UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

MAESTRÍA EN

SOFTWARE CON MENCIÓN EN DESARROLLO WEB Y MÓVIL

RPC-SO-34-NO.778-2021

OPCIÓN DE TITULACIÓN:

PROYECTO DE TITULACIÓN CON COMPONENTES DE INVESTIGACIÓN APLICADA Y/O DE DESARROLLO

TEMA:

DISEÑO DE LA ARQUITECTURA Y MEJORA DE LA APLICACIÓN MÓVIL "BUSITY" PARA LA INFORMACIÓN DEL TRANSPORTE PÚBLICO EN QUITO

AUTOR(ES)

DANIEL ALEXANDER BURBANO POZO

DIRECTOR:

HOLGER RAÚL ORTEGA MARTÍNEZ

QUITO – ECUADOR

2023

Autor(es):

Daniel Alexander Burbano Pozo
Licenciado en Diseño Gráfico
Candidato a Magíster en Software con Mención en Desarrollo Web y Móvil por la Universidad Politécnica Salesiana – Sede Quito.
dburbanop@est.ups.edu.ec

Dirigido por:

Holger Raúl Ortega Martínez
Físico
Master of Science in Machine Learning
hortega@ups.edu.ec

Daniel Alexander Burbano Pozo
DISEÑO DE LA ARQUITECTURA Y MEJORA DE LA APLICACIÓN MÓVIL "BUSITY" PARA LA INFORMACIÓN DEL TRANSPORTE PÚBLICO EN QUITO.

DEDICATORIA

Dedico este trabajo de titulación a mis padres, quienes me apoyaron y me motivaron a seguir estudiando, a no rendirme y a ser mejor de lo que fui ayer. Gracias por inculcarme los valores que hoy definen quien soy y gracias por todo su apoyo incondicional.

A mis hermanos y familia por brindarme su apoyo y alentarme a continuar con mis estudios a pesar de haber sacrificado tiempo en familia.

Finalmente, a mi novia, por su apoyo incondicional, su amor y su paciencia durante todo este proceso.

AGRADECIMIENTO

Quiero expresar mi más sincero agradecimiento a todas las personas que contribuyeron al éxito de este trabajo de titulación.

En primer lugar, quisiera agradecer a mi tutor académico, el Físico Holger Ortega M. por su orientación y apoyo constante durante todo el proceso. Gracias a su sabiduría y experiencia, pude enfrentar los desafíos y superar las dificultades que se presentaron a lo largo del camino.

También me gustaría agradecer a todas las personas que estuvieron conmigo todo este tiempo como amigos, familia y mi novia por su apoyo incondicional y su paciencia en los momentos en que tuve que dedicar largas horas de trabajo en este proyecto. Su apoyo y consideración me ayudaron a seguir con un equilibrio en los estudios, trabajo y la familia.

Finalmente, agradezco a la Universidad Politécnica Salesiana por brindarme la posibilidad de continuar con mis estudios y seguir creciendo profesionalmente.

Tabla de Contenido

Tabla de ilustraciones

Diseño de la arquitectura y mejora de la aplicación móvil "Busity" para la información del transporte público en Quito

Autor(es):

Daniel Alexander Burbano Pozo

Resumen

Este proyecto tuvo como objetivo analizar, diseñar y aplicar una arquitectura de software eficiente y escalable para su uso en aplicativos móviles y web. Para ello, se definió las funcionalidades, requerimientos funcionales/no funcionales y un modelo c4 para describir y documentar la arquitectura de software.

Se procedió a refactorizar una aplicación existente para que cumpla con los lineamientos de la arquitectura de software diseñada, lo que permitió reducir el número de dependencias un 60%, pasando de 48 dependencias a solamente 19. Por otro lado, se refactorizó los controladores de todos los módulos donde pudimos obtener una reducción significativa de la complejidad de la lógica de negocio reduciendo en un 59.62% el número de líneas de código del controlador de la página principal, mejorando el rendimiento y la estabilidad de la aplicación.

Se construyó un prototipo de las principales funcionalidades de la aplicación utilizando la arquitectura diseñada, el cual fue probado por un grupo de usuarios para medir su funcionalidad y facilidad de uso sacando datos mediante una prueba de usabilidad en el prototipo, donde se obtuvo resultados positivos como una aplicación intuitiva, fácil de usar y que estarían dispuestos a utilizar la aplicación en un futuro cercano para movilizarse con el transporte público de Quito.

Abstract

This project aimed to analyze, design, and apply an efficient and scalable software architecture for use in mobile and web applications. To do so, the functionalities, functional/non-functional requirements, and a C4 model were defined to describe and document the software architecture.

An existing application was refactored to comply with the guidelines of the designed software architecture, which reduced the number of dependencies by 60%, from 48 dependencies to only 19. Additionally, the controllers of all modules were refactored to significantly reduce the complexity of the business logic, reducing the number of lines of code in the main page controller by 59.62% and improving application performance and stability.

A prototype of the application's main functionalities was built using the designed architecture, which was tested by a group of users to measure its functionality and ease of use, extracting data through a usability test on the prototype. Positive results were obtained, indicating an intuitive and easy-to-use application that users would be willing to use in the near future to move around with public transport in Quito.

1. Introducción

1.1. Antecedentes

Un estudio sobre la movilidad realizado por Quito Cómo Vamos (vamos, 2020), menciona que en el año 2020 el servicio de transporte público municipal es el servicio con más insatisfacción con un 66,1 % y con el 63,7 % le sigue el servicio de transporte público de cooperativas. Es decir que tres cuartas partes de los encuestados están insatisfechas con estos servicios, ya sea por inseguridad, maltrato, accidentes, etc. Por otra parte, la información del trasporte público es incierta, la mayoría de personas estima el tiempo de llegada de los buses, pero esta información no existe en ningún lado. Puede haber la posibilidad de que pasen dos unidades seguidas o que la siguiente unidad pase después de cierto tiempo. Todas las personas se movilizan con esta incertidumbre con la esperanza de que el bus que necesitan esté por llegar. La aplicación de Busity intenta solventar este problema de incertidumbre de las unidades, brindando a los usuarios del aplicativo, la información GPS de las unidades que necesitan y de las estaciones que ya han pasado, es decir, mostrando el proceso de la ruta de cada bus.

Busity fue el proyecto de titulación de la carrera de Diseño Gráfico Medios Interactivos de la Universidad San Francisco de Quito de Daniel Burbano y Sebastián Burbano (2021). El proyecto “Busity” es una aplicación híbrida diseñada en el framework de Ionic. Este framework nos permite programar aplicaciones tanto para IOS como para Android, tiene características para cada sistema operativo dando una apariencia nativa, para una mejor experiencia de usuario. El proyecto intenta solventar la brecha de información que existe con el transporte público de Quito. Lo que busca la aplicación es brindar la información de las unidades de bus en tiempo real, para que el usuario pueda saber la posición de los buses, en qué estación se encuentran, y que estaciones han pasado. Con esta información, tomar decisiones para llegar a tiempo a su destino y minimizar la desinformación del transporte público, brindando una movilidad más digna.

El proyecto no cuenta con un diseño de arquitectura, por lo tanto, es difícil la mantenibilidad y la escalabilidad del proyecto. La falta de arquitectura genera el famoso código llamado espagueti en el proyecto, donde el código fuente tiene funciones escritas por todas partes y no cuenta con una estructura base. Si se pretende migrar la aplicación móvil a una plataforma web, se tendrá que crear un nuevo proyecto tomando las funcionalidades y creando desde cero, para lo cual esto no es viable ni mucho menos práctico.

1.2. Justificación

Se busca diseñar una arquitectura que sustente el código fuente para que su mantenibilidad y escalabilidad a nuevas plataformas sea de una manera más práctica, con estándares de programación y sobre todo fácil de entender para poder replicar en nuevas tecnologías si alguna vez se busca la migración del proyecto. Con esto también nos aseguramos de que las personas que trabajen a futuro en el proyecto entiendan de mejor manera las funcionalidades y como trabajan entre ellas. De esta manera, podemos asegurar que el proyecto siga adelante con nuevos desarrolladores, nuevas tecnologías, nuevos alcances a otras plataformas para que la aplicación pueda llegar a ser multi dispositivo y llegar a nuestro público objetivo desde cualquier dispositivo, mejorando así la experiencia de usuario.

Aparte de la arquitectura, se desea refactorizar el código para reducir el código espagueti que existe dentro de la aplicación. De esta manera, el proyecto estará mejor estructurado ya que seguirá la arquitectura planteada anteriormente. Con esto, el proyecto tomará forma, con bases sólidas, con código limpio y estructurado.

1.3. Objetivos

1.3.1 Objetivo General

Diseñar una arquitectura para la aplicación Busity con el fin de mejorar las funcionalidades de la aplicación.

1.3.2 Objetivos Específicos

- Analizar y definir la arquitectura para que pueda ser utilizada en aplicativos móviles y web.
- Refactorizar la aplicación para que cumpla con el lineamiento de la arquitectura de software.
- Construir un prototipo de aplicación de las principales funcionalidades con la arquitectura diseñada.
- Realizar pruebas de usabilidad del prototipo para medir y analizar si el prototipo es funcional.

1.4. Alcance

En el presente proyecto se contemplará el diseño de arquitectura web y móvil de la aplicación Busity. Dentro de la arquitectura estará definido:

- Requerimientos funcionales y no funcionales.
- Análisis de las funcionalidades de la aplicación “Busity”.
- Construcción de modelos C4.

Por otro lado, se va a refactorizar el código fuente para organizar y estructurar el código en función a la arquitectura diseñada. Se analizará y utilizará los patrones de diseño para que el código fuente cumpla estándares de programación, sea fácil de leer, y cumpla con principios de diseño. También se creará un prototipo de aplicación web aplicando la arquitectura diseñada, en base a eso analizar los resultados y sacar conclusiones sobre la arquitectura. Por último, se propone pruebas de usabilidad del prototipo para obtener retroalimentación y poder analizar y concluir si el prototipo cumple con las funcionalidades del proyecto Busity.

Se dejará por fuera del proyecto los siguientes puntos:

- El despliegue de las aplicaciones en un ambiente productivo, ya que solamente será un prototipo con las principales funcionalidades.

- La publicación del prototipo en cualquier tienda de aplicaciones (Play Store/App Store).

1.5. Metodología

Para poder visualizar y garantizar el avance del proyecto, se implementará una metodología que se adapte al tema de investigación. La metodología que se utilizará es Scrum, que nos permite medir y definir en que etapa del proyecto nos encontramos. También con esta metodología podemos utilizar las herramientas y funciones para estructurar y gestionar el desarrollo del proyecto. Al ser una investigación de tipo experimental, podemos aplicar una metodología empírica para generar una serie de resultados con relación al tema de investigación. Para ello se busca realizar un prototipo de aplicación web utilizando la arquitectura diseñada en esta investigación y ponerla en práctica, obteniendo así un objeto de estudio para poder analizar y sacar resultados tangibles.

Para entender mejor el proceso de la metodología Scrum, se muestra a detalle los sprints que tendrá el proyecto.

Sprint 1:

- Analizar patrones de diseño de software.
- Analizar las principales funcionalidades de la aplicación.

Sprint 2:

- Definir los requerimientos funcionales y los no funcionales.
- Diseñar modelos de diagramas C4.

Sprint 3:

- Refactorizar el código fuente de la aplicación “Busity”.
- Utilizar patrones de diseño de software.

Sprint 4:

- Crear prototipo de una aplicación web con la arquitectura diseñada.
- Pruebas de usabilidad.

Para analizar los patrones de diseño, nos enfocaremos en los patrones vistos en el módulo “Patrones de diseño de software” impartidas en la maestría de software. En cuanto al análisis de las principales funcionalidades del aplicativo, observaremos y definiremos sus funcionalidades de la aplicación.

Por otro lado, para diseñar esta arquitectura de una aplicación web y móvil, se buscará analizar y definir los requerimientos funcionales tanto como los no funcionales para determinar todas las funcionalidades de la app Busity. Una vez definidos los requerimientos, diseñar una arquitectura proponiendo usar un modelo de vista de arquitectura parecida a la 4+1. En este caso, usaremos el modelo de arquitectura C4, este cuenta con 4 niveles que describen el sistema, nivel de contexto, de contenedores, de componentes, y de código (Holgado , Peñalvo, & Ingelmo, 2020), pero en este caso solamente usaremos los 3 primeros niveles.

En la refactorización del código fuente, nos basaremos en la arquitectura diseñada y en los patrones de diseño de software analizados para que el código tenga una mejor estructura y se pueda entender de mejor manera, evitando la acumulación de código repetitivo. Por otra parte, para el prototipo se usará alguna de los nuevos frameworks que existen en el mercado y que se usan actualmente para el desarrollo de proyectos como lo son Angular, React o Vue.js. Para terminar, las pruebas de usabilidad se realizarán con personas externas mediante encuestas después de probar el prototipo

2 Marco teórico referencial

2.1. Marco Conceptual

2.1.1 Arquitectura de software

De acuerdo al (SEI) Software Engineering Institute, citado por Cervantes (2010) menciona que la arquitectura de software se refiere a la estructura de algún sistema o programa con componentes claramente identificables y cómo estos se relacionan entre sí.

2.1.2 Aplicación móvil

Según el desarrollador Luis Herazo, define a una aplicación móvil o app móvil como un tipo de aplicación diseñada para ser ejecutada en dispositivos móviles o smart phones. Estas aplicaciones puede contar con pequeñas cualidades, pero pueden aportar grandes servicios y una gran experiencia de calidad.

2.1.3 Ionic 6

Ionic es un framework para desarrollar aplicaciones móviles modernas, según la página oficial de Ionic (2022), el SDK móvil para la web, son herramientas de interfaz de usuario móvil para crear aplicaciones multiplataforma y de alta calidad a partir de una base de código como Vue/React/Angular, sin olvidar que es una framework de código abierto.

2.1.4 Firebase

Firebase, como se menciona en la página oficial, es una plataforma de desarrollo de apps que nos ayuda a compilar y desarrollar aplicaciones o juegos con el respaldo de Google. Esta plataforma brinda muchos servicios como lo es el almacenamiento de datos y también nos ayuda con la creación del (CRUD) create, read, update and delete del login y el registro de la aplicación.

2.1.5 Google Maps

Google Maps es una aplicación que se encarga de proporcionar toda la información a los usuarios sobre su ubicación actual, cualquier dirección especifica de todo el mundo entero, cabe resaltar que los mapas pueden a veces estar desactualizados algunos años, pero aun así la aplicación tiene una veracidad con un porcentaje alto. Y, por último, la aplicación también nos ayuda con el trazado de recorridos y direcciones para que los usuarios puedan llegar al lugar deseado desde su posición actual o desde cualquier posición inicial.

2.1.6 API

API es un acrónimo en inglés que significa "Application Programming Interface". En informática, una API es una colección de reglas y estándares que determinan cómo dos sistemas informáticos se comunicarán entre sí. Las APIs permiten que los desarrolladores creen aplicaciones que interactúen con otros servicios, como bases de datos o plataformas en línea, sin tener que conocer los detalles técnicos de cómo funcionan estos servicios.

En resumen, una API es un puente entre diferentes aplicaciones que les permite interactuar entre ellas de una manera controlada y estandarizada. (Richardson & Amundsen, 2013)

2.1.7 Patrones de diseño

Existen varios patrones de diseño de software diferentes, cada uno con su propósito y aplicaciones específicos. (Gamma, Helm, Johnson, & Vlissides, 1994). Los patrones estructurales son aquellos patrones de diseño que se enfocan en la organización y estructuración de los componentes de una aplicación. Algunos de los patrones estructurales más comunes son:

1. Patrón de Decorador: permite agregar responsabilidades adicionales a un objeto dinámicamente sin afectar su comportamiento.
2. Patrón de Composición: permite construir objetos más complejos a partir de objetos simples.

3. Patrón de Proxy: proporciona un objeto sustituto o intermediario que controla el acceso a otro objeto.
4. Patrón de Puente: desacopla una abstracción de su implementación para que ambas puedan evolucionar independientemente.
5. Patrón de Fachada: proporciona una interfaz simplificada para un subsistema complejo.

Los patrones de creación son patrones de diseño que se enfocan en la forma en que se crean objetos y se gestionan sus dependencias. Algunos de los patrones de creación más comunes son:

1. Patrón de Fabrica: proporciona una interfaz para crear objetos en una clase o familia de clases sin especificar su clase concreta.
2. Patrón de Constructor: proporciona una forma de crear objetos complejos a partir de objetos más simples.
3. Patrón de Singleton: garantiza que una clase tenga solo una instancia y proporciona un acceso global para esta clase.
4. Patrón de Prototipo: permite crear nuevos objetos a partir de objetos existentes sin especificar la clase concreta de objeto a crear.
5. Patrón de Fábrica Abstracta: proporciona una interfaz para crear objetos en una familia de clases sin especificar su clase concreta.

Los patrones de comportamiento son patrones de diseño que se enfocan en la comunicación y la colaboración entre objetos. Los patrones de comportamiento más comúnmente usados son:

1. Patrón de Observador: permite que objetos puedan recibir notificaciones de cambios en otros objetos. En otras palabras, es una forma de que un objeto "observe" a otro objeto y reaccione a los cambios que ocurren en él. Este patrón es útil en situaciones donde se requiere que varios objetos estén sincronizados con los cambios en un objeto central.

2. Patrón de Comando: encapsula una solicitud como un objeto, lo que le permite parametrizar un objeto con una solicitud, enviar la solicitud como un objeto y poner en cola o registrar la solicitud.
3. Patrón de Iterador: permite acceder secuencialmente a los elementos de un objeto de colección sin exponer su representación subyacente.
4. Patrón de Mediador: define un objeto que actúa como un intermediario entre dos o más objetos para simplificar su comunicación y mejorar la cohesión.
5. Patrón de Estado: permite que un objeto altere su comportamiento cuando su estado interno cambia.

Los patrones de arquitectura de software son patrones de diseño que describen cómo organizar y estructurar un software de manera eficiente. Algunos de los patrones de arquitectura de software más comunes son:

1. Patrón de Modelo-Vista-Controlador (MVC): la aplicación está organizada en tres componentes: modelo, vista y controlador.
2. Patrón de Capas: divide la aplicación en capas lógicas, y cada una se encarga de una tarea en específico.
3. Patrón de Microservicios: divide una aplicación en pequeños servicios autónomos que trabajan juntos para cumplir con un objetivo común.
4. Patrón de Event-Driven Architecture (EDA): se basa en el envío y recepción de eventos para comunicar componentes de la aplicación.
5. Patrón de Arquitectura de N-Capas: divide la aplicación en varias capas, cada una de las cuales se encarga de una tarea específica y se comunica con la siguiente capa.

2.2. Estado del Arte

En estos últimos años, las aplicaciones móviles han ido tomando fuerza a un nivel que dependemos prácticamente para todo nuestro día a día, para el trabajo, para la comunicación, para pagar servicios básicos, pagar cualquier compra, para movilizarnos en auto de alguna dirección a otra que se desconozca, se utiliza

aplicaciones para el hogar como abrir la puerta de la casa o controlar las luces del hogar, etc. Ahora también existen aplicaciones móviles que facilitan la movilización del transporte público para diferentes partes del mundo y en Ecuador no es la excepción.

Existen varias aplicaciones que solventan el tema del transporte público, como, por ejemplo, se encontró un proyecto llamado “BusTap” que solventa la misma problemática que la aplicación “Busity” pero con algunas diferencias. El Proyecto “BusTap: A Real-Time Bus Tracking Android Application” solventa las necesidades de la ciudad de Baguio Filipinas, de la desinformación de autobuses en tiempo real para los pasajeros y operadores de autobuses (Ascueta, Bautista, Quilala, & Beninsig, 2021). Este proyecto usa el mismo sistema de almacenamiento “Firebase” para todos los archivos, imágenes, autenticación de usuarios, y la recopilación de almacenamiento de datos en tiempo real, como por ejemplo las ubicaciones en tiempo exacto de la línea de buses Victory Liner en Baguio, Filipinas. La desventaja de este proyecto es que solo está disponible como una aplicación desarrollada para Android. El proyecto de “Busity” es una aplicación híbrida, es decir que está diseñada tanto para Android como para IOS, y con una visión a una escalabilidad para plataformas web. Lo interesante del proyecto de Ascueta, es el uso de una metodología Scrum, que facilitó el desarrollo del proyecto debido a las características de esta metodología de roles, artefactos y eventos que permitió la gestión de tareas y roles de los autores.

Otro dato interesante del proyecto es el uso de la vista de modelo arquitectónico 4+1, donde se puede observar cuatro vistas principales: lógica, proceso, física y de desarrollo con escenarios extendidos, ayudando a separar los puntos de vista de los stakeholders y abordar los concers del proyecto de “BusTap”.

El proyecto muestra las siguientes conclusiones: a) La información identificada sobre la empresa, los autobuses, los horarios, los pasajeros y los conductores, así como los requisitos del sistema y los servicios de biblioteca, son necesarios para diseñar, desarrollar y utilizar con éxito todas las funciones de BusTap. b) El uso de la arquitectura del modelo de vista 4+1 para describir el marco arquitectónico de la

aplicación es importante para la cohesión y organización del proyecto. c) Las características identificadas, diseñadas y desarrolladas son necesarias para cerrar la brecha de conocimiento entre los operadores de autobuses, conductores y pasajeros.

Otro proyecto similar a “Busity” es el proyecto de Steven Hidalgo y Eduardo Mena (2022). El proyecto llamado “Desarrollo y evaluación de una aplicación móvil para monitorear el transporte público en Quito” busca monitorizar el transporte público con los llamados “busters” el cual consiste en que los mismos usuarios son los que proporcionen información de los buses en quito, y por otro lado los “ruters” pueden utilizar esa información de todos los buses que se encuentren a 5km a la redonda en tiempo real.

Este proyecto también utiliza una base no relacional “Firebase”, para que la información se vaya actualizando dinámicamente sin tener que recargar toda la aplicación. El proyecto está creado con el lenguaje java en Android Studio y fue desplegado únicamente para la plataforma de Play Store para Android.

La desventaja de este proyecto es que depende 100 % de los usuarios “busters” para la información del transporte público, sin ellos la aplicación no tendría información para brindar a los usuarios “ruters”.

Este proyecto también utilizó la metodología de Scrum para el desarrollo del proyecto, proponiendo algunos requerimientos y dividiéndolas en sprints para una entrega más ágil y con la facilidad de medir el cumplimiento del proyecto. Mirar si está retrasado, si está a tiempo, si existen complicaciones que retrasen alguna historia e ir visualizando las tareas en el backlog de un tablero de Scrum.

Se puede observar que existen varias aplicaciones tratando de solventar la misma desinformación del transporte público, pero de diferentes maneras y con distintas tecnologías. Lo que caracteriza a Busity sobre las demás aplicaciones son las diferentes funcionalidades adicionales sobre la funcionalidad principal. Además de brindar la información organizada de las unidades del transporte público, Busity busca brindar información de las estaciones de Quito, dentro de la información se

puede encontrar las diferentes unidades que pasan por el lugar, el nombre de la estación y la posición dentro de un mapa. Otra funcionalidad es “Llévame”, esta brinda la posibilidad de movilizarte desde un punto a otro otorgando la información sobre qué bus tomar, dónde bajarte, si debes tomar otro bus y el tiempo estimado de llegada al lugar. Por último, la aplicación brinda la posibilidad de agregar a favoritos las estaciones de interés para la fácil navegación dentro de la app y que el usuario encuentre la información que necesita con pocos clics dentro de la app pensando en la seguridad del usuario.

Como vemos, los servicios y funcionalidades de las aplicaciones pueden brindar mejores experiencias de usuario, pero el desarrollo de la aplicación no lo es todo. También se observa que la arquitectura de software es igual de importante que el desarrollo del código fuente, es el esqueleto de las aplicaciones. Es la guía para que los desarrolladores sepan por dónde encaminar las funcionalidades y en cómo. En la investigación de (Alibasa, Santos, Glozier, Harvey, & Calvo, 2018), se observa una arquitectura para separar los datos identificables de los no identificables que pueden mantenerse anónimos y una arquitectura diseñada para aplicaciones móviles y web.

La investigación intenta crear una discusión sobre la arquitectura de software para almacenar y administrar datos recopilados en aplicaciones móviles enfocadas a la salud. Para ello intentan diseñar una aplicación web y móvil que guarde esta información de modo seguro. Para la seguridad se utilizó el algoritmo estándar de cifrado avanzado AES. Para ello esta investigación analiza los aspectos más importantes de la seguridad como la confidencialidad, integridad, control de acceso, disponibilidad y no repudio. Con este proyecto Alibasa no solo muestra la importancia de diseñar la arquitectura de una aplicación, también para la seguridad de la misma.

3 DESARROLLO DEL PROYECTO

3.1. SPRINT 1: ANÁLISIS DE LOS PATRONES DE DISEÑO

La primera unidad de análisis de este trabajo práctico se enfocó en el estudio de los patrones de diseño para el desarrollo de software. El objetivo de este análisis fue evaluar y posteriormente mejorar la calidad del software con los patrones de diseño más frecuentes. Se realizó una investigación para identificar los patrones de diseño más utilizados.

En primer lugar, se recolectó datos sobre los patrones de diseño, así como sobre su implementación y uso. A continuación, se analizó estos datos para evaluar cómo los patrones de diseño afectan a la calidad del sistema, especialmente en cuanto a su facilidad de uso, fiabilidad y rendimiento.

Se ha considerado utilizar los siguientes patrones de diseño:

Model-View-Controller (MVC): Este patrón se puede utilizar para separar la lógica de negocios de la aplicación de la representación visual de la información. El modelo representa la información y la lógica de negocios, la vista se encarga de la presentación visual y el controlador trabaja como intermediario de la vista y el modelo.

Singleton: Este patrón se puede utilizar para asegurarse de que solo exista una única instancia de un objeto en toda la aplicación. Por ejemplo, se lo puede utilizar para crear una única instancia de la base de datos o de la API de Google Maps en toda la aplicación.

Factory: Este patrón se puede utilizar para crear objetos en tiempo de ejecución sin tener que definir una clase. Por ejemplo, se puede utilizar para crear una estación o una ruta en función de los datos recibidos desde el servidor.

Observer: Este patrón se puede utilizar para notificar a los objetos interesados cuando se producen cambios en un objeto específico. Por ejemplo, se puede utilizar para notificar a la vista cuando se reciba una nueva posición GPS desde el servidor.

Estos son algunos patrones de diseño que pueden ayudar para mejorar la escalabilidad, la mantenibilidad y la reutilización del código en la aplicación.

El análisis de patrones de diseño es un proceso clave para entender cómo los patrones de diseño afectarán al desempeño y la calidad del software, y para tomar decisiones informadas sobre su uso en proyectos futuros.

3.2. Sprint 1: Análisis de las principales funcionalidades de la aplicación "Busity"

Dentro de la aplicación se observó cuatro funcionalidades que caracterizan a la aplicación de las demás aplicaciones vistas anteriormente en el estado del arte. Las funcionalidades son la parte más importante de una aplicación ya que, pueden captar más la atención y principalmente generar un servicio útil para los usuarios. Las funcionalidades del aplicativo "Busity", son muy ambiciosas y podrían ser muy útiles para los usuarios del transporte público de Quito. Con estas funcionalidades se podrían mejorar de manera muy notable la calidad del transporte público mediante un servicio externo. Entre las funcionalidades de la aplicación "Busity" está la funcionalidad de "Rutas" (observar las rutas del transporte público de Quito); funcionalidad "Llévame" (te ayudará a movilizarte con los buses de Quito); funcionalidad de "GPS" (se observará los buses en tiempo real); por último, pero no menos importante, la funcionalidad de "estaciones" (ofrece la posibilidad de visualizar en un mapa interactivo cualquier estación de Quito). A continuación, se detallará y se aclarará las cuatro funcionalidades del aplicativo.

3.2.1 Rutas

En la funcionalidad de rutas, se podrá conocer cualquier ruta del transporte público. En la página oficial de la secretaría de movilidad de Quito (Secretaría de movilidad, 2023) se puede observar las rutas de buses según las operadores, las operadores son compañías privadas que ofrecen servicio de transporte público por todo Quito, entre ellas está la operadora Compañía de Transportes 21 de Julio Cía. Ltda. y tiene como rutas las siguientes:

- Chorrera – San Juan – Bolivia
- San Salvador – Colegio Mejía
- Chorrera – Tejar – Bolivia
- Reserva

Si se cuenta con cierto conocimiento de los sectores de Quito, es posible tener una noción general de las rutas de la mencionada compañía. Sin embargo, en el caso de turistas o personas que están aprendiendo a movilizarse en Quito, puede resultar complicado familiarizarse con todos los sectores y no se obtendrá una comprensión clara hasta que se suban a dichos autobuses y recorran la ciudad durante un período considerable.

La funcionalidad de Rutas permite conocer las rutas de las operadoras que brindan servicios de transporte público. En la siguiente ilustración se puede observar las pantallas que el aplicativo “Busity” brinda a sus usuarios para que conozcan de inicio a fin cada ruta.

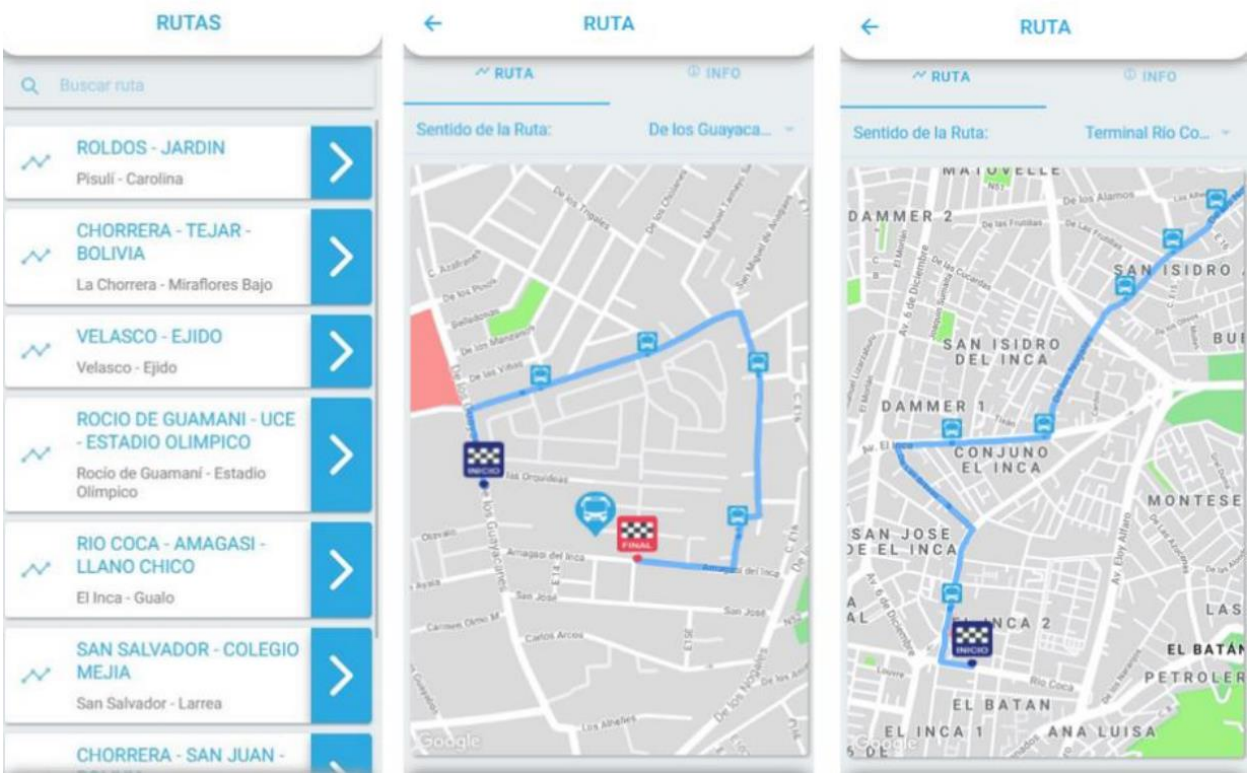

Ilustración 1 Funcionalidad de rutas

Observando la Ilustración 1, se puede ver que existen varias rutas, pero apenas se conocerá el nombre de cada ruta, pero al dar clic llevará a una pantalla donde se visualizará de inicio a fin la ruta y de igual manera el sentido de la ruta ya que puede ser de norte-sur o este-oeste o viceversa, también se tendrá una pestaña de información que mostrará los datos importantes de la ruta, como la operadora, el inicio y fin de la ruta, etc.

3.2.2 Llévame

Para la funcionalidad de Llévame, es necesario el uso del API de Google Maps, específicamente el servicio de Directions API y Maps JavaScript API, el primer API nos ayudará 100% para desarrollar esta funcionalidad, ya que la aplicación usará estas funcionalidades para brindar la funcionalidad de llévame dentro de la aplicación de Busity. La funcionalidad está basada en un punto de inicio, originalmente es la posición del GPS del usuario, pero puede ser un punto inicial distinto. Luego las direcciones y la guía serán brindadas por Directions API de Google Maps e integradas en el aplicativo. De esta manera, el usuario podrá saber cómo llegar de un punto a otro usando el transporte público de Quito. En la Ilustración 2,

se puede apreciar de qué manera el aplicativo muestra la información.

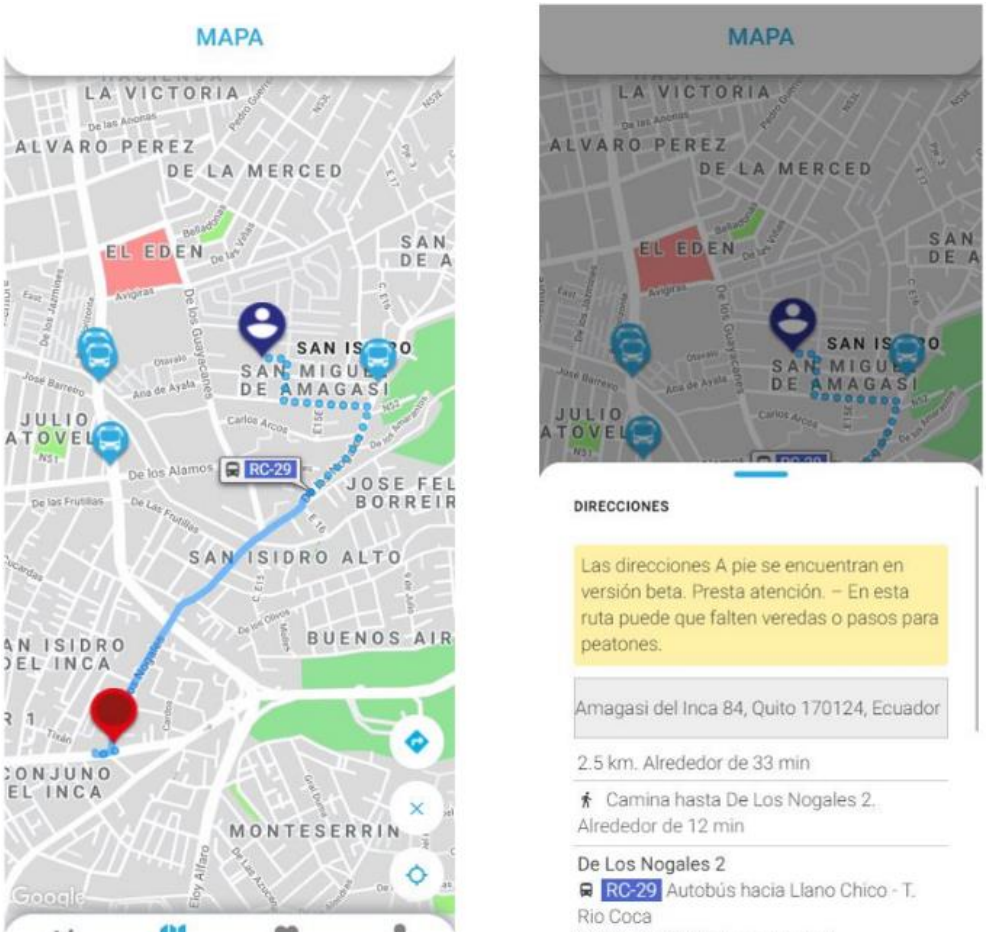

Ilustración 2 Funcionalidad de llévame

3.2.3 GPS DEL BUS

La funcionalidad principal de la aplicación es el GPS de los buses. Estos GPS estarían integrados a las unidades y de esta manera se sacaría la información que se mostraría a los usuarios. Para no sobrecargar el mapa principal, se diseñó de tal manera para que el usuario pueda entrar a las rutas y ver los buses que se encuentran en el recorrido y si se da clic al bus se abrirá un modal con las estaciones que el bus ya ha pasado. Para entender de mejor manera esta funcionalidad se presenta en la Ilustración 3 que detalla las pantallas de la aplicación “Busity”.

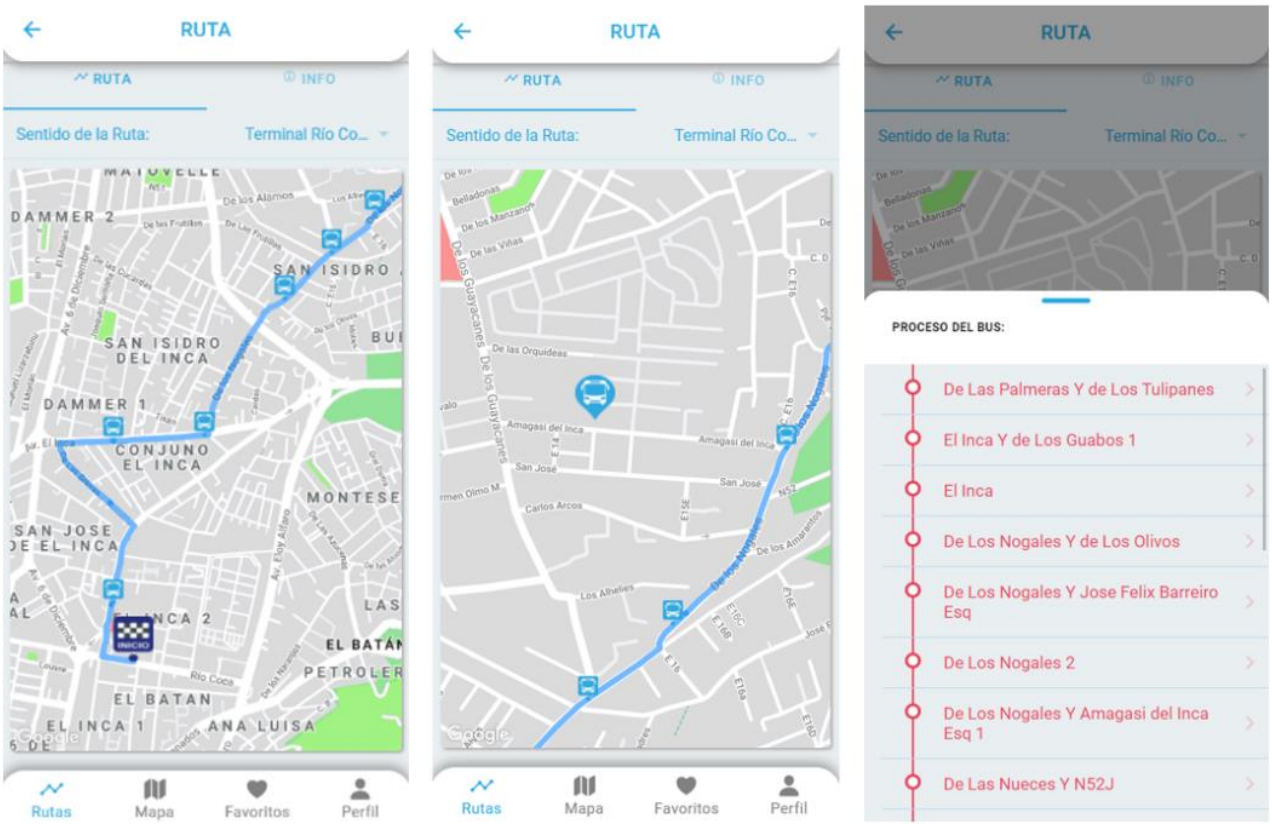

Ilustración 3 Funcionalidad de GPS

Al observar la ilustración, se puede entender de mejor manera que la funcionalidad del GPS va de la mano con la funcionalidad de la ruta, al ingresar a una ruta especifica, se pueden observar el o los buses que están en ese momento haciendo el recorrido de esa ruta y al dar clic en el bus se ve el proceso del bus y las estaciones que ya pasó, se pondrán de color verde.

3.2.4 Estaciones

La funcionalidad de estaciones ayudará al usuario a que identifique estaciones cercanas y lejanas a él para que pueda saber si por una estación cercana pasa una ruta que le podría llevar a su destino, también le dice el nombre de la estación y la dirección de donde se encuentra ubicada. También se podrá agregar estaciones a favoritos para que el usuario tenga un acceso rápido a las estaciones. De esta manera poder hacer que la navegación dentro de la aplicación sea más rápida y que obtenga la información que necesita pensando en la seguridad del usuario, ya que existe inseguridad dentro del transporte público y el riesgo se incrementa si el

usuario está utilizando su celular. La sección de favoritos ayuda a que el usuario con solo pocos clics llegue a la información que necesite. La información que se encuentra dentro de las estaciones son las rutas de buses que pasa por la estación y también si se da clic en las rutas le lleva directamente a la funcionalidad de rutas desde la estación que agregó a favoritos.

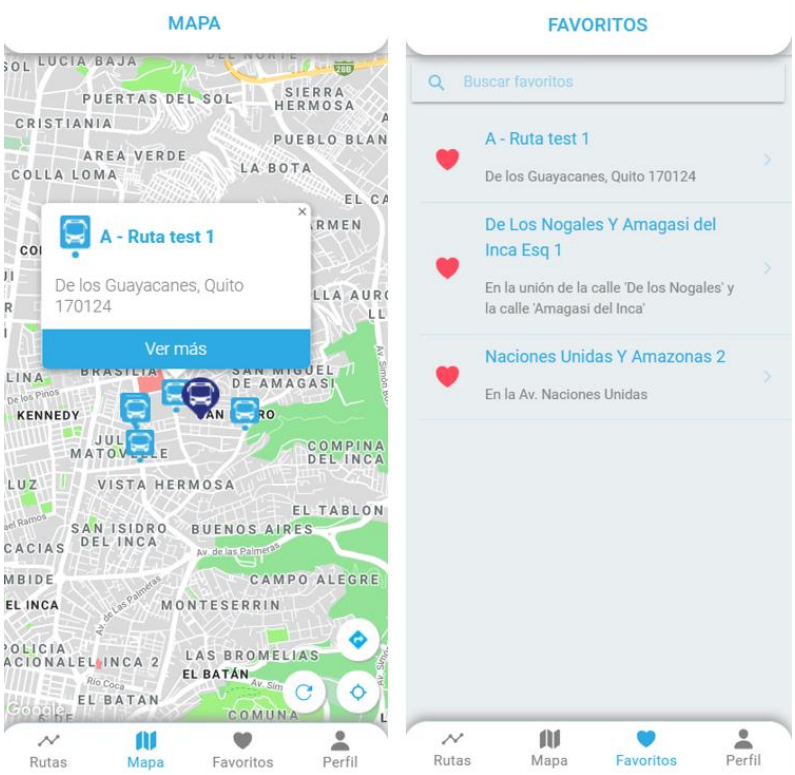

Ilustración 4 Funcionalidad de Estaciones

3.3. Sprint 2: Definir los requerimientos funcionales y los no funcionales

3.3.1 Requisitos funcionales:

Visualización de estaciones: La aplicación debe permitir al usuario ver todas las estaciones de transporte público en un mapa interactivo y también proporcionar información detallada sobre las rutas de transporte que pasan por cada estación al hacer clic en ella.

Visualización de rutas: La aplicación debe permitir al usuario ver las diferentes rutas de transporte público en un mapa interactivo y proporcionar información detallada sobre las estaciones que se encuentran en la ruta seleccionada.

GPS de los buses: La aplicación debe permitir al usuario ver en tiempo real la ubicación de los buses en la ruta seleccionada y proporcionar información sobre las unidades de bus en operación.

Función "Llévame": La aplicación debe permitir al usuario ingresar una dirección y proporcionar direcciones detalladas para llegar a esa ubicación utilizando el transporte público, incluyendo un mapa interactivo y una guía de direcciones.

3.3.2 Requisitos no funcionales:

Tiempo de respuesta rápido: La aplicación debe responder de manera rápida y eficiente sin interrupciones ni retrasos.

Facilidad de uso: La aplicación debe ser fácil de usar y no requerir una formación especializada para su uso.

Estabilidad: La aplicación debe ser estable y no presentar errores frecuentes.

Compatibilidad: La aplicación debe ser compatible con diferentes navegadores y sistemas operativos.

Disponibilidad: La aplicación debe tener una alta disponibilidad y estar disponible para su uso en cualquier momento.

Seguridad y privacidad: La aplicación debe ser segura y proteger la privacidad y los datos personales del usuario.

3.4. Sprint 2: Diseñar modelos de diagramas C4.

En el diseño de la aplicación de transporte público, se decidió utilizar una arquitectura cliente-servidor para garantizar una mejor escalabilidad, seguridad y rendimiento.

En una arquitectura cliente-servidor, el lado del cliente se encarga de mostrar e interactuar con el usuario, mientras que el lado del servidor se encarga de almacenar y gestionar la información y los recursos. (Taylor & Medvidovic, 2017) Esto permite que el sistema se pueda mantener y mejorar más fácilmente en el futuro, ya que los cambios en el servidor no afectarán la experiencia del usuario en el lado del cliente.

Además, al separar estas responsabilidades, se pueden implementar factores de seguridad más efectivas para asegurar la información y los recursos de la aplicación. También permite una mejor gestión de la carga, ya que el servidor puede utilizar técnicas de balanceo de carga para asegurarse de que todas las peticiones se procesen de manera eficiente.

La arquitectura cliente-servidor se utilizó en el diseño de la aplicación de transporte público para mejorar su escalabilidad, seguridad, rendimiento y sobre todo garantizar una buena experiencia de usuario.

El diseño de la arquitectura cliente-servidor para la aplicación de transporte público fue realizado con el objetivo de garantizar una solución escalable y eficiente para los usuarios. Se utilizó el modelo C4 para representar de manera clara y visual la estructura de los componentes de la aplicación y la relación que tienen entre ellos.

La arquitectura estaba formada por un servidor central encargado de procesar y almacenar la información, y varios clientes, que podían ser dispositivos móviles o páginas web, encargados de mostrar e interactuar con los datos. La distribución de responsabilidades entre el servidor y los clientes permitió a la aplicación adaptarse de manera eficiente a las necesidades de los usuarios, manteniendo una buena velocidad y disponibilidad.

Para diseñar los modelos C4 de la arquitectura cliente-servidor se usó la aplicación de Structurizr para los diagramas, se diseñó utilizando Structurizr DSL Language Reference, que nos ayuda a crear de una manera más rápida los diagramas de arquitectura. Los modelos se renderizaron utilizando el código que se encuentra en el enlace del repositorio Github https://github.com/dan984git/BusityC4Model/blob/master/busity.dsl o dando clic aquí.

3.4.1 Nivel 0: Contexto

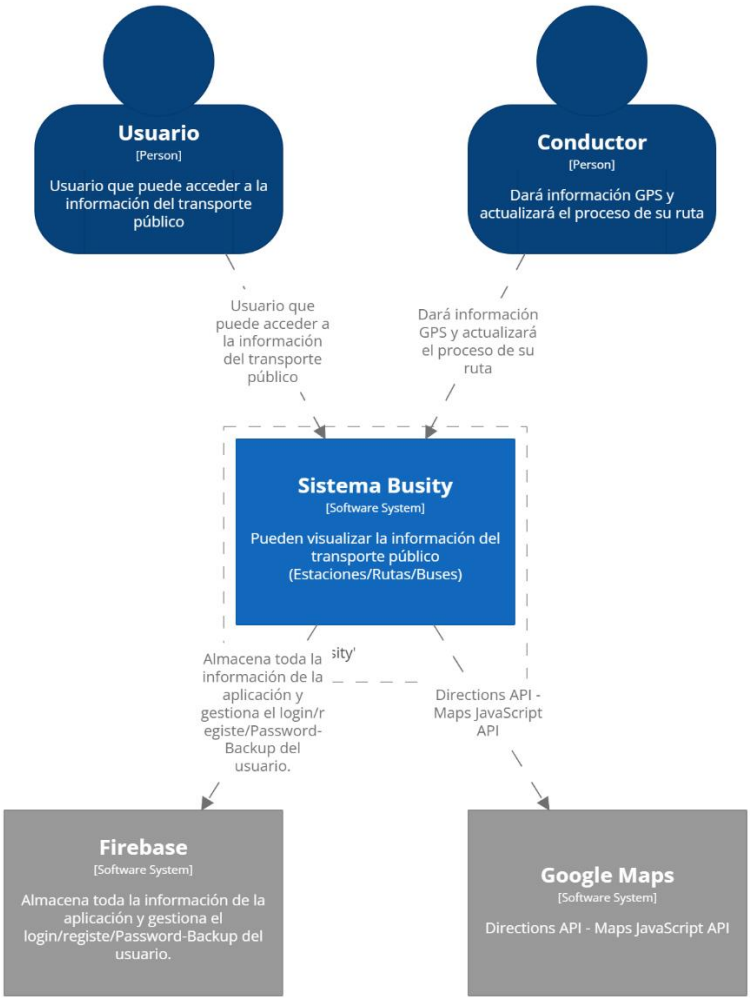

Ilustración 5 Modelo C4 Nivel Contexto

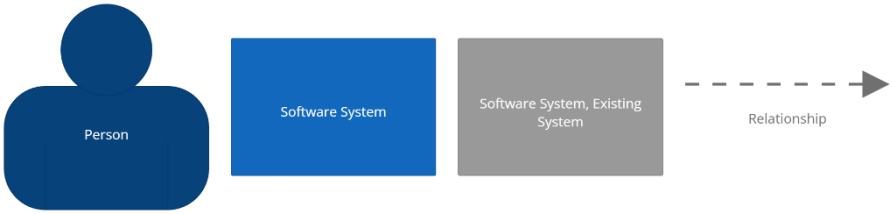

Ilustración 6 Modelo C4 Nivel Contexto Key

En este nivel del modelo C4, se pueden observar los servicios y dependencias de la aplicación Busity. Al tratarse de un nivel de contexto, se muestra únicamente cómo los usuarios utilizarán la aplicación. Además de utilizar Firebase para la base de datos y la autenticación, la aplicación también hace uso de las APIs de Google Maps. Estas APIs permiten obtener direcciones, calcular rutas y mostrar un mapa interactivo en la aplicación mediante JavaScript. Con esta integración, los usuarios de Busity pueden explorar y navegar de manera eficiente por los diferentes lugares y puntos de interés. La combinación de Firebase y las APIs de Google Maps brinda una experiencia fluida y completa a los usuarios, facilitando tanto la autenticación y gestión de datos como la navegación geográfica en la aplicación.

3.4.2 Nivel 1: Bloques de sistema

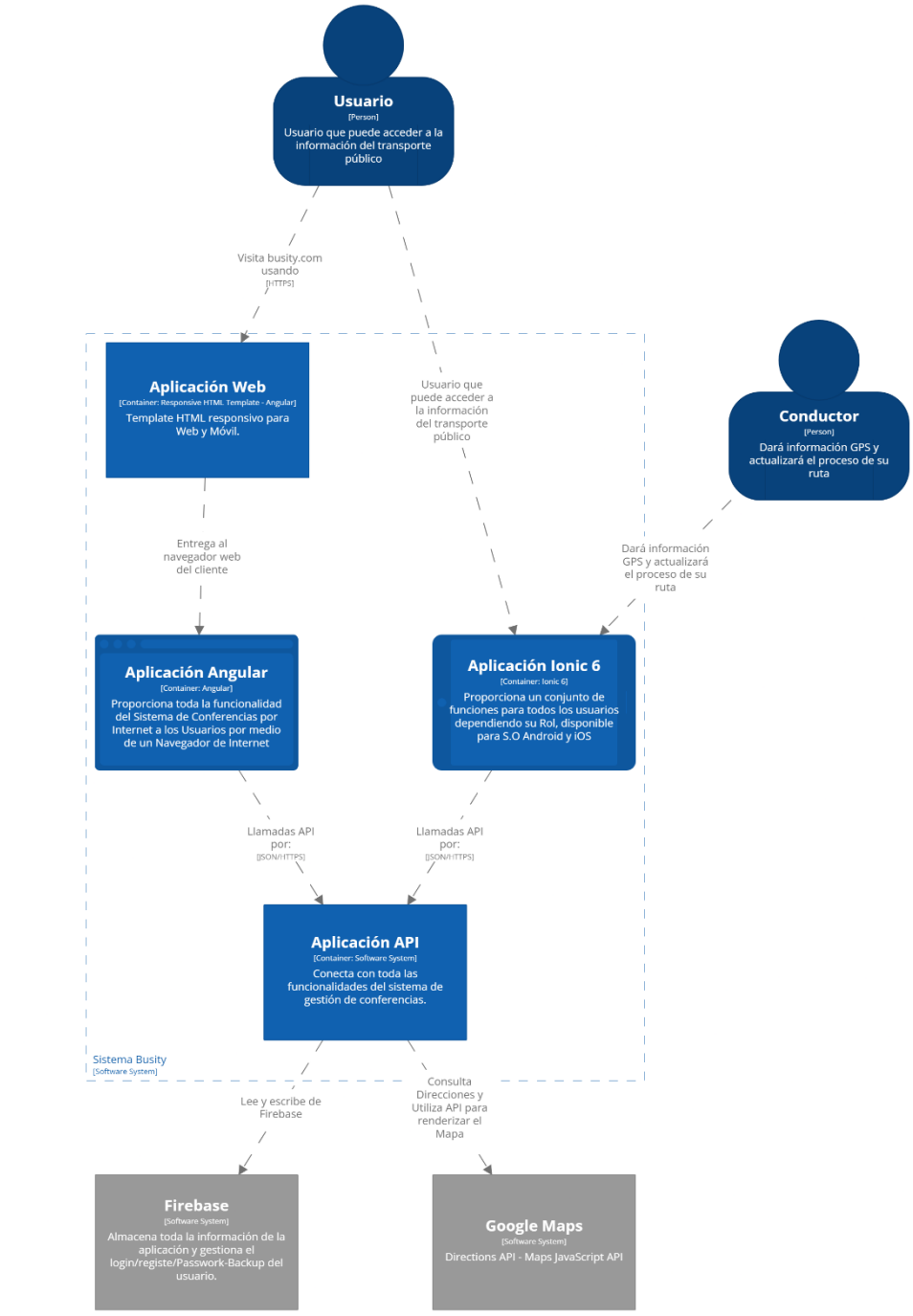

Ilustración 7 Modelo C4 Nivel Contenedor

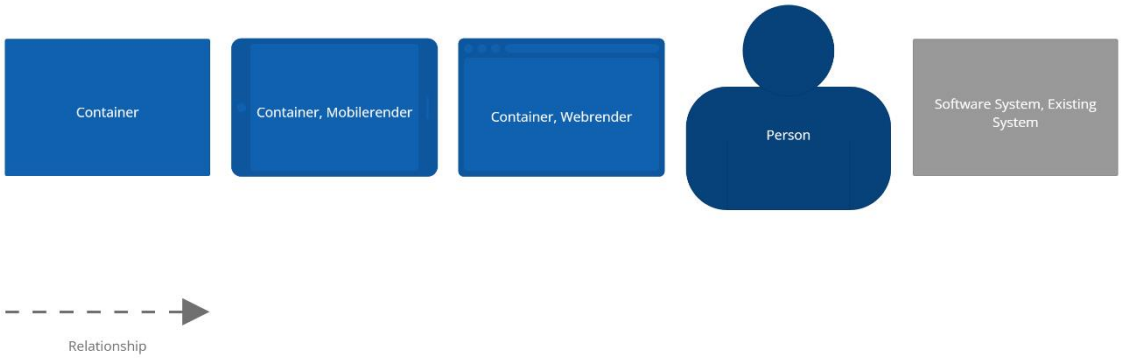

Ilustración 8 Modelo C4 Nivel Contenedor Key

Usuario desde el navegador: los usuarios pueden acceder a la aplicación Busity a través de un navegador web. Este bloque representa la interfaz de usuario web que permite a los usuarios interactuar con la aplicación desde sus dispositivos de escritorio o móviles mediante una interfaz amigable. Por otro lado, los usuarios también pueden acceder a la aplicación Busity mediante una aplicación móvil dedicada. Este bloque representa la interfaz de usuario específica para dispositivos móviles, lo que brinda a los usuarios una experiencia adaptada y optimizada para sus smartphones o tablets.

Conductores desde la aplicación móvil: los conductores tienen acceso a la funcionalidad específica de la aplicación Busity a través de la aplicación móvil. Este bloque se refiere a la interfaz de usuario diseñada para los conductores, que les permite proporcionar actualizaciones en tiempo real sobre su ubicación y estado.

Aplicación API: este bloque representa la capa de lógica y procesamiento central de la aplicación Busity, implementada utilizando el framework Ionic 6. Ionic 6 proporciona una plataforma para desarrollar aplicaciones híbridas que pueden ejecutarse tanto en navegadores web como en dispositivos móviles. Aquí es donde se encuentra la lógica empresarial y la gestión de datos de la aplicación Busity.

Firebase: este bloque se conecta a la capa de aplicación API y proporciona funcionalidades clave, como almacenamiento de datos en la base de datos y autenticación para la aplicación Busity. Firebase es una plataforma de desarrollo de aplicaciones móviles y web que facilita la gestión de la base de datos en tiempo real y la autenticación de usuarios.

Google Maps: la aplicación Busity hace uso de las APIs de Google Maps para obtener datos de ubicación, calcular rutas y mostrar mapas interactivos. Estas APIs permiten mostrar información geográfica relevante tanto para los usuarios como para los conductores, lo que mejora la experiencia general de navegación y localización en la aplicación.

3.4.3 Nivel 2: Componentes

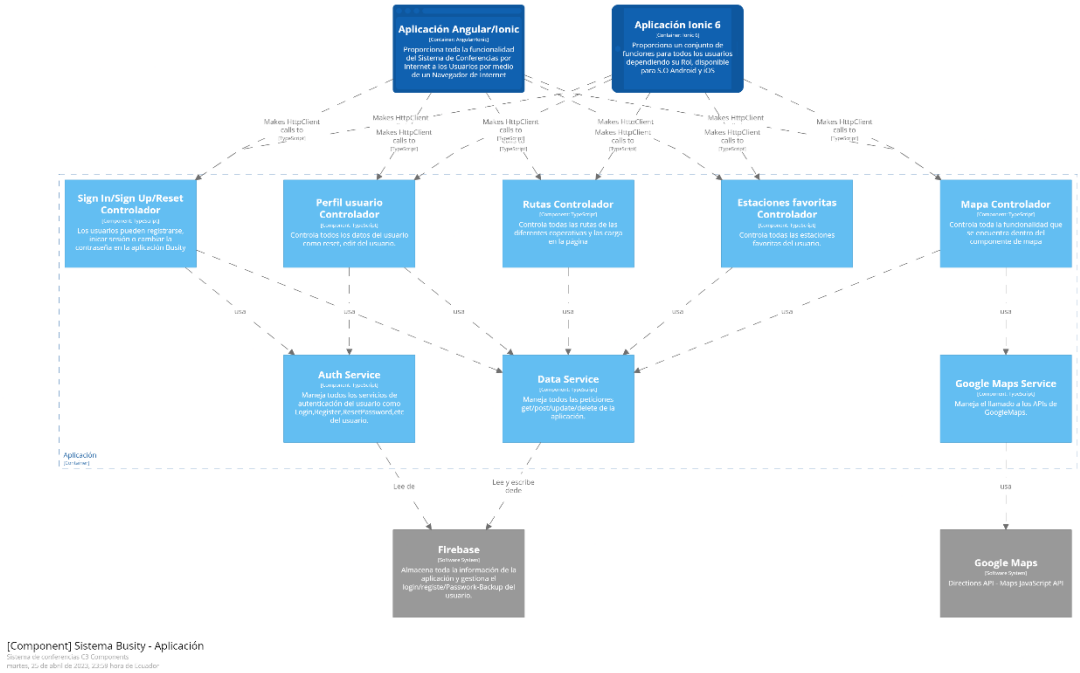

Ilustración 9 Modelo C4 Nivel Componente

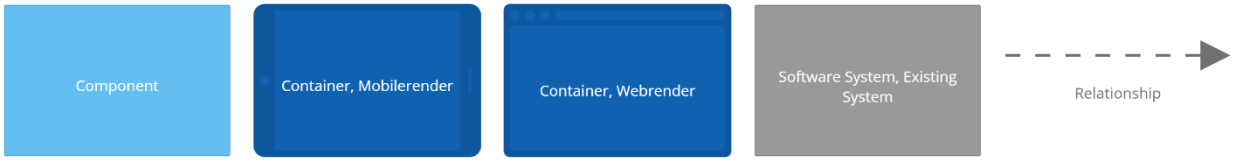

Ilustración 10 Modelo C4 Nivel Componentes Key

Aplicación Angular/Ionic 6: Este componente representa la interfaz de usuario de la aplicación Busity que los usuarios acceden a través de un navegador web. Los controladores de cada pantalla, implementados en TypeScript, se encargan de manejar la información y la lógica de la aplicación en este entorno.

Aplicación Ionic 6: Similar a la aplicación para navegadores web, este componente representa la interfaz de usuario específica para dispositivos móviles de la aplicación Busity. Los controladores de las diferentes pantallas, desarrollados en TypeScript, se utilizan para gestionar la información y la lógica de la aplicación en el contexto móvil.

Auth Service: Este componente de TypeScript se utiliza en los controladores de las pantallas de inicio de sesión, registro y perfil de usuario. Su función principal es llamar a las funciones proporcionadas por el servicio de autenticación de Firebase, permitiendo el registro, inicio de sesión y actualización de datos del usuario.

Data Service: En los controladores de las pantallas de inicio de sesión, registro, perfil de usuario, rutas, estaciones y mapa, se hace uso del Data Service. Este componente de TypeScript contiene todas las funciones necesarias para realizar llamadas a Firebase, permitiendo la lectura y escritura a la base de datos.

Google Maps Service: El controlador de la pantalla del mapa hace uso de este componente de TypeScript. El Google Maps Service proporciona funciones que realizan llamadas a las APIs de Google Maps, permitiendo obtener información de direcciones y mostrar mapas interactivos.

3.5. Sprint 3: Refactorizar el código fuente de la aplicación "Busity"

La refactorización de una aplicación móvil es una tarea crucial en el desarrollo de software. Este proceso consiste en optimizar el código de la aplicación para que sea más fácil de mantener, entender y mejorar en el futuro. La refactorización es especialmente importante en aplicaciones móviles de gran tamaño o que se espera que crezcan en el futuro, ya que puede mejorar su rendimiento y escalabilidad.

La refactorización implica técnicas como la eliminación de código redundante, la simplificación de la lógica del código y la reorganización de las clases y métodos para

que sean más coherentes y fáciles de entender. Los desarrolladores de aplicaciones móviles deben realizar la refactorización regularmente para garantizar la calidad del código de la aplicación.

La refactorización es un proceso esencial para mantener la eficiencia y efectividad del software. La refactorización de una aplicación móvil puede mejorar la calidad del código, reducir los errores y mejorar el rendimiento y la escalabilidad. Es una tarea que se debe hacer con regularidad para garantizar que la aplicación funcione sin problemas.

Durante la etapa de refactorización del código fuente de la aplicación móvil, se realizaron varias tareas para reestructurar los archivos y el código. Se actualizaron las versiones de las librerías y dependencias de la aplicación, se eliminaron dependencias obsoletas y redundantes, y se refactorizaron los métodos para que fueran más simples y eficientes. También se reorganizaron las clases, métodos, páginas, servicios y otros elementos en sus correspondientes carpetas.

Actualizar las dependencias de la aplicación fue un gran desafío, ya que se tuvo que reemplazar las clases y métodos obsoletos por otros más actualizados con una sintaxis diferente. Por ejemplo, se actualizó la dependencia de Firebase de la versión 8 a la 9, lo que implicó modificar la sintaxis de muchos de los métodos utilizados. Otro reto importante fue la actualización de Angular de la versión 12 a la 15, que requería un salto de versiones importante sin afectar el funcionamiento normal de la aplicación.

Al eliminar las dependencias deprecadas, se logró reducir a la mitad el número total de dependencias utilizadas, pasando de 48 a 19. Esto permitió que la aplicación funcionara de manera más eficiente y con menos posibilidades de errores o fallos.

Durante la tarea de eliminación de código redundante, se trabajó en toda la aplicación, pero sobre todo en la página principal ya que se logró una gran mejora donde la cantidad de líneas de código se redujo significativamente de 966 a 390, manteniendo la misma funcionalidad y con una lógica más liviana, evitando así el código redundante y el conocido "código espagueti".

Por otro lado, durante el proceso de refactorización también se trabajó en la reorganización de las carpetas de la aplicación. Se realizó un esfuerzo para reubicar todas las páginas, componentes, servicios, pipes, interfaces, y demás archivos en sus carpetas correspondientes, con el objetivo de hacer más fácil y eficiente la ubicación de los archivos necesarios para el desarrollo y mantenimiento de la aplicación.

Esta tarea fue importante, ya que la estructura de carpetas adecuada puede mejorar significativamente la organización del proyecto y la productividad del equipo de desarrollo. Además, al tener una estructura de carpetas clara y bien definida, se facilita la tarea de localizar y modificar el código, lo que a su vez puede acelerar el proceso de desarrollo y reducir la posibilidad de errores. En general, la reorganización de las carpetas es una tarea crucial dentro del proceso de refactorización, y puede marcar la diferencia entre un proyecto bien organizado y uno confuso y difícil de mantener.

En la etapa de refactorización del código fuente se logró reducir la cantidad de dependencias de 48 a 19, lo que significa que existían dependencias que no eran esenciales para el funcionamiento de la aplicación, lo que a su vez mejoró el rendimiento de la aplicación. Además, al momento de actualizar las versiones de las librerías, se enfrentó con desafíos, como el cambio de sintaxis de métodos y la deprecación de algunos otros métodos y para ello se tuvo que investigar cuál era la nueva sintaxis y los nuevos métodos que se utilizaban para las funciones que se requerían especialmente en la dependencia de Firebase. Por otro lado, la reorganización de las carpetas y la eliminación del código redundante simplificó el código y mejoró la mantenibilidad del mismo. En general, la refactorización de la aplicación móvil permitió mejorar significativamente su calidad y rendimiento, lo que a futuro resultará en una mejor experiencia de usuario.

3.6. Sprint 3: Utilizar patrones de diseño de software.

En esta fase del proyecto, se empleó la información previamente analizada acerca de los patrones de diseño. Se consideró cada patrón estudiado, prestando especial atención a dos patrones principales para la aplicación. El primero de ellos fue el patrón Singleton, que se utiliza para garantizar que solo haya una instancia de ciertos objetos en la aplicación y facilitar el acceso a ellos desde cualquier parte del programa. El segundo patrón utilizado fue el patrón Observer, que permite mantener los datos de la aplicación sincronizados en tiempo real y notificar a los objetos interesados sobre cualquier cambio en dichos datos. Ambos patrones son muy útiles en el desarrollo de la aplicación de Busity, mejorando la eficiencia y organización del código, y su uso adecuado puede reducir errores y facilitar el mantenimiento de la aplicación en el futuro. Por tanto, se consideró importante incorporarlos en el diseño de la aplicación para lograr un resultado óptimo.

En esta etapa del proyecto, se utilizó el patrón Singleton para definir las instancias de Google Maps. Esto permitió asegurar la existencia de una única instancia y facilitar el acceso a ella desde cualquier componente dentro de la aplicación. Antes de la refactorización, se debía declarar el servicio de Google Maps repetidamente en cada uno de los componentes que lo requerían, lo que generaba un código redundante. Con el uso del patrón Singleton, se logró simplificar el código y hacerlo más eficiente.

Cabe destacar que el patrón Singleton no solo se utiliza en el contexto de Google Maps, sino que es ampliamente utilizado en el diseño de software para asegurar la existencia de una única instancia de un objeto, permitiendo su acceso desde cualquier parte del programa. Además, este patrón puede mejorar la eficiencia y reducir la complejidad del código, que debe ser primordial para cualquier proyecto de desarrollo de software.

En la aplicación, se utilizó el patrón Observer como segundo patrón de diseño de software, el cual resultó muy útil para hacer llamados a la base de datos. Con la reactividad que proporciona Firebase, los datos pueden actualizarse automáticamente sin necesidad de recargar la página o hacer una llamada API a la base de datos. Al usar el patrón Observer, los métodos y componentes que están subscriptos a él reciben la nueva respuesta sin tener que realizar una nueva petición. Esto es gracias a la funcionalidad que Firebase proporciona de forma nativa mediante Firebase Realtime Database o Cloud Firestore. En la aplicación, se optó por utilizar la segunda opción.

El patrón Observer es una técnica de programación que permite a un objeto observado (observable) realizar una notificación automáticamente a otros objetos que se han registrado para ser notificados (observadores) cuando se produce algún cambio en su estado. En Angular, se utiliza la clase RxJS Observable para implementar el patrón Observer en los servicios que realizan llamadas a la base de datos.

Además, cabe mencionar que en el aplicativo se implementó el patrón Observer en todos los llamados get y getById, lo que permitió suscribirse a los observadores y obtener la información cada vez que la data cambia. Esto mejoró la eficiencia y la experiencia de usuario al reducir el tiempo de carga para que retorne la información actualizada.

En la aplicación que se desarrolló, se utilizó una metodología de programación llamada patrón de arquitectura de componentes. Este enfoque fue posible gracias al uso del framework Angular, que permitió estructurar cada página de la aplicación de manera única y con responsabilidades específicas.

Con este patrón, cada componente tiene una tarea concreta que cumple mediante la combinación de otros componentes independientes y reutilizables. Al utilizar componentes modulares, se pueden emplear en diferentes partes de la aplicación, con solo llamarlos de manera sencilla y con diferentes datos.

La gran ventaja de este patrón es la alta reutilización de los componentes, lo que reduce el tiempo de trabajo y mejora la calidad del software. Es decir, cada componente puede ser utilizado en diferentes contextos y en diferentes partes de la aplicación. De esta manera, se crea una aplicación modular, escalable y fácil de mantener.

En conclusión, la metodología de programación de patrón de arquitectura de componentes permitió estructurar cada página de la aplicación con responsabilidades específicas y tareas concretas. Además, con la ayuda de Angular, se logró crear componentes reutilizables que se pueden utilizar en diferentes contextos y partes de la aplicación. Todo esto resultó en una aplicación modular, escalable y fácil de mantener, lo que redujo el tiempo de desarrollo y mejoró la calidad del software.

3.7. Sprint 4: Crear prototipo de una aplicación web con la arquitectura diseñada.

Para crear la aplicación web con la arquitectura diseñada, se necesitó del framework de Ionic conjuntamente con el framework de Angular que permitió la creación del prototipo web. Cabe destacar que la aplicación móvil creada anteriormente a la cual se refactorizó, no era responsiva para los demás dispositivos como tablets o computadoras. Entonces para ello se realizó las respectivas modificaciones para que la aplicación pudiera ser responsiva, visualizarse desde cualquier dispositivo y acceder desde internet.

Para la arquitectura cliente-servidor se utilizó el framework de Ionic y de Angular como cliente y a Firebase como servidor. La arquitectura cliente-servidor funciona de la siguiente manera:

- El cliente (en este caso la aplicación web) envía solicitudes al servidor a través de la API de Firebase para realizar operaciones CRUD.

- El servidor (Firebase) procesa estas solicitudes y responde con los datos solicitados.
- Los datos se sincronizan automáticamente entre el cliente y el servidor en tiempo real, lo que permite ver la información actualizada hecha desde un dispositivo en todos los demás dispositivos que utilicen la aplicación.

Esta arquitectura es altamente escalable y fácil de mantener, ya que Firebase se encarga de administrar la base de datos y la infraestructura de backend, mientras que a los desarrolladores nos permite enfocarnos plenamente en el desarrollo de la aplicación.

La aplicación que desarrollamos para realizar pruebas fue desplegada utilizando Appflow de Ionic. Esta herramienta permitió automatizar el proceso de construcción, prueba y despliegue de la aplicación, lo que simplificó en gran medida la tarea de asegurar de que la aplicación funcionara correctamente antes de ser lanzada. Además, permitió realizar pruebas en diferentes entornos y plataformas, lo que permitió identificar y corregir problemas específicos de cada plataforma. En general, el uso de Appflow fue muy beneficioso, ya que ayudó acortar el tiempo de desarrollo y mejorar la calidad de la aplicación visualizando posibles errores desde diferentes dispositivos.

El link de aplicación desplegada temporalmente se puede encontrar en el siguiente repositorio GitHub, https://github.com/dan984git/App-Busity o dando clic aquí.

3.8. Sprint 4: Pruebas de usabilidad.

Durante la fase de pruebas de usabilidad, se decidió emplear una aplicación web para recopilar información sobre el prototipo y hacer algunas preguntas a los usuarios. Para ello, se optó por Maze, una aplicación capaz de recopilar datos relevantes sobre el prototipo, como el heatmap de cada pantalla, que muestra dónde los usuarios hacen clic para interactuar con la aplicación, y la lista de pantallas que el usuario visita. Esta herramienta resultó invaluable para obtener información útil y valiosa sobre nuestros usuarios de prueba.

Pero Maze no se limita a recopilar información sobre el prototipo. También se puede usar para hacer preguntas a los usuarios, lo que proporciona más información sobre su experiencia con el prototipo. Los comentarios y sugerencias recibidos a través de estas preguntas resultaron muy valiosos para mejorar el diseño y la funcionalidad del prototipo.

Una de las ventajas de Maze es que es fácil de usar para los usuarios, lo que les permite interactuar con el prototipo de manera natural y sin esfuerzo. De esta manera, la información recopilada es más precisa y representa mejor la experiencia del usuario.

Maze resultó ser una herramienta excelente para recopilar información valiosa sobre la usabilidad de nuestro prototipo. La información recopilada nos permitió identificar los aspectos que debían mejorarse y realizar los cambios necesarios para ofrecer una experiencia de usuario satisfactoria.

3.8.1 Test de usabilidad

La prueba de usabilidad constó de nueve pasos que nuestros usuarios de prueba siguieron para completar el test. La prueba incluye una serie de preguntas y tareas que permiten evaluar la usabilidad del prototipo en diferentes situaciones y escenarios.

El primer paso para los usuarios fue responder algunas preguntas para poner en contexto el prototipo que se iba a probar. Estas preguntas incluyeron: ¿Crees que el servicio de transporte público de Quito es adecuado para los pasajeros? ¿Te gustaría usar una aplicación web/móvil que te ayude a movilizarte utilizando el transporte público?

Después, se procedió a realizar una primera aproximación a la aplicación, específicamente el proceso de registro e inicio de sesión. Los usuarios recibieron la tarea de crear una cuenta y luego iniciar sesión. Una vez que completaron la tarea, se les preguntó si encontraron el proceso de registro complicado o no.

A continuación, se les pidió a los usuarios que probaran la funcionalidad de "Llévame", que les permitía interactuar con un mapa para trazar una ruta desde un punto a otro utilizando el transporte público. Una vez que los usuarios completaron la tarea, se les preguntó si encontraron la funcionalidad fácil de utilizar.

Luego, se les animó a navegar por el prototipo para que pudieran descubrir nuevas funcionalidades y contenido dentro de la aplicación. Se les pidió que exploraran libremente la aplicación y luego respondieran a la pregunta si la aplicación era intuitiva o no. El objetivo de esta tarea fue determinar si a medida que los usuarios navegaban por la aplicación, podían entender todo lo que veían y si había algún elemento que no entendieran.

Finalmente, se les preguntó si usarían la aplicación para movilizarse utilizando el transporte público.

La prueba de usabilidad constó de nueve pasos que permitió evaluar la usabilidad del prototipo en diferentes situaciones y escenarios. Los usuarios realizaron una serie de tareas y respondieron preguntas para proporcionar retroalimentación sobre la funcionalidad, facilidad de uso e intuición de la aplicación. Esta información se utilizó para mejorar y optimizar la aplicación antes de su lanzamiento al público.

4 Resultados y discusión

4.1. Implementación de la arquitectura cliente-servidor

La arquitectura cliente-servidor que se ha propuesto para los aplicativos móviles y web del transporte público de Quito ha arrojado resultados positivos en diversos aspectos.

En términos de eficiencia y escalabilidad, esta arquitectura ha permitido una distribución eficiente de las tareas entre el cliente (la aplicación móvil o web) y el servidor. Además, la arquitectura ha sido diseñada de manera escalable, lo que nos

permite adaptarnos a futuras expansiones y al crecimiento de la aplicación. Gracias a Firebase, podemos escalar rápidamente los servicios que nos proporciona y satisfacer la demanda de los usuarios cuando exista un alto consumo de los servicios.

Así mismo, la arquitectura cliente-servidor ha brindado mayor flexibilidad y modularidad en el desarrollo de los aplicativos móviles y web. Al separar las responsabilidades entre el cliente y el servidor, se puede facilitar la actualización y mantenimiento de los aplicativos de manera independiente sin afectar al sistema completo. Esto puede brindar mayor agilidad en el desarrollo y actualización de los aplicativos, lo que permite adaptarse rápidamente a cualquier cambio.

4.1.1 Implementación de la arquitectura a nivel 2 del modelo C4

A nivel de código, la arquitectura cliente-servidor se implementó mediante el uso de controladores en cada página de la aplicación. Estos controladores actúan como intermediarios entre el cliente y el servidor, y son responsables de manejar la funcionalidad de la página en cuestión. Por ejemplo, la página de mapas tiene un controlador que maneja la visualización del mapa llamando al servicio de Google Maps, también el uso del API de direcciones para que el usuario haga uso de la funcionalidad de llévame, y también para manejar las estaciones que se visualizan en el mapa. De esta manera, el controlador hace uso del servidor y del cliente para el correcto funcionamiento de cada página.

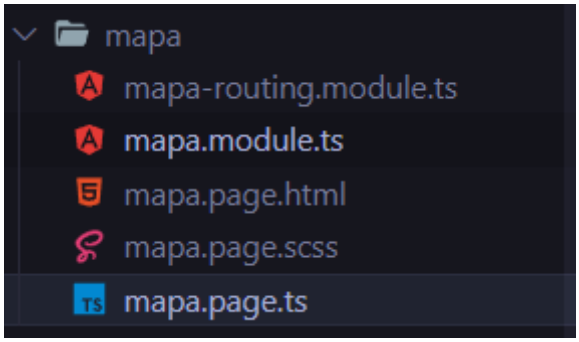

Ilustración 11 Controlador de mapa

En la ilustración anterior, se puede observar el controlador del componente mapa "mapa.page.ts", que para su implementación se utiliza el lenguaje de programación TypeScript, que ofrece una serie de características avanzadas y herramientas para el desarrollo de aplicaciones web modernas. Los controladores creados en TypeScript se comunican con otros servicios importantes, como la autenticación de usuarios, el acceso a los datos almacenados en Firebase y la integración de los APIs de Google Maps.

Por otro lado, los servicios son también programados en TypeScript. En estos, se llevan a cabo las operaciones CRUD y los llamados a las APIs necesarios. Luego, los controladores utilizan estos servicios para implementar la lógica de la aplicación.

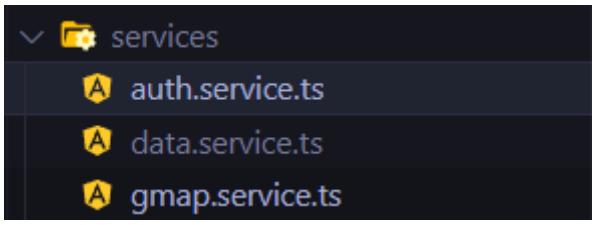

Ilustración 12 Servicios de Busity

4.2. Resultados de la refactorización de la aplicación "Busity"

Gracias a la refactorización de la aplicación móvil, esta ha experimentado una mejora en el rendimiento y la eficiencia debido a la implementación de patrones de diseño. El patrón Observer ha permitido una mejor gestión de eventos y notificaciones, lo que ha optimizado la comunicación y la actualización de datos

entre componentes de la aplicación. Por otro lado, el patrón Singleton ha ayudado a centralizar la gestión de ciertos recursos y servicios, evitando redundancias y mejorando la gestión de recursos en la aplicación.

Además, la arquitectura cliente-servidor ha permitido una distribución más eficiente de las tareas entre el cliente (la aplicación móvil) y el servidor central. Esto ha optimizado la carga de datos y el procesamiento en ambos lados, lo que ha mejorado el rendimiento y la velocidad de respuesta de la aplicación.

La eliminación de código redundante y dependencias que ya no hacían falta ha contribuido a la simplificación del código y la mejora de la legibilidad y mantenibilidad del mismo. Esto ha facilitado la identificación y corrección de posibles errores y ha agilizado el proceso de desarrollo y actualización de la aplicación.

La actualización de las dependencias de Firebase y Angular a las versiones más recientes (Firebase 9 y Angular 15) ha brindado acceso a nuevas funcionalidades y mejoras de rendimiento y seguridad. Esto ha permitido mantener la aplicación actualizada y alineada con las últimas tecnologías y mejores prácticas, lo que contribuye a una mayor estabilidad y seguridad del sistema.

La refactorización de la aplicación "Busity" en Ionic 6 con la implementación de patrones de diseño como el patrón Observer y el patrón Singleton, junto con una arquitectura cliente-servidor, y la eliminación de código redundante, actualización de la lógica de negocio y la actualización de dependencias ha generado resultados positivos en términos de rendimiento, eficiencia, legibilidad, mantenibilidad y actualización de la aplicación. Estos cambios han mejorado la calidad de la aplicación y la experiencia del usuario, lo que ha llevado a una mayor satisfacción del usuario y un desarrollo más eficiente.

Una de las principales mejoras que se lograron tras la refactorización del código, fue la reducción significativa en el número de dependencias del proyecto. Antes de la refactorización, el proyecto contaba con un total de 48 dependencias, lo que dificultaba la gestión y mantenimiento del código.

Luego de la refactorización, se logró reducir el número de dependencias en un 60%, quedando solamente 19 dependencias necesarias para el correcto funcionamiento del proyecto. Esta reducción no solo facilita la gestión del código, sino que también mejora la eficiencia del proyecto y reduce los riesgos de conflictos y errores en el futuro.

Además, la refactorización permitió una mejor organización y estructuración del código, lo que hace que sea más fácil de mantener y actualizar en el futuro.

Ilustración 13 Dependencias previas a la refactorización

Ilustración 14 Dependencias posterior a la refactorización

La optimización de la lógica de negocio de la aplicación es uno de los logros más destacados de la refactorización del código. En concreto, se logró reducir la cantidad de líneas de código necesarias para implementar la misma funcionalidad en un impresionante 59.62%. Antes de la refactorización, la cantidad de líneas de código era de 966, mientras que después se logró reducir a solamente 390 líneas de código.

Es importante destacar que esta reducción de la cantidad de líneas de código no afectó en absoluto la funcionalidad de la aplicación. Todas las herramientas y características se mantuvieron intactas, y la aplicación sigue ofreciendo la misma calidad y rendimiento que antes de la refactorización.

La reducción de la cantidad de líneas de código tiene un impacto directo en el rendimiento de la aplicación. Con menos líneas de código, la aplicación se ejecuta de manera más rápida y eficiente, lo que se traduce en una mejor experiencia de usuario y una mayor velocidad de respuesta.

Además, la simplificación del código también hace que sea más fácil de entender y mantener. Esto significa que futuras actualizaciones y mejoras serán más fáciles de implementar, lo que garantiza la escalabilidad y sostenibilidad del proyecto en el futuro.

Con la ayuda de la aplicación Araxis Merge, se obtuvo estadísticas comparando solamente el código del controlador del componente de mapa, dándonos datos muy significativos para notar la refactorización que hubo solamente en un controlador.

	Between Files 1 and 2	
Description	Text Blocks	Lines
Unchanged	70	228
Changed	49	946
Inserted	10	11
Removed	12	171

Ilustración 15 Estadísticas en Araxis Merge del archivo mapa.page.ts pre y post refactorización.

4.3. Implementación del prototipo de aplicación web de Busity

En el prototipado de la aplicación web de Busity, la implementación de la arquitectura cliente-servidor ha permitido una separación clara de responsabilidades entre el cliente (la aplicación web) y el servidor (Firebase). Esto ha facilitado la gestión de datos y la lógica de negocio, permitiendo una mayor flexibilidad y escalabilidad en el desarrollo de la aplicación. El uso de Ionic 6 y Angular como tecnologías del lado del cliente ha proporcionado un marco de desarrollo robusto y moderno. Ionic 6, junto con Angular, ha permitido la creación de una interfaz de usuario atractiva y funcional, con una navegación fluida y una experiencia de usuario intuitiva.

La elección de Firebase como servidor ha aportado ventajas significativas, como la facilidad de uso, la escalabilidad y la disponibilidad de herramientas y servicios para la gestión de bases de datos en tiempo real, autenticación de usuarios, almacenamiento de archivos y notificaciones push, entre otros. Esto ha agilizado la implementación de la lógica del servidor y la integración con el cliente, permitiendo un desarrollo rápido y eficiente del prototipo.

El uso de una arquitectura cliente-servidor ha permitido una mayor flexibilidad en la evolución del prototipo, ya que los cambios en la lógica del servidor se pueden realizar sin afectar la interfaz de usuario del cliente. Esto ha facilitado la iteración y mejora continua del prototipo, permitiendo adaptarse rápidamente a los requisitos y necesidades cambiantes del proyecto.

El uso de Ionic 6, Angular, Firebase y la arquitectura cliente-servidor han contribuido a la creación de un prototipo funcional y atractivo que cumple con los objetivos del proyecto y sienta las bases para el desarrollo de una aplicación web completa y exitosa.

Luego de completar esta fase del proyecto, se logró obtener un prototipo completamente funcional de la aplicación web. Este prototipo puede ser accedido tanto desde navegadores web como desde dispositivos móviles, y cuenta con una interfaz completamente responsiva para brindar al usuario una experiencia de usuario óptima.

Gracias a la implementación de una arquitectura cliente-servidor clara y bien estructurada, la aplicación cuenta con un alto nivel de eficiencia y escalabilidad. Además, se ha puesto especial atención en el diseño y la experiencia de usuario para garantizar que la aplicación sea atractiva y fácil de utilizar para todo tipo de usuarios.

El prototipo web lo pueden encontrar en el siguiente repositorio Github, https://github.com/dan984git/App-Busity o dando clic aquí.

4.4. Resultados de las pruebas de usabilidad del prototipo

La realización de pruebas de usabilidad del prototipo web utilizando la aplicación de Maze ha proporcionado resultados positivos en varias áreas clave. En primer lugar, el uso de mapas de calor generados por Maze ha brindado información valiosa sobre las áreas en las que los usuarios han hecho clic y cómo han

interactuado con la aplicación. Esto ha permitido identificar patrones de comportamiento, identificar áreas de interés y comprender cómo los usuarios interactúan con la interfaz de usuario.

Además, las preguntas antes y después de la prueba de usabilidad han brindado información valiosa sobre la percepción de los usuarios en términos de facilidad de uso de las funcionalidades, la complejidad del registro y login, y su disposición a usar la aplicación en el futuro. Los resultados positivos en estas preguntas indican que los usuarios encuentran las funcionalidades fáciles de utilizar y que el proceso de registro y login no es complicado o tedioso debido a la verificación del correo electrónico.

Estos resultados positivos en las pruebas de usabilidad son indicativos de que el prototipo web ha sido diseñado teniendo en cuenta las necesidades y preferencias de los usuarios, lo que ha resultado en una interfaz de usuario intuitiva y fácil de usar. Esto ha mejorado la experiencia del usuario y ha aumentado la probabilidad de que los usuarios utilicen la aplicación en el futuro.

La información obtenida de las pruebas de usabilidad y las respuestas de los usuarios antes y después de probar el prototipo han permitido realizar mejoras y ajustes en la interfaz de usuario y la experiencia del usuario, lo que ha contribuido a la creación de una aplicación web más atractiva y funcional. Además, estos resultados positivos también respaldan la toma de decisiones informadas en el desarrollo futuro de la aplicación, lo que puede ayudar a garantizar una mayor adopción y satisfacción del usuario en el futuro. Las pruebas de usabilidad han brindado resultados positivos en términos de la experiencia del usuario y la aceptación de la aplicación, lo que respalda el desarrollo exitoso del prototipo web.

Los resultados obtenidos de cada paso de la prueba de usabilidad son:

Primer paso:

¿Crees que el servicio de transporte público de Quito es digno para los pasajeros?

Ilustración 16 Respuesta a primera pregunta

Segundo paso:

¿Te gustaría una aplicación web/móvil que te ayude a movilizarte usando el transporte público?

Ilustración 17 Respuesta a segunda pregunta

Tercer paso:

Regístrate e Inicia sesión en la aplicación de Busity

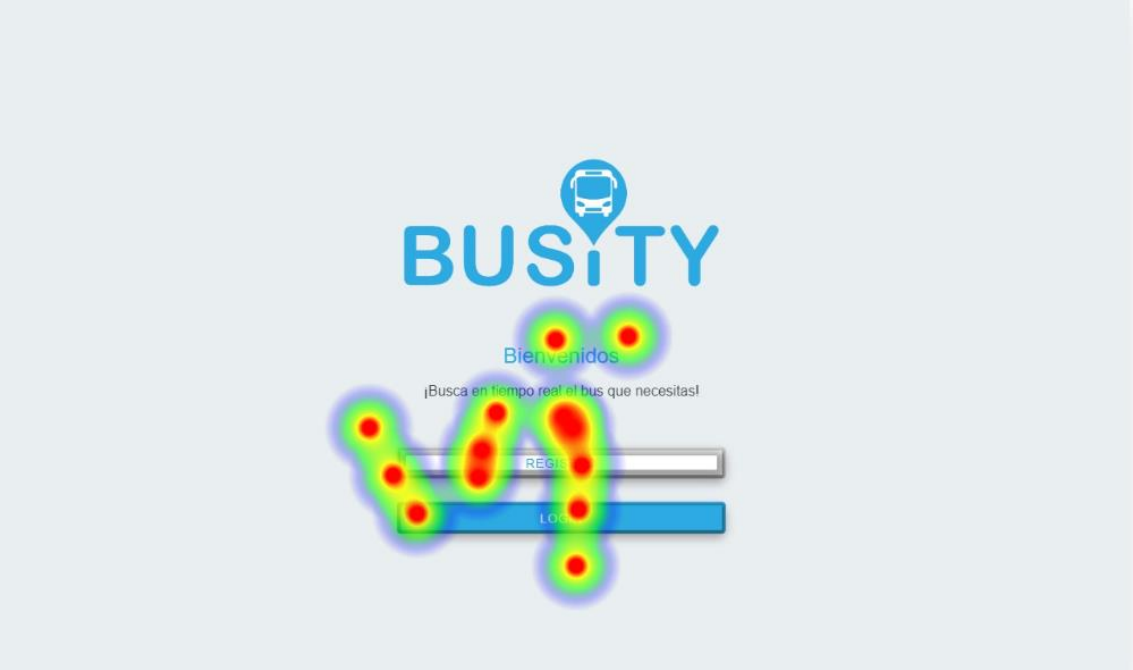

Ilustración 18 Mapa de calor registro e inicio de sesión

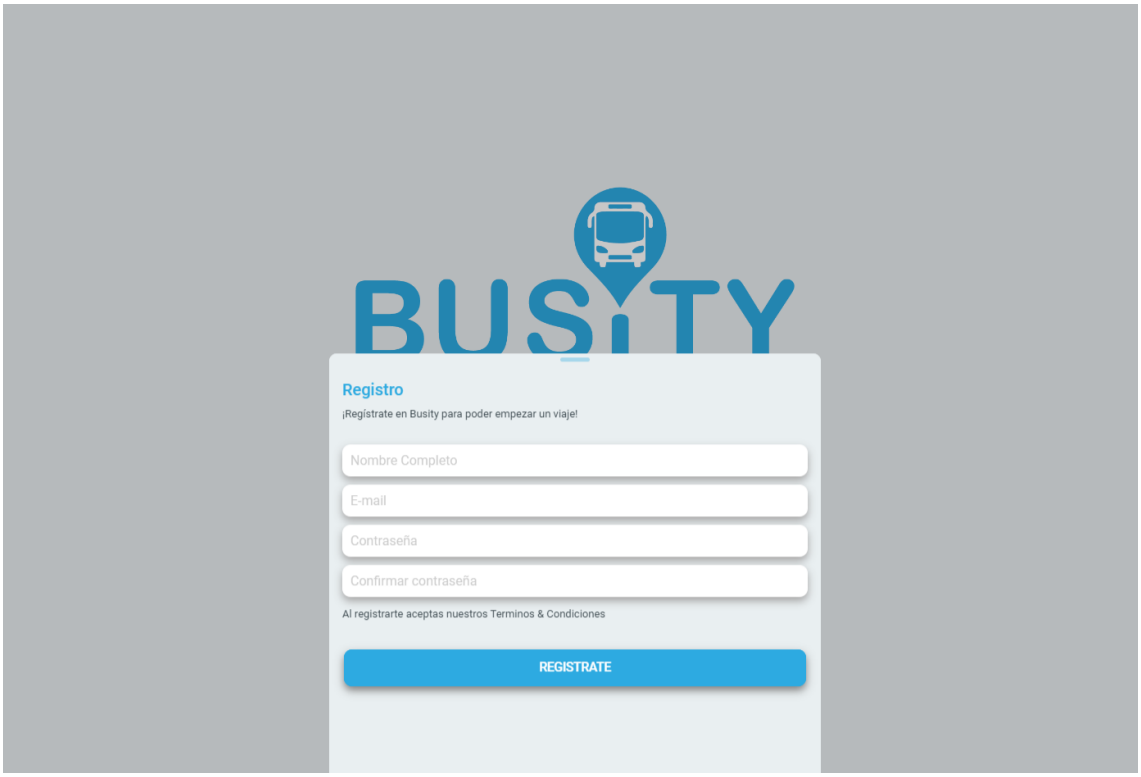

Ilustración 19 Registro aplicación Busity

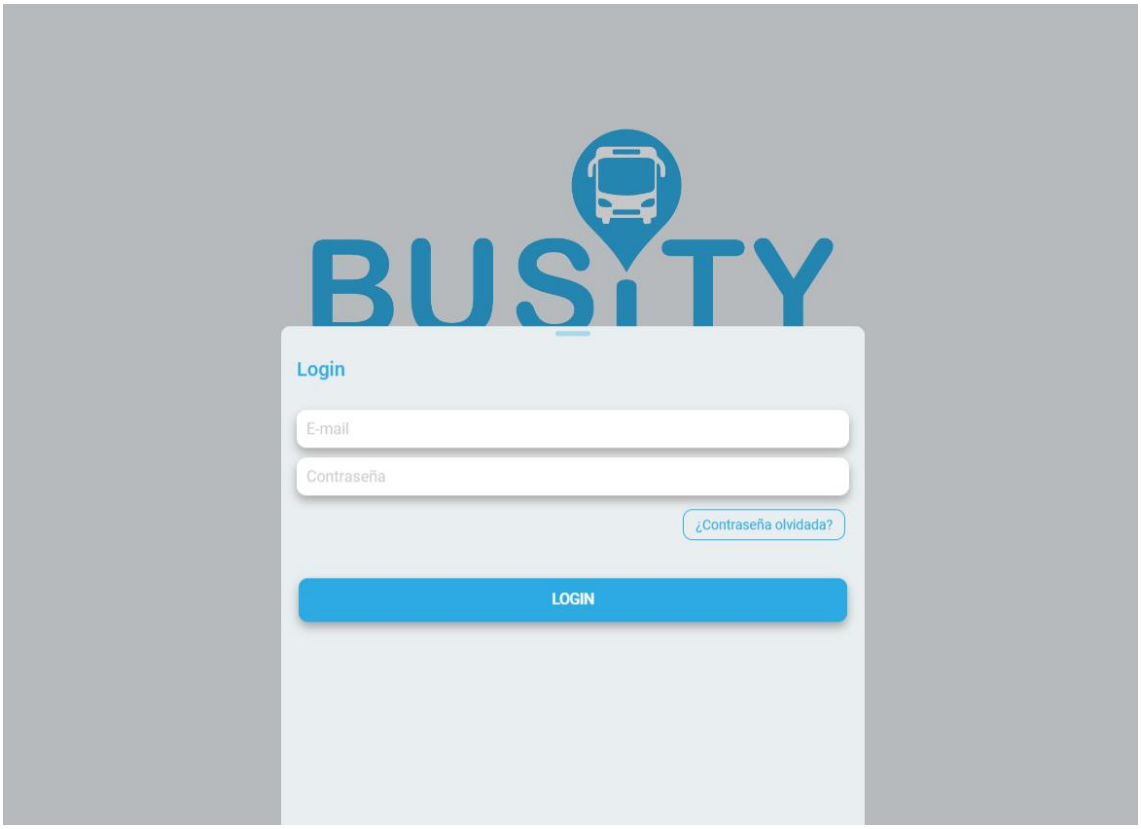

Ilustración 20 Inicio de sesión Busity

Se puede notar que, durante el proceso de prueba de nuestra aplicación, se utilizó la herramienta Maze para recopilar información importante acerca del comportamiento de nuestros usuarios al interactuar con el prototipo. Sin embargo, al hacer uso de esta herramienta, notamos que no es capaz de registrar pantallas modales dentro de la aplicación, lo que implica que no podemos recopilar información del heatmap en este tipo de pantallas. A pesar de esto, Maze sí puede capturar el comportamiento de los usuarios al dar clic en elementos dentro de los modales abiertos, lo que nos permite tener una idea general del flujo de interacción que los usuarios tienen con nuestra aplicación.

Es importante mencionar que, aunque la información que podemos obtener de los modales no es tan completa como la que podemos obtener de otras secciones de la aplicación, sigue siendo una parte importante del proceso de prueba de usabilidad. Esto nos permite detectar cualquier problema o dificultad que los usuarios puedan tener al interactuar con los modales, y hacer los ajustes necesarios para mejorar la experiencia general de usuario. Además, cabe destacar que la

herramienta Maze sigue siendo una herramienta muy poderosa para la recopilación de información y feedback de los usuarios, lo que nos permite seguir mejorando nuestra aplicación y brindar un mejor servicio a los usuarios.

Cuarto paso:

¿Fue complicado registrarte e iniciar sesión en la aplicación de Busity?

Ilustración 21 Respuesta del cuarto paso

Quinto Paso:

Usa la funcionalidad de Llévame

Para esto se les pidió a los usuario interactuar dentro del Mapa y puedan marcar un punto de destino para que Google Maps proporcione la información para poder llegar al destino.

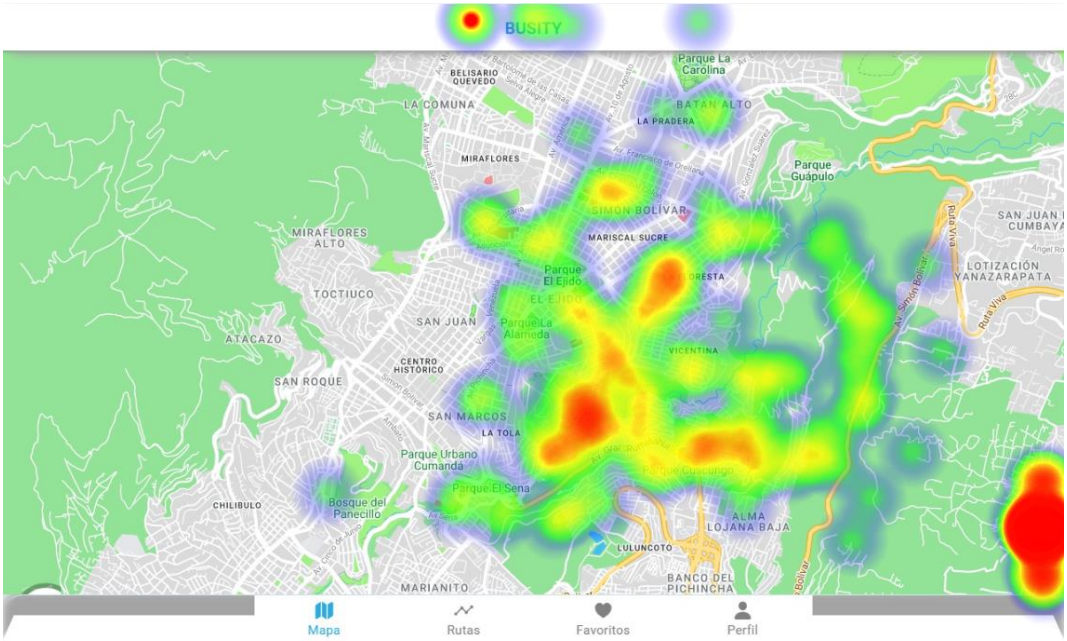

Ilustración 22 Mapa de calor funcionalidad llévame

Sexto paso:

¿Fue fácil utilizar la funcionalidad de llévame?

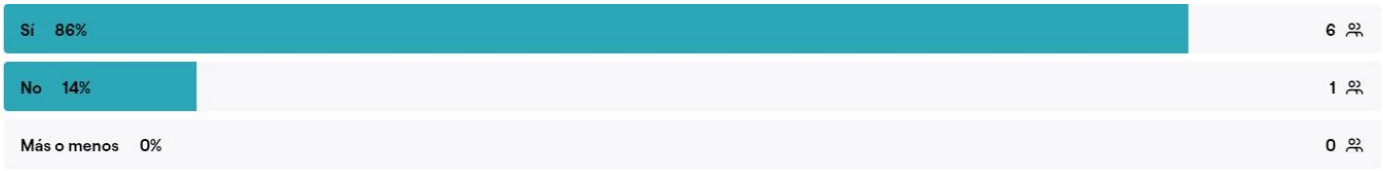

Ilustración 23 Respuesta sexto paso

Séptimo paso:

En este paso se les pidió a los usuarios de prueba que navegaran dentro de la aplicación para que pudieran encontrar nuevas funcionalidades y para comprobar si el prototipo es intuitivo o no.

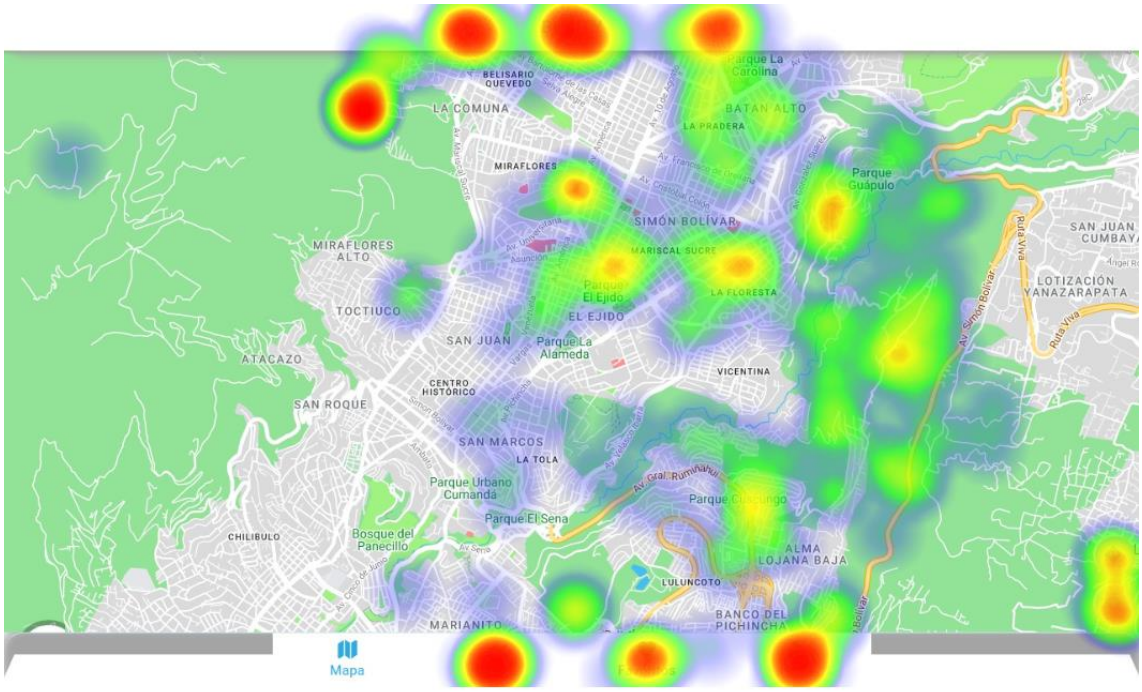

Ilustración 24 Mapa de calor de la navegación dentro de la aplicación

Octavo paso:

¿La aplicación web es intuitiva?

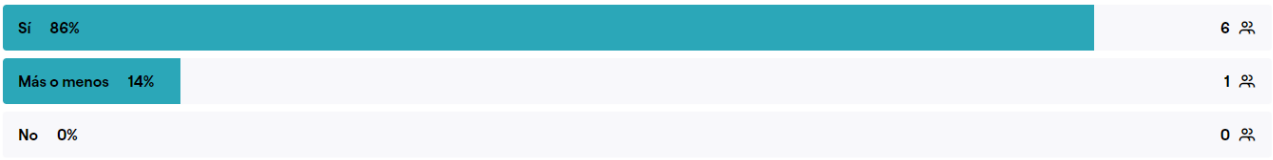

Ilustración 25 Respuesta del octavo paso

En base a los resultados de nuestra encuesta de usabilidad, podemos afirmar que la gran mayoría de los usuarios encontró nuestra aplicación bastante intuitiva y fácil de utilizar. Esto es un indicador muy positivo, ya que uno de nuestros principales objetivos es ofrecer una experiencia de usuario óptima y satisfactoria.

No obstante, durante el proceso de recolección de datos, se encontró un caso en el que un usuario reportó tener dificultades para utilizar ciertas funcionalidades de la aplicación. Al profundizar en el asunto, descubrimos que la razón detrás de este problema era que la persona en cuestión no había activado la función de GPS dentro de la aplicación. Esto limitó su capacidad para acceder a ciertas características, lo que podría haber generado la impresión de que la aplicación no era tan intuitiva como se esperaba.

Noveno paso:

¿Usarías la aplicación para movilizarte con el transporte público?

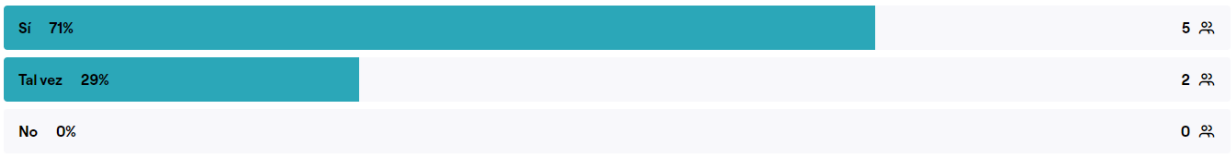

Ilustración 26 Respuesta noveno paso

La respuesta mayoritaria de los encuestados fue afirmativa acerca del uso de la aplicación para movilizarse con el transporte público de Quito. Sin embargo, hubo algunas personas que respondieron con un tal vez. Para poder entender mejor las razones detrás de esta respuesta, se les preguntó a estas personas el porqué de su indecisión. Al profundizar en sus respuestas, se pudo notar que para algunos

usuarios la inseguridad en Quito se vuelve cada vez más notoria, sobre todo en el transporte público, lo que les genera cierta preocupación. Es por eso que prefieren utilizar otros medios de transporte más seguros, y aunque la aplicación puede ayudarles a moverse de manera más eficiente, la preocupación por su seguridad sigue siendo una prioridad. Es importante tener en cuenta estos aspectos a la hora de desarrollar y mejorar la aplicación, para poder abordar y solucionar los problemas que preocupan a los usuarios y ofrecerles una experiencia de movilidad segura y satisfactoria.

En conclusión, las pruebas de usabilidad resultaron ser una herramienta valiosa para recopilar información y obtener retroalimentación sobre nuestro prototipo de aplicación web para el transporte público en Quito. A través de estos procesos, pudimos identificar tanto los puntos fuertes como los débiles de nuestra aplicación, y, en consecuencia, realizar mejoras significativas. Además, pudimos comprobar que la mayoría de los usuarios encontraron nuestra aplicación intuitiva y estarían dispuestos a utilizarla en un futuro cercano. Sin embargo, algunos participantes destacaron la inseguridad en el transporte público de Quito como un obstáculo para su uso. Por lo tanto, consideramos que estas pruebas de usabilidad fueron un paso importante en la creación de una aplicación web efectiva y accesible para los usuarios del transporte público en la ciudad.

5 Conclusiones

Después de llevar a cabo el proyecto, se pueden establecer las siguientes conclusiones:

En cuanto al objetivo de analizar y definir la arquitectura para su uso en aplicativos móviles y web, se logró diseñar una arquitectura eficiente y escalable que cumple con los requerimientos técnicos y de rendimiento necesarios para una aplicación moderna. La arquitectura fue diseñada con lenguaje TypeScript y se utilizaron servicios como la autenticación y los APIs de Google Maps.

En cuanto al objetivo de refactorizar la aplicación para cumplir con la arquitectura de software, se logró reducir el número de dependencias en un 60% y la lógica de negocio se redujo en un 59.62%. Esto permitió mejorar el rendimiento y la estabilidad de la aplicación, al mismo tiempo que se mantuvo su funcionalidad.

En cuanto al objetivo de construir un prototipo de las principales funcionalidades con la arquitectura diseñada, se logró desarrollar un prototipo funcional que se puede utilizar en navegadores web y en dispositivos móviles. La aplicación es responsiva y ofrece una experiencia de usuario satisfactoria.

En cuanto al objetivo de realizar pruebas de usabilidad del prototipo, se realizaron pruebas con un grupo de usuarios y se analizaron los resultados obtenidos. Las pruebas demostraron que el prototipo es funcional y fácil de usar, lo que sugiere que la arquitectura de software diseñada y aplicada en la aplicación puede ser una buena solución para aplicativos móviles y web.

Este proyecto presenta limitaciones en el despliegue de las aplicaciones móvil y web de Busity. La aplicación web solo pudo estar disponible de forma temporal durante 7 días, mientras que la aplicación móvil no pudo ser publicada en ninguna tienda de aplicaciones para Android y iOS.

Con los datos obtenidos de nuestros usuarios, podemos observar una tendencia en los resultados, pero esto solo representa una aproximación. Sería beneficioso

contar con un mayor número de usuarios de prueba para obtener datos más precisos y cercanos a la realidad.

En conclusión, el proyecto logró cumplir con los objetivos establecidos, permitiendo diseñar una arquitectura eficiente y escalable para aplicativos móviles y web, refactorizar la aplicación para cumplir con los lineamientos de la arquitectura de software, construir un prototipo funcional de las principales funcionalidades con la arquitectura diseñada y realizar pruebas de usabilidad del prototipo para demostrar su funcionalidad y facilidad de uso.

6 Recomendaciones

- Se recomienda en un futuro el despliegue de las aplicaciones de tal manera que tenga un dominio para la aplicación web y que esté publicada en las tiendas de apps de Android y iOS para la aplicación móvil.
- Se recomienda realizar pruebas de usabilidad con una mayor cantidad de usuarios para obtener resultados más representativos y datos más precisos.
- Se recomienda utilizar hardware para obtener la ubicación GPS de los autobuses, de manera que la aplicación no dependa de que los conductores de bus cuenten con un dispositivo móvil siempre conectado a internet para conocer su ubicación en tiempo real.

Referencias

Alibasa, Santos, Glozier, Harvey, & Calvo. (2018). Designing a secure architecture for m-health applications. *IEEE Xplore*, 91-94.

Ascueta, Bautista, Quilala, & Beninsig. (2021). BusTap: A Real-Time Bus Tracking Android Application. *IEEE Xplore*, 175-180.

Burbano, D., & Burbano, S. (2021). *Desarrollo de una aplicación móvil enfocada en la información de buses convencionales de quito "busity".*

Cervantes, H. (2019). *Arquitectura de Software.* Obtenido de https://sg.com.mx/revista/27/arquitectura-software

Firebase. (2022). *firebase.google.* Obtenido de https://firebase.google.com/

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). En *Design patterns: elements of reusable object-oriented software. Addison-Wesley Professional.*

Herazo, L. (2019). *anincubator*. Recuperado el 2022, de https://anincubator.com/que-es-una-aplicacion-movil/

Hidalgo, & Mena. (2022). Desarrollo y evaluación de una aplicación móvil para monitorear el transporte público en quito. *Repositorio Institucional de la universidad Politécnica salesiana*.

Holgado , A., Peñalvo, J., & Ingelmo, A. (2020). *Modelo c4.*

Ionic. (2022). *ionicframework.com*. Obtenido de https://ionicframework.com/

NeoAttack. (2020). *NeoAttack.* Obtenido de https://neoattack.com/neowiki/google-maps/

Richardson, L., & Amundsen, M. (2013). *Restful Web APIs.* O'Reilly Media, Inc.

Secretaría de movilidad. (2023). *Secretaría de Movilidad Quito Intracantonal Urbano*. Obtenido de https://secretariademovilidad.quito.gob.ec/index.php/intracantonal-urbano

Taylor, R. N., & Medvidovic, N. (2017). *Arquitectura de software en el mundo real.* Pearson Education.

vamos, Q. c. (2020). *Información sobre movilidad quito como vamos 2020*. Obtenido de https://quitocomovamos.org/wp-content/uploads/2021/05/7.MOVILIIDAD.pdf