabilidad: Las bases de datos relacionales están diseñadas para ser ejecutadas en un solo servidor, esto para mantener la integridad de los datos y evitar problemas con datos distribuidos. Con este diseño, si el sistema necesita ser escalado, los clientes deben comprar hardware mucho más grande, más complejo y más caro, que tenga mayor poder de procesamiento, memoria y almacenamiento. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 2[6].

Disponibilidad: Debido a la teoría CAP (Consistency, Availability, or Partition Tolerance.), sabemos que cuando un computador se ejecuta en una red, necesita decidir si priorizar resultados consistentes o la disponibilidad. Las bases de datos relacionales sacrifican esa disponibilidad para obtener consistencia de los datos. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 2[7].

LLAVE-VALOR:

Rendimiento: El modelo llave-valor usa una llave para apuntar directamente a la dirección de memoria del dato, dando un rendimiento excepcional y velocidad. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 5[8].

Escalabilidad: El modelo llave-valor, está diseñado para escalar de manera fácil y para manejar grandes cantidades de datos, mientras que al mismo tiempo mantiene una de sus ventajas principales: la velocidad de respuesta. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 5[8].

Disponibilidad: Para garantizar disponibilidad, se necesita mantener y replicar un registro de escritura anticipada, de lo contrario, rendimiento se ve obstaculizado. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 2[9].

ORIENTADA A COLUMNAS:

Rendimiento: Este modelo es muy rápido al cargar y al realizar queries. Una tabla de un millón de datos, puede ser cargada en algunos segundos. Por lo tanto, a este

modelo de bases de datos se le dará un puntaje de 5[10].

Escalabilidad: Las bases de datos basadas en columnas son muy escalables. Están bien implementadas para procesamiento masivo paralelo, el cual involucra tener datos distribuidos en un gran clúster de máquinas. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 5[10].

Disponibilidad: Las bases orientadas a columnas pueden tener alta disponibilidad cuando se hace replicación entre nodos, lo cual al utilizar este modelo es lo más común. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 4[11].

ORIENTADA A DOCUMENTOS:

Rendimiento: Muchas bases de datos orientadas a documentos cuentan con potentes motores de búsqueda con propiedades de indexación. Lo que garantiza que las consultas sean veloces, cortas de tiempo pero de gran eficiencia. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 5[13].

Escalabilidad: Este modelo de bases de datos esta diseñado para ser escalable horizontalmente. Es decir, se consigue escalabilidad, añadiendo más servidores al clúster. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 5[12].

Disponibilidad: Ya que las bases de datos basadas en documentos tienen un esquema flexible, este puede ser modificado sin causar tiempo de inactividad, lo que aumenta la disponibilidad. Por lo tanto, a este modelo de bases de datos se le dará un puntaje de 5[12].

CONCLUSIÓN:

Al realizar un promedio ponderado, como el que se explicó anteriormente, con los datos de los puntos asignados a cada modelo de base de datos, llegamos a la conclusión de que la opción más acertada es implementar el sistema con una base orientada a documentos. Este promedio ponderado se puede ver en el archivo llamado "Comparación de bases de datos". Además, se seleccionó el motor de base de datos MongoDB, debido a que nos brinda ventajas con respecto a los atributos de calidad y de su facilidad de uso nos ayuda a completar el desarrollo en menor tiempo.

3 Metas y Restricciones de la Arquitectura

Una de las características más importantes de la arquitectura basada en microservicios es la escalabilidad, por lo cual, una de las metas con el uso de esta es que se nos haga posible, como programadores, escalar el sistema y encontrar la manera de que se adecúe a la llegada de más usuarios, sin que esto afecte su correcto funcionamiento. Además de la escalabilidad, esta arquitectura tiene grandes ventajas en disponibilidad, modificabilidad y desplegabilidad, por lo cual se espera que al querer hacer modificaciones al sistema su gran capacidad de modificabilidad y desplegabilidad, lo permitan sin que los usuarios se vean afectados, es decir, sin tener que sacar el sistema de producción. La disponibilidad del sistema también fue un factor importante al escoger esta arquitectura, y se espera que el correcto desarrollo de este, permita que los desarrolladores agreguen la redundancia necesaria, para que el sistema esté disponible en la mayor parte del tiempo posible.

Una de las desventajas de esta arquitectura basada en microservicios es el rendimiento, debido a que a medida que el sistema vaya creciendo, la comunicación entre microservicios puede hacerse más compleja y tomar más tiempo del esperado, por esto los programadores deben evaluar qué está causando el bajo rendimiento del sistema, y tomar medidas para solucionar estos problemas.

4 Vista de Casos de Uso

Los diagramas de casos de uso y la formalización de esos casos de uso, se encuentran dentro de la carpeta "Casos de uso".

5 Vista Lógica

La vista lógica del sistema se encuentra en las carpetas "Diagramas de clases" y "Diagrama de Secuencia".

6 Vista de Procesos

La vista de procesos se encuentra en la carpeta "Diagrama de actividades".

7 Vista Física

La vista física del sistema está detallada en el diagrama de despliegue, debido a que en ese diagrama, se pueden ver todos los servicios de la arquitectura, incluidas sus bases de datos y sus APIs. Además, tiene un nivel de detalle que hace que se entienda cada elemento de la arquitectura de manera más profunda.

8 Vista de Despliegue

La vista de despliegue se encuentra en la carpeta "Diagrama de Despliegue".