



**UNIVERSIDAD POLITÉCNICA SALESIANA**  
**SEDE CUENCA**  
**CARRERA DE INGENIERÍA DE SISTEMAS**

“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA RECOMENDADOR BASADO EN  
FILTROS COLABORATIVOS PARA EL ACOPIO Y DISTRIBUCIÓN DE  
PRODUCTOS UTILIZANDO APLICACIONES WEB PROGRESIVAS Y GRAPHQL  
SOBRE DEVOPS PARA PLATAFORMAS BASADAS EN SERVICIOS”

Trabajo de titulación previo a la obtención  
del título de Ingeniero de Sistemas

AUTORES: JORDAN FERNANDO MURILLO VALAREZO  
EDUARDO ISMAEL CASTILLO CARDENAZ  
TUTOR: ING. DIEGO FERNANDO QUISI PERALTA

Cuenca - Ecuador

2022

**CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO  
DE TITULACIÓN**

Nosotros, Jordan Fernando Murillo Valarezo con documento de identificación N° 0706148509 y Eduardo Ismael Castillo Cardenaz con documento de identificación N° 0106711724; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 13 de mayo del 2022

Atentamente,



---

Jordan Fernando Murillo Valarezo

0706148509



---

Eduardo Ismael Castillo Cardenaz

0106711724

## **CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICASALESIANA**

Nosotros, Jordan Fernando Murillo Valarezo con documento de identificación N° 0706148509, y Eduardo Ismael Castillo Cardenaz con documento de identificación N° 0106711724, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto Técnico: “Diseño e implementación de un sistema recomendador basado en filtros colaborativos para el acopio y distribución de productos utilizando aplicaciones web progresivas y GraphQL sobre DevOps para plataformas basadas en servicios”, el cual ha sido desarrollado para optar por el título de: Ingeniero de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos entrega del trabajo en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 13 de mayo del 2022

Atentamente,



---

Jordan Fernando Murillo Valarezo

0706148509



---

Eduardo Ismael Castillo Cardenaz

0106711724

## CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Diego Fernando Quisi Peralta con documento de identificación N° 0104616461, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: “DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA RECOMENDADOR BASADO EN FILTROS COLABORATIVOS PARA EL ACOPIO Y DISTRIBUCIÓN DE PRODUCTOS UTILIZANDO APLICACIONES WEB PROGRESIVAS Y GRAPHQL SOBRE DEVOPS PARA PLATAFORMAS BASADAS EN SERVICIOS”, realizado por Jordan Fernando Murillo Valarezo con documento de identificación N° 0706148509 y por Eduardo Ismael Castillo Cardenaz con documento de identificación N° 0106711724, obteniendo cómo resultado final el trabajo de titulación bajo la opción Proyecto Técnico que cumple con todos los requerimientos estipulados por la Universidad Politécnica Salesiana.

Cuenca, 13 de mayo del 2022

Atentamente,



---

Ing. Diego Fernando Quisi Peralta, Mst.

0104616461

## **DEDICATORIA**

*Este trabajo está dedicado primeramente a Dios, quién me ha dado salud y las fuerzas necesarias para lograr mis objetivos.*

*A mi madre Carmen Valarezo y mi padre Miguel Murillo, quienes que con su esfuerzo, apoyo incondicional y palabras de aliento me ayudaron a luchar por cumplir esta meta tan importante en mi vida, gracias por enseñarme los valores necesarios que me permiten luchar por mis sueños día a día.*

*A cada uno de mis hermanos, por su apoyo brindado en cada etapa de mi vida académica, su cariño, su compañía y todas las acciones que hicieron por mí.*

*A mis familiares, abuelitos, tíos y primos, que han estado para mí haciéndome notar su aprecio y compartiéndome mensajes de perseverancia cuando lo requería.*

*A mis compañeros, por convertir mi etapa universitaria en momentos de alegría, quiero hacerles saber que su amistad me ayudado a comprender lo divertida que es la vida con una buena compañía y la gran ayuda que ha representado que compartan sus conocimientos.*

**Jordan Fernando Murillo Valarezo**

*Dedico este trabajo a mi padre Francisco Castillo por su esfuerzo, apoyo y consejos que me han dado, lo que me ha permitido poder alcanzar esta meta la cual es muy importante en mi vida. Me siento muy afortunado de tenerlo a mi lado apoyándome en cada paso que doy.*

*A mi abuelo Humberto Castillo por ser un gran apoyo, y motivarme a realizar cada paso que he realizado, los cuales me permitieron alcanzar esta meta.*

*A mis abuelas Lastenia Yanza, y Rosa Villa por estar apoyándome en cada momento de diferentes maneras, pero que me ha servido mucho para poder alcanzar esta meta tan importante.*

*A cada uno de mis compañeros quienes me ayudaron con sus conocimientos cuando más lo necesitaba sin pedir nada a cambio.*

**Eduardo Ismael Castillo Cardenaz**

## **AGRADECIMIENTO**

*Agradezco a Dios por permitirme completar una meta más que es culminar con éxito mi carrera, de la misma manera a mis padres, hermanos, familiares y amigos; gracias a ellos por confiar y creer en mis sueños.*

*Así mismo, de manera especial al Mst. Diego Quisi, quién me ha compartido su sabiduría durante el desarrollo de este proyecto, aquel que con su dedicación me transmitió amor por esta carrera, gracias por su motivación constante y las diversas oportunidades brindadas para adquirir habilidades que me ayudarán en el mundo profesional y laboral.*

*Agradezco al Dr. Gabriel León por el sinnúmero de enseñanzas que compartió al incluirme en actividades donde aplicaba los conocimientos impartidos en las clases.*

*Agradezco a los docentes Dr. Gustavo Bravo, Dr. Remigio Hurtado, Mag. María Leguízamo, aquellos que, me apoyaron y presentaron ideas para mejorar el desarrollo de mi formación académica.*

*Finalmente, a todos los docentes de la Universidad Politécnica Salesiana que fueron importantes en mi proceso de enseñanza.*

**Jordan Fernando Murillo Valarezo**

*Quiero agradecer primeramente a Dios por guiarme y permitir que haya cumplido esta meta que es tan importante en mi vida, al Mst. Diego Quisi por poder compartirme sus múltiples conocimientos y poder guiarme en el desarrollo de este trabajo de titulación.*

*Agradezco a mi padre Francisco Castillo por todo su apoyo, cariño y consejos, lo que me ha permitido nunca rendirme y hacer con el mayor esfuerzo cada uno de los pasos para poder culminar esta etapa de mi vida.*

*Agradezco a toda mi familia más cercana quienes han estado presentes y pendientes de mi preocupándose por mí, motivándome a seguir adelante y alcanzar esta meta que es muy importante en mi vida.*

*Finalmente quiero agradecer a todos los docentes que forman parte de la Universidad Politécnica Salesiana que me han compartido sus conocimientos para poder realizar de la mejor manera este trabajo de titulación, y que me ayudaron con cualquier inquietud cuando más me hacía falta.*

**Eduardo Ismael Castillo Cárdenaz**

## **Resumen**

En la actualidad, con el crecimiento de la población los sistemas transaccionales, los sistemas de apoyo a las decisiones y sistemas estratégicos almacenan considerables cantidades de información, cada dato representa antecedentes relevantes de su propietario.

Al representar un crecimiento de grandes magnitudes, dichas entidades empresariales pueden aprovechar este conjunto de información de maneras óptimas, consiguiendo así mejorar sus estrategias de venta y sus procesos de distribución. En consecuencia, extraer eficientemente de información útil sea más difícil. Realizar un filtrado eficiente de esta información conlleva la aplicación de mecanismos efectivos. Una técnica para resolver este problema es el filtrado colaborativo.

En base a esta problemática, el presente proyecto tiene como propósito crear un sistema con recomendación basado en filtros colaborativos para el acopio y distribución de productos, usando tecnologías nuevas como lo son: PWA, GraphQL, Docker y DevOps. Para llevar a cabo esto, se tiene el uso de la metodología SCRUM donde se dividió cada parte del diseño, desarrollo y puesta en marcha del sistema, para cumplir con los procesos de ejecución y verificación de los mismos.

Con el sistema implementado a partir de las diversas tecnologías aplicadas, en donde cada una de sus partes cumplen con el funcionamiento, como el alto rendimiento, tolerancia a fallos y la seguridad.

Al final para comprobar las características del sistema se realizaron pruebas de carga, unitarias y una encuesta. En las de carga se simularon 100, 500, 1000, 1500 y 2000 peticiones concurrentemente, dando resultados mayores a 97.5 % sin fallos y latencias promedio menores a 1 segundo. Las unitarias se realizaron a nivel de la aplicación cliente y se aplicó una encuesta a 10 personas, donde se obtuvieron resultados muy positivos sobre la temática y la aplicación en general. Finalmente, el sistema recomendador basado en filtros colaborativos se obtuvo que el error medio cuadrático de prueba y entrenamiento están cerca, por lo que concluye que el

sistema genera una recomendación acorde al perfil del usuario que permita generar y sugerir nuevos lugares.

**Palabras clave:** GraphQL, PWA, DevOps, Sistema Recomendador, Filtros Colaborativos, Middleware, API, Docker.



## **Abstract**

Today, with population growth, transactional systems, decision support systems and strategic systems store considerable amounts of information, each piece of data representing relevant background information on its owner.

Representing a growth of great magnitude, such business entities can leverage this information set in optimal ways, thereby improving their sales strategies and distribution processes. Consequently, efficiently extracting useful information is more difficult. Efficient filtering of this information requires the application of effective mechanisms. One technique to solve this problem is collaborative filtering.

Based on this problem, the present project aims to create a recommendation system based on collaborative filtering for the collection and distribution of products, using new technologies such as: PWA, GraphQL, Docker and DevOps. To carry out this, we have the use of the SCRUM methodology where each part of the design, development and implementation of the system was divided, to comply with the processes of execution and verification of the same.

With the system, implemented from the various technologies applied, each of its parts comply with the operation, such as high performance, fault tolerance and security.

At the end, in order to verify the system characteristics, load tests, unit tests and a survey were performed. In the load tests, 100, 500, 1000, 1000, 1500 and 2000 requests were simulated concurrently, giving results higher than 97.5% without failures and average latencies lower than 1 second. The unitary tests were carried out at the client application level and a survey was applied to 10 people, where very positive results were obtained on the subject and the application in general. Finally, the recommender system based on collaborative filters showed that the mean squared error of testing and training are close, which concludes that the system generates a recommendation according to the user's profile that allows generating and suggesting new places.

**Keywords:** GraphQL, PWA, DevOps, Recommender System, Collaborative Filters, Middleware, API, Docker.

# ÍNDICE

|           |   |           |
|-----------|---|-----------|
| <b>I</b>  | <b>INTRODUCCIÓN</b>                           | <b>13</b> |
| 1.1       | Descripción del Problema.....                 | 15        |
| 1.1.1     | Antecedentes .....                            | 15        |
| 1.1.2     | Importancia y alcances .....                  | 15        |
| 1.2       | Objetivos.....                                | 17        |
| 1.2.1     | General .....                                 | 17        |
| 1.2.2     | Específicos .....                             | 17        |
| <b>II</b> | <b>MARCO TEÓRICO</b>                          | <b>18</b> |
| 2.1       | Sistemas de Recomendación . . . . .           | 18        |
| 2.1.1     | Filtros Colaborativos . . . . .               | 19        |
| 2.2       | Aplicaciones Web Progresivas (PWAs) . . . . . | 21        |
| 2.2.1     | Componentes de una PWA . . . . .              | 21        |
| 2.2.2     | Flutter . . . . .                             | 22        |
| 2.2.3     | PWA vs Aplicaciones Nativas . . . . .         | 23        |
| 2.3       | GraphQL . . . . .                             | 23        |
| 2.3.1     | Operaciones . . . . .                         | 24        |
| 2.3.1.1   | Query . . . . .                               | 24        |
| 2.3.1.2   | Mutation . . . . .                            | 25        |
| 2.3.1.3   | Subscription . . . . .                        | 25        |

|            |  |           |
|------------|--|-----------|
| 2.3.2      | GraphQL vs REST .....                      | 25        |
| 2.3.3      | Herramientas GraphQL .....                 | 26        |
| 2.3.3.1    | Hasura.....                                | 26        |
| 2.4        | JWT.....                                   | 27        |
| 2.5        | Odoo .....                                 | 27        |
| 2.6        | FastApi.....                               | 28        |
| 2.6.1      | Características .....                      | 28        |
| 2.7        | Docker.....                                | 28        |
| 2.7.1      | Componentes .....                          | 29        |
| 2.7.2      | Dockerfile.....                            | 29        |
| 2.7.3      | Docker Compose .....                       | 30        |
| 2.8        | DevOps .....                               | 30        |
| <b>III</b> | <b>ANÁLISIS Y REQUERIMIENTOS</b> .....     | <b>31</b> |
| 3.1        | Elicitación de Requerimientos .....        | 31        |
| 3.1.1      | Requerimientos Funcionales .....           | 31        |
| 3.1.2      | Requisitos No Funcionales.....             | 37        |
| <b>IV</b>  | <b>METODOLOGÍA</b> .....                   | <b>38</b> |
| 4.1        | Metodología de Desarrollo ágil SCRUM ..... | 38        |
| 4.1.1      | Características .....                      | 39        |
| 4.1.2      | Roles.....                                 | 40        |
| 4.1.3      | Artefactos .....                           | 40        |
| 4.1.4      | Actividades.....                           | 41        |
| 4.1.5      | Fases de SCRUM .....                       | 41        |
| 4.2        | Fases .....                                | 42        |
| 4.2.1      | Primera fase.....                          | 45        |
| 4.2.1.1    | Estado del arte .....                      | 45        |

|             |                                |            |
|-------------|--------------------------------|------------|
| 4.2.2       | Segunda fase . . . . .         | 47         |
| 4.2.3       | Tercera fase . . . . .         | 59         |
| 4.2.4       | Cuarta fase . . . . .          | 61         |
| 4.2.5       | Quinta Fase . . . . .          | 71         |
| 4.3         | Arquitectura .....             | 76         |
| 4.3.1       | Backend.....                   | 76         |
| 4.3.2       | Middleware.....                | 77         |
| 4.3.3       | FrontEnd.....                  | 77         |
| <b>V</b>    | <b>ANÁLISIS DE RESULTADOS</b>  | <b>81</b>  |
| 5.1         | Pruebas de carga .....         | 81         |
| 5.2         | Pruebas unitarias .....        | 83         |
| 5.3         | Encuesta.....                  | 87         |
| <b>VI</b>   | <b>CONCLUSIONES</b>            | <b>92</b>  |
| <b>VII</b>  | <b>RECOMENDACIONES</b>         | <b>95</b>  |
| <b>VIII</b> | <b>TRABAJOS FUTUROS</b>        | <b>96</b>  |
|             | <b>Anexos</b>                  | <b>101</b> |
| <b>A</b>    | <b>Formato de la encuesta</b>  | <b>102</b> |
| 1.1         | Información de la empresa..... | 102        |

# Índice de tablas

|    |  |    |
|----|--|----|
| 1  | Algoritmos de Filtrado Colaborativo .....                  | 20 |
| 2  | PWA vs Aplicación Nativa. ....                             | 23 |
| 3  | GraphQL vs REST.....                                       | 26 |
| 4  | Requisito funcional 1.....                                 | 32 |
| 5  | Requisito funcional 2.....                                 | 32 |
| 6  | Requisito funcional 3.....                                 | 33 |
| 7  | Requisito funcional 4.....                                 | 33 |
| 8  | Requisito funcional 5.....                                 | 34 |
| 9  | Requisito funcional 6.....                                 | 34 |
| 10 | Requisito funcional 7.....                                 | 35 |
| 11 | Requisito funcional 8.....                                 | 35 |
| 12 | Requisito funcional 9.....                                 | 36 |
| 13 | Requisito funcional 10.....                                | 36 |
| 14 | Requisito no funcional 1.....                              | 37 |
| 15 | Requisito no funcional 2.....                              | 37 |
| 16 | Roles en el proyecto.....                                  | 43 |
| 17 | Tabla generalizada de los resultados <i>query</i> .....    | 83 |
| 18 | Tabla generalizada de los resultados <i>mutation</i> ..... | 83 |

|    |   |    |
|----|---|----|
| 19 | Pregunta 1. Conteo respuestas . . . . . | 87 |
| 20 | Pregunta 2. Conteo respuestas . . . . . | 88 |
| 21 | Pregunta 3. Conteo respuestas . . . . . | 89 |
| 22 | Pregunta 4. Conteo respuestas . . . . . | 90 |
| 23 | Pregunta 5. Conteo respuestas . . . . . | 90 |

# Índice de figuras

|    |  |    |
|----|--|----|
| 1  | Filtrado colaborativo y basados en el contenido.....                   | 20 |
| 2  | Funcionamiento del service worker.....                                 | 21 |
| 3  | Archivo Web App Manifest.....  | 22 |
| 4  | Estructura de Query en GraphQL.....                                    | 24 |
| 5  | Consola de Hasura.....   | 27 |
| 6  | Arquitectura Docker.....   | 29 |
| 7  | Ciclo de Vida SCRUM.....   | 39 |
| 8  | Sprints del proyecto.....  | 44 |
| 9  | Distribución de la lógica de negocio.....                              | 47 |
| 10 | Distribución del código para el módulo.....                            | 48 |
| 11 | Dockerfile Odoos.....  | 49 |
| 12 | Distribución de la lógica de negocio para la API de autenticación..... | 49 |
| 13 | Resolver para un Query.....  | 50 |
| 14 | Conexión con la base de datos.....                                     | 50 |
| 15 | Esquema para la respuesta de autenticación.....                        | 51 |
| 16 | Código para definir el servidor FastAPI.....                           | 51 |
| 17 | Dockerfile FastAPI Sistema Recomendador.....                           | 51 |
| 18 | Matriz de Correlación.....   | 52 |
| 19 | Histogramas de los datos.....  | 53 |



|    |  |    |
|----|--|----|
| 20 | Mapa de Puntos.....  | 54 |
| 21 | Precisiones para generar <i>clúster</i> basados en la distancias ..... | 55 |
| 22 | Similitud entre usuarios .....   | 57 |
| 23 | Pronóstico de similitud entre usuarios .....                           | 58 |
| 24 | Dockerfile FastAPI Sistema Recomendador .....                          | 59 |
| 25 | Diagrama del middleware Hasura.....                                    | 60 |
| 26 | Formulario para conectar la base de datos en hasura.....               | 60 |
| 27 | Formulario para conectar una API de GraphQL externa .....              | 61 |
| 28 | Aplicación Web Progresiva .....  | 62 |
| 29 | Pantalla de inicio de sesión.....                                      | 64 |
| 30 | Pantalla de registro .....   | 65 |
| 31 | Pantalla de tickets de viaje.....                                      | 66 |
| 32 | Pantalla de perfil de usuario.....                                     | 67 |
| 33 | Pantalla de reserva de ticket.....                                     | 68 |
| 34 | Pantalla de recomendaciones .....                                      | 69 |
| 35 | Pantalla del destino mejor recomendado .....                           | 70 |
| 36 | Estructura del backend.....  | 71 |
| 37 | Parte del archivo docker-compose.....                                  | 72 |
| 38 | Estructura de la técnica DevOps.....                                   | 73 |
| 39 | Estructura de la técnica DevOps.....                                   | 75 |
| 40 | Tabla de Pipelines.....  | 75 |
| 41 | Arquitectura del sistema .....   | 76 |
| 42 | Arquitectura Frontend.....   | 79 |
| 43 | Estructura del proyecto .....  | 80 |
| 44 | Pruebas de Carga .....   | 82 |
| 45 | Latencia.....  | 82 |

|    |  |    |
|----|--|----|
| 46 | Pruebas unitarias - Iniciar Sesión Exitoso ..... | 84 |
| 47 | Pruebas unitarias - Iniciar Sesión Error .....   | 85 |
| 48 | Pruebas unitarias - Comprar Ticket Exitoso ..... | 86 |
| 49 | Pruebas unitarias - Comprar Ticket Error .....   | 87 |
| 50 | Representación porcentual pregunta 1 . . . . .   | 88 |
| 51 | Representación porcentual pregunta 2 . . . . .   | 88 |
| 52 | Representación porcentual pregunta 3 . . . . .   | 89 |
| 53 | Representación porcentual pregunta 4 . . . . .   | 90 |
| 54 | Representación porcentual pregunta 5 . . . . .   | 91 |

# Capítulo I

## INTRODUCCIÓN

En la actualidad la mayoría de las transacciones que las personas realizan se dan a través Internet, a raíz de esto los clientes realizan búsquedas para encontrar lo que necesitan, el problema surge en las búsquedas, porque generalmente devuelven una gran cantidad de resultados que no tiene relación y además no son los esperados. En base a ello, nace la iniciativa de sistemas de recomendación basados en los gustos de usuarios similares para filtrar información relacionada, que se conoce cómo el modelo de filtros colaborativos.

De acuerdo a Galán (2007), "los primeros sistemas de recomendación eran conocidos tan sólo cómo filtros colaborativos y los trabajos iniciales datan de principios de los años 90. Uno de los pioneros en el desarrollo del filtrado colaborativo fue el grupo de investigación del proyecto GroupLens de la Universidad de Minnesota a mediados de los noventa".

Por otro lado, las aplicaciones web progresivas (PWA) permiten funcionar a través de una página web ejecutándose sobre un navegador sin la necesidad de instalarlas directamente en el dispositivo, resultando ser más interoperables y productivas. Este tipo de aplicaciones presenta ventajas cómo: disponibilidad independientemente de la conexión, tiempos rápidos de carga, notificaciones push, acceso directo en la pantalla de inicio, aspecto nativo, ligeras, funcionan en todos los navegadores(progresivas) al adaptarse a cualquier dispositivo (responsivas) y tienen un nivel de seguridad muy alto debido a que usan el protocolo TLS (Seguridad de la Capa de

Transporte) (Ater, 2017).

En virtud de las ventajas de la PWA, se desarrollan este nuevo tipo de aplicación que ayudan con las dificultades de las aplicaciones tradicionales cómo: la compatibilidad, el espacio requerido y el proceso de instalación, la forma de compartir, entre otras. Agregando en el proceso técnicas que mejoren el desarrollo, la comunicación, la distribución y el despliegue de los sistemas informáticos (Ater, 2017).

Al conocer las grandes capacidades de estas nuevas formas de desarrollo, se pretende construir un sistema que permita aprovechar estas cualidades para llevar a cabo el acopio y distribución de productos. Este sistema consiste en una forma de llevar la administración del centro de acopio que es el almacenamiento de la producción de pequeños productores para alcanzar la cantidad y calidad que se requiere para su posterior distribución en el sector urbano, con el que se busca agilizar la gestión de estos centros.

## **1.1 Descripción del Problema**

### **1.1.1 Antecedentes**

Según Del Pino et al. (2011), “durante los últimos años, los sistemas de recomendación han sido herramientas que han contribuido a mejorar la experiencia de los usuarios en las aplicaciones que estos utilizan. Numerosas empresas han apostado por el uso de estas herramientas para lograr que sus sitios web posean un ambiente personalizado. En donde, el contenido que se les presenta a sus usuarios está estrechamente atado a sus preferencias personales. Algunas empresas que usan recomendadores en sus sitios web son Amazon, Netflix, Facebook y Twitter; los cuales sugieren compras de productos, películas, amistades y personas a seguir, respectivamente. La gran cantidad de información filtrada de algunos usuarios sirve para ayudar a otros, resultando en un entorno colaborativo”.

Para poder abrir las puertas a los filtros colaborativos se introduce un nuevo concepto Aplicaciones Web Progresivas (PWA). Una aplicación web progresiva es un tipo de software que se entrega a través de la web, elaborado utilizando tecnologías web comunes como HTML, CSS y JavaScript. El objetivo de una PWA es el estudio de los nuevos estándares que permiten a las aplicaciones web forjar el uso del hardware, esto impulsa el desarrollo de aplicaciones web que funcionen en diferentes plataformas ganando portabilidad sin perder ningún elemento o atractivo que tiene una aplicación nativa (Rodríguez, 2018).

### **1.1.2 Importancia y alcances**

Los sistemas recomendadores favorecen a las empresas con la administración de la información ayudando en la toma de decisiones, permitiendo a sus usuarios encontrar resultados adecuados a las especificaciones descritas al basarse en recomendaciones de otros usuarios, con

el objetivo de ayudar al consumidor a encontrar productos que de otro modo probablemente no encontrarán (Li et al., 2005).

El sistema está enfocado para empresas u organizaciones que reciben y distribuyen gran variedad de productos a través del Internet, es decir, que trabajan haciendo uso del comercio electrónico. En donde los propietarios de estas organizaciones esperan un resultado positivo que refleje el aumento de las ganancias a través de sus portales generando así una ventaja competitiva por el uso de recomendaciones que conducen a amplios márgenes de utilidad.

En base a los planteamientos anteriores, se hace uso de nuevas tecnologías para mejorar los procesos en el desarrollo de software. Las cuales han mejorado y optimizado el proceso de diseño e implementación para lograr buenos resultados en menos tiempo, cómo: mayor cantidad de peticiones, lenguajes de consulta sencillos y potentes, nuevas técnicas de despliegue, etc.

Entre las nuevas tecnologías se tiene el uso de GraphQL mediante Hasura para agilizar el proceso de creación de APIs, con conexión a múltiples fuentes de datos. Luego tenemos Docker para distribuir las diferentes partes de un sistema en pequeños ambientes específicos para un mejor funcionamiento, otorgando la característica de acelerar el proceso de puesta en escena.

## **1.2 Objetivos**

### **1.2.1 General**

Diseñar e implementar un sistema recomendador basado en filtros colaborativos para el acopio y distribución de productos utilizando aplicaciones web progresivas y GraphQL sobre DevOps para plataformas basadas en servicios.

### **1.2.2 Específicos**

- Revisar el estado del arte sobre sistemas recomendadores basados en filtros colaborativos.
- Diseñar e implementar una arquitectura basada en técnicas DevOps para el sistema de acopio y distribución.
- Implementar una aplicación PWA para la gestión de acopio y distribución de los productos consumiendo servicios GraphQL.
- Definir y desarrollar un sistema recomendador basado en filtros colaborativos para el sistema de acopio y distribución.
- Diseñar y ejecutar un plan de experimentación para validar el sistema recomendador utilizando encuestas.
- Validar y probar la arquitectura del sistema utilizando métricas de rendimiento, calidad y tiempos de respuesta.
- Automatizar el despliegue de la plataforma y el sistema recomendador en la nube.

# Capítulo II

## MARCO TEÓRICO

### 2.1 Sistemas de Recomendación

La finalidad de los sistemas de recomendación es generar predicciones a un usuario tomando en cuenta sus valoraciones previas y en una retroalimentación de usuarios con un historial similar de gustos y valoraciones. Existen dos formas de hacer retroalimentaciones para obtener un historial de información que ayuden a recomendar (Martinez, 2007):

- De manera explícita puntuar cada elemento que representara un valor discreto en un rango de mínimo o máximo.
- Escoger valoraciones implícitamente, de esta manera extraer la información pertinente de las acciones que realice el usuario.

Existe una diferencia clave para poner en práctica los sistemas de recomendación una vez se tenga la información necesaria y está es que predicción hace referencia a ponderar que valoración otorgaría el usuario a cada elemento que este considere pertinente, mientras que recomendación hace referencia a la extracción de los N elementos más recomendables de un conjunto de información.



Generar buenas recomendaciones demanda de una extensa cantidad de información y de algoritmos bien optimizados para su tratamiento, todo esto realiza un consumo de memoria y CPU de manera considerable por lo que buscar un buen rendimiento se ha vuelto primordial en este campo. Este cómo otros aspectos son esenciales para tener en cuenta cuando se desarrolla un sistema recomendador.

### **2.1.1 Filtros Colaborativos**

La literatura de hoy en día apoya la idea de que los sistemas basados en filtros colaborativos funcionan almacenando juicios humanos, tomados a cabo cómo votaciones de cada individuo para una serie de elementos en un dominio dado, con el objetivo de emparejar personas que comparten las mismas necesidades o gustos (Castellano, 2007).

Estos sistemas brindan diversas ventajas con respecto a otros, por eso es importante tener en cuenta que en el filtrado colaborativo van a ser los usuarios los que lleguen a determinar la relevancia, cualidad e interés de estos elementos (Castellano, 2007).

Además, tienen la capacidad de discernir cómo se adapta un elemento a las necesidades o intereses de los usuarios por medio de la capacidad de los mismos, cómo es la calidad y el gusto lo que sería difícil realizar por medio de procesos computacionales. Finalmente, este sistema recomienda elementos que no se esperaban encontrar haciendo referencia a un hallazgo afortunado, casual y no esperado.

Es enorme el potencial del filtrado colaborativo en el ámbito de discriminar y tener en cuenta comportamientos subjetivos, que combinado con las tecnologías de la información su potencial completo puede ser explotado (Castellano, 2007).

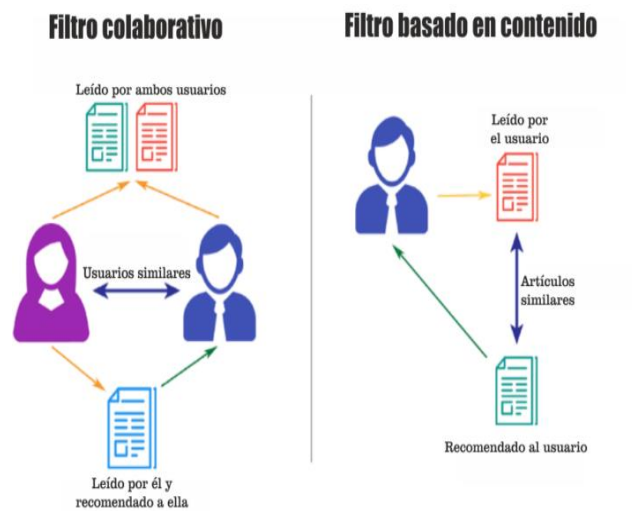


Figura 1. Filtrado colaborativo y basados en el contenido.  
(Unipython, sf)

Según (Galán, 2007), un sistema de recomendación basado en filtros colaborativos se basa en dos tipos de algoritmos que son:

- **Basados en memoria:** Hacen uso de una base de datos completa, con elementos y usuarios, para realizar predicciones.
- **Basados en modelo:** Estos empiezan creando un modelo de valoraciones en base al usuario. Analizan la problemática cómo una predicción estadística y calculan el valor esperado para cada elemento en base a las valoraciones previas.

| Tipo                                   | Algoritmos                             |
|--|--|
| Algoritmos basados en vecinos cercanos | Coefficiente de Correlación de Pearson |
| Algoritmos basados en elementos        | Similitudes Basadas en Coseno          |
|  | Similitudes Basadas en Correlación     |
|  | Similitudes Basadas en Coseno Ajustado |

Tabla 1. Algoritmos de Filtrado Colaborativo.  
(Galán, 2007)

## 2.2 Aplicaciones Web Progresivas (PWAs)

Según (Rodríguez, 2018), son aplicaciones web que, mediante una serie de nuevas tecnologías, se integran a dispositivos, como si de aplicaciones nativas se tratase, pueden funcionar de manera offline, tienen acceso a tecnologías de bajo nivel, cómo sensores, cámara, almacenamiento interno, etc.; además, pueden recibir notificaciones push; se pueden agregar a la pantalla de inicio y al buscador de aplicaciones del sistema operativo, etc.

### 2.2.1 Componentes de una PWA

- **Service worker:** Son proxies del lado del cliente programados en Javascript que se pueden desarrollar para controlar las distintas estrategias a seguir cuando se realizan peticiones HTTP. Esto permite que la primera vez que se ingrese en una página se introduzcan en caché los recursos necesarios para que la próxima vez, en caso de no tener conexión, se devuelve la versión cacheada de tal forma que sea transparente para el resto de la aplicación. Además, permite conjuntamente con la tecnología WebPush, recibir y mostrar notificaciones push cómo cualquier aplicación nativa.

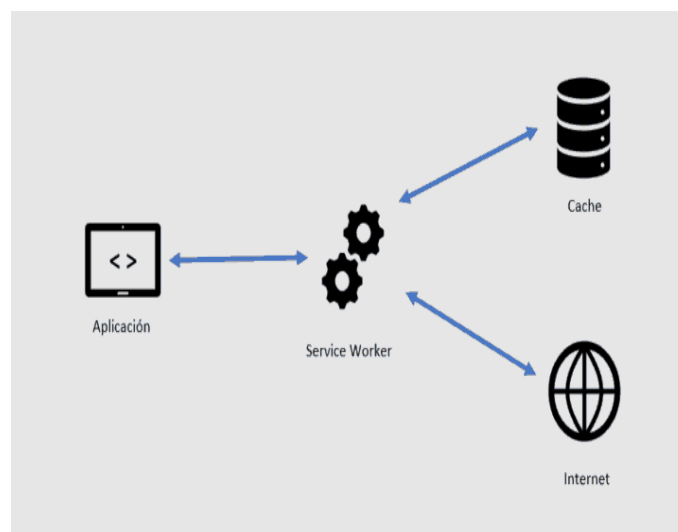


Figura 2. Funcionamiento del service worker.  
(Triguero, 2018)

- **Web Manifest:** Es un archivo JSON que describe a la aplicación y que es utilizada tanto por el navegador cómo por el sistema operativo. Aquí se definen atributos que formarán parte de la aplicación una vez esté instalada en el dispositivo: nombre, icono, color del tema, etc.

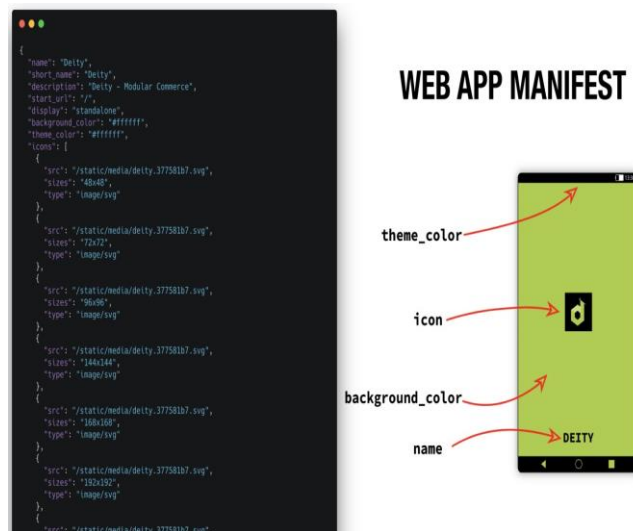


Figura 3. Archivo Web App Manifest.  
(Schouren, 2019)

## 2.2.2 Flutter

Es un kit de desarrollo de Google (SDK) para el desarrollo de aplicaciones multiplataforma especializándose en el ambiente móvil, con el objetivo de crear aplicaciones de alto rendimiento y alta resolución para iOS, Android, Web y Windows a partir de una única base de código. El propósito de esta herramienta es facilitar a los desarrolladores la construcción de aplicaciones de alto rendimiento que se adapten de manera natural a diferentes plataformas. Permite reducir significativamente los tiempos de costos y producción, a través de desarrollar un solo proyecto para todos los sistemas operativos. Flutter para hacer todo esto posible necesita del lenguaje de programación Dart, donde este también es creado por Google, conformando así una herramienta útil para el desarrollo en la actualidad (Olivo et al., 2020).

### 2.2.3 PWA vs Aplicaciones Nativas

Al avanzar la tecnología móvil se a vuelto el centro de la actividad en internet, desde prácticamente cualquier dispositivo móvil se puede acceder a internet ya sea a través de aplicaciones instaladas o navegadores quedando a elección del usuario elegir el medio por el cual conectarse, es aquí donde las PWA pretenden tomar lo mejor de dos mundos, empezando por el alcance ilimitado de las aplicaciones web que permiten tener los contenidos debidamente actualizados, el rendimiento y experiencia no se ven afectados porque responden de una manera efectiva y rápida.

Por otra parte, las aplicaciones nativas gozan de gran rendimiento y fiabilidad para el usuario sobre la plataforma en la cual está se encuentra, las mismas son capaces de acceder a los distintos periféricos del dispositivo, cámara, Bluetooth, archivos, etc. sin restricción alguna a excepción de las que ya interponga dicha aplicación, generando así una experiencia de usuario codependiente de la plataforma móvil en cual está se ejecute.

| <b>PWA</b>                                    | <b>Aplicación nativa</b>                        |
|---|---|
| Código HTML/CSS/JS.                           | Código y funciones nativas.                     |
| Rendimiento bajo.                             | Rendimiento alto.                               |
| No se puede poner en las tiendas.             | Aplicación en las tiendas.                      |
| Proceso de compilación y distribución rápido. | Procesos de compilación y distribución tedioso. |
| Una sola base de código.                      | Programación para cada plataforma.              |

Tabla 2. PWA vs Aplicación Nativa..

## 2.3 GraphQL

Según (Garrido, 2018) indica que es una capa API igual que REST que sirve para administrar peticiones por medio del cliente a distintas bases de datos heterogéneas, esta tecnología revolucionó la forma de realizar peticiones y su costo debido a que se necesita un único punto de acceso, pudiendo utilizarse tanto como Wrapper de su símil que es REST como sustituirla

por completo; para que el cliente pueda realizar nuevas consultas se aumenta su esquema en el servidor y su versatilidad se enfoca en pedir varios datos a la vez eliminando el problema del *roundtrip* time en donde se necesita el primer dato antes de pedir el segundo; además, genera una auto documentación rellenando el campo de descripción del atributo.

Entre los principios fundamentales de GraphQL, se encuentra SWAP, debido a que es uno de los ejemplos más relevantes al momento de su implementación basado en la película de Star Wars. Aunque es un lenguaje de “consultas” no se expone la estructura de la base de datos cómo tal y trabaja de forma que describe la petición del cliente de manera estructurada, cuando se recibe la consulta del cliente la petición se realiza en el backend. Es así como GraphQL con estas capacidades mejora a REST.



```
{
  user(id: 4802170) {
    id
    name
    isViewerFriend
    profilePicture(size: 50) {
      uri
      width
      height
    }
    friendConnection(first: 5) {
      totalCount
      friends {
        id
        name
      }
    }
  }
}
```

```
{
  "data": {
    "user": {
      "id": "4802170",
      "name": "Lee Byron",
      "isViewerFriend": true,
      "profilePicture": {
        "uri": "cdn://pic/4802170/50",
        "width": 50,
        "height": 50
      }
    },
    "friendConnection": {
      "totalCount": 13,
      "friends": [
        {
          "id": "305249",
          "name": "Stephen Schwink"
        },
        {
          "id": "3108935",
          "name": "Nathaniel Roman"
        }
      ]
    }
  }
}
```

Figura 4. Estructura de Query en GraphQL.  
(Byron, 2018)

## 2.3.1 Operaciones

### 2.3.1.1 Query

Es la operación que permite hacer las consultas para obtener y dar lectura a los datos del servidor. Estas consultas son la naturaleza de GraphQL, puesto que permite al cliente modelar

objetos con propiedades esenciales para recibir la información que este necesita.

### **2.3.1.2 Mutation**

Es una acción desarrollada por GraphQL que permite la alteración de los datos en el servidor (escritura, actualización o eliminación de los datos). Su estructura es igual al de las consultas, devuelve un tipo de objeto en base a la operación que se ejecute, el objeto contiene las propiedades que sean de interés para el cliente. Está característica, provee un nuevo estado del objeto en cuestión después de una modificación.

### **2.3.1.3 Subscription**

Es esencialmente una consulta de GraphQL, donde el cliente recibe una actualización cada vez que el valor de la consulta cambia en cualquier sentido. Es decir, son consultas en tiempo real que devolverán el resultado más reciente de haber ejecutado la misma. Este resultado puede variar en base a la creación o actualización de los objetos en cuestión.

## **2.3.2 GraphQL vs REST**

cómo se sabe REST es el enfoque estándar para el desarrollo de APIs, por lo cual se vio desafiado por el surgimiento GraphQL en 2016 porque planteaba una nueva forma de construir una API. Una de sus principales diferencias es que GraphQL utilizaba un lenguaje de consulta que permitía el uso de varias herramientas a través de un solo punto final con respecto a HTTP. En la Tabla 3 se visualiza una comparativa entre GraphQL y REST.

| <b>GraphQL</b>                                      | <b>REST</b>  |
|---|--|
| Una sola url para obtener los datos.                | Varias url para obtener los datos.                 |
| Eficaz y rápido. Obtiene solo los datos necesarios. | Trae todos los datos, afectando en el rendimiento. |
| Comunicación síncrona y asíncrona                   | Comunicación síncrona.                             |
| Protocolos HTTP, AMQP, MQTT                         | Protocolo HTTP                                     |

Tabla 3. GraphQL vs REST..

### 2.3.3 Herramientas GraphQL

Para trabajar con GraphQL existe una gran cantidad de herramientas, unas que permiten crear la API desde cero y otras que modelan toda la estructura de la base de datos a través de GraphQL, minimizando los tiempos y costos de desarrollo. A partir de esto dependiendo del objetivo final del proyecto se debe elegir la herramienta adecuada para el desarrollo.

#### 2.3.3.1 Hasura

Es una herramienta de código abierto en su versión local, permite a los datos ser accedidos al instante mediante una API GraphQL en tiempo real. Hasura puede conectarse directamente a base de datos, servidores REST, servidores GraphQL y otras API de terceros, que proporciona una API GraphQL unificada en tiempo real de todas las fuentes de datos.

Actualmente permite las siguientes bases de datos: Postgres, MS SQL Server, Citus / Hyperscale, BigQuery and MySQL(versión preliminar).



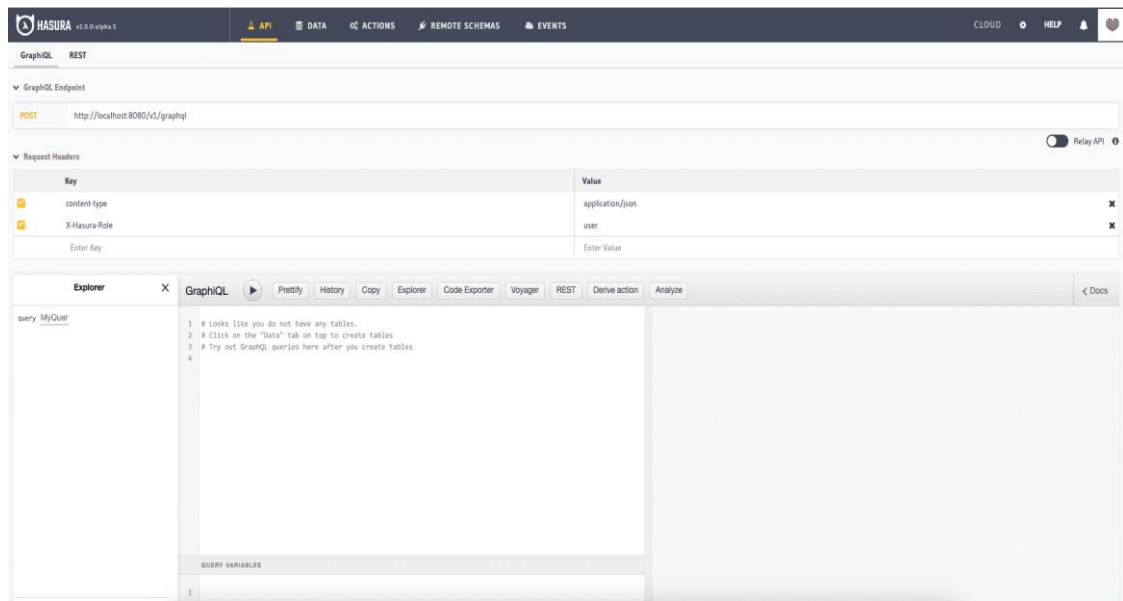


Figura 5. Consola de Hasura.  
(Hasura, sf)

## 2.4 JWT

JWT<sup>1</sup> es un bloque de información en formato JSON que está firmado. La firma permite que pueda ser verificado y es factible de entender al estar en un formato pequeño y familiar, cómo lo es JSON. Es una herramienta muy importante en temas de seguridad y autenticación para las empresas que ofrecen servicios a través de la Web.

## 2.5 Odoo

Odoo(anteriormente OpenERP) es un software ERP empresarial que integra una gran cantidad de aplicaciones y módulos de gestión empresarial, así como un sitio web integrado. Presenta dos versiones la Community (software libre) y la Enterprise. Contiene aplicaciones cómo CRM, Ventas, Inventario, Gestión de Proyectos, Gestión Financiera, Recursos Humanos, entre otros. Utiliza cómo lenguaje de programación base a Python y para la parte gráfica hace uso de

<sup>1</sup>Json Web Token

JavaScript, incluyendo Bootstrap, además su motor de base de datos es PostgreSQL.

## **2.6 FastApi**

FastAPI es un framework web moderno y rápido (alto rendimiento) para construir APIs con Python 3.6+ basado en las anotaciones de tipos estándar de Python.

### **2.6.1 Características**

FastAPI presenta las siguientes características:

- Alto Rendimiento
- Programación rápida
- Menos errores
- Intuitivo
- Fácil
- Corto
- Robusto
- Basado en estándares (OpenAPI y JSON Schema)

## **2.7 Docker**

Es una herramienta de código abierto que automatiza el despliegue de las aplicaciones dentro de contenedores. Está construido para proveer un entorno liviano y rápido en donde ejecutar el código de programación, facilitando las pruebas y las puestas en producción de los sistemas. Su misión es proporcionar: una forma fácil y ligera de poner en marcha las

aplicaciones o sistemas, crear entornos adecuados a los requerimientos de las aplicaciones a ejecutar y un ciclo de vida de desarrollo rápido y eficiente (Turnbull, 2014).

### 2.7.1 Componentes

Docker presenta los siguientes componentes:

- Cliente y Servidor de Docker
- Imágenes
- Registros
- Contenedores

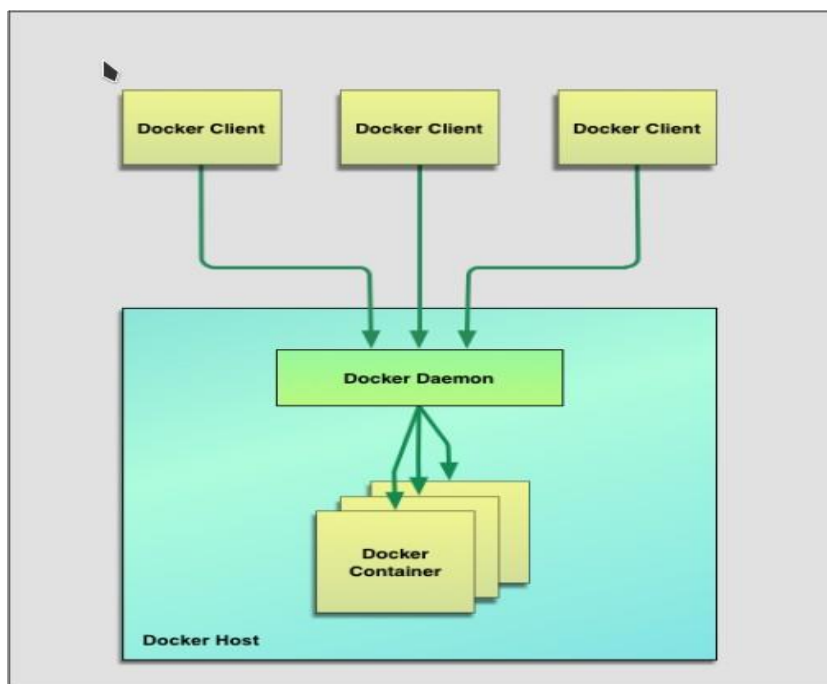


Figura 6. Arquitectura Docker.  
(Turnbull, 2014)

### 2.7.2 Dockerfile

Es un archivo que usa DSL básico con instrucciones para la construcción de nuevas imá-

genes Docker. Docker recomienda que para el despliegue de nuevas aplicaciones se haga uso de esta herramienta para crear contenedores adecuados para las aplicaciones a ejecutar (Turnbull, 2014).

### **2.7.3 Docker Compose**

Es una herramienta para declarar y ejecutar aplicaciones Docker de varios contenedores relacionados entre sí, es decir, cada aplicación corre en un contenedor diferente, todo esto se define mediante un archivo cuya extensión es yaml, a cada contenedor en el archivo se le denomina cómo servicio. Al final con un solo comando se levanta cada uno de los servicios y se pone en marcha el sistema completo (Turnbull, 2014).

## **2.8 DevOps**

Es una técnica empleada para automatizar el despliegue de los sistemas. Según Basilio (2020), “debido a los cambios económicos y sociales que fueron provocados por la transformación digital y el auge de las metodologías ágiles en las empresas han impulsado el uso de Development + Operations (DevOps) para el desarrollo de software; este término es un movimiento que aborda el conflicto natural entre el desarrollo de software y las operaciones, para mejorar la colaboración entre los departamentos de desarrollo y operaciones, agilizando el proceso completo de entrega de una manera integral; además, conduce al enfoque de "un equipo" en donde los programadores, evaluadores y administradores de sistemas participan en el desarrollo de la solución al problema”.

La importancia de adoptar cómo filosofía de trabajo admite la reducción de riesgos para que existan cambios mediante el uso de herramientas adecuadas, buscando entregar aplicaciones a un menor costo, de tal manera que sea más rápida y con menores riesgos. Por este motivo, es vital la necesidad de difundir más este concepto, de forma clara, sintetizada y entendible (Basilio, 2020).

## **Capítulo III**

# **ANÁLISIS Y REQUERIMIENTOS**

Este proceso es uno de los más importantes cuando de un proyecto de desarrollo de software se trata, con él se obtiene las bases para iniciar las actividades, además ayuda a comprender de mejor forma el problema de estudio. Al final se tendrá una idea clara de lo que abarca el sistema, de la misma manera permitirá corroborar si se cumplieron con los objetivos planteados. Una buena especificación de requerimientos es la clave para el triunfo de los proyectos de software.

### **3.1 Elicitación de Requerimientos**

#### **3.1.1 Requerimientos Funcionales**

- Requisitos funcionales las actividades que los usuarios deben realizar.

|                    |   |
|--------------------|---|
| <b>Código</b>      | RE01  |
| <b>Nombre</b>      | Registro de usuarios  |
| <b>Descripción</b> | Se deberá registrar en la herramienta los datos necesarios.   |
| <b>Entradas</b>    | Correo electrónico, Contraseña  |
| <b>Proceso</b>     | Se debe contar con un botón el cual permitirá al usuario registrarse, contando con un formulario en donde el usuario ingrese sus datos personales para el registro. La herramienta almacenara la información en la base de datos. |
| <b>Salidas</b>     | Mensaje exitoso al registrar un usuario.<br>Mensaje de error al no completar todos los campos necesarios.   |
| <b>Prioridad</b>   | Alta  |

Tabla 4. Requisito funcional 1.

|                    |  |
|--------------------|--|
| <b>Código</b>      | RE02   |
| <b>Nombre</b>      | Inicio de sesión   |
| <b>Descripción</b> | La herramienta permitirá realizar un inicio de sesión obligatoria para acceder a sus funcionalidades.  |
| <b>Entradas</b>    | Correo electrónico, Contraseña   |
| <b>Proceso</b>     | Se contará con un formulario donde el usuario llenara con los datos que se piden, después la herramienta realizara la verificación de credenciales ingresadas con las de la base de datos. En caso de realizar una verificación exitosa se continuará en la siguiente interfaz, caso contrario deberá ingresar los datos nuevamente. |
| <b>Salidas</b>     | Mensaje de error al no completar los campos necesarios.<br>Mensaje de error al no haber ingresado correctamente los datos.   |
| <b>Prioridad</b>   | Alta   |

Tabla 5. Requisito funcional 2.

|                    |  |
|--------------------|--|
| <b>Código</b>      | RE03   |
| <b>Nombre</b>      | Cierre de sesión   |
| <b>Descripción</b> | La herramienta permitirá realizar un cierre de sesión para salir de la aplicación  |
| <b>Entradas</b>    |  |
| <b>Proceso</b>     | Se contará con un botón donde el usuario proporcionara un click para salir de su cuenta, denegando el acceso a la misma y garantizando la privacidad de los datos. |
| <b>Salidas</b>     | Mensaje de error al no poseer un acceso a internet.  |
| <b>Prioridad</b>   | Alta   |

Tabla 6. Requisito funcional 3.

|                    |   |
|--------------------|---|
| <b>Código</b>      | RE04  |
| <b>Nombre</b>      | Actualización de contraseña   |
| <b>Descripción</b> | La herramienta permitirá realizar una actualización de contraseña en caso de que el usuario así lo requiera.  |
| <b>Entradas</b>    | Contraseña nueva, contraseña actual   |
| <b>Proceso</b>     | Se contará con un botón el cual permitirá al usuario el cambio de su contraseña, contando con un formulario en donde el usuario ingrese la contraseña actual y la nueva contraseña para su posterior envío. |
| <b>Salidas</b>     | Mensaje de error al no poseer un acceso a internet.   |
| <b>Prioridad</b>   | Alta  |

Tabla 7. Requisito funcional 4.

|                    |  |
|--------------------|--|
| <b>Código</b>      | RE05   |
| <b>Nombre</b>      | Detalles del usuario en sesión   |
| <b>Descripción</b> | La herramienta permitirá visualizar los datos del usuario en sesión.   |
| <b>Entradas</b>    |  |
| <b>Proceso</b>     | Se contará con apartado con el nombre perfil de usuario, donde el usuario podrá visualizar su información para que el mismo tenga una retroalimentación de los datos que actualmente registra. |
| <b>Salidas</b>     | Mensaje de error al no poseer un acceso a internet.  |
| <b>Prioridad</b>   | Alta   |

Tabla 8. Requisito funcional 5.

- Requisitos funcionales correspondientes al manejo de tickets.

|                    |   |
|--------------------|---|
| <b>Código</b>      | RE06  |
| <b>Nombre</b>      | Mantenimiento de tickets de viaje   |
| <b>Descripción</b> | La herramienta debe permitir la creación, modificación, eliminación y visualización de los tickets de viaje.  |
| <b>Entradas</b>    | Ticket de viaje   |
| <b>Proceso</b>     | Se contará con un formulario donde el administrador del sistema llenara los datos que se piden, después a través de un botón se guardara dicha información en la base de datos para previamente ser tratados. |
| <b>Salidas</b>     | Mensaje de error al no poseer un acceso a internet.   |
| <b>Prioridad</b>   | Alta  |

Tabla 9. Requisito funcional 6.



|                    |  |
|--------------------|--|
| <b>Código</b>      | RE07   |
| <b>Nombre</b>      | Compra de un ticket de viaje   |
| <b>Descripción</b> | La herramienta permitirá realizar la compra de un ticket de viaje.   |
| <b>Entradas</b>    | Ticket de viaje, Fecha de viaje  |
| <b>Proceso</b>     | Se contará con un formulario donde el usuario deberá seleccionar la fecha de viaje tanto de ida y de retorno para visualizar la disponibilidad de tickets, en base a esto el usuario podrá seleccionar un asiento y por último proceder a su pago para la compra de su ticket. |
| <b>Salidas</b>     | Mensaje de error al no poseer un acceso a internet.  |
| <b>Prioridad</b>   | Alta   |

Tabla 10. Requisito funcional 7.

|                    |  |
|--------------------|--|
| <b>Código</b>      | RE08   |
| <b>Nombre</b>      | Reserva de un ticket de viaje  |
| <b>Descripción</b> | La herramienta permitirá realizar la reserva de un ticket de viaje.  |
| <b>Entradas</b>    | Contraseña nueva, contraseña actual  |
| <b>Proceso</b>     | Se contará con un formulario donde el usuario deberá seleccionar la fecha de viaje tanto de ida y de retorno para visualizar la disponibilidad de tickets, en base a esto el usuario podrá seleccionar un asiento y por último proceder a la reserva de su ticket. |
| <b>Salidas</b>     | Mensaje de error al no poseer un acceso a internet.  |
| <b>Prioridad</b>   | Alta   |

Tabla 11. Requisito funcional 8.

|                    |  |
|--------------------|--|
| <b>Código</b>      | RE09   |
| <b>Nombre</b>      | Detalles de un ticket de viaje   |
| <b>Descripción</b> | La herramienta permitirá visualizar los detalles de un ticket de viaje disponible.   |
| <b>Entradas</b>    | Contraseña nueva, contraseña actual  |
| <b>Proceso</b>     | Se contará con un apartado para que el usuario pueda visualizar los detalles del ticket de viaje ya sea este reservado, comprado o disponible. |
| <b>Salidas</b>     | Mensaje de error al no poseer un acceso a internet.  |
| <b>Prioridad</b>   | Alta   |

Tabla 12. Requisito funcional 9.

- Requisitos funcionales correspondientes al manejo de la información.

|                    |   |
|--------------------|---|
| <b>Código</b>      | RE10  |
| <b>Nombre</b>      | Almacenamiento de la información  |
| <b>Descripción</b> | La herramienta permitirá guardar toda la información manejada en sus diferentes procesos.   |
| <b>Entradas</b>    | Datos de usuarios, Datos de tickets   |
| <b>Proceso</b>     | Se contará con una base de datos en donde quedará la evidencia de toda la información manejada en los diferentes procesos realizados en el herramienta. |
| <b>Salidas</b>     | Mensaje de error al no poseer una conexión a una base de datos.   |
| <b>Prioridad</b>   | Alta  |

Tabla 13. Requisito funcional 10.

### 3.1.2 Requisitos No Funcionales

|                    |   |
|--------------------|---|
| <b>Código</b>      | RE01  |
| <b>Nombre</b>      | Diseño responsive.  |
| <b>Descripción</b> | Las interfaces del sistema deben acoplarse a los diferentes tamaños de pantalla en los dispositivos.  |
| <b>Entradas</b>    |   |
| <b>Proceso</b>     | Al momento de visualizar las interfaces de la herramienta de cualquier pantalla, la misma se adaptará y mostrara un diseño ordenado y elegante. |
| <b>Salidas</b>     |   |
| <b>Prioridad</b>   | Media   |

Tabla 14. Requisito no funcional 1.

|                    |   |
|--------------------|---|
| <b>Código</b>      | RE02  |
| <b>Nombre</b>      | Tiempo de respuesta.  |
| <b>Descripción</b> | La herramienta no debe demorarse más de 10 segundos en brindar una respuesta al usuario.  |
| <b>Entradas</b>    |   |
| <b>Proceso</b>     | Al momento que el usuario interactúe con la herramienta, la misma debe dar sus respuestas a los diferentes procesos de manera rápida excediendo cómo tiempo de respuesta 10 segundos. |
| <b>Salidas</b>     |   |
| <b>Prioridad</b>   | Media   |

Tabla 15. Requisito no funcional 2.

## **Capítulo IV**

# **METODOLOGÍA**

La metodología propuesta para el desarrollo de este proyecto es “SCRUM” por ser una metodología ágil, constituye en una solución que aporta una elevada simplificación y no renuncia a las prácticas esenciales para asegurar la calidad del producto.

### **4.1 Metodología de Desarrollo ágil SCRUM**

SCRUM es un proceso que sirve para gestionar proyectos que ayuda a reducir la complejidad en el desarrollo de forma que se pueda llegar a satisfacer las necesidades de los clientes. Si bien en si no es una metodología, es un modelo de proceso empírico que se basa en la autoorganización de los equipos de trabajo para resolver y poder lidiar con problemas complejos que se van inspeccionando y adaptando continuamente cómo se aprecia en la Figura 7 (Gallego, 2012).

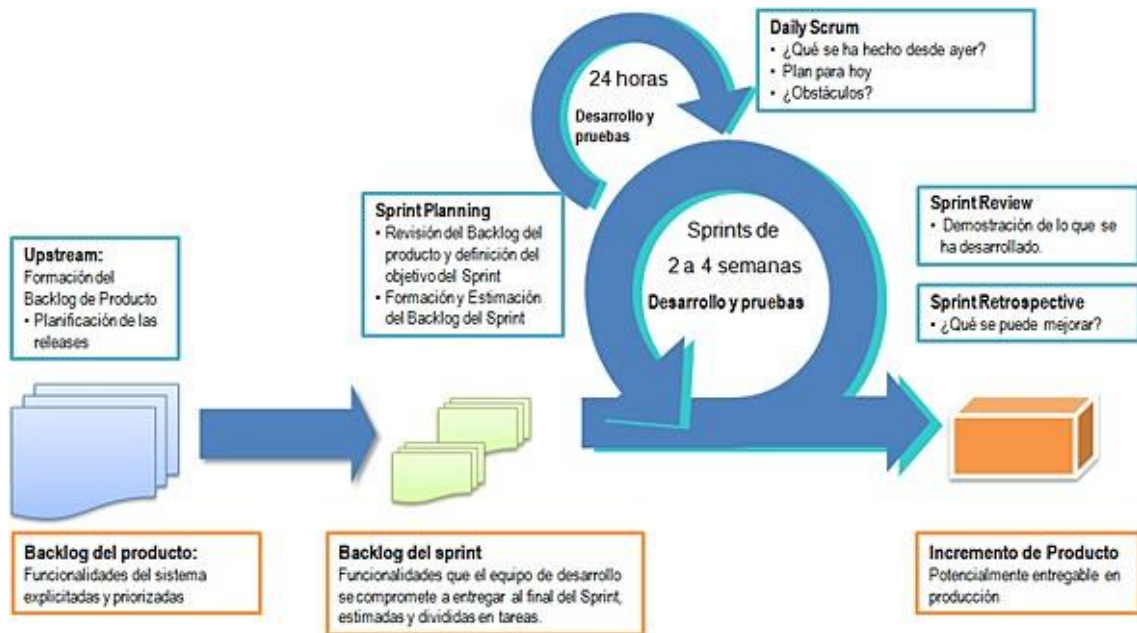


Figura 7. Ciclo de Vida SCRUM.  
(Subra, 2018)

#### 4.1.1 Características

Según (Gallego, 2012), SCRUM presenta las siguientes características:

- Incertidumbre: se plantea el objetivo sin que se haga un plan detallado del producto.
- Auto organización: cada equipo es capaz de organizarse por sí solo.
- Autonomía: son quienes se encargan de encontrar una solución en base a una estrategia adecuada.
- Autosuperación: los cambios o mejoras que sufrirán con el tiempo.
- Auto enriquecimiento: el equipo se enriquece de forma mutua, cada miembro aporta con soluciones que puedan complementar al resto del equipo.
- Control moderado: se establece el control moderado para evitar problemas, se crea un escenario de autocontrol e igualdad para no afectar a la espontaneidad y creatividad de los otros miembros.

- **Transmisión del conocimiento:** se busca que los conocimientos se puedan trasladar entre todos, para que todos puedan aprender durante el proceso.

### 4.1.2 Roles

Los roles establecen las responsabilidades dentro de un proyecto. Es así cómo se ve reflejado un equipo de trabajo (Monte, 2016):

- **Product owner (PO):** representa el enlace entre el cliente y el equipo de desarrollo. Esta persona puede estar enfocada a las TIC o al negocio donde sus funciones son varias cómo, levantar una estrategia, aclarar los objetivos, mantener el product backlog, revisar los alcances con el cliente y junto con el SCRUM máster definir los criterios de aceptación del proyecto.
- **SCRUM máster (SM):** representa el liderazgo del proyecto, una de sus funciones es ser un mentor para el equipo de desarrollo, es quien brinda un soporte a los problemas del equipo de desarrollo, además reporta, archiva y lleva un registro sobre todos los procesos que se llevan a cabo.
- **Development team (DT):** cómo su nombre lo dice, son el equipo de desarrollo, representa la flexibilidad, autoorganización y la multidisciplinaria para hacer frente y garantizar la ejecución del proyecto.
- **Stakeholder:** Son los receptores finales y son quienes hacen la aceptación del producto final acabado.

### 4.1.3 Artefactos

Son las herramientas para que los roles definidos puedan coordinar y trabajar de la mejor manera.

- **Product backlog (PB):** es el listado de funcionalidades, productos y acciones que conforman el producto que se va a construir, todo esto se representa por medio de historias de usuario con su respectiva información para el cliente.
- **Sprint backlog (SB):** son las actividades que se extraen del product backlog y se dividen en base a los sprints que se definan con el equipo de trabajo.
- **Graphs:** son los gráficos que sirven como una herramienta para poder visualizar la evolución del proyecto.
- **SCRUM board:** muestra el estado del sprint en curso donde se lo va actualizando en base a las actividades completadas y no completadas según avance del proyecto.

#### 4.1.4 Actividades

Según (Monte, 2016) Las actividades que SCRUM propone son:

- **Sprint 0:** es la fase inicial del proyecto donde el equipo hace la planificación inicial y se establecen ciertas reglas.
- **Sprint planning:** es la unidad de tiempo dentro de SCRUM que determina un ciclo de desarrollo con SCRUM y durante este tiempo se deben llevar a cabo las actividades definidas.
- **Daily meeting:** representa una reunión, la mismas se debe realizar a la misma hora y en el mismo lugar y está no debe durar más de 15 minutos y el tema principal es cómo se está llevando a cabo las actividades del proyecto.

#### 4.1.5 Fases de SCRUM

SCRUM consta de 5 fases para un ciclo desarrollo ágil.

- **Concepto:** se designa de forma general las características del producto y se asigna el cargo de desarrollo del equipo.
- **Especulación:** se establecen los límites que van a regir el proyecto cómo: los costes y agendas, en esta fase se repiten cada iteración que consiste en: desarrollar y revisar requisitos, mantener la lista de funcionalidades esperadas, plan de entrega que consta de las fechas de entrega, versiones anteriores, hitos e iteraciones, se mide el esfuerzo.
- **Exploración:** se añaden funcionalidades a la fase de especulación.
- **Revisión:** se revisa todo lo que se ha construido y se compara con los objetivos.
- **Auto enriquecimiento:** el equipo se enriquece de forma mutua, cada miembro aporta con soluciones que puedan complementar al resto del equipo.
- **Control moderado:** se establece el control moderado para evitar problemas, se crea un escenario de autocontrol e igualdad para no afectar a la espontaneidad y creatividad de los otros miembros.
- **Cierre:** se hace la entrega en la fecha acordada en la versión deseada del producto, se debe considerar que el cierre de una versión no significa que el producto llegó a su fin, sino que puede llegar a tener cambios futuros, conocidos cómo mantenimiento.

## **4.2 Fases**

Para realizar el proyecto, se aplicó la reconocida metodología SCRUM, puesto que permite la entrega de avances de forma incremental. cómo se describía anteriormente para el uso de la metodología en cuestión, se definieron los roles principales.



| <b>Rol</b>       | <b>Encargado</b>                 |
|------------------|----------------------------------|
| Product Owner    | Ing. Diego Quisi                 |
| Cliente          |                                  |
| SCRUM Máster     | Jordan Murillo                   |
| Deveploment Team | Ismael Castillo y Jordan Murillo |

Tabla 16. Roles en el proyecto.

Una vez que se definen los roles, se procede a definir los sprint para llevar acabo el proyecto, para facilitar la creación de los sprint se utiliza cómo punto de partida cada uno de los objetivos específicos definidos previamente.

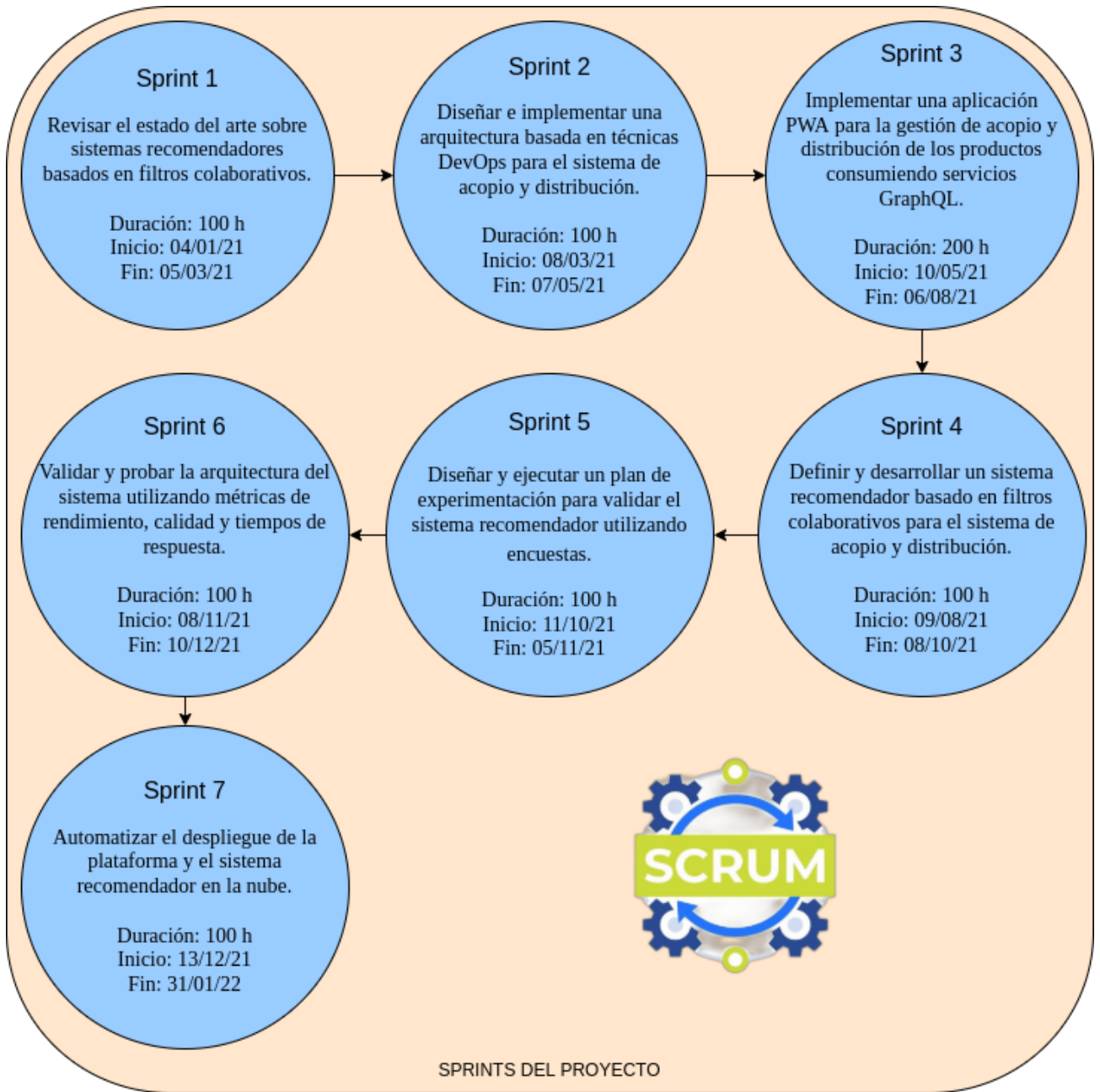


Figura 8. Sprints del proyecto.

En base a las actividades propuestas para cada sprint se estableció un lapso de tiempo que se puede visualizar en la Figura 8. Con esta planificación se obtuvo la distribución de tiempo adecuada para llevar a cabo los procesos de desarrollo, revisión, validación, y reestructuración

de cada sprint en caso de ser necesario, para posteriormente continuar con el siguiente sprint.

### 4.2.1 Primera fase

Consiste en estudiar y comprender el estado del arte de los sistemas de recomendación basados en filtros colaborativos, los diferentes tipos, implementación y demás características importantes.

En esta fase, se hizo una investigación acerca de las tecnologías a usar y el estado del arte de los sistemas de recomendación basados en filtros colaborativos, para luego hacer un análisis y generar un resumen de la información y posterior uso de la misma en el desarrollo del proyecto.

#### 4.2.1.1 Estado del arte

- En el artículo de Galán (2007) **“Filtrado Colaborativo y Sistemas de Recomendación”** se describe los diferentes tipos de algoritmos de recomendación haciendo énfasis en el filtrado colaborativo y cómo este hace uso de las valoraciones de los usuarios sobre algunos elementos de todo el conjunto de datos para predecir valoraciones a otros usuarios y recomendar en base a la mayor valoración predecir. También, se describe las principales características de estos sistemas, cómo lo son: aproximaciones estadísticas, basados en modelos y datos, métodos de evaluación y problemas referente a la implementación. El autor concluye que los sistemas de recomendación son usados principalmente en Internet, por la gran cantidad de información que existe, dando a entender que su uso aumentará a futuro.
- En el trabajo de fin de grado de Castellano (2007) **“Evaluación del uso de algoritmos colaborativos para orientar académicamente al alumnado en bachillerato”** se da a conocer la importancia que tienen los sistemas de recomendación para la toma de decisiones, en el proceso de filtrado para las posibilidades que se presentan. Por consiguiente se centra en los sistemas de recomendación en el ámbito de la educación. Se describen

ejemplos de cómo estos sistemas ayudan en la elección de tipos de enseñanza o contenidos de cursos. La finalidad del trabajo es hacer uso de los sistemas de recomendación en los alumnos y personal de orientación al momento de determinar los diferentes caminos formativos que se presentan en base a las distintas posibilidades educativas. Como conclusión el autor, señala la capacidad de los sistemas colaborativos para dar recomendaciones adecuadas de ítems a usuarios según sus preferencias sobre elementos de difícil tratamiento, evitando el análisis de los mismos, solo utilizando las valoraciones de otros usuarios.

- En el artículo de Rodríguez et al. (2016) “**Sistema de recomendación de objetos de aprendizaje a través de filtrado colaborativo**” se relata cómo los sistemas de recomendación de objetos de aprendizaje basados en filtros colaborativos, representan un apoyo a los estudiantes en su proceso de aprendizaje autónomo, objetando cómo recursos los gustos e intereses de un estudiante y tomando como base para una futura recomendación aplicando el concepto de que si dos personas se parecen y a una de ellas le gusta un ítem, hay una alta probabilidad de que a la otra también le guste ese ítem, entendiéndose ítem cómo cualquier material disponible (documentos, vídeos, imágenes, recursos, entre otros). En donde para encontrar la similitud entre usuarios se utiliza una combinación de varias métricas que miden este valor.
- En el artículo de Prada (2018) “**Sistemas de recomendación de productos para banca empresarial y corporativa: Un análisis comparativo**” se describe varias metodologías para la realización de un sistema recomendador para una entidad bancaria colombiana. Dichas metodologías se centran en utilizar información actual de los productos, su actividad económica junto con el comportamiento financiero de cada cliente, esto con el fin de generar recomendaciones a través de productos populares, filtros colaborativos, factorización de matrices y similitud de clientes. Todas estas metodologías de comparan mediante un error absoluto medio que indica la diversificación de los productos vendidos.

## 4.2.2 Segunda fase

Está conformada por el diseño de la capa de acceso a datos conjuntamente con el sistema de recomendación y la capa de negocio a través de un software ERP como es Odoo y el framework FastAPI. Esta fase será clave porque contiene la lógica que permitirá el funcionamiento del sistema.

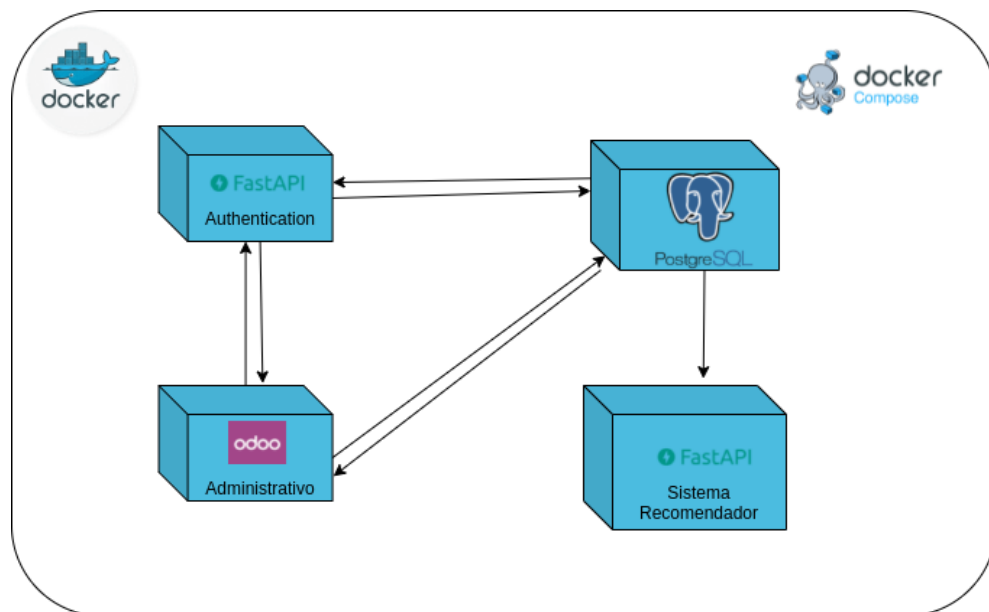


Figura 9. Distribución de la lógica de negocio.

Para iniciar, en base a las funcionalidades que presentará el sistema se define las tecnologías a usar en la construcción, recordando que todo esto pasará sobre la capa de middleware que se desarrolla en la siguiente fase.

Para la parte administrativa del sistema, se hace uso del software ERP Odoo en su versión 13.0, con este se genera las bases del sistema e inicializa la base de datos.

Se hace uso de esta herramienta, puesto que posee la configuración para manejo de usuarios y presenta una forma de desarrollo adecuada para la parte de administración, por defecto.

En este se construirá la base del sistema, que será accedido desde la aplicación, aquí se

tiene las compras y ventas para el sistema. El principal uso de este software es para llevar la parte administrativa del sistema, donde se tendrá un registro detallado de las transacciones a realizarse.

Para cumplir con los objetivos planteados se procederá a desarrollar lo relacionado al acopio y distribución de los productos. Con esto como punto de partida, se procede a trabajar en el manejo de productos. Luego de esto, se continúa con la parte de las compras para el acopio y la parte de ventas para la distribución. Para ello, se toma en cuenta los fundamentos de cómo funciona una factura, mediante la cabecera y sus detalles.

Para diseñar se crea un paquete que en Odoo se llama módulo, donde se localiza tanto la lógica como la vista del programa a desarrollar, en la Figura 10 se muestra la estructura del módulo.

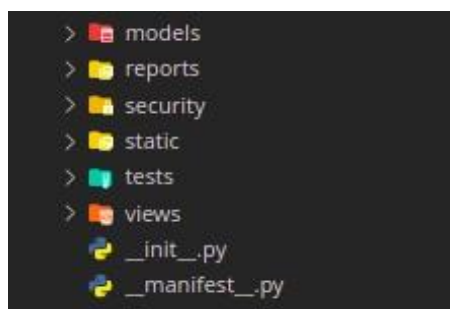


Figura 10. Distribución del código para el módulo.

Cuando ya está desarrollado esta parte, se configura el código para su despliegue, mediante Docker. Se creará el respectivo Dockerfile para montar el sistema. En este caso, se hace uso de librerías externas a las de Odoo, también se define un archivo de requerimientos que será ejecutado en el momento de construir la imagen.

```
FROM odoo:13.0

USER odoo

COPY ./odoo_requirements.txt /etc/odoo

USER root

RUN apt-get update && apt-get -y upgrade

RUN apt-get install -y python3-pandas

RUN python3 -m pip install --no-cache-dir --upgrade -r /etc/odoo/odoo_requirements.txt

USER odoo
```

Figura 11. Dockerfile Odoo.

El resultado de ejecutar el Dockerfile en la Figura 10 será la imagen del ERP Odoo para la lógica de negocio, que será usada posteriormente en el archivo yaml del *docker-compose* para su puesta en marcha.

Para la parte de la autenticación de los usuarios se hace uso de FastAPI para crear una API con las características básicas como el registro e ingreso de usuarios y algunas funciones necesarias requeridas para el funcionamiento del sistema.

Entonces se estableció la creación de varios archivos de Python, para manejar cada una de las partes de la API, que se aprecia en la Figura 12.

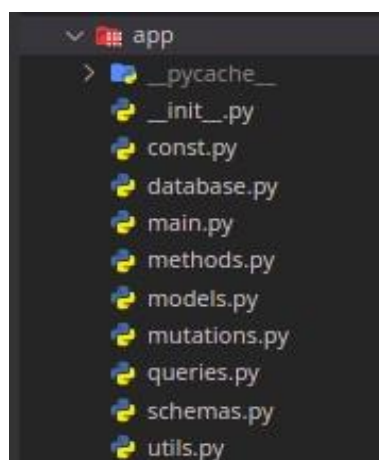


Figura 12. Distribución de la lógica de negocio para la API de autenticación.

Esta API se construyó bajo las reglas de GraphQL de FastAPI, donde se determina los *queries*, *mutations* y el *schema* a utilizar. En el caso de los *queries* y *mutations*, estos devuelven sus respuestas en base a los métodos que se generan que en GraphQL se conoce cómo *resolvers* visualizado en la Figura 13.

```
async def resolve_get_token(self, info):
    token = info.context['request'].headers.get('x-token')
    verify_auth = await verificar(token)
    return verify_auth
```

Figura 13. Resolver para un Query.

Para realizar las diferentes funciones la API se conecta a la base de datos y a servicios web de terceros, cómo es el caso de Open Street Map(OSM), para determinar nombres de calles, determinar direcciones y dar ubicaciones. En el caso de la base datos, para facilitar su manejo se emplea un ORM que se llama *SQLAlchemy*, donde se refleja solo lo necesario de la base para la autenticación de los usuarios, cómo lo son las tablas *res\_users* y *res\_partner*, que Odoogenera para guardar la información de los mismos, lo que se puede observar en la Figura 14.

```
SQLALCHEMY_DATABASE_URL = f"postgresql://{db_user}:{db_password}@db:5432/{db_name}"
engine = create_engine(SQLALCHEMY_DATABASE_URL)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
session = SessionLocal()

metadata = MetaData()
metadata.reflect(engine, only=['res_users', 'res_partner'])
Base = automap_base(metadata=metadata)
Base.prepare()
```

Figura 14. Conexión con la base de datos.

Con la ayuda del ORM la comunicación entre la base de datos y Python se hace a través de los modelos que se auto generan directamente de la base de datos. Al disponer de esto ya se pueden generar los métodos o funciones que se llaman en los *resolvers*, aquí también se agregara la parte de JWT para la verificación y validación de la autenticación de los usuarios.

Cada *query* o *mutation* devuelve su respuesta en base al esquema definido para la respuesta,



que se muestra a continuación, en la Figura 15.

```
class RespuestaAuth(ObjectType):
    estado = Boolean(required=True)
    usuario = Field(UserType)
    mensaje = String(required=True)
    token = String(required=False)
```

Figura 15. Esquema para la respuesta de autenticación.

cómo siguiente paso se tiene ya la definición del servidor de FastAPI GraphQL que se hace con el código que se encuentra en la Figura 16.

```
app = FastAPI()
app.add_route("/graphql", GraphQLApp(
    schema=Schema(query=queries.Query, mutation=mutations.Mutation),
    executor_class=AsyncioExecutor
))
```

Figura 16. Código para definir el servidor FastAPI.

cómo se trabaja con Docker se define el archivo de construcción de la imagen del API de autenticación mediante el archivo *Dockerfile*, que se muestra en la Figura 17

```
FROM python:3.8

ENV PYTHONUNBUFFERED=1

WORKDIR /code

COPY ./requirements.txt /code/requirements.txt

RUN python -m pip install --upgrade pip

RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

COPY ./app /code/app
```

Figura 17. Dockerfile FastAPI Sistema Recomendador.

El resultado de ejecutar el Dockerfile será la imagen del API GraphQL para la autenticación,

que será usada posteriormente en el archivo yaml del *docker-compose* para su puesta en marcha.

Continuando con el desarrollo de esta fase se tiene la creación del sistema recomendador que se describirá a continuación.

Para empezar, se carga la fuente de datos de la cuál hará uso el sistema recomendador. Estos datos son dados por la empresa, que corresponde a datos de usuarios y sus respectivos viajes a través de ocho ciudades del Ecuador. Posteriormente para el uso de estos datos se debe hacer un análisis y limpieza de los mismos.

Debido a que la información se encuentra en archivos separados se procede a unirla, para luego realizar la matriz de correlación y analizar cómo se relacionan los datos y obtener criterios para realizar las recomendaciones que se visualiza en la Figura 18.

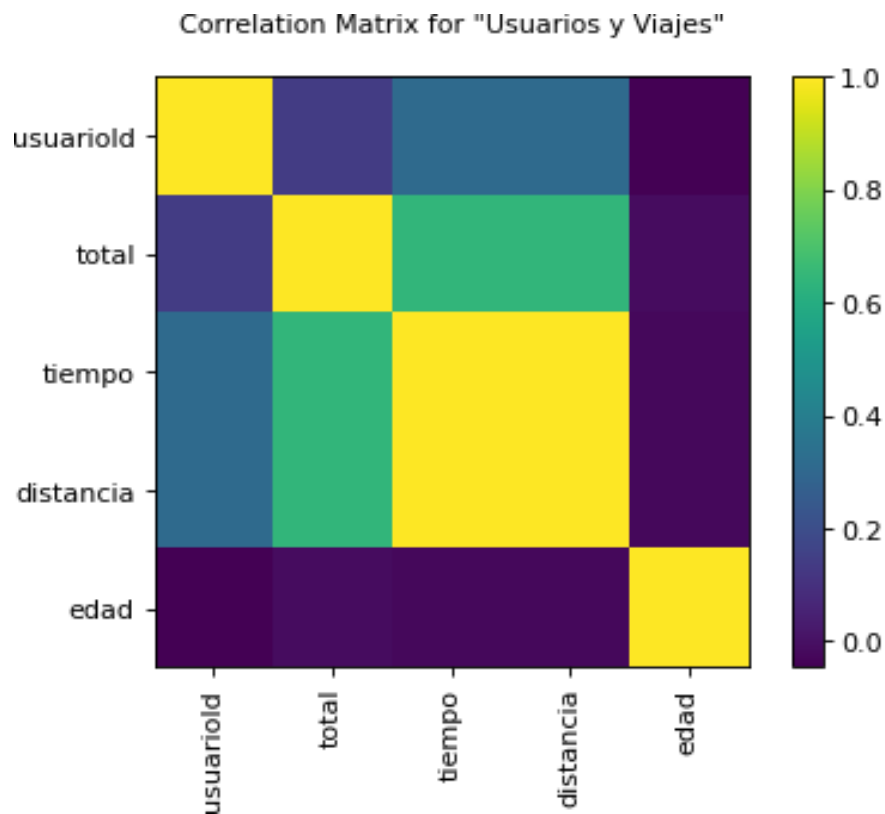
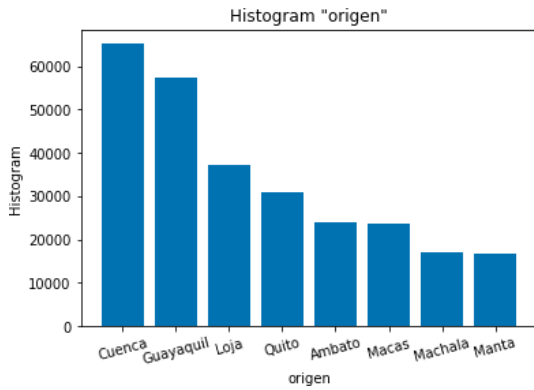


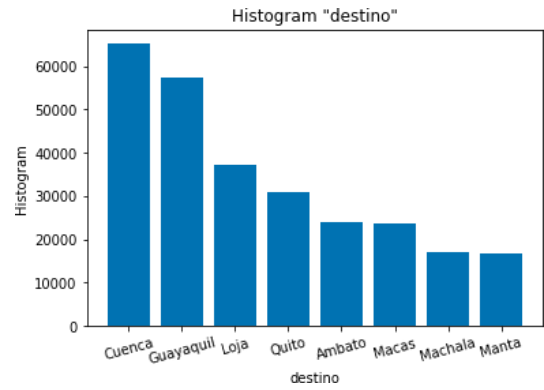
Figura 18. Matriz de Correlación.

Con ayuda de la gráfica se puede determinar que hay una correlación entre el tiempo y la dis-

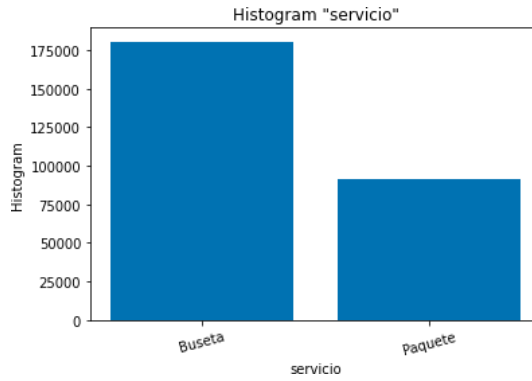
tancia, debido a que son variables dependientes. Para describir los datos se realizó histogramas de las ciudades de origen y destino.



(a) Histogramas del origen.



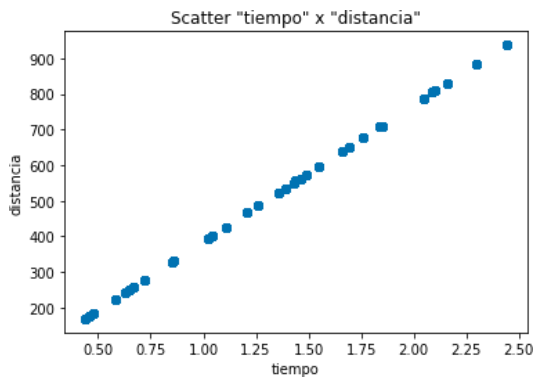
(b) Histograma del destino.



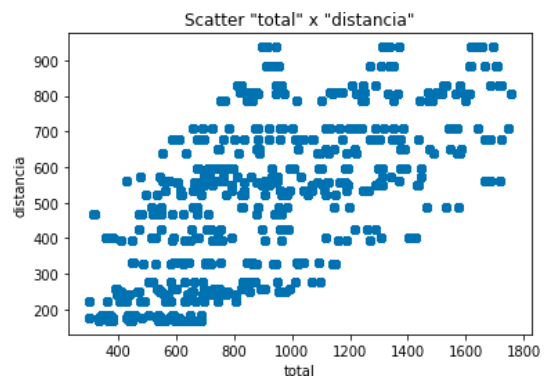
(c) Histograma de los servicios.

Figura 19. Histogramas de los datos.

Para terminar el análisis en base a la correlación se gráfica un mapa de puntos para las relaciones tiempo-distancia y total-distancia.



(a) Mapa de puntos tiempo-distancia.



(b) Mapa de puntos total-distancia.

Figura 20. Mapa de Puntos.

Después de analizar se determina descartar la técnica de regresión lineal porque no hay datos correlacionados para el sistema recomendador.

Luego de esto se procedió a aplicar una recomendación sencilla basada en KNN <sup>1</sup>, agrupando a los usuarios por el origen, destino y la edad. Para aplicar esta recomendación se debe codificar las columnas categóricas en numéricas, lo cual es conocido como codificación de etiquetas y la librería usada ya presenta una herramienta conveniente para automatizar este proceso. Dicha librería es *sklearn*, que ya implementa algunos algoritmos, como es el caso de KMeans.

El principio de este algoritmo es agrupar los datos basados en usuarios, en base a los criterios a los siguientes criterios: se usará la edad, el origen y el destino para agrupar, a estos grupos se les llama *clúster*. Luego se procede con la gráfica para determinar la cantidad de grupos a utilizar que se visualiza en la Figura 21.

<sup>1</sup>K-Nearest-Neighbor - Vecinos cercanos

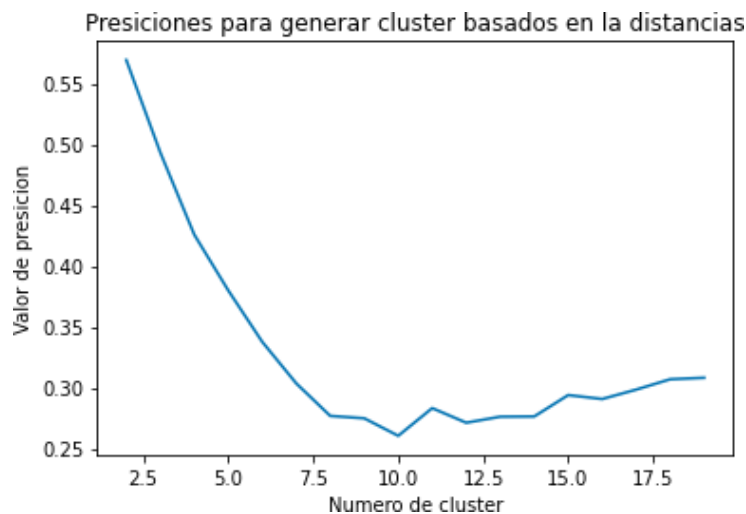


Figura 21. Precisiones para generar *clúster* basados en la distancias.

Se puede observar que la mejor precisión es obtener cuatros grupos, optando por está configuración, lo que se recomienda, en base a estos datos, es usuarios que tengan la misma edad, el origen y destino, es una recomendación bastante sencilla cómo experimento.

Se procede a probar el sistema dándole los parámetros necesarios, cómo lo son origen, destino y edad. Luego de ver los resultados, se determina que no puede ser una buena recomendación, puesto que sólo genera grupos en base a las ciudades, lo cual resulta en una sugerencia básica.

Al no obtener buenos resultados de recomendación, se decide realizar un proceso de filtrado colaborativo para mejorar la recomendación. Para el filtrado colaborativo se tiene en cuenta los enfoques (usuario e ítem), en el filtrado basado en ítem es la similitud entre ambos. En el ámbito actual no se puede usar el de ítem, porque al partir de un conjunto de datos para las predicciones los ítems disponibles son estáticos y son muy pocos, lo que causaría que no genere una recomendación óptima debido a las 8 ciudades que integran la base de datos.

Se intenta encontrar usuarios que tengan preferencias de origen y destino parecidas, para entonces recomendar lugares que se hayan parecido al ingreso anterior.

A continuación, se presenta el proceso para crear el sistema recomendador basado en el

usuario:

- Elegir un usuario con los lugares de origen y destino que el usuario ha mirado.
- Basado en su índice de selección de compras de boletos, encuentra a los primeros X vecinos.
- Obtener el registro de los lugares que miró el usuario para cada vecino.
- Calcular un puntaje de similitud utilizando alguna fórmula.
- Recomendar los lugares con los puntajes más altos.

cómo primer punto, en base a los datos se genera una nueva matriz que identifique la preferencias, que, en términos de viajes, van a ser la cantidad de veces que un usuario a comprado un boleto cómo origen o destino desde una ciudad.

Se procede con el filtrado colaborativo en base al vecindario, creando una matriz usuario-ciudad. Se obtiene una matriz de escala estándar a partir del número total de usuarios y lugares origen o destino que se obtuvieron anteriormente y luego se forma una matriz de elementos de usuario asignando la preferencia.

Se normalizan los datos basados en la media, una técnica simple para normalizar los datos, que consiste es restar el promedio del valor de la característica en cuestión de cada usuario de todo el valor.

Luego de normalizar los datos se hace uso de la función `pairwise_distances` para calcular la similitud del coseno. Con los resultados obtenidos, cuanto más cercano a uno, mayor similitud entre aquellos usuarios. Los resultados se muestran en la Figura 22.

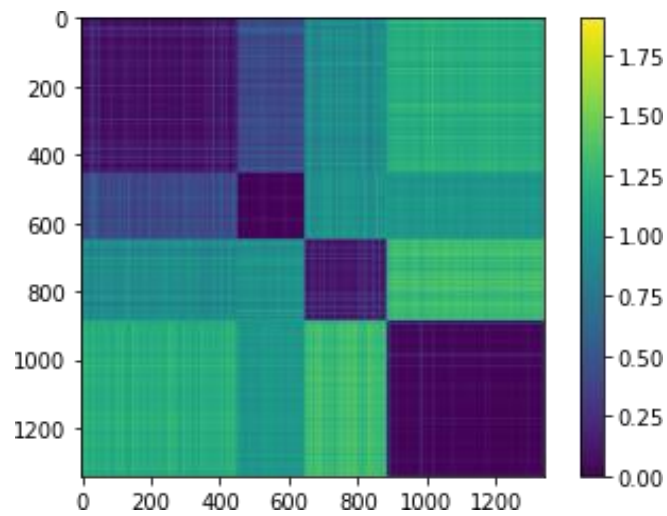


Figura 22. Similitud entre usuarios.

En base a lo obtenido, para empezar, se calcula la similitud del usuario mediante la multiplicación de la matriz de similitud y la matriz de puntuación, con esto se obtiene el peso final recomendado de acuerdo a la fórmula de la Ecuación 1, es decir, la similitud del usuario se multiplica por la preferencia del usuario.

$$\hat{r}_{ui} = \frac{\sum_{u'} sim(u, u')r_{u'i}}{\sum_{u'} |sim(u, u')|} \quad (1)$$

Después se crea un nuevo DataFrame, que lleva la similitud pronosticada de los usuarios en base al origen y al destino, teniendo cómo preferencia el número de veces que la ciudad tiene cómo origen o destino, que se muestra en la Figura 23.

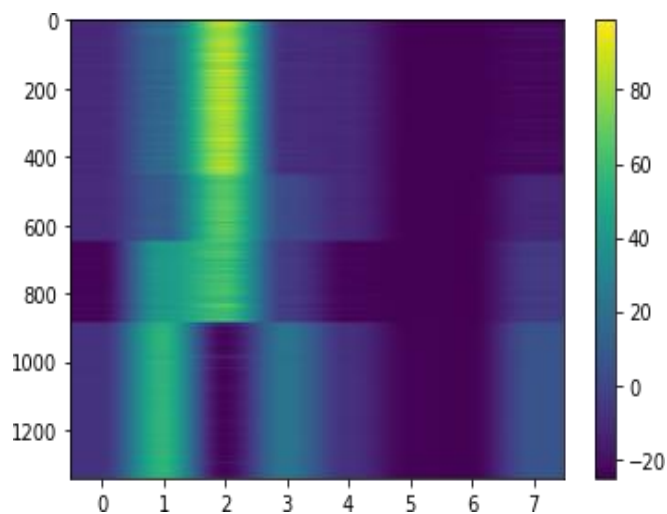


Figura 23. Pronóstico de similitud entre usuarios.

Luego, se divide los datos en entrenamiento(80%) y prueba(20%), y sobre los datos de prueba se aplica la fórmula para obtener el error, denominada Mean Squared Error(MSE), debido a que no se puede afirmar que el resultado sea correcto del todo ya que es una predicción. Por el contrario si se trata de sistemas de clasificación se pueden aplicar otras métricas de precisión. Para el error de los datos de entrenamiento se tiene un error de 52.04 y para los datos de prueba un error de 53.27, los cuales al estar cerca se puede determinar que producirán buenas sugerencias, caso contrario fuera que el MSE de los datos de prueba fuera el doble o la mitad respecto al valor de los datos de entrenamiento.

Para finalizar solo queda agregar el nuevo usuario al DataFrame, para que pase por la predicción y se le pueda recomendar en base a los datos de entrada, que son en este caso, las veces que ha viajado a una ciudad, tanto de origen cómo destino.

Por otro lado, cómo se trabaja con Docker se define el archivo de construcción de la imagen del API de autenticación mediante el archivo *Dockerfile*, que se muestra en la Figura 24.



```
FROM python:3.8

ENV PYTHONUNBUFFERED=1

WORKDIR /code

COPY ./requirements.txt /code/requirements.txt

RUN python -m pip install --upgrade pip

RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt

COPY ./app /code/app
```

Figura 24. Dockerfile FastAPI Sistema Recomendador.

El resultado de ejecutar el Dockerfile será la imagen del API GraphQL para la autenticación, que será usada posteriormente en el archivo yaml del *docker-compose* para su puesta en marcha.

### 4.2.3 Tercera fase

Consiste en el diseño del middleware (GraphQL), que facilita a los usuarios hacer solicitudes como el envío de formularios desde los navegadores web, así como la devolución de información del servidor a las peticiones determinadas, es decir, es el encargado de la comunicación de la aplicación web progresiva con la lógica de negocio.



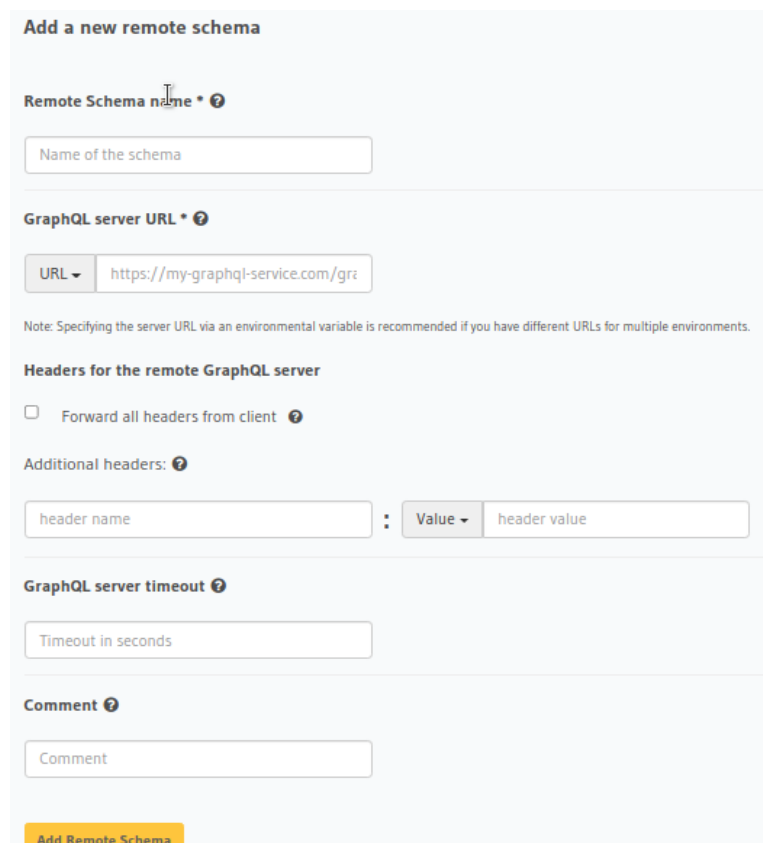
Figura 25. Diagrama del middleware Hasura.

Para dar el funcionamiento correcto al middleware se procede a conectar cada uno de las APIs creadas y la conexión con la base de datos PostgreSQL. Para conectar la base se dirige a la opción DATA en la consola de Hasura y se procede a conectar la base, mediante el formulario que se ve en la Figura 26.

Formulario para conectar la base de datos en Hasura. Incluye campos para Database Display Name, Data Source Driver (PostgreSQL), Database URL y un botón Connect Database.

Figura 26. Formulario para conectar la base de datos en hasura.

Para conectar las API externas de GraphQL, se dirige a REMOTE SCHEMAS en la consola de Hasura y se procede a conectar las API, mediante el formulario que se ve en la Figura 27.



The image shows a web form titled "Add a new remote schema". It contains several sections: "Remote Schema name" with a text input field; "GraphQL server URL" with a dropdown menu set to "URL" and a text input field containing "https://my-graphql-service.com/graphql"; a note about using environmental variables; "Headers for the remote GraphQL server" with a checkbox for "Forward all headers from client" and an "Additional headers" section with a key-value pair input; "GraphQL server timeout" with a text input field; and "Comment" with a text input field. At the bottom is a yellow "Add Remote Schema" button.

Figura 27. Formulario para conectar una API de GraphQL externa.

Luego de realizar los pasos anteriores, ya se tiene unificado en un solo lugar las diferentes funcionalidades del sistema.

#### 4.2.4 Cuarta fase

Está conformada por el diseño e implementación de una aplicación web progresiva (PWA), consta de la presentación de la información a los usuarios con los servicios que ofrece el sistema, de esta forma su funcionamiento se basa en el consumo de los servicios proporcionados por GraphQL (middleware).

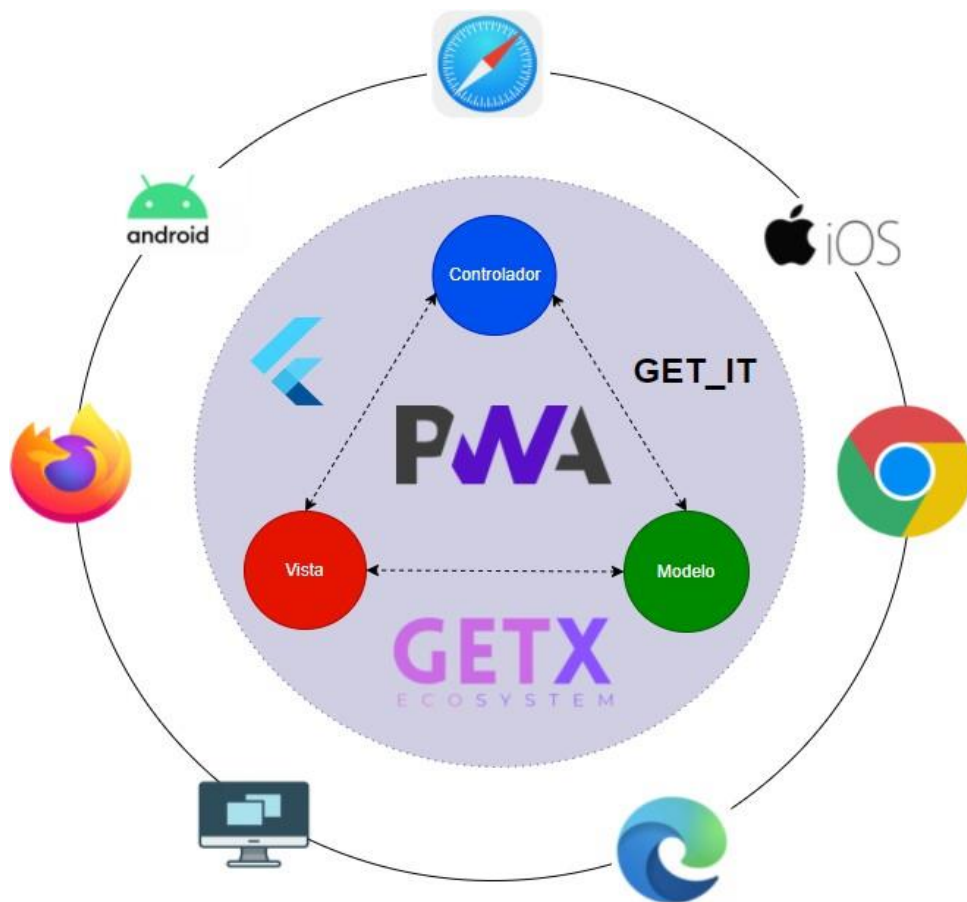


Figura 28. Aplicación Web Progresiva.

Flutter es un framework de desarrollo en auge que se emplea para el desarrollo multi-plataforma, entre sus principales ventajas se puede mencionar el rendimiento y la interfaz de usuario. Creado por Google y su primera versión estable fue lanzada a finales de 2018, a partir del primer lanzamiento ha sufrido grandes cambios beneficiando a la comunidad de desarrolladores donde a partir de la versión 2, se incluyó el soporte para PWA (Aplicaciones Web Progresivas) con el fin de cerrar la brecha, entre las aplicaciones móviles y la web. Brindando así la posibilidad de trabajar con una misma base de código en un solo proyecto cubriendo el funcionamiento en la mayoría de las plataformas.

El proceso para el desarrollo de una aplicación con el framework implica la base funda-

mental que en Flutter todo es un widget incluso la propia aplicación, estos widgets son los que brindan la posibilidad de realizar diseños intuitivos y profesionales que es lo que caracteriza a esta herramienta. Los widgets pueden ser agregados, editados, remplazados y eliminados dinámicamente de la interfaz. Lo que facilita a los desarrolladores al momento de realizar un proyecto.

Para el lado del frontend, al igual que otros frameworks, busca imponer un código limpio a través de gestores de estado cómo lo son BLOC, GETX, GETIT, PROVIDER, etc. Evitando así el famoso *boilerplate*, generalmente esto queda a elección del desarrollador, en este caso buscando generar un código limpio y de utilidad se utilizó uno de los gestores de estado más usados para el ecosistema, el cual es representado por GETX, es un paquete desarrollado por la comunidad de Flutter que brinda estándares proporcionando el manejo de rutas, inyección de dependencias, control de carga y además proporciona controladores para la interfaz de usuario, que permite la capacidad de acelerar y facilitar el desarrollo.

Presentar un código limpio no solo con Flutter si no en un ámbito general siempre va de la mano de una arquitectura limpia la cual se ancla a un patrón de diseño, en nuestro caso se empleo una arquitectura de 3 capas que son Data, Domain y View la cual trabaja internamente de manera similar al patrón Modelo Vista Controlador, pero no igual. Empleando esto se logra una separación de todos los componentes y generando código reutilizable, facilitando así generar una óptima gestión del proyecto donde a futuro su mantenimiento y escalabilidad queda garantizada evitando programación de funcionalidades y apegándose al gestor de estado empleado y dando la posibilidad de emplear otros gestores de estado en caso de ser necesarios.

Toda herramienta de desarrollo posee amplios conjuntos de librerías para reutilización de código y Flutter no se queda atrás, teniendo cómo repositorio de librerías a pub.dev el cual contiene un amplio repertorio que facilita el desarrollo, y la librería principal para el funcionamiento de nuestra aplicación es la librería `graphql-flutter`, esta dependencia brinda la posibilidad de consumir toda la lógica que provee el middleware (backend) a través de queries, mutations y

subscriptions que combinado con un gestor de estado se logra un código netamente limpio aprovechando la funcionalidad de los widgets en su totalidad sin alguna limitación.

Finalmente, algunas de las interfaces que se desarrolló se presentan a continuación:

- **Login:** está pantalla permite al usuario ingresar a la aplicación por medio de sus credenciales, en caso de que el mismo no tenga una cuenta se le presenta un botón que le llevara a realizar dicho proceso de registro.

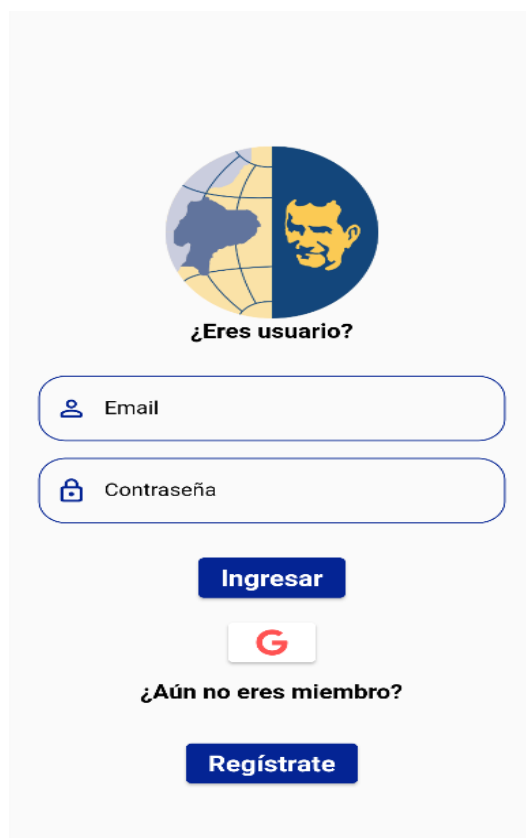


Figura 29. Pantalla de inicio de sesión.

- **Registro:** está pantalla le permite al usuario realizar el proceso de registro para la creación de su cuenta en la aplicación, para esto se solicitan los datos personales básicos del usuario cómo lo es nombres, apellidos, correo, contraseña, etc. Una vez completado el proceso de registro el usuario podrá ingresar directamente a la aplicación que se ve en la Figura 29.

The image shows a mobile application registration screen. At the top, there is a dark blue header with a white back arrow and the text 'Regístrate'. Below the header, there are six rounded rectangular input fields stacked vertically, each containing a label: 'Nombres y apellidos', 'Correo electrónico', 'Telefono', 'Celular', 'Direccion', and 'Contraseña'. At the bottom center of the form area, there is a dark blue button with the white text 'Crear'.

Figura 30. Pantalla de registro.

- **Tickets de viaje:** la siguiente pantalla muestra los tickets de viaje disponibles para una ruta especificada por el usuario, en base a esto el usuario podrá realizar la compra o reserva de su ticket para una fecha determinada.

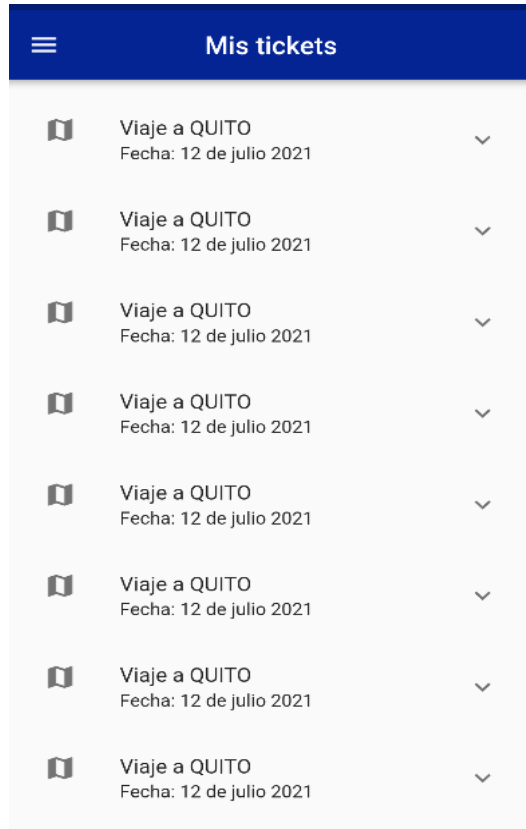


Figura 31. Pantalla de tickets de viaje.

- **Perfil:** en esta pantalla se muestran los datos de sesión del usuario, así como la posibilidad de actualizarlos si el usuario lo requiere.





Figura 32. Pantalla de perfil de usuario.

- **Reserva:** en esta pantalla el usuario podrá realizar la reserva de su ticket, según una fecha especificada, tomando en cuenta los lugares recomendados que ofrece el sistema recomendador para el usuario.



Figura 33. Pantalla de reserva de ticket.

- **Recomendaciones:** en esta pantalla el usuario podrá visualizar los destinos más visitados y con mejores recomendaciones, y así aprovechar los tickets si su destino se encuentra en el conjunto de mejores recomendados.



Figura 34. Pantalla de recomendaciones.

- **Mejor recomendación:** en esta pantalla el usuario podrá visualizar el destino con la mejor recomendación y así visualizar información del destino y aprovechar ofertas de tickets o promociones.



Figura 35. Pantalla del destino mejor recomendado.

Finalmente, en la Figura 36, se puede observar la distribución del código fuente.

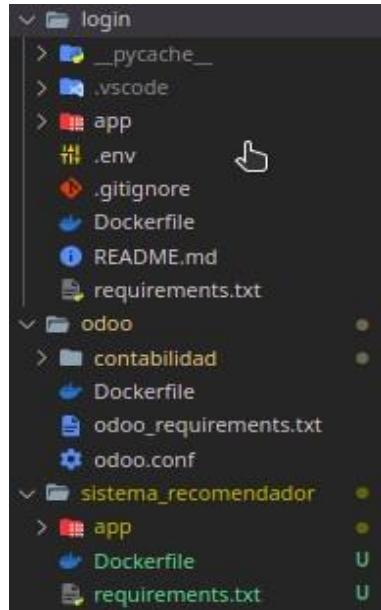


Figura 36. Estructura del backend.

#### 4.2.5 Quinta Fase

Consiste en el diseño e implementación de una técnica sobre DevOps para plataformas basadas en servicios, en este caso se tiene acceso aun servidor de la Universidad para el backend y para el frontend, se usará un servidor en Google Cloud; para el desarrollo y despliegue del sistema.

Antes de empezar a emplear la técnica DevOps, se hará la configuración del archivo yaml de *docker-compose*, donde se describirá cada uno de los subsistemas que conforman el backend junto al middleware. Como en el proceso de desarrollo ya se creó los correspondientes Dockerfile de cada componente, ahora solo falta crear el *docker-compose*, en consecuencia, se hace uso del mismo para construir las imágenes de cada componente, que luego pasarán a ejecutarse en su respectivo contenedor. En la Figura 37, se puede apreciar una parte del archivo.

```

version: '3.8'
services:
  db:
    image: postgres:10
    container_name: postgres
    environment:
      - POSTGRES_DB=${DB_NAME}
      - POSTGRES_PASSWORD=${DB_PW}
      - POSTGRES_USER=${DB_USER}
      - PGDATA=/var/lib/postgresql/data/pgdata
    ports:
      - "5432:5432"
    restart: always
    volumes:
      - data-odoo-db:/var/lib/postgresql/data
  web:
    build:
      context: odoo
      dockerfile: Dockerfile
    image: serverodoo:latest
    container_name: serverodoo
    command: -u all -d odooscafp
    depends_on:
      - db
    ports:
      - "8069:8069"
    restart: always
    volumes:
      - data-odoo-web:/var/lib/odoo
      - ./odoo/odoo.conf:/etc/odoo/odoo.conf
      - ./odoo/contabilidad:/mnt/extra-addons

```

Figura 37. Parte del archivo docker-compose.

Todos las variables estáticas, se definieron en un archivo `.env`, para tener una mejor organización.

Una vez configurado lo anterior, se procede a aplicar la técnica para DevOps que provee el sistema de control de versiones, donde se aloja el código del proyecto, que és GitLab.

Para resumir el proceso, se busca lo siguiente:

- Se procede a confirmar y subir los cambios a la rama *main* del repositorio.
- GitLab CI detecta si hay cambios en la rama y procede a reflejarlos en el repositorio.
- GitLab CI activará el GitLab Runner para iniciar el proceso de canalización(pipeline) de

la rama en cuestión, la cual buscara el código más reciente, procederá a construir y poner en marcha el proceso *docker-compose*.

- Si no se produce ningún error en el proceso, al final se tiene el sistema desplegado en la última versión, según la rama.

Para tener una visualización del proceso, se realizara un diagrama, que se muestra a continuación en la Figura 38.

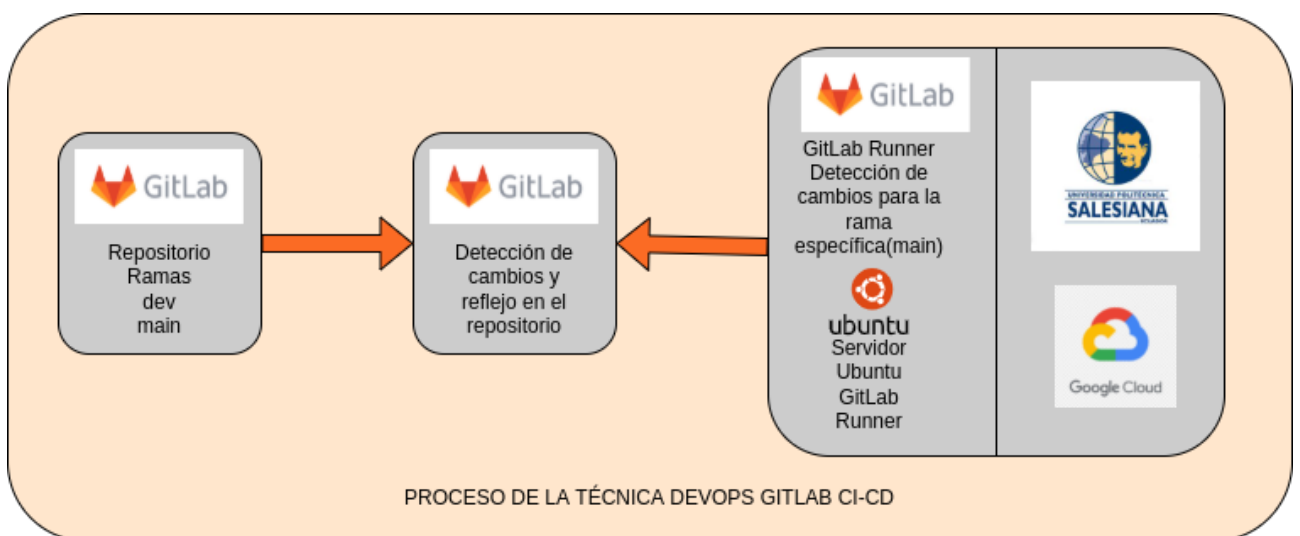


Figura 38. Estructura de la técnica DevOps.

Para la realización de los siguientes pasos, se toma cómo guía lo descrito por (Bradley, 2019). Se debe instalar y configurar el GitLab Runner en el servidor, mediante los siguientes pasos:

- Ejecutar:

```
$ curl -L https://packages.gitlab.com/install/repositories/runner/  
gitlab-runner/script.deb.sh | sudo bash
```

- Luego se procede a la instalación:

```
$ sudo apt-get install gitlab-runner
```

- Después de eso, en el repositorio de GitLab se dirige a Mi Repositorio → Settings → CI/CD → Runners. Una vez en la ubicación se desactiva la opción *Shared Runners* y se revisa la configuración de *Specific Runners*, se sigue los pasos con el token de registro para el servidor y se procede con el siguiente comando:

```
$ sudo gitlab-runner register --url https://gitlab.com/ --registration  
-token $REGISTRATION_TOKEN
```

Se debe reemplazar el token generado en el comando en la parte de *\$REGISTRATION\_TOKEN*.

Para culminar este proceso, se siguen los pasos que genera el comando, se recuerda que la información dada en ese proceso se usa luego en la configuración final de la técnica en el archivo *.gitlab-ci*, de extensión *yml*.

- Una vez terminada la configuración anterior, se ejecuta:

```
$ sudo nano /etc/sudoers
```

aquí se agrega la siguiente línea:

```
gitlab-runner ALL=(ALL) NOPASSWD: ALL
```

se guarda los cambios y se cierra el archivo.

- Para finalizar se procede a crear y configurar el archivo *.gitlab-ci*, que se puede apreciar en la Figura 39.



```

image: docker:latest
services:
  - docker:dind

stages:
  - test
  - deploy

step-main:
  stage: deploy
  only:
    - main
  tags:
    - dmain
  script:
    - sudo apt-get install -y python3-pip
    - sudo pip install docker-compose
    - sudo docker image prune -f
    - sudo docker-compose -f docker-compose.yml build --no-cache
    - sudo docker-compose -f docker-compose.yml down && sudo docker-compose -f docker-compose.yml up -d

```

Figura 39. Estructura de la técnica DevOps.

Después de seguir los pasos anteriores, cuando se ejecute un *push* a la rama establecida, esto automáticamente ejecutará el proceso de la técnica en base al archivo configurado.

| Status  | Pipeline   | Triggerer | Stages            |
|---|--|-----------|-------------------|
| <p>passed</p> <p>00:01:54</p> <p>1 week ago</p> | <p>Actualizacion de Ventas - Compras - Retenciones - Gu...</p> <p>#472912974 main → 1c1acc83</p> <p>latest</p> |           | <p>✓</p> <p>⋮</p> |
| <p>passed</p> <p>00:01:44</p> <p>1 week ago</p> | <p>Correccion tiempo de duracion token</p> <p>#471016623 main → 0c011406</p>                                   |           | <p>✓</p> <p>⋮</p> |
| <p>passed</p> <p>00:01:51</p> <p>1 week ago</p> | <p>Url Database Correction</p> <p>#470963873 main → 52df2152</p>   |           | <p>✓</p> <p>⋮</p> |
| <p>passed</p> <p>00:02:00</p> <p>1 week ago</p> | <p>Add Update SQL Gestion Init</p> <p>#470960571 main → 0d13cf07</p>   |           | <p>✓</p> <p>⋮</p> |
| <p>passed</p> <p>00:01:50</p> <p>1 week ago</p> | <p>Correccion Error Vistas Inventario</p> <p>#470797581 main → ac188224</p>                                    |           | <p>✓</p> <p>⋮</p> |

Figura 40. Tabla de Pipelines.

## 4.3 Arquitectura

Se le puede denominar como el proceso para diseñar la estructura de todo el proyecto, donde se especifican la distribución, comunicación y tecnologías a utilizar en el desarrollo. Es un paso muy importante a que ayuda a visualizar de forma generalizada cómo va a estar estructurado el sistema completo, cómo se puede apreciar en la Figura 41.

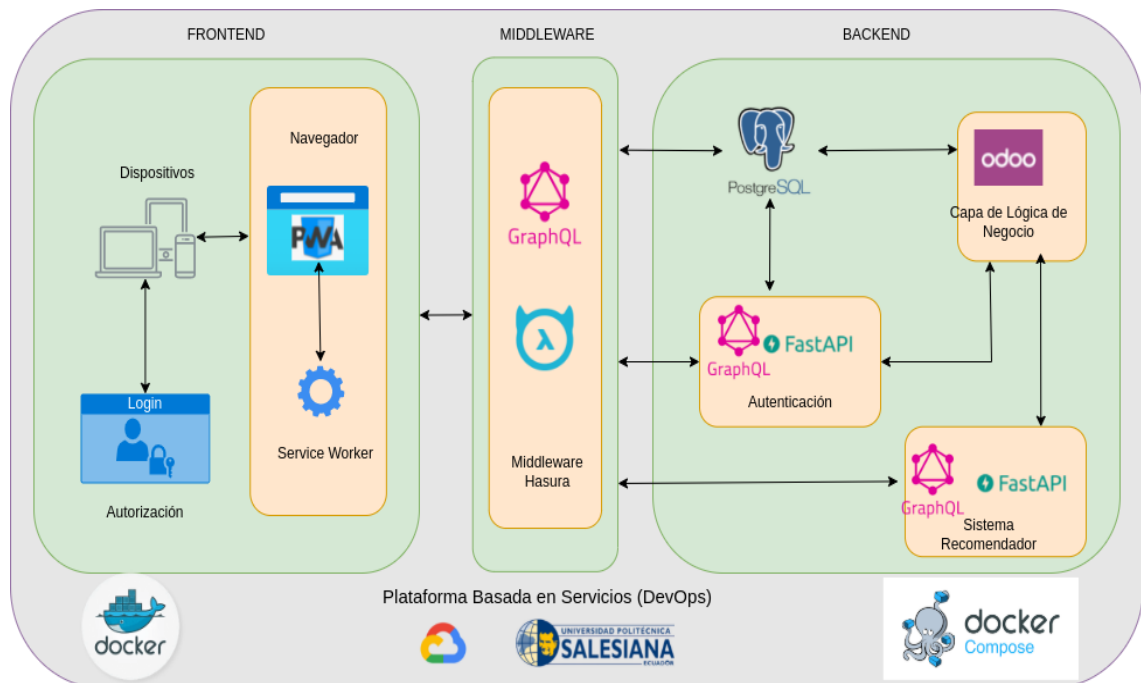


Figura 41. Arquitectura del sistema.

### 4.3.1 Backend

Es el elemento que comprende la entrada a las operaciones y funcionalidades del sistema, puede ser accedido por los usuarios, dependiendo sus permisos. Conformado por la lógica de negocio y la interacción de la información de la base de datos. A continuación, se presenta una lista de los principales módulos del proyecto:

- **Lógica de negocio o capa de negocio (Odoo):** Es la capa que hace referencia a la parte del sistema que se encargará de codificar las reglas de negocio de la realidad, está capa

determina el flujo de la información en el ámbito de programación. Mediante el software ERP Odoo, ya se accede directamente a la base de datos PostgreSQL para el guardado de la información. Como se determinó en el proceso de desarrollo aquí se encuentra la parte de inventario de productos y la facturación (compras y ventas).

- **Sistema de Autenticación (FastAPI):** Es la capa encargada de permitir el ingreso al sistema por parte de los usuarios. Con la ayuda de Python y su framework FastAPI se desarrolló una API GraphQL que se unirá al middleware en Hasura.
- **Sistema Recomendador (FastAPI):** Es el motor que permite el análisis de los datos y la entrega de resultados basados en filtros colaborativos, cálculos matemáticos que son interpretados a través de un algoritmo para generar información útil para los usuarios. De la misma manera, con la ayuda de Python y su framework FastAPI se desarrolló una API GraphQL que se unirá al middleware en Hasura, para acceder a sus funciones.

### 4.3.2 Middleware

Esta parte del sistema permite la comunicación entre los distintos módulos que se generarían en el desarrollo del sistema, funcionando como un intérprete que establece el intercambio y la administración de datos en aplicaciones distribuidas. Esta implementación hace uso de Hasura, para de esta manera comunicar el backend con el frontend.

### 4.3.3 FrontEnd

Es el elemento del sistema que representa la presentación al usuario que contiene las funcionalidades dadas por los desarrolladores, tanto para acceso general o de registro, que se obtiene con algún tipo de autenticación.

Para la construcción de la aplicación web progresiva, se empleó una arquitectura limpia DDV (Data-Domain-View), es un conjunto de reglas, principios y patrones de diseño que sirven

para proveer facilidad al proceso de construcción del software, así como su escalabilidad y mantenimiento. Al seguir este conjunto de recomendaciones las capas por las que se encuentra conformada la aplicación se vuelven agnósticas a la plataforma y al lenguaje de programación empleados, porque de esta manera en el futuro agregar cambios a cualquiera de los componentes se vuelve netamente sencillo y no se afecta a los componentes entre sí, debido a que el cambio solo va afectar a la lógica del funcionamiento nuevo que se desee emplear.

Los componentes de la arquitectura son:

- **Data:** es la capa que se encarga de comunicarse con las dependencias externas que requiere la aplicación para obtener los datos. Son los repositorios de información para el funcionamiento de la aplicación los cuales pueden consistir en servicios REST, SOAP, Middleware GraphQL, Firebase, etc.
- **Domain:** representa la capa que abarca toda la lógica de negocio de la aplicación, en la cual se encuentra el código que debe ser escéptico de cualquier otra parte de la aplicación es decir es el código reutilizable en base a los casos de uso que presente cada funcionamiento del software.
- **View:** es la capa encargada de presentar la información al usuario final. En nuestro caso, consiste en todo el código de Flutter que ayuda en este proceso a través del empleo de páginas en conjunto con widgets y el empleo de navegación para la entrada y salida de los datos.

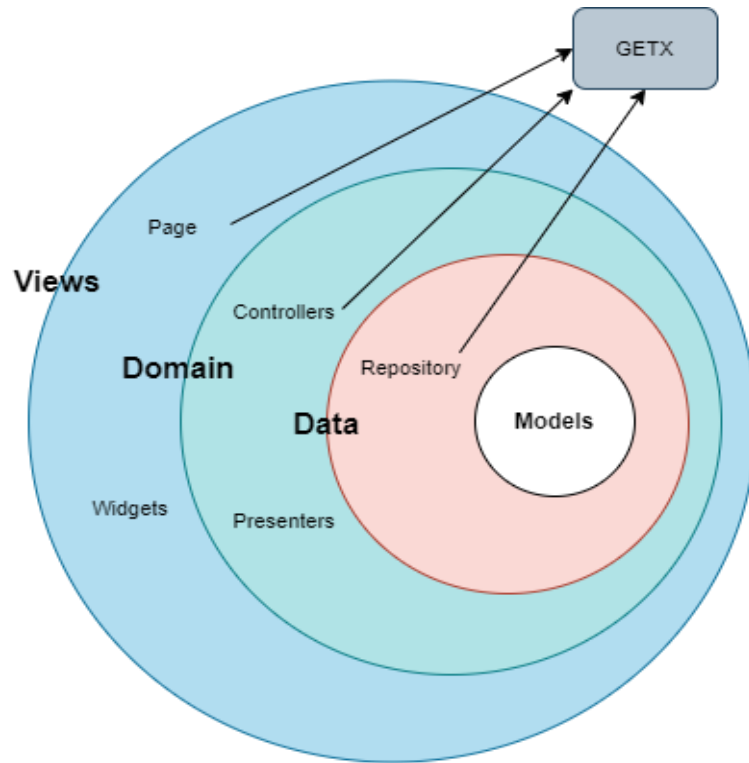


Figura 42. Arquitectura Frontend.

La arquitectura incorpora GetX como gestor de estado, al complementar la arquitectura con este micro framework, se puede simplificar el manejo de estados, enrutamiento y llevar a cabo la inyección de dependencias de una manera eficiente. Al usar GETX dentro de la capa domain por medio del controlador evita el uso de StatefulWidget, puesto que posee su propio ciclo de vida, facilitando separar la lógica de negocio de la vista y siempre que sucede algún cambio cuando se usa el controlador, la vista se actualiza automáticamente.

A través del manejo de esta arquitectura las carpetas del proyecto quedan estructuradas de la siguiente manera como se muestra en la Figura 43.

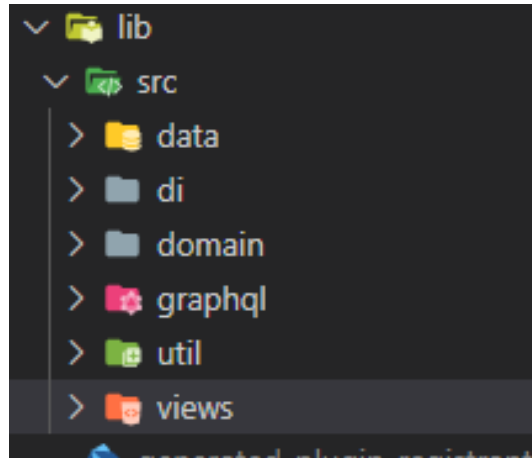


Figura 43. Estructura del proyecto.

# Capítulo V

## ANÁLISIS DE RESULTADOS

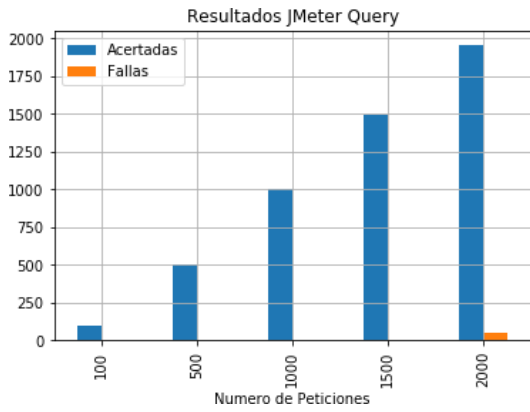
En este capítulo se describe el proceso de las pruebas realizadas hacia el sistema. Se realizaron pruebas de carga, unitarias y una validación de la aplicación mediante una encuesta.

### 5.1 Pruebas de carga

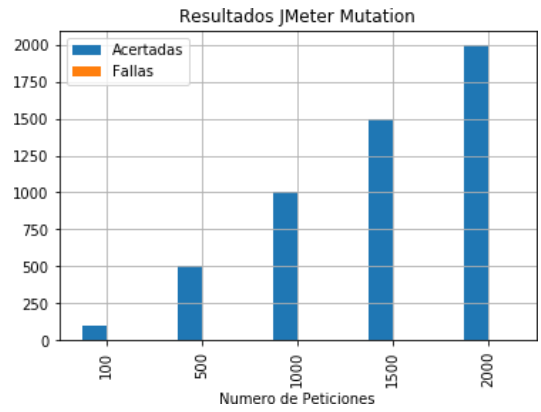
Son las pruebas que validan cómo reacciona el sistema frente a múltiples peticiones concurrentes, donde se obtiene respuestas sobre el rendimiento, el tiempo que demora una petición, porcentajes de error al existir varias peticiones.

Para ejecutar las pruebas de carga se utilizó JMeter, el cual permite automatizar los procesos de prueba, sobre el middleware GraphQL de Hasura que unifica todos los servicios que ofrece el backend.

Las pruebas se basaron en tomar un *query* y una *mutation* del sistema, simulando 100, 500, 1000, 2000, 3000 peticiones concurrentemente. Con la ayuda de la herramienta, se obtienen los resultados que se muestran en la Figura 44, para las diferentes cantidades de solicitudes y en la Figura 45, para ver la latencia.

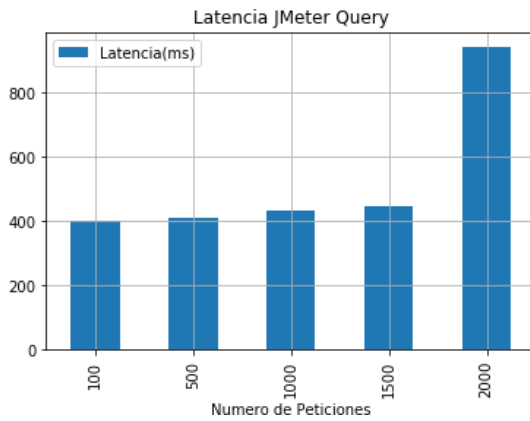


(a) Resultados *query*.

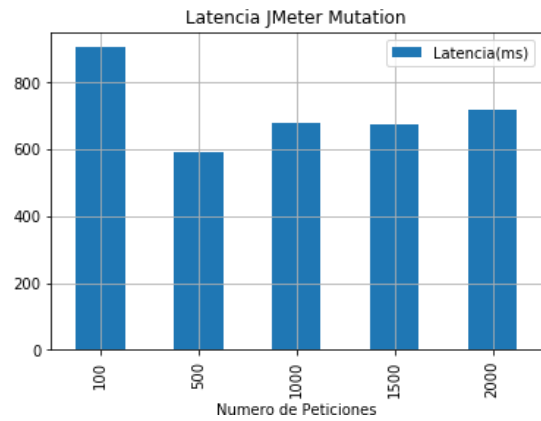


(b) Resultados *mutation*.

Figura 44. Pruebas de Carga.



(a) Resultados *query*.



(b) Resultados *mutation*.

Figura 45. Latencia.

Al final se puede determinar que el middleware GraphQL mediante Hasura, presenta unos buenos resultados, porque presenta una buena tolerancia a fallos ante altas peticiones concurrentes, para visualizar se genera una tabla con los resultados tanto para el *query* y la *mutation* que se visualizan en las Tablas 17 y 18 respectivamente.



| Peticiones | Acertadas     | Fallos     | Tiempo mínimo (ms) | Tiempo máximo (ms) |
|------------|---------------|------------|--------------------|--------------------|
| 100        | 100 (100%)    | 0 (0%)     | 96                 | 3134               |
| 500        | 500 (100%)    | 0 (0%)     | 93                 | 7126               |
| 1000       | 999 (99.9%)   | 1 (0.1%)   | 92                 | 21025              |
| 1500       | 1497 (99.8%)  | 3 (0.2%)   | 92                 | 21046              |
| 2000       | 1953 (97.65%) | 47 (2.35%) | 92                 | 21048              |

Tabla 17. Tabla generalizada de los resultados *query*.

| Peticiones | Acertadas     | Fallos    | Tiempo mínimo (ms) | Tiempo máximo (ms) |
|------------|---------------|-----------|--------------------|--------------------|
| 100        | 100 (100%)    | 0 (0%)    | 95                 | 7130               |
| 500        | 500 (100%)    | 0 (0%)    | 93                 | 7129               |
| 1000       | 1000 (100%)   | 0 (0%)    | 92                 | 15145              |
| 1500       | 1496 (99.73%) | 4 (0.27%) | 92                 | 21046              |
| 2000       | 1995 (99.75%) | 5 (0.25%) | 93                 | 21054              |

Tabla 18. Tabla generalizada de los resultados *mutation*.

## 5.2 Pruebas unitarias

Las pruebas unitarias son procesos automáticos que tienen como objetivo constatar el correcto funcionamiento del código de un software. Consisten en separar y verificar una pequeña porción de código para validar la ejecución y comportamiento de un objeto y lógica de programación.

La generación de pruebas unitarias brinda la capacidad de encontrar errores en el código y además brindar una solución a estos sin tener la necesidad de rehacer el trabajo ya implementado. A continuación, se dan a conocer los resultados de los métodos:

- **Iniciar Sesión:** se realizan dos pruebas, la primera con credenciales correctas y la segunda con credenciales incorrectas como se muestra en las Figuras 46 y 47.

- **Comprar Ticket:** se realizan dos pruebas, la primera con un objeto con todos sus atributos válidos y la segunda con un objeto con sus atributos inválidos cómo se muestra en las Figuras 48 y 49.

De esta manera se puede constatar la correcta funcionalidad de los métodos implementados.

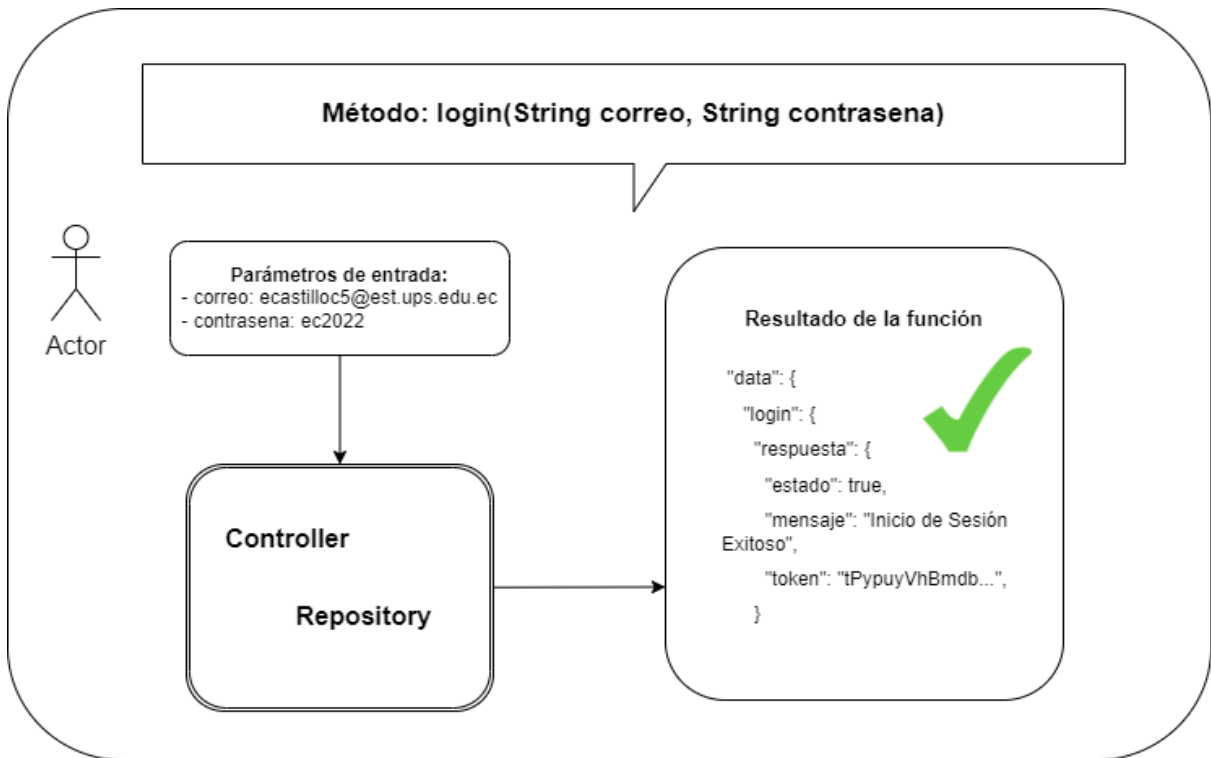


Figura 46. Pruebas unitarias - Iniciar Sesión Exitoso.

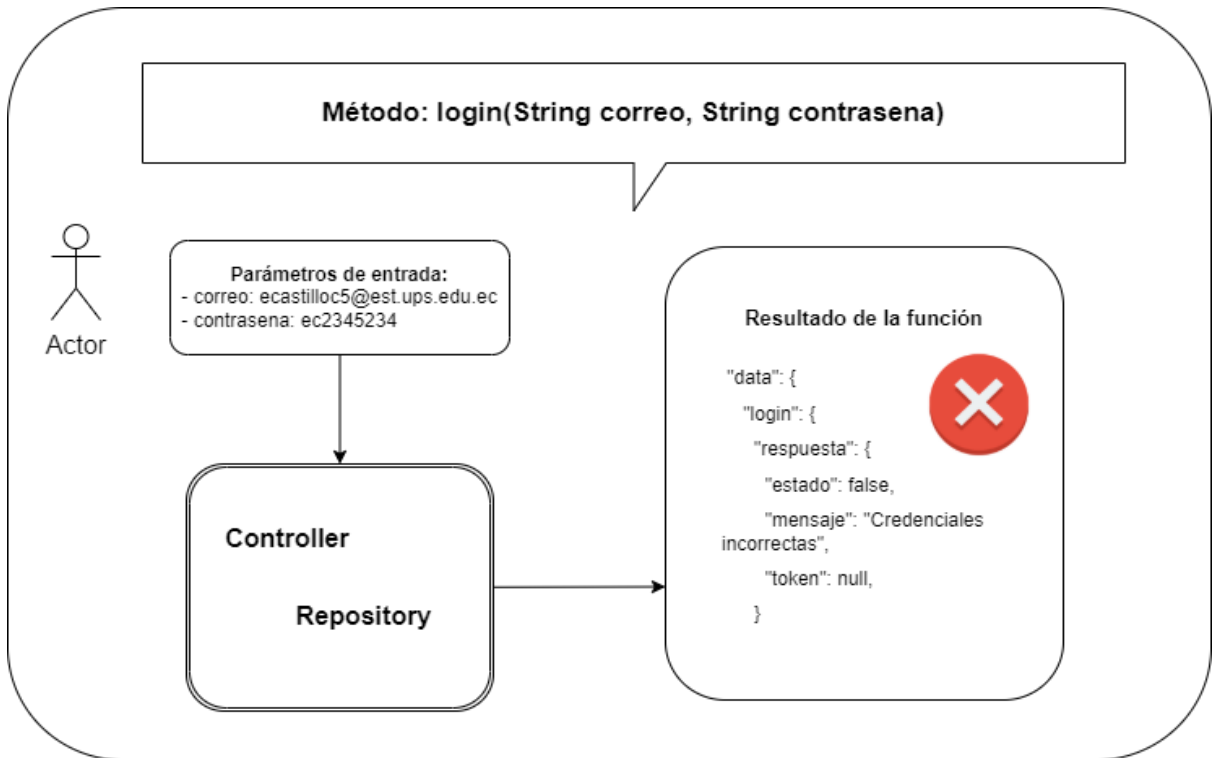


Figura 47. Pruebas unitarias - Iniciar Sesión Error.

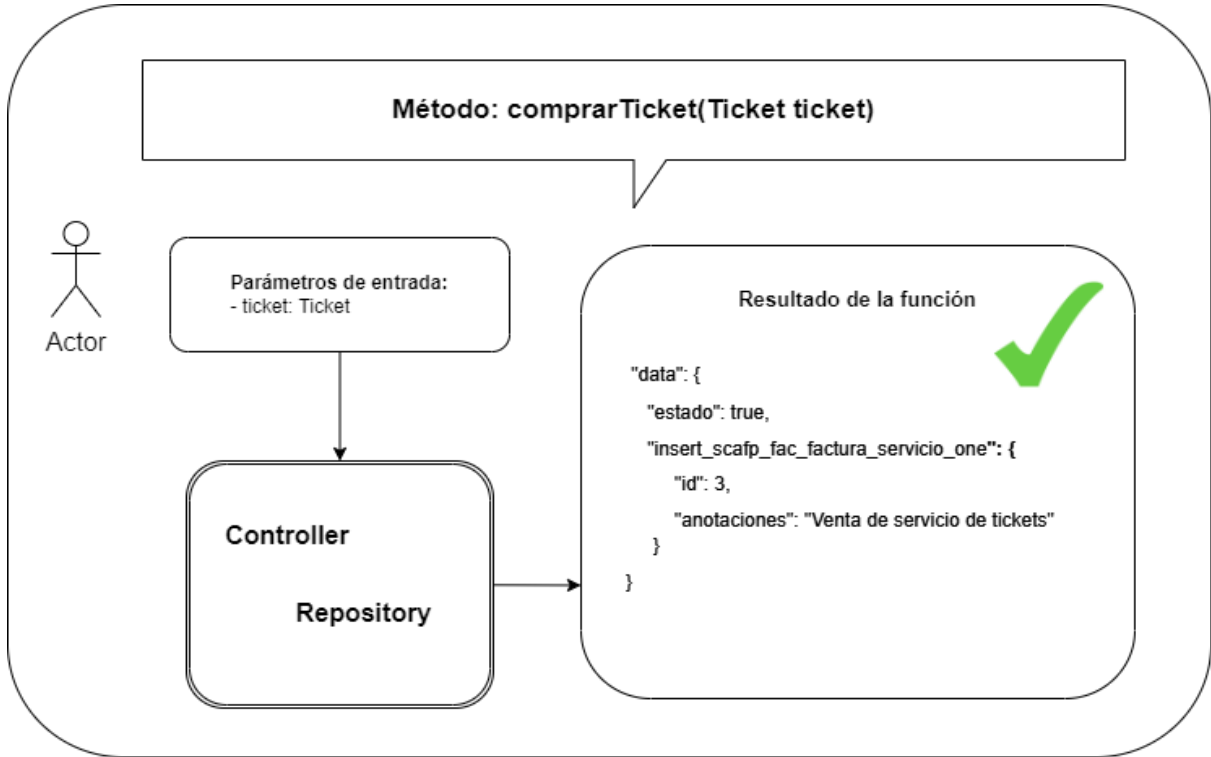


Figura 48. Pruebas unitarias - Comprar Ticket Exitoso.

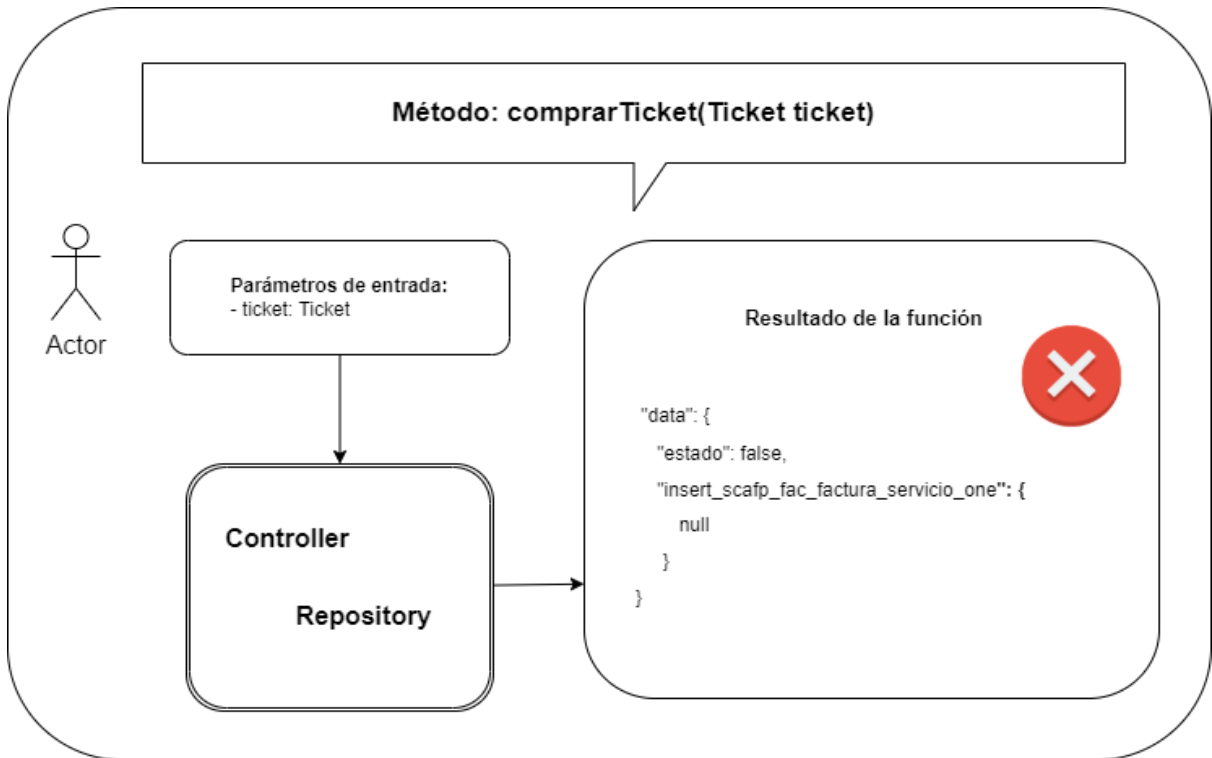


Figura 49. Pruebas unitarias - Comprar Ticket Error.

### 5.3 Encuesta

A continuación, se presentan los resultados de la encuesta con su respectiva interpretación, la cual está se realizó a 10 personas.

**Pregunta 1. ¿Cuál es su percepción sobre el impacto que tienen las aplicaciones móviles para ayudar en los procesos comunes, cómo lo es la compra y venta de productos y servicios?**

| Alternativas           | Respuesta | Porcentaje |
|------------------------|-----------|------------|
| Absolutamente positivo | 4         | 40%        |
| Muy positivo           | 2         | 20%        |
| Positivo               | 4         | 40%        |
| Poco negativo          | 0         | 0%         |
| Absolutamente negativo | 0         | 0%         |

Tabla 19. Pregunta 1. Conteo respuestas.

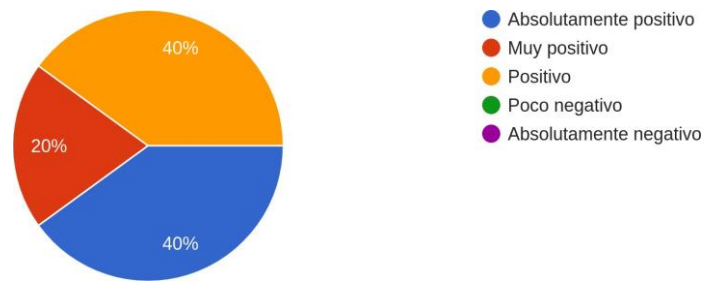


Figura 50. Representación porcentual pregunta 1.

*Interpretación:*

Del 100% de los encuestados, el 40% considera absolutamente positivo el impacto de las aplicaciones en actividades cotidianas, otro 40% que es positivo y el 20% restante que es muy positivo. Esto demuestra que las aplicaciones móviles tienen un efecto positivo en la ayuda de los procesos comunes para la mayoría de los usuarios.

**Pregunta 2. ¿Qué tan importante consideras que son las aplicaciones móviles con recomendaciones para facilitar/acelerar los procesos cotidianos?**

| Alternativas     | Respuesta | Porcentaje |
|------------------|-----------|------------|
| Muy importante   | 4         | 40%        |
| Importante       | 4         | 40%        |
| Neutral          | 2         | 20%        |
| Poco importante  | 0         | 0%         |
| No es importante | 0         | 0%         |

Tabla 20. Pregunta 2. Conteo respuestas.

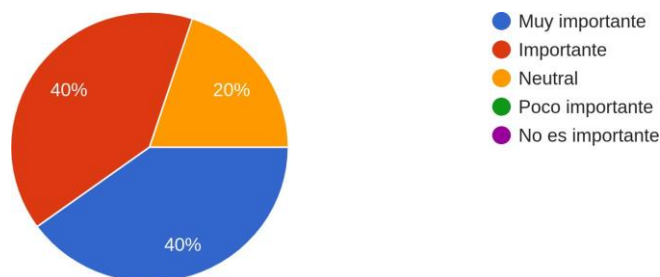


Figura 51. Representación porcentual pregunta 2.

*Interpretación:*

Del 100% de los encuestados, el 40% considera que es muy importante la presencia de recomendaciones en las aplicaciones, otro 40% considera que es importante y el 20% restante se mantienen neutrales respecto al tema. Esto demuestra que las aplicaciones móviles con recomendaciones son de gran importancia y beneficio para la mayoría de las personas.

**Pregunta 3. ¿Estás de acuerdo en que los sistemas ofrezcan sugerencias?**

| Alternativas                   | Respuesta | Porcentaje |
|--------------------------------|-----------|------------|
| Totalmente de acuerdo          | 3         | 30%        |
| De acuerdo                     | 6         | 60%        |
| Ni de acuerdo ni en desacuerdo | 1         | 10%        |
| En desacuerdo                  | 0         | 0%         |
| Totalmente en desacuerdo       | 0         | 0%         |

Tabla 21. Pregunta 3. Conteo respuestas.

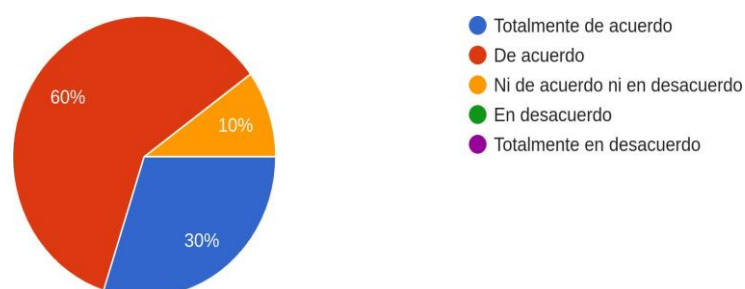


Figura 52. Representación porcentual pregunta 3.

*Interpretación:*

Del 100% de los encuestados, el 30% está totalmente de acuerdo en que los sistemas generen sugerencias, el 60% está de acuerdo y un 10% no está de acuerdo ni en desacuerdo. Esto indica que las aplicaciones que los usuarios apoyan la idea de presentar recomendaciones en los sistemas.

**Pregunta 4. ¿Estás de acuerdo con las recomendaciones de ciudades a visitar?**

| Alternativas                   | Respuesta | Porcentaje |
|--------------------------------|-----------|------------|
| Totalmente de acuerdo          | 4         | 40%        |
| De acuerdo                     | 5         | 50%        |
| Ni de acuerdo ni en desacuerdo | 1         | 10%        |
| En desacuerdo                  | 0         | 0%         |
| Totalmente en desacuerdo       | 0         | 0%         |

Tabla 22. Pregunta 4. Conteo respuestas.

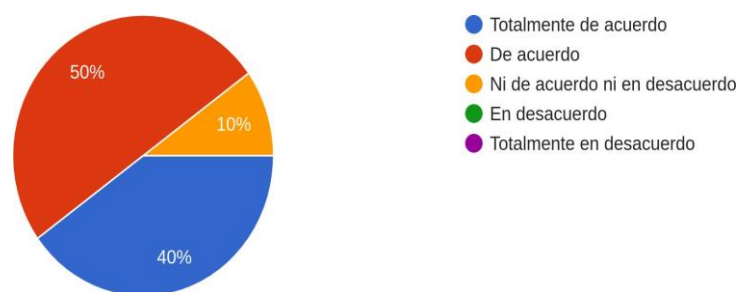


Figura 53. Representación porcentual pregunta 4.

*Interpretación:*

Del 100% de los encuestados, el 50% está de acuerdo con la recomendación de las ciudades, el 40% está totalmente de acuerdo y un 10% no está de acuerdo ni en desacuerdo. Esto implica que la mayoría está de acuerdo con las recomendaciones generadas por el sistema.

**Pregunta 5. ¿Qué tan satisfecho está con su experiencia en la aplicación?**

| Alternativas             | Respuesta | Porcentaje |
|--------------------------|-----------|------------|
| Totalmente satisfecho    | 6         | 60%        |
| Satisfecho               | 4         | 40%        |
| Moderadamente satisfecho | 0         | 0%         |
| Poco satisfecho          | 0         | 0%         |
| No satisfecho            | 0         | 0%         |

Tabla 23. Pregunta 5. Conteo respuestas.



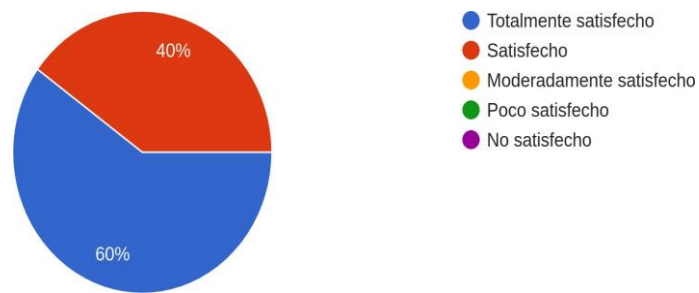


Figura 54. Representación porcentual pregunta 5.

*Interpretación:*

Del 100% de los encuestados, el 60% están totalmente satisfechos con la aplicación y el 40% restante están satisfechos. Esto indica que las aplicaciones cumplen con las necesidades del usuario y es fácil de usar por los mismos.

# Capítulo VI

## CONCLUSIONES

- Después de revisar el estado del arte de los sistemas de recomendación y una de sus técnicas cómo lo es el filtrado colaborativo, se comprobó la importancia de los mismos a través de los años y la ayuda que prestan a los usuarios en las diferentes aplicaciones que existen así como las búsquedas en Internet. En la información obtenida, se encuentran los conceptos base, cómo ejemplos de implementación de los mismos, que ayudan a formar conceptos claros y robustos en menor tiempo.
- La arquitectura implementada se desarrolló de forma distribuida con Docker, mediante la creación de imágenes con Dockerfile y los diferentes servicios que formaran contenedores adecuados con docker-compose, para cada uno de los módulos que conforman el sistema y cumplen con los requisitos planteados, lo cual facilitó el proceso de despliegue a través de la técnica DevOps aplicada, formando una distribución correcta y fuerte que soporta múltiples peticiones y facilita los procesos de conexión. Esta arquitectura se montó con la ayuda de software muy potente cómo el ERP Odoo y frameworks rápidos cómo FastAPI, que se comunicaban a través de una sola conexión generada con la herramienta Hasura que usa GraphQL para generar una API robusta y fácil de usar, que además provee una documentación generada automáticamente.

- La implementación de la PWA se llevó a cabo con Flutter, uno de los principales frameworks más usados actualmente debido a su corta curva de aprendizaje y además brinda soporte para aplicaciones web progresivas, con el plus que el fundamento primordial se basa en los sistemas operativos iOS y Android. Se selecciono esta herramienta debido a que presenta una amplia documentación sostenida por Google y una extensa comunidad de desarrolladores. Así como un vasto repertorio de librerías que aceleran aún más el desarrollo. El uso de la librería "graphql-flutter" permite actuar como un cliente del servidor GraphQL que a su vez satisface el funcionamiento necesario como crear, leer, actualizar y eliminar la información del middleware GraphQL, permitiendo así proveer el uso tecnologías innovadoras y de bajo costo de implementación.
- Tomando en cuenta la información obtenida en el estudio del estado del arte, se procedió con la definición y desarrollo del sistema recomendador, mediante una base de datos de la empresa con viajes realizados entre las ciudades principales del Ecuador, donde se comprueba que los procesos de recomendación básicos como los vecinos cercanos no resulta muy óptimo para el sistema, por lo que así se puede sustentar el uso de la técnica de filtros colaborativos, donde se consiguieron predicciones más acertadas, pero teniendo en cuenta que al ser predicciones, no son completamente seguras, pero se tiene que al separar los datos en prueba y entrenamiento el error medio cuadrático están de estos grupos son cercanos, concluyendo que el sistema generar una correcta sugerencia.
- En base a la encuesta realizada para la validación del sistema, se tiene que la mayoría de las personas aprobarían las aplicaciones que presenten sugerencias de los productos o servicios que ofrecen, agilizando de esta forma el proceso de uso de la aplicación para sus respectivos usuarios, además del 100% de los encuestados, el 90% está de acuerdo con la recomendación de las ciudades y/o totalmente de acuerdo.
- Para probar y validar la arquitectura se consumió desde el cliente (Flutter) sin compli-

caciones, así como la realizaron de pruebas de carga y unitarias, todo esto cuando el sistema se encuentra en un ambiente de producción, dando resultados muy buenos, soportando incluso con un alto tráfico de solicitudes hacia el servidor según las pruebas realizadas, dando resultados favorables, donde el porcentaje de error máximo fue 2.35% y 0.27% para el *query* y la *mutation*, respectivamente. De la misma forma obteniendo 3134 ms y 21048 ms de tiempo mínimo y máximo de respuesta, respectivamente para el *query* y 7130 ms y 21054 ms de tiempo mínimo y máximo de respuesta, respectivamente para la *mutation*, con lo que se afirma que el sistema puede responder bien en cuanto a concurrencia en el uso en entornos de 100 a 2000 usuarios concurrentes.

- Con la automatización del despliegue mediante la técnica DevOps implementada a través de GitLab, se consigue acelerar los procesos de puesta en escena(producción), al mismo tiempo que se guarda una versión estable del sistema, proporcionando un proceso más limpio y optimizado en el desarrollo, despliegue de proyectos.

## Capítulo VII

# RECOMENDACIONES

En base al desarrollo del proyecto, para el sistema recomendador se debe tener una mayor fuente de datos para entrenar y proveer predicciones acertadas, cuando el sistema está en sus estados iniciales donde posee poca información.

Con lo que respecta al servidor en términos de despliegue a producción, para el middleware GraphQL y el Odoon se debe contar con los respectivos certificados de seguridad (SSL), para que funcionen sin problemas.

Con respecto a Flutter tener en cuenta el manejo adecuado de las rutas, así como la selección de un manejador de estados eficiente en calidad de rendimiento y escalabilidad. Esto evita la invocación frecuente del método *build* cuando los widgets se reconstruyen. Para lograr un terminado eficaz buscando una combinación de rendimiento e interfaz intuitiva, se recomienda tener un conocimiento robusto en el diseño de interfaces debido a que el framework representa un espacio extenso y complejo en este campo.

## Capítulo VIII

# TRABAJOS FUTUROS

En este trabajo se desarrolló e implementó un sistema recomendador basado en filtros colaborativos, sobre una aplicación para acopio y distribución de productos utilizando Odoon y Flutter (PWA), estos se comunicaron a través de un middleware GraphQL con ayuda de Hasura, el cuál conecta las diferentes fuentes de datos y funcionalidades, poniendo todo esto en funcionamiento a través de una técnica DevOps implementada con GitLab CI-CD y Docker (Dockerfile - docker-compose). En base a ello, se describen a continuación ideas para mejorar el sistema en lo respectivo a rendimiento y despliegue del mismo:

- Proveer de más datos al sistema recomendador para generar mejores predicciones.
- Llevar al siguiente nivel la técnica de despliegue DevOps, migrando a Kubernetes, y automatizando el proceso de pruebas en esta fase.
- Desarrollar el proceso de despliegue del sistema en diferentes plataformas basadas en servicios.

# REFERENCIAS

Ater, T. (2017). *Building Progressive Web Apps*. O'Reilly Media, Inc.

Bagnato, J. I. (2019). Recomendador de Repositorios de Github. [https://github.com/jbagnato/machine-learning/blob/master/Ejercicio\\_Sistemas\\_Recomendacion.ipynb](https://github.com/jbagnato/machine-learning/blob/master/Ejercicio_Sistemas_Recomendacion.ipynb). Online; recuperado 20 de febrero de 2022.

Basilio, D. M. (2020). El paradigma DevOps y su implementación en el desarrollo de software. *Universidad de Ciego de Ávila Máximo Gómez Báez*.

Bradley, S. (2019). You can automate DevOps yourself using GitLab CI and Docker-Compose. <https://sean-bradley.medium.com/auto-devops-with-gitlab-ci-and-docker-compose-f931233f080f>. Online; recuperado 15 de enero de 2022.

Byron, L. (2018). GraphQL: A data query language. <https://engineering.fb.com/2015/09/14/core-data/graphql-a-data-query-language/>. Online; recuperado 20 de febrero de 2022.

Castell, G. (2020). Desarrollo e implementación de una aplicación web progresiva (pwa). Master's thesis, Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona.

Castellano, E. J. (2007). *Evaluación del uso de algoritmos colaborativos para orientar académicamente al alumnado en bachillerato*. PhD thesis, Universidad de Jaén.

- Cerro, I. (2019). Implementación de una aplicación web progresiva. Master's thesis, Universidad Pública de Navarra. Online; recuperado 20 de febrero de 2022.
- Del Pino, J., Salazar, J., and Cedeño, V. (2011). Análisis de Métricas de Similitud Usadas en el Algoritmo de Filtro Colaborativo Basado en el Usuario Para Recomendar Materias de Pregado. *DSpace Repositorio de ESPOL*.
- Gallego, M. T. (2012). Gestión de Proyectos informáticos Metodología SCRUM. *Universitat Oberta de Catalunya*.
- Galán, S. M. (2007). Filtrado Colaborativo y Sistemas de Recomendación. *Universidad Carlos III de Madrid*.
- Garrido, A. (2018). Integrando fuentes heterogéneas de datos en Web con GraphQL. Master's thesis, Universidad de Extremadura.
- Hartig, O. and Pérez, J. (2018). Semantics and Complexity of GraphQL. *Universidad de Chile and Linköping University*.
- Hasura (s.f). Migrations and Metadata. <https://hasura.io/learn/graphql/hasura-advanced/migrations-metadata/>. Online; recuperado 20 de febrero de 2022.
- Li, Y., Lu, L., and Xuefeng, L. (2005). A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in e-commerce, expert syst. Online; recuperado 20 de febrero de 2022.
- Martinez, L. (2007). A multigranular linguistic content-based recommendation model research articles. Online; recuperado 20 de febrero de 2022.
- Monte, J. (2016). *Implantar scrum con éxito*. Editorial UOC. Online; recuperado 20 de febrero de 2022.



- Moreno, I. (s.f.). Recomendaciones basado en filtrado colaborativo en Python. <https://www.statdeveloper.com/recomendaciones-basado-en-filtrado-colaborativo-en-python/>. Online; recuperado 20 de febrero de 2022.
- Olivo, M., Marilyn, M., Silva, S., and Miguel, J. (2020). Desarrollo de un prototipo de aplicación móvil utilizando sdk flutter y lenguaje de código abierto dart para promover actividades deportivas en guayaquil. Master's thesis, Universidad de Guayaquil. Online; recuperado 20 de febrero de 2022.
- Pipis, G. (2020). Item-based collaborative filtering in python. <https://predictivehacks.com/item-based-collaborative-filtering-in-python/>. Online; recuperado 20 de febrero de 2022.
- Prada, D. E. A. (2018). Sistemas de recomendación de productos para banca empresarial y corporativa: Un análisis comparativo. *Universidad de los Andes*.
- Programador clic (2020). Demo de Python para el filtrado colaborativo basado en puntuación. <https://programmerclick.com/article/9885185935/>. Online; recuperado 20 de febrero de 2022.
- Rodríguez, P. (2018). Desarrollo de un cliente web mediante aplicaciones web progresivas. *Universidad de Vigo*.
- Rodríguez, P. A., Pérez, A. M., Londoño, L. F., and Duque, N. D. (2016). Sistema de recomendación de objetos de aprendizaje a través de filtrado colaborativo. *TEKNOS*, 16(2):85–94.
- Schouren, J. M. (2019). Web App Manifests — A Guide to get your website on a user's Home Screen. <https://medium.com/deity-io/web-app-manifests-a-guide-to-get-your-website-on-a-users-home-screen-6baf108538e7>. Online; recuperado 20 de febrero de 2022.

Subra, J.-P. (2018). *Scrum Un Método Ágil Para Sus Proyectos*. Ediciones ENI.

Triguero, D. (2018). Progressive Web Apps (PWA): la nueva generación de aplicaciones móviles. <https://profile.es/blog/progressive-web-apps-la-nueva-generacion-de-aplicaciones>. Online; recuperado 20 de febrero de 2022.

TRLogic (2018). Docker Compose. <https://medium.com/@trlogic/docker-compose-57a51a34526c>. Online; recuperado 20 de febrero de 2022.

Turnbull, J. (2014). *The Docker Book*. The Docker Book.

Unipython (s.f). Como desarrollar un sistema de recomendación en Python. <https://unipython.com/como-desarrollar-un-sistema-de-recomendacion-en-python/>. Online; recuperado 20 de febrero de 2022.

# **Anexos**

# **Anexos A**

## **Formato de la encuesta**

**Diseño e implementación de un sistema recomendador basados en filtros colaborativos para el acopio y distribución de productos utilizando aplicaciones web progresivas y GraphQL sobre DevOps para plataformas basadas en servicios.**

Con la presente encuesta se pretende validar el funcionamiento y la utilidad del sistema.

### **1.1 Información de la empresa**

**1. ¿Cuál es su percepción sobre el impacto que tienen las aplicaciones móviles para ayudar en los procesos comunes, cómo lo es la compra y venta de productos y servicios?**

- Absolutamente positivo
- Muy positivo
- Positivo
- Poco negativo

Absolutamente negativo

**2. ¿Qué tan importante consideras que son las aplicaciones móviles con recomendaciones para facilitar/acelerar los procesos cotidianos?**

Muy importante

Importante

Neutral

Poco importante

No es importante

**3. ¿Estás de acuerdo en que los sistemas ofrezcan sugerencias?**

Totalmente de acuerdo

De acuerdo

Ni de acuerdo ni en desacuerdo

En desacuerdo

Totalmente en desacuerdo

**4. ¿Estás de acuerdo con las recomendaciones de ciudades a visitar?**

Totalmente de acuerdo

De acuerdo

Ni de acuerdo ni en desacuerdo

En desacuerdo

Totalmente en desacuerdo

**5. ¿Qué tan satisfecho está con su experiencia en la aplicación?**

- Totalmente de acuerdo
- De acuerdo
- Ni de acuerdo ni en desacuerdo
- En desacuerdo
- Totalmente en desacuerdo