ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN SISTEMA PROTOTIPO DE MOVILIDAD COLABORATIVA UTILIZANDO SERVICIOS WEB Y EL SISTEMA OPERATIVO ANDROID

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

ANDRÉS SANTIAGO BAUS LOGROÑO

andres.baus@epn.edu.ec

DIEGO FERNANDO LLERENA RENGEL

diego.llerena@epn.edu.ec

DIRECTOR: Ing. GABRIEL ROBERTO LÓPEZ FONSECA, MSc.

gabriel.lopez@epn.edu.ec

CODIRECTOR: Ing. FRANKLIN LEONEL SÁNCHEZ CATOTA, MSc.

franklin.sanchez@epn.edu.ec

Quito, abril 2019

AVAL

Certificamos que el presente trabajo fue desarrollado por Andrés Santiago Baus Logroño y Diego Fernando Llerena Rengel, bajo nuestra supervisión.

Ing. GABRIEL ROBERTO LÓPEZ FONSECA, MSc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

Ing. FRANKLIN LEONEL SÁNCHEZ CATOTA, MSc.
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Nosotros, Andrés Santiago Baus Logroño y Diego Fernando Llerena Rengel, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

ANDRÉS SANTIAGO BAUS LOGROÑO	DIEGO FERNANDO LLERENA RENGEL

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente Trabajo de Titulación tiene como objetivo principal desarrollar un sistema prototipo de movilidad colaborativa utilizando servicios web y el sistema operativo Android.

La arquitectura del sistema se divide en Servicios Web hospedados en un servidor en la nube y una Aplicación Móvil. Los Servicios Web se encargan de gestionar la capa de de datos y la lógica de todas las transacciones que se realicen desde la aplicación Móvil. Estos servicios se encuentran desarrollados en el lenguaje de programación JavaScript y hace uso del framework NodeJS. El servidor en la nube tiene publicados estos servicios web y la aplicación utiliza peticiones del tipo REST para consumir dichos servicios y transaccionar con la base de datos.

La aplicación móvil se encuentra desarrollada utilizando el lenguaje de programación JavaScript y el framework React Native para aplicaciones híbridas. El diseño de esta aplicación fue realizado creando sketches de las interfaces que cumplan con los requerimientos propuestos por el cliente, dentro de sus funcionalidades más importantes se encuentra la administración de usuarios, creación de rutas e intercambio de mensajes. La aplicación se puede conectar con el servicio web a través de peticiones en las que envía los datos recolectados a través de la interfaz y recibe mensajes que permiten notificar al usuario del estado de la aplicación.

Capítulo 1: Fundamento teórico del proyecto.

Capítulo 2: Detalle de la metodología empleada para el desarrollo del proyecto y diseño del prototipo.

Capítulo 3: Detalle de la implementación del prototipo.

Capítulo 4: Pruebas realizadas, resultados obtenidos y conclusiones luego de concluido el proyecto.

PALABRAS CLAVE: Servicio Web, Servidor, Rest, Aplicaciones Hibridas, JavaScript, React Native.

ABSTRACT

The main objective of this degree work is to develop a collaborative mobility prototype system using web services and the Android operating system.

The architecture of the system is divided into Web Services hosted on a server in the cloud and a Mobile Application. Web Services are responsible for managing the database layer and the logic of all transactions made from the Mobile application. These services are developed in the JavaScript programming language and makes use of the NodeJS framework. The server in the cloud has published these web services and the application uses requests of the REST type to consume those services and to transact with the database.

The mobile application is developed using the JavaScript programming language and the React Native framework for hybrid applications. The design of this application was made by creating sketches of the interfaces that meet the requirements proposed by the client, within its most important functionalities are the administration of users, creation of routes and exchange of messages. The application can connect to the web service through requests in which it sends the collected data through the interface and receives messages that allow the user to be notified of the status of the application.

Chapter 1: Theoretical basis of the project.

Chapter 2: Detail of the methodology used for the development of the project and prototype design.

Chapter 3: Detail of the prototype implementation.

Chapter 4: Tests carried out, results obtained and conclusions after the conclusion of the project.

KEYWORDS: Web Service, Server, Rest, Hybrid Applications, JavaScript, React Native

1. INTRODUCCIÓN

El continuo y agresivo crecimiento del parque automotor en la ciudad de Quito perjudica notablemente la movilidad de los ciudadanos; en vista de eso se han realizado y planteado varias soluciones viales como paliativos para tratar de aliviar la congestión vehicular.

Dentro de las medidas adoptadas se encuentra la implementación de contraflujos en horas pico [1] y el programa de Pico y Placa lanzado en el año 2010.

Para el 2017, luego de analizar la ciudad por un periodo de 6 meses, la Universidad Internacional SEK publica un análisis de la aplicación del Pico y Placa en la ciudad de Quito [2], de donde se extrae lo siguiente: "Debido al veloz incremento del parque automotor en Quito, la medida solo fue efectiva durante los primeros 3 años desde su implementación, pero luego de ello esta medida está siendo obsoleta, por lo que se sugiere buscar otras soluciones como fomentar el uso de transporte público".

La movilidad compartida se muestra como una alternativa para reducir la cantidad de vehículos que circulan en una ciudad; esta se basa en compartir un vehículo con colegas o personas de un círculo cercano que compartan un mismo destino. Según el estudio [3] sobre la ciudad de Manhattan indica que si los conductores recogieran uno o más pasajeros en su ruta se incrementaría la eficiencia del transporte en un 40%.

En Quito no se cuenta con programas que busquen el compartir vehículos, y de esta manera brindar una alternativa que facilite la movilidad de los ciudadanos. Aprovechando las facilidades tecnológicas se puede desarrollar un sistema compatible con la tecnología móvil, y en base a esto generar una herramienta que brinde información y fomente la integración de diversos actores en esta iniciativa.

En este proyecto se plantea la creación de una aplicación de carpooling[1], la cual tiene como objetivo, en primera instancia, convertirse en una alternativa para la movilidad, y con esto mejorar la calidad de transportación de las personas. El prototipo de este programa será aplicado en la Escuela Politécnica Nacional.

[1] Carpooling: Es la compartición de viajes en un automóvil para que más de una persona viaje en el mismo y evitar la necesidad de que otros tengan que conducir al mismo lugar en varios vehículos [3].

1.1 Objetivos

El objetivo general de este Proyecto es desarrollar un sistema prototipo de movilidad colaborativa utilizando servicios web y el sistema operativo Android.

Los objetivos específicos son:

- Analizar la fundamentación teórica necesaria para el desarrollo del proyecto.
- Diseñar los módulos que conforman el prototipo.
- Implementar todos los módulos diseñados para el prototipo.
- Analizar los resultados obtenidos en las pruebas de funcionamiento del prototipo.

1.2 Alcance

El presente proyecto de titulación plantea el diseño e implementación de un sistema prototipo de movilidad utilizando un servicio web, una base de datos y una aplicación cliente sobre Android.

Los usuarios al usar por primera vez el sistema deberán crear una cuenta con una dirección de correo electrónico a fin de registrarse en el sistema.

En base a lo descrito anteriormente se definen los siguientes roles para los usuarios del sistema:

- **Conductor:** Es el usuario que registra su vehículo particular, establece una ruta fija y un horario a través de la aplicación móvil. Al ofertar este servicio podrán obtener bonificaciones de acuerdo con su participación.
- **Pasajero:** Es el usuario que utiliza la aplicación cliente para registrarse en el servicio, buscar una ruta en un horario establecido y suscribirse a esta.
- **Administrador:** Es el usuario tiene facultades para editar y eliminar usuarios o su información.

El sistema permitirá a los conductores registrar un vehículo particular, establecer una ruta fija, la cual puede ser aprovechada por los pasajeros.

El servicio será implementado en un servidor en la nube y contará con al menos los siguientes módulos:

- **Administración de Usuarios:** Este módulo contará la capacidad para crear, leer, editar y eliminar a los usuarios del sistema.
- **Autenticación:** Este módulo permitirá autenticar a los usuarios de la aplicación móvil con el sistema.
- **Gestión de Rutas:** Este módulo habilita la publicación de rutas fijas y horarios. Una ruta fija es una descripción de un camino a seguir entre una dirección origen y un destino a definir por los conductores. Una ruta fija contará con horarios, los cuales son una descripción de la fecha de inicio de la ruta, fecha de finalización de la ruta y el horario de inicio del recorrido. Todos estos indicadores permiten a los usuarios determinar si la ruta es conveniente para ellos y que puedan suscribirse a las mismas.
- **Buzón de mensajes:** Este módulo permitirá el intercambio de mensajes entre los pasajeros que se encuentren suscritos a una ruta con el conductor que publicó dicha ruta.

La aplicación cliente estará desarrollada sobre Android y al menos contará con los mismos módulos implementados en el servidor web.

En la figura 1.1 se puede ver el esquema del proyecto donde se representan los componentes principales de la arquitectura de manera general.

Figura 1.1. Esquema del proyecto.

El sistema prototipo se diseñará utilizando la metodología ágil de diseño de software SCRUM. Además, la aplicación cliente se implementará usando el framework React Native, que permite generar aplicaciones nativas. El servicio web será desarrollado sobre Node.js utilizando el framework Express con una base de datos MySQL, ambos se encontrarán funcionando sobre infraestructura en la nube, utilizando AWS.

El presente proyecto de titulación tendrá un producto final demostrable, mismo que consta de la Aplicación móvil instalable en dispositivos Android v5 o superior, además de los archivos de configuración del Servicio Web.

1.3 Marco Teórico

A continuación, se definirán los conceptos claves de bases de datos, como lenguajes de consulta, modelos relacionales. En el campo de los servicios web se tratarán conceptos como la arquitectura de capas, servicios en la nube, APIs, para luego pasar a definir conceptos fundamentales de aplicaciones móviles como la arquitectura del sistema Android, aplicaciones híbridas y versiones del sistema operativo.

También se detallarán conceptos respecto a la metodología SCRUM que será usada para el desarrollo del prototipo y finalmente se detallará las herramientas de desarrollo utilizadas.

1.3.1 Bases de Datos

Se denomina base de datos a un conjunto de información o datos que se encuentran organizados específicamente de tal forma que facilitan el almacenamiento, recuperación, modificación y borrado de los datos mediante operaciones de procesamiento de datos [4].

Para realizar las operaciones de procesamiento de datos, se requiere de un Sistema de Administración de Base de Datos (DBMS), el cual extrae la información de la base como respuesta a consultas sobre la misma. Las consultas se realizan con los denominados lenguajes de consulta (DDL). SQL [5] es uno de los más difundidos, y dentro de los DBMS que utilizan SQL para gestionar la base datos, una de las opciones gratuitas más usadas es MYSQL [6].

A. SQL

Es un lenguaje de consulta de base de datos, que se caracteriza por ser estructurado. Su principio de funcionamiento se basa en sentencias, las cuales mediante comandos que se asemejan a oraciones escritas en inglés se puede insertar, leer o modificar datos en una base de datos [5].

El lenguaje ha sido definido y estandarizado por la ANSI/ISO [7] desde el año 1986 donde se han emitido varias actualizaciones al estándar siendo la versión SQL:2003 [8] la versión actual del standard.

B. MySQL

Es un Sistema de Administración de Base de Datos desarrollado y distribuido por Oracle Corporation [9].

MySQL permite el acceso y procesamiento de la información almacenada en una base de datos gracias a su motor MySQL Server. Este sistema crea modelos de bases de datos relacionales, trabajando sobre entornos cliente/servidor o sistemas embebidos; además es un producto Open Source [10], lo cual ha permitido que el producto sea muy difundido en los sistemas de base de datos.

MySQL como parte de una solución integral ha construido una herramienta que permite la administración del DBMS. Esta herramienta se denomina MySQL Workbench y está pensada para proveer una solución integral de administración mediante una interfaz gráfica. Permite la configuración del servidor, administración del usuario, copias de seguridad, creación del modelo relacional, entre otras [6].

C. Modelo Relacional

El Modelo Relacional es un concepto que se maneja en el diseño de bases de datos, gracias a este se puede obtener una abstracción gráfica de la estructura de la base de datos que se está diseñando, conteniendo la información de las tablas y las relaciones entre datos [11].

En el modelo relacional se tienen los siguientes componentes:

A. **Tablas:** Estas representan las entidades del modelo a través de una matriz bidimensional de columnas y filas. Las columnas representan los atributos y las filas los objetos del mundo real almacenados en la base. Cada tabla tiene un atributo principal denominado clave primaria.

B. **Claves:** Existen dos tipos de claves:

- **Clave Primaria:** Nace de la necesidad de tener un identificador único que sea no ambiguo. Usualmente se escoge como clave primaria el atributo de menor tamaño.
- **Clave Foránea:** Un atributo de una tabla que proviene de la relación con otra tabla. Una clave foránea es una clave primaria en su tabla de origen.

C. **Restricciones:** En un modelo relacional existen 3 tipos de restricciones:

- **Integridad de la clave:** La clave primaria no puede tomar valores nulos.
- **Integridad Referencial:** Una clave foránea no puede tomar un valor nulo. Y se debe actualizar conforme la información de la tabla de origen.
- **Restricciones Varias:** Aquí se aplican restricciones a los demás atributos de la tabla, como solicitar que se puedan usar ingresar valores nulos, definir máximos y mínimos, definir el tipo de dato a guardar en un atributo, etc.

1.3.2 Arquitectura en Capas

En programación se considera un estilo de desarrollo en el cual el enfoque principal es la separación de la lógica del sistema. Generalmente esta separación divide al sistema en una capa de datos, capa de negocio y capa de presentación [12].

A. Capa de Datos

En esta capa se encuentran los datos almacenados en una base. Esta base puede ser estructurada o no estructurada, y puede estar formada por uno o varios gestores de bases de datos. Esta capa interactúa directamente con la capa de negocio [13].

B. Capa de Negocio

Esta capa es la encargada de recibir las peticiones de los usuario, ejecuta los procesos necesarios y envía las respuestas de vuelta tras el procesamiento realizado. Aquí se establecen las reglas para la recepción y entrega de datos, así como toda la lógica de ejecución de los programas.

C. Capa de Presentación

Es la capa con la que el usuario interactúa directamente, presenta el sistema al usuario, captura la información de este y le muestra los resultados de sus transacciones.

En el presente trabajo de titulación la capa de Datos y la Capa de Negocio físicamente se encuentran en un servidor en la nube, de aquí en adelante llamado “Servidor”. Este servidor ha publicado Servicios Web que son accesibles por la Capa de Presentación, llamada de aquí en adelante también “Aplicación Móvil”. La aplicación móvil es la que se encarga de interactuar con los requerimientos del usuario para enviarlos al servidor y que se pueda obtener la información o los resultados de las operaciones realizadas.

1.3.3 Servicios en la Nube

Los servicios en la nube se constituyen como la entrega bajo demanda de la capacidad informática para almacenamiento, procesamiento u otros recursos a través de una plataforma de servicios en la nube.

Dentro de los servicios en la nube se pueden identificar 3 modelos de servicio principales:

A. **IaaS (Infraestructura como Servicio):** Es un modelo de servicio que se encarga de entregar almacenamiento y procesamiento de computo. Los servidores que se utilizan en este esquema se encuentran concentrados y generalmente son virtualizados dentro de un clúster de ordenadores físicos.

   Gracias a estas características pueden manejar diferentes tipos de cargas de trabajo bajo demanda, donde se puede dinámicamente aumentar la capacidad de dichos servidores de acuerdo con la cantidad de procesamiento o transacciones que reciben.

   En los ejemplos más conocidos de empresas que ofrecen este servicio se encuentran Amazon Web Services (AWS) y Microsoft Azure [14].

B. **PaaS (Plataforma como Servicio):** Este modelo de servicio permite el desarrollo, manejo y entrega de aplicaciones. Los usuarios utilizan una suite de herramientas previamente desarrolladas y cargadas en la plataforma. Provee un entorno de desarrollo y pruebas para estas aplicaciones [14].

C. **SaaS (Software como Servicio):** En este modelo se tiene acceso a un software específico hospedado en un servidor en la nube. Los usuarios finales pueden acceder a este software a través de la web o de un API. La instalación y actualización del software la maneja directamente el proveedor [14].

En el presente trabajo de titulación se ha utilizado los servicios de Amazon Web Services, bajo el esquema IaaS para publicar los servicios web que utiliza la Aplicación Móvil.

1.3.4 Servicios Web

El término Servicios Web hace referencia a un software que permite la comunicación directa entre dos máquinas usando una función que se encuentra hospedada en una ubicación direccionable en la red [15].

Un servicio web esconde todos los detalles de implementación a través de una interfaz, la cual permite que el servicio pueda ser usado independientemente del hardware, software o el lenguaje de codificación de este.

La interfaz de los Servicios Web esta descrita en un formato procesable a nivel de maquina llamado Web Service Definition Language (WSDL) [16]. Los Web Services usados en la actualidad pueden ser del tipo SOAP o del tipo REST.

Un Web Service del tipo SOAP esta descrito usando notación estándar XML[2], posee la ventaja de ser extensible y puede ser transportado por una variedad de protocolos [17].

A. REST Web Services

REST[3] es un estilo de arquitectura que define requisitos específicos para el diseño de servicios web, tales como una interfaz uniforme, escalabilidad y posibilidad de modificación.

Un Web Service del tipo REST transporta la información en objetos del tipo JSON[4], los cuales tienen su esquema de notación definido y requiere el uso del protocolo HTTP para su transporte. JSON actualmente es un formato de representación de datos usando ampliamente en aplicaciones móviles y aplicaciones basadas en JavaScript[5] [18].

Objetos JSON

JSON está constituido por dos estructuras:

- **Pares de Nombre y Valor:** Conocidos también como objetos, tienen una estructura tipo diccionario que permite asociar el nombre de la variable con su valor.
- **Lista ordenada de Valores:** Tal como lo implementan otros lenguajes de programación esto es implementado como listas, arreglos o secuencias.

Un objeto JSON es un conjunto desordenado de pares nombre/valor. Empieza con "{" (llave de apertura) y termina con "}" (llave de cierre). Cada nombre denota su valor usando ":" (dos puntos) y los pares nombre/valor se encuentran separados por "," (coma) [18].

---

[2] XML (Extensible Markup Language): Es un lenguaje de marcado que permite el transporte y almacenamiento de datos. [44]
[3] REST: REpresentational State Transfer o Transferencia de estado representacional.
[4] JSON (JavaScript Object Notation): Es un formato de intercambio de datos considerado como ligero, que permite una fácil interacción para ser leído y escrito por Humanos, además de ser fácil de procesar y generar por una máquina. [18]
[5] JavaScript: Es un lenguaje de programación que permite crear efectos interactivos entre buscadores web. [45]

En la figura 1.2 se puede observar un ejemplo de un paquete JSON, el cual contiene mencionados objetos del tipo nombre/valor.

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

Figura 1.2. Ejemplo de notación de un objeto JSON. [18]

El conjunto de estos servicios web que permiten acceder a la capa de lógica del negocio conocida como servidor, se denomina API[6].

1.3.5 Sistema Operativo Android

Android, es un stack de tecnologías de software basadas sobre un Kernel de Linux[7] [19]. Este stack ha sido creado para soportar una gran cantidad de dispositivos, además de distribuirse como open source. A continuación, se describirán los elementos:

A. Kernel Linux

La base de la plataforma Android es el kernel de Linux, es la capa más baja representando al núcleo del sistema operativo y está construida sobre el kernel de Linux con ciertos cambios arquitectónicos [20]. Dentro de las características que proporciona se encuentra la administración de memoria, administración de seguridad, pila de red, administración de procesos y administración de dispositivos.

[6] API: Conjunto de reglas (código/funciones) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas.
[7] Kernel de Linux: Es la parte básica de cualquier sistema operativo, en el caso de Linux el kernel contiene funciones esenciales como la gestión de memoria y de procesos, conexión con dispositivos de entrada y salida entre otros [48].

En sí el kernel representa a una lista de controladores de dispositivo que facilitan la comunicación de un dispositivo Android con otros dispositivos periféricos. Un controlador de dispositivo es un software que proporciona una interfaz de software para los dispositivos de hardware. Al hacerlo, el sistema operativo y otros programas pueden acceder a estos dispositivos de hardware.

B. Capa de Abstracción del Hardware (HAL)

Esta capa proporciona interfaces de tipo estándar que exponen las capacidades de hardware del dispositivo al marco de API de Java de nivel superior [19].

La HAL consta de varios módulos, cada uno de los cuales implementa una interfaz para un tipo específico de componente de hardware, como la cámara o el bluetooth. Cuando una API del marco realiza una llamada para acceder al hardware del dispositivo, el sistema Android carga el módulo de biblioteca para ese componente de hardware.

C. Android Runtime y librerías nativas de C/C++

La capa está compuesta principalmente por las librerías nativas principales y la máquina virtual Dalvik [20]. Las librerías nativas de C/C++ son responsables de proporcionar soporte para las funciones principales como el maneja de gráficos en 2D y en 3D o la implementación de código nativo para interactuar con la HAL.

La máquina virtual de Dalvik o también llamada máquina virtual de Java (JVM), es una máquina virtual basada en registros que proporciona las optimizaciones necesarias para ejecutar en entornos de poca memoria.

La máquina virtual de Dalvik convierte los códigos de bytes (archivos de clase Java) que tienen extensiones *.class* generadas por el compilador de Java en Dalvik: archivos ejecutables que tienen extensiones "*.dex"*. Este tipo de binarios están optimizados para ejecutarse en procesadores más pequeños y en entornos de poca memoria.

D. Java API Framework

El conjunto completo de características del sistema operativo Android está disponible a través de API escritas en el lenguaje Java [19].

Estas API forman los componentes básicos que necesita para crear aplicaciones de Android al simplificar la reutilización de los componentes y servicios centrales del sistema modular, dentro de las características se encuentra a:

- Un sistema de visualización enriquecido y extensible que puede utilizar para crear la interfaz de usuario de una aplicación, incluidas listas, cuadrículas, cuadros de texto, botones e incluso un navegador web que se puede insertar.
- Un administrador de recursos, que proporciona acceso a recursos no codificados, como cadenas localizadas, gráficos y archivos de diseño.
- Un administrador de notificaciones que permite que todas las aplicaciones muestren alertas personalizadas en la barra de estado.
- Un administrador de actividades que administra el ciclo de vida de las aplicaciones y proporciona una pila de navegación trasera de navegación común.
- Proveedores de contenido que permiten que las aplicaciones accedan a datos de otras aplicaciones, como la aplicación Contactos, o que compartan sus propios datos.

E. Aplicaciones del Sistema

Android viene con un conjunto de aplicaciones base como: cámara, teclado, SMS, entre otros [19]. Estas permiten utilizar funciones básicas a través del sistema y a otras aplicaciones para que no se requiera implementar este tipo de funcionalidad entre las demás aplicaciones.

En la figura 1.3 se puede ver una representación gráfica del stack de Software Android, en el que se engloban todos los módulos listados anteriormente.

1.3.6 Framework de desarrollo de aplicaciones híbridas React Native

React Native es un framework de JavaScript para escribir aplicaciones móviles nativas y del tipo híbrido, es decir que el código genera aplicaciones tanto para iOS como para Android [21].

Está basada en React, la librería de JavaScript de Facebook utilizada para crear interfaces de usuario que serán cargadas en un navegador web, en el caso de React Native está dirigido a plataformas móviles. La principal ventaja de este enfoque es que los desarrolladores web pueden escribir aplicaciones móviles nativas, a partir del uso de JavaScript reutilizando el código para iOS y Android.

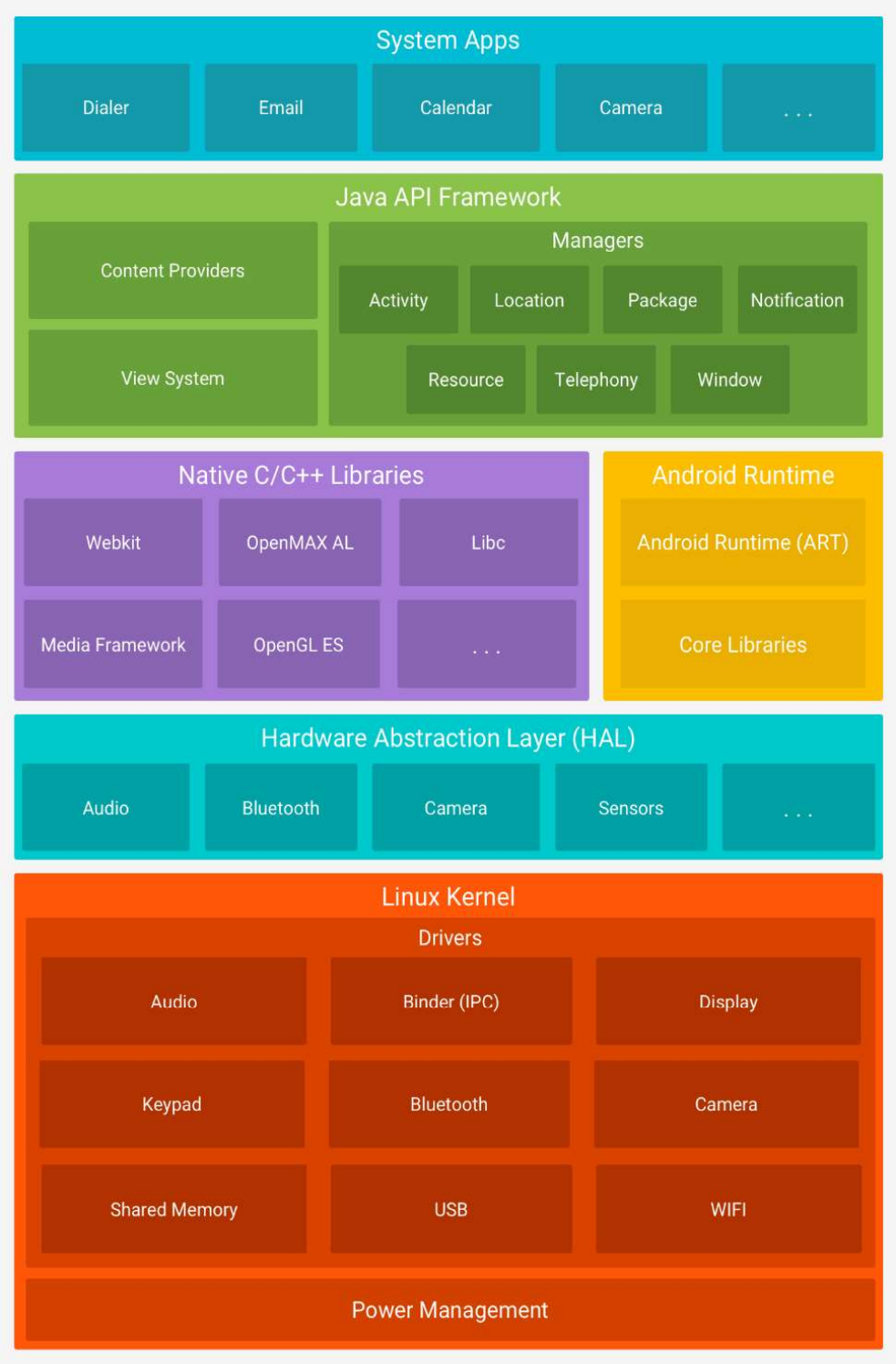


Figura 1.3. Stack de Software Android [19]

Arquitectura del Framework React Native

El framework React Native al utilizar JavaScript como su lenguaje de programación base, no tiene directamente un método para compilar el código sobre las plataformas móviles ya sea iOS o Android. Es por esto que React Native aprovecha la existencia de compiladores de código para cada una de las plataformas; es decir que existe un compilador de código para Java o Kotlin que generan aplicaciones instalables en Android y de la misma manera existe otro compilador para Objective-C o Switft que generan aplicaciones instalables en iOS [22].

React Native se aprovecha de esto y crea un puente hacia estos compiladores y dependiendo de la plataforma se puede crear la aplicación de iOS o Android. La figura 1.4 muestra la arquitectura de React Native y el paso de un código JavaScript hacia artefactos multiplataforma.

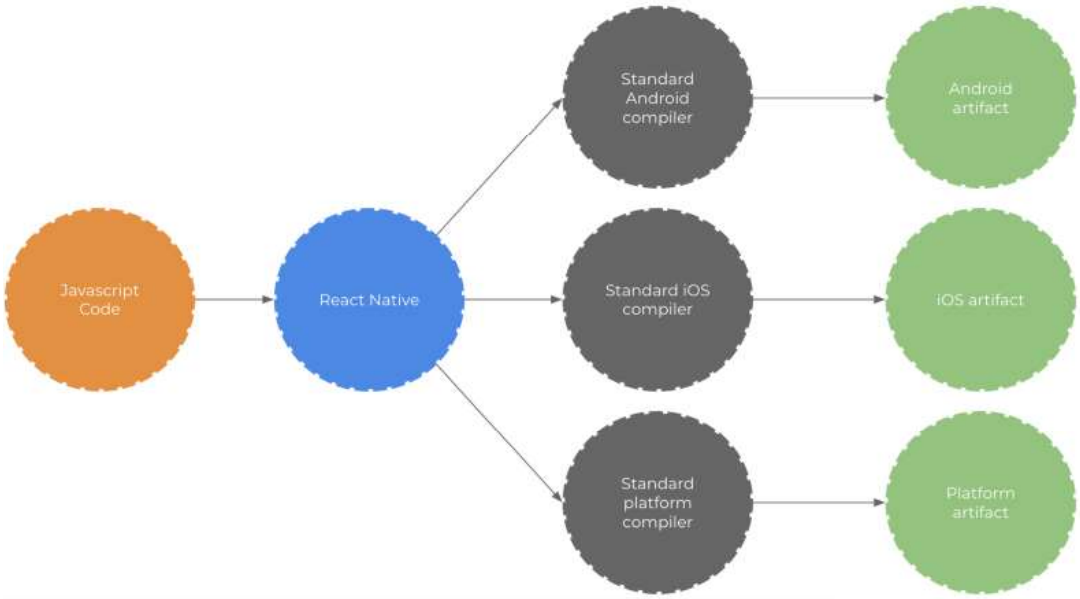


Figura 1.4. Arquitectura de React Native [22]

El puente permite a React Native interactuar con las librerías nativas C/C++ tanto de iOS como de Android a través de código Java o C++, de esta manera todo el código de JavaScript se mantiene ejecutando dentro del JavaScript Core, equivalente a JVM en Java.

En la figura 1.5 se muestra la arquitectura que permite la interconexión de los módulos Nativos de cada sistema operativo y el core de JavaScript.

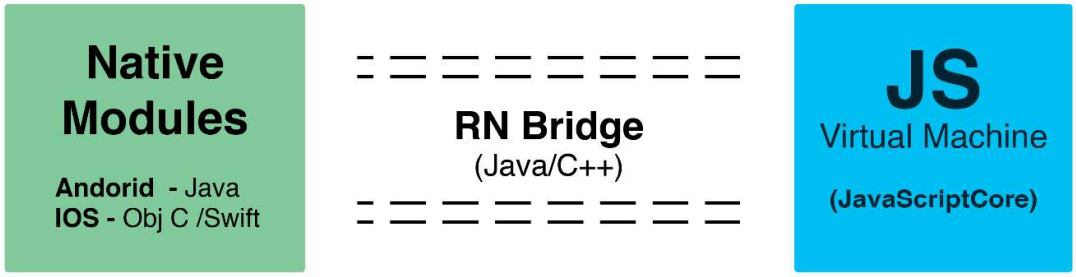


Figura 1.5. Flujo entre el JavaScript Core y las librerías nativas C/C++ a través del puente de React Native.

Versiones de React Native en relación con Android

Las versiones de los sistemas operativos soportadas por React Native son [23]:

- Android 4.1 o mayor (API 16)
- iOS 9.0 o mayor

El desarrollo de las aplicaciones puede ser realizado en Windows, GNU/Linux o macOS, pero la creación y depuración de las aplicaciones en iOS está limitado a macOS.

1.3.7 Metodología de desarrollo ágil SCRUM

Scrum es una metodología ágil del tipo iterativo o incremental utilizado en proyectos, productos o desarrollo de software [24]. Se ha utilizado para gestionar el desarrollo de productos complejos desde principios de los años noventa [25]. Scrum no es un proceso o una técnica para construir productos; más bien, es un conjunto de técnicas y procesos.

Dentro de Scrum se encuentra especificado distintos componentes que cumplen con un propósito específico, dentro de ellos se tiene a los grupos de trabajo con sus miembros y sus roles, eventos, artefactos y reglas asociadas.

Scrum se basa en procesos empíricos, el empirismo afirma que el conocimiento proviene de la experiencia y la toma de decisiones basadas en lo que se conoce. Scrum emplea un enfoque iterativo e incremental para optimizar la previsibilidad y controlar el riesgo.

Estos son los componentes y eventos que forman parte de la metodología:

A. Scrum Team

El Scrum Team está compuesto por los siguientes roles: Product Owner, Equipo de Desarrollo y el Scrum Máster [25].

Este equipo tiene la característica de que puede auto organizarse y sus miembros pueden llegar a tener funciones mixtas; además, están diseñados para tener flexibilidad, creatividad y productividad.

Scrum Teams entregan productos continuamente lo que les permite obtener retroalimentación continua y oportuna, de esta manera se puede asegurar que exista una versión del producto disponible.

B. Product Owner

Es el responsable de maximizar el valor del producto y el trabajo del equipo de desarrollo [25]. Posee las siguientes responsabilidades:

- Expresar claramente las historias de usuario o PBIs (Product Backlog Items).
- Ordenan los PBIs para alcanzar metas y objetivos.
- Optimizan el trabajo realizado por el equipo de desarrollo.
- Se aseguran de que los PBIs sean visibles y claros; que demuestren en lo que el Scrum Team va a trabajar.

El Product Owner no es un comité o un grupo, es una sola persona que representa los deseos de los verdaderos dueños del producto. El equipo de desarrollo solo podrá trabajar a partir de las especificaciones realizadas por los PBIs definidos por el Product Owner.

C. Scrum Master

Es la persona encargada de que todo el proceso de Scrum sea entendido y divulgado entre los miembros del Scrum Team, es decir se asegura de que los miembros conozcan la teoría, prácticas y reglas.

Además, cumple la función de líder de servicio [24], es decir que ayuda a cada uno de los miembros del equipo a maximizar el valor creado por su trabajo en cada una de las iteraciones.

Dentro de las responsabilidades del Scrum Master se encuentra que:

- Utilizar técnicas efectivas para el manejo del Product Backlog.
- Ayudar a entender los PBIs a los miembros del equipo.
- Entender la planeación del proyecto desde un punto de vista empírico.
- Asegurarse que el Product Owner sabe cómo maximizar el valor que ofrecen los PBIs.
- Facilitar los eventos de Scrum como sean necesitados.
- Entrenar y organizar al equipo de desarrollo.
- Ayudar al equipo de desarrollo a crear productos de alto valor.
- Encargarse de los impedimentos del equipo de desarrollo.

D. Equipo de Desarrollo

El equipo de desarrollo consiste en un grupo de profesionales que realizan el trabajo para generar un producto terminado en cada iteración de Scrum [25]. Solo los miembros del equipo de desarrollo crean estos incrementos sobre el producto. El equipo de desarrollo debe de estar organizado para generar eficiencia y eficacia sobre el producto.

El equipo de desarrollo tiene las siguientes características:

- Sus miembros se auto organizan; es decir que nadie, incluso el Scrum Master, decide cómo tratar los PBIs para generar progresos incrementales.
- Tienen las capacidades necesarias para generar incrementos del producto en cada iteración.
- Miembros individuales del equipo pueden llegar a tener especialidades o áreas de enfoque.

E. Sprint

Es la parte principal y esencial de Scrum, es delimitado por una medida de tiempo que puede ser un mes o menos en el que una iteración del producto llega al estado de terminado [25], una vez que un Sprint termina inicia inmediatamente el siguiente.

Cada Sprint debe de ser considerado como un proyecto pequeño con un horizonte de tiempo definido, dentro de los artefactos usados dentro del Sprint se encuentra a:

- **Sprint Planning Meeting:** Es el momento en el que se define las metas y objetivos para el Sprint, es decir los PBIs que permitirán obtener un producto definido.
- **Daily Scrums:** Son reuniones diarias donde se revisa el trabajo realizado en el día anterior, se planifican las tareas del día para los miembros del equipo y se tratan dificultades o impedimentos.
- **Sprint Review y Restrospective:** Son reuniones entre los miembros del equipo donde se realiza una retroalimentación de lo sucedido durante el Sprint. Además, se obtienen notas y definen actividades que permitan mejorar el siguiente Sprint.

1.3.8 Herramientas de Desarrollo

A. Visual Studio Code

Es un editor de texto con desarrollado por Microsoft para los sistemas operativos Windows, GNU/Linux y macOS [26]. Dentro de las funcionalidades se encuentra la capacidad para realizar debugging, control de versiones Git, resaltado de sintaxis, intellisense, plantillas, refactorización de código, entro otros.

Se encuentra desarrollado utilizando Electron, librería de JavaScript, además de incluir un modelo extensivo público que permite a los desarrolladores crear extensiones para la herramienta que permitan aumentar su funcionalidad.

En la figura 1.6 se puede ver la interfaz de Visual Studio Code, donde una de sus características es la identificación de funciones y variables con diferentes colores de texto.

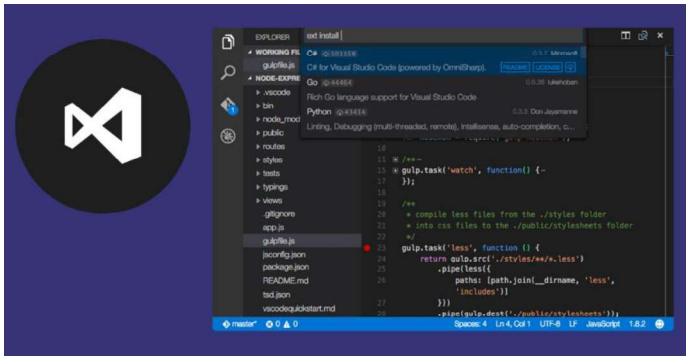


Figura 1.6. Logo de Visual Studio Code e imagen del editor [26]

B. Genymotion

Es una herramienta que permite la creación de dispositivos virtuales que funcionen con el sistema operativo Android, de esta manera se puede acelerar el proceso de desarrollo y automatizar las pruebas [27]. Dentro de las características que permite están:

- Administración de la batería del dispositivo virtual.
- Uso de GPS del dispositivo virtual para probar ubicaciones.
- Emulación de la cámara del dispositivo a través de dispositivos externos.
- Manejo de interfaces de red como wifi o radio del dispositivo virtual.
- Emulación de disco duro y memoria RAM.

C. Expo

Es un grupo de herramientas open source que facilitan el desarrollo de aplicaciones con el framework React Native [28]. Provee de un SDK como una librería nativa y de JavaScript que provee acceso a las funcionalidades de sistema del dispositivo como:

- Cámara
- Acelerometro
- Localización, entre otros.

De esta manera no es necesario utilizar las herramientas como XCode y Android Studio para poder probar y hacer debugging de la aplicación en desarrollo.

D. Gravit Designer

Es una aplicación enfocada en el diseño gráfico [29], esta aplicación permite importar bocetos, modelos para diseño, transformaciones, entre otras. Contiene herramientas que facilitan la creación de interfaces o experiencia de usuario. Además, permite exportar archivos PDF, IVS, creando imágenes de alta calidad.

Este programa es gratuito y multiplataforma, por lo que es compatible con Windows, Mac OS, Linux y tiene soporte para Chrome OS, una característica adicional de esta aplicación es que cuenta con una versión web compatible con los navegadores Chrome, Firefox, Internet Explorer, entre otros.

2. METODOLOGÍA

En este capítulo se definirán los requerimientos que debe cumplir el prototipo, los cuales establecerán el comportamiento del sistema. Además, se realizará una descripción de la metodología de desarrollo ágil SCRUM que consiste en desarrollo iterativo e incremental, reduciendo de esta manera procesos y documentación innecesaria.

Como punto de partida se procederá a definir un backlog a través del cual se determinarán los roles de los miembros del proyecto. A continuación, se implementarán las historias de usuario, la cuales dentro de la metodología Scrum toman el nombre de PBI (Product Backlog Items). Estos PBI serán trabajados dentro de ciclos de codificación conocidos como Sprint.

A partir de los PBI se procederá con el diseño de la base de datos empleando un modelo relacional. Posteriormente, se continuará con diseño del servicio web implementado a través de una API REST. Finalmente, se realizará el diseño de la aplicación móvil, la cual funcionará sobre el sistema operativo Android.

2.1. Definición del Backlog

La metodología SCRUM [25] menciona que, es necesario definir los requerimientos en un proyecto.

El mecanismo utilizado es el backlog [24] que se encuentra definido en la tabla 2.2, y en él residirán todas las peticiones que el cliente desea del software a diseñarse. Previo a ello, es necesario definir los roles de las personas que intervendrán en el proyecto, para que sean ellas las encargadas de definir los requerimientos que formarán parte del backlog.

2.2. Roles de Scrum

- **Product Owner:** Msc. Gabriel López, Docente del DETRI EPN.
- **Scrum Master:** Msc. Franklin Sánchez, Docente del DETRI EPN.
- **Equipo de Desarrollo:** Diego Llerena

  Andrés Baus

2.3. Valoración de Requerimientos

Es necesario definir la forma en la que se va a dar equivalencia a las prioridades y nivel de complejidad de los PBI, en la tabla 2.1 se encuentran definidas las equivalencias.

Una vez realizado esto se procede a definir el backlog con los requerimientos del proyecto en términos generales, los elementos de la tabla fueron definidos considerando las características requeridas para completar la funcionalidad descrita por el Product Owner y los criterios de complejidad y prioridad fueron definidos por el Scrum Master. La tabla 2.2 muestra el backlog con su identificador, prioridad y complejidad, estos valores fueron establecidos en base a la experiencia del equipo de desarrollo y el Scrum Master.

Tabla 2.1. Valoración de Requerimientos

Número	Complejidad	Prioridad
1	Fácil	Baja
2	Media	Media
3	Difícil	Alta

2.4. Definición de número de Sprints para el desarrollo del proyecto

Para este proyecto la información recolectada se definió en los PBI a través de las historias de usuario. Este trabajo se realizó en conjunto con el Product Owner, el Scrum Master y el equipo de desarrollo.

El detalle del número de Sprints del proyecto y su respectiva descripción se encuentra descrito en la tabla 2.3.

En cada una de los PBI de cada Sprint se definirá un identificador único, nombre, miembro del equipo asignado, sprint al que pertenece, prioridad de negocio, complejidad, descripción y validación.

Tabla 2.2. Backlog

Identificador	Descripción	Prioridad	Complejidad
A01	Acceso a través de Aplicación móvil Android	3	3
A02	Usuarios con un rol de conductores	2	2
A03	Usuarios con un rol de pasajeros	2	2
A04	Usuarios con un rol de administración	2	2
A05	Rutas de usuario fijas	2	2
A06	Buzón de mensajes para los usuarios	2	2
A07	Registro de usuarios	2	1
B01	Infraestructura de servicios en la nube	3	1
D01	Diseño de diagrama de secuencia	3	2
D02	Selección del proveedor IaaS	1	3
B02	Diseño de la base de datos	3	3
B03	Diseño del API REST	3	3
B04	Diseño de la aplicación móvil	3	3
B05	Codificación de la Base de Datos	3	1
B06	Codificación del módulo de autenticación en el servicio web	2	2
B07	Codificación del módulo de autenticación en la aplicación móvil	2	2
B08	Codificación del módulo de administración de usuarios en el servicio web	2	2
B09	Codificación del módulo de administración de usuarios en la aplicación móvil	2	2
B10	Codificación del módulo de gestión de rutas de usuarios en el servicio web	2	2
B11	Codificación del módulo de gestión de rutas de usuarios en la aplicación móvil	2	2
B12	Codificación del módulo de buzón de mensajes de usuarios en el servicio web	2	2
B13	Codificación del módulo de buzón de mensajes de usuarios en la aplicación móvil	2	2
B14	Codificación de las interfaces en la aplicación móvil y conexión con los módulos.	3	3
B15	Prototipo con información en BDD para pruebas	2	1
B16	Despliegue de infraestructura en la nube	2	1

Tabla 2.3. Detalle de Sprints

Detalle de Sprints	
Identificador	**Descripción**
Sprint 01	Definición de Requerimientos del Product Owner.
Sprint 02	Definición de Requerimientos del Proveedor y Diseño de secuencia.
Sprint 03	Diseño de la Capa de Datos, Negocio y Diseño de Secuencia.
Sprint 04	Implementación de la BDD y módulos de autenticación y administración de usuarios en el servicio web y aplicación móvil.
Sprint 05	Implementación de los módulos de gestión de rutas y de buzón de mensajes de usuarios en el servicio web y aplicación móvil.
Sprint 06	Implementación de las interfaces de la aplicación cliente e información para pruebas en BDD.
Sprint 07	Pruebas de Funcionamiento.

2.5. Sprint 01 - Definición de Requerimientos del Product Owner

Dentro de este Sprint se procederán a definir dentro de los PBI las historias de usuarios de los requerimientos funcionales del cliente o Product Owner. A continuación, de la tabla 2.4 a la 2.10 se describirán los diferentes PBI desde la perspectiva del Product Owner, estos permitirán definir de forma global los requerimientos del prototipo.

La forma de acceso del usuario al sistema se encuentra definido en la tabla 2.4.

Tabla 2.4. PBI A01: Acceso a través de Aplicación móvil Android

PBI A01: Acceso a través de Aplicación móvil Android	
Sprint: 01	**Prioridad:** 3
Asignado a: Diego Llerena	**Complejidad:** 3
Descripción El usuario puede acceder al sistema a través de una aplicación móvil conectada al internet.	
Validación Acceso a los servicios a través de una aplicación móvil Android.	

El rol para un conductor y sus permisos en la aplicación se definen en la tabla 2.5.

Tabla 2.5. PBI A02: Usuarios con un rol de conductores

PBI A02: Usuarios con un rol de conductores	
Sprint: 01	**Prioridad:** 2
Asignado a: Andrés Baus	**Complejidad:** 2
Descripción Es el usuario que puede publicar rutas. Puede registrar su vehículo particular, establece una ruta fija y un horario a través de la aplicación móvil.	
Validación Dentro de la aplicación móvil existirán formularios para registrar uno o varios vehículos; publicar o eliminar rutas.	

El rol para un pasajero y sus permisos en la aplicación se encuentran definidos en la tabla 2.6.

Tabla 2.6. PBI A03: Usuarios con un rol de pasajeros

PBI A03: Usuarios con un rol de pasajeros	
Sprint: 01	**Prioridad:** 2
Asignado a: Diego Llerena	**Complejidad:** 2
Descripción Es el usuario que puede buscar y suscribirse a rutas.	
Validación Dentro de la aplicación móvil existirán formularios para buscar rutas; suscribirse o desuscribirse de rutas.	

El rol para un administrador y sus permisos en la aplicación se encuentran definidos en la tabla 2.7.

Tabla 2.7. PBI A04: Usuarios con un rol de administración

PBI A04: Usuarios con un rol de administración	
Sprint: 01	**Prioridad:** 2
Asignado a: Andrés Baus	**Complejidad:** 2
Descripción Es el usuario que puede leer, agregar, editar o eliminar usuarios y rutas.	
Validación Dentro de la aplicación móvil existirán formularios que permitan leer, agregar, editar o eliminar usuarios y rutas. Solo podrán acceder a ellos los usuarios con el rol de administrador.	

El concepto de ruta fija y los parámetros necesarios para identificarla en el sistema se encuentran definidos en la tabla 2.8.

Tabla 2.8. PBI A05: Rutas de usuario fijas

PBI A05: Rutas de usuario fijas	
Sprint: 01	**Prioridad:** 2
Asignado a: Diego Llerena	**Complejidad:** 2
Descripción Una ruta fija es una descripción de un camino a seguir entre una dirección origen y un destino a definir por los conductores. Una ruta fija contará con horarios, los cuales son una descripción de la fecha de inicio de la ruta, fecha de finalización de la ruta y el horario de inicio del recorrido.	
Validación Dentro de la aplicación móvil existirán formularios que permitan observar la información descrita en la descripción de la ruta.	

Las características del buzón de mensajes para un usuario que permitirá la comunicación entre conductores y pasajeros se encuentran definidos en la tabla 2.9.

Tabla 2.9. PBI A06: Buzón de mensajes para los usuarios

PBI A06: Buzón de mensajes para los usuarios	
Sprint: 01	**Prioridad:** 2
Asignado a: Andrés Baus	**Complejidad:** 2
Descripción Un buzón de mensajes que permitirá el intercambio de mensajes entre los pasajeros que se encuentren suscritos a una ruta con el conductor que publicó dicha ruta.	
Validación Dentro de la aplicación móvil existirán formularios que permitan enviar y recibir mensajes dentro de la aplicación entre pasajeros y conductores.	

La funcionalidad de registro de usuarios, en donde un usuario se registra para acceder a los servicios del sistema se encuentran definidos en la tabla 2.10.

Tabla 2.10. PBI A07: Registro de usuarios

PBI A07: Registro de usuarios	
Sprint: 01	**Prioridad:** 2
Asignado a: Diego Llerena	**Complejidad:** 1
Descripción Nuevos usuarios podrán registrarse en el sistema a través de la aplicación móvil, ingresando su información personal.	
Validación Dentro de la aplicación móvil existirán formularios que permitan registrarse en el sistema. Después del registro, los usuarios podrán acceder a los servicios de la aplicación.	

2.6. Sprint 02 - Definición de Requerimientos del Proveedor y Diseño de secuencia.

Dentro de los siguientes Sprint se procederán a definir dentro de los PBI las historias de usuarios de los requerimientos del Development Team. Estos describirán los requerimientos no funcionales.

Las características técnicas de los servicios en la nube a ser utilizados por el sistema se encuentran definidos en la tabla 2.11.

Tabla 2.11. PBI B01: Infraestructura de servicios en la nube

PBI B01: Infraestructura de servicios en la nube	
Sprint: 02	**Prioridad:** 3
Asignado a: Andrés Baus	**Complejidad:** 1
Descripción El proveedor de los servicios debe de estar localizado en una arquitectura en la nube, es decir a través de un servicio. Se requiere utilizar IaaS ya que permite mantener los componentes de BDD y servicios web en un solo equipo administrable.	
Validación Verificar que los servicios se encuentren alojados en la nube y esté sobre un ambiente del tipo IaaS.	

2.6.1. Diagrama de Secuencia

Los diagramas de secuencia son encargados de mostrar dentro de un sistema funcional, la forma en que los objetos interactúan entre sí, y los momentos en el tiempo [30].

El método de definición de los diagramas de secuencia en base a los componentes del sistema se encuentra definido en la tabla 2.12.

Tabla 2.12. PBI D01: Diseño de diagrama de secuencia

PBI D01: Diseño de diagrama de secuencia	
Sprint: 02	**Prioridad:** 3
Asignado a: Diego Llerena	**Complejidad:** 2
Descripción Utilizar los diagramas de secuencia para definir la interacción de los componentes que forman parte del sistema y la interacción entre ellos.	
Validación Presentación de diagrama de secuencia.	

El flujo comienza con el cliente utilizando la aplicación móvil, a través de su interfaz, para agregar, editar o eliminar información, ya sea sobre usuarios, rutas, vehículos o mensajes. La aplicación se comunicará con el servicio web a través de peticiones web.

Una petición web es una acción que indica la necesidad a acceder a un recurso del servicio web. [31] A través del protocolo http la aplicación enviará peticiones web hacia el servicio, este procesará la información y realizará operaciones con la base de datos, al concluir este proceso se generará una respuesta dirigida a la aplicación móvil.

Finalmente, la interfaz de la aplicación se encargará de mostrar un resultado ante la petición realizada, a través de estos criterios se realiza el diagrama de secuencia del sistema presente en la figura 2.1.

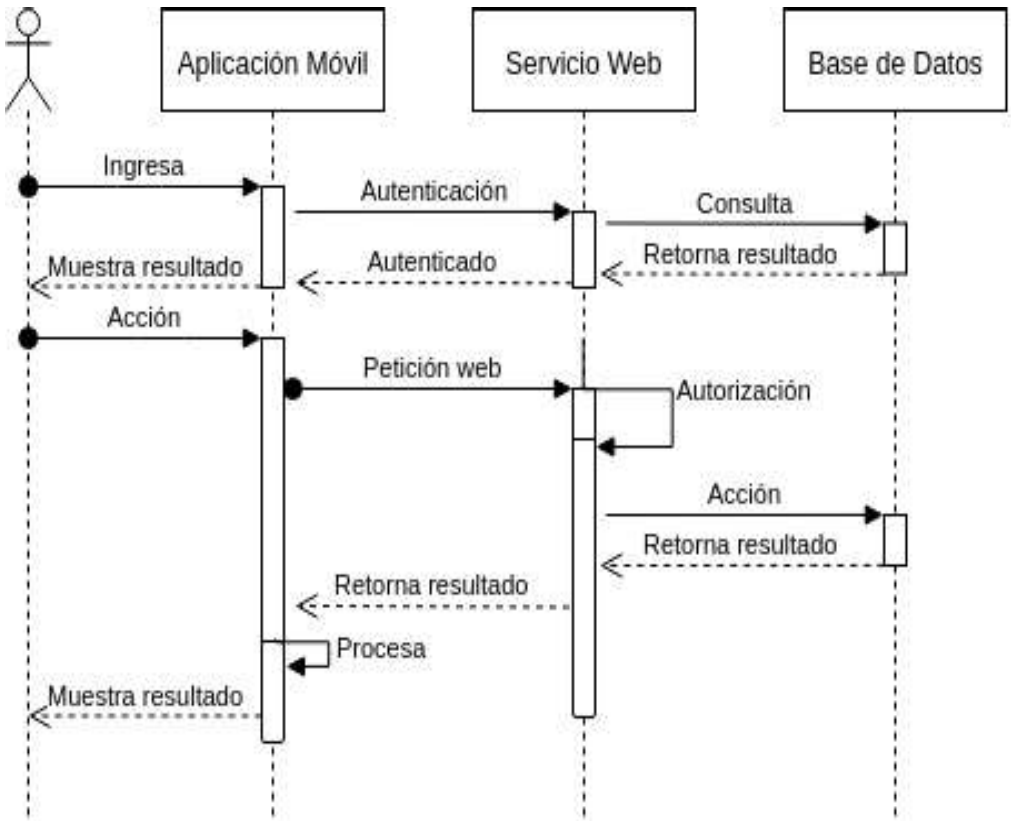


Figura 2.1. Diagrama de Secuencia

2.7. Sprint 03 – Diseño de la Capa de Datos, Negocio y Presentación

2.7.1. Diseño de la Base de Datos

El diseño de la base de datos contempla la funcionalidad que debe de cubrir; los modelos que va a almacenar y sus relaciones, características que se muestran a detalle en la tabla 2.13.

Tabla 2.13. PBI B02: Diseño de la base de datos

PBI B02: Diseño de la base de datos	
Sprint: 03	**Prioridad:** 3
Asignado a: Diego Llerena	**Complejidad:** 3

Descripción

La capa de datos se implementará a través de una base de datos, esta contiene los modelos y relaciones respectivas, además de las multiplicidades asignadas a cada relación.

La BDD debe de establecer los modelos necesarios para almacenar la información de rutas, usuarios, mensajes y vehículos. Dentro de la base pueden existir uno o varios usuarios, estos usuarios pueden a su vez tener uno o varios vehículos, y estos vehículos pueden ser de uno o más tipos de vehículo.

Un usuario puede publicar una o más rutas, donde cada ruta tiene un solo vehículo asociado a esta. Un usuario puede crear una o varias conversaciones, donde estas pueden tener uno o más mensajes.

Las tablas por implementarse son las siguientes:

- **Usuario:** Almacenará toda la información referente al usuario como su datos personales e información de vivienda.
- **Vehículo:** Almacenará la información de los vehículos de un usuario.
- **TipoVehiculo:** Almacena los distintos tipos de vehículos.
- Ruta: Almacenará los datos de una ruta definida por un usuario con el rol de conductor.
- **UsuarioRuta:** Almacenará los usuarios asociados con una ruta tanto el conductor como los pasajeros.
- **Mensaje:** Define los atributos de cada mensaje a ser intercambiado entre conductor y pasajero.
- **Conversación:** Define identificadores para asociar los usuarios y la conversación a la que pertenecen.
- **UsuarioConversación:** Define los parámetros para asociar cada uno de los usuarios participantes de una misma conversación.

Validación

Diagrama del modelo relacional de la BDD.

2.7.2. Diagrama Entidad – Relación

El diagrama entidad – relación posee varias entidades, entre las principales se encuentra la entidad usuario y ruta, la relación existente entre ellas comprende a la funcionalidad principal del prototipo desarrollado. En la figura 2.2, se muestra el diagrama entidad relación el cual recoge todos los criterios mencionados en el PBI B02 y se detalla la estructura de las tablas de la base de datos.

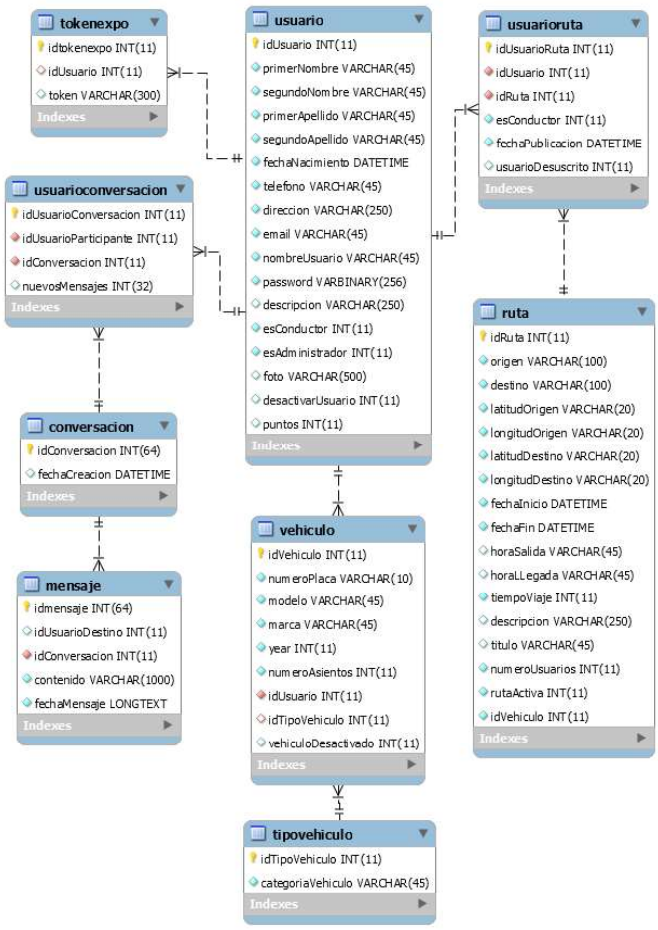


Figura 2.2. Diagrama Entidad – Relación de la Base de Datos

Los requerimientos del proyecto indican que es necesario almacenar la información sobre los usuarios, rutas, vehículos y mensajes. Para esto se han establecido los diferentes modelos con sus atributos especificados por el Product Owner.

2.7.3. Diseño del API REST

El diseño del API será implementado utilizando un servicio del tipo REST. Las características de este se definen en la tabla 2.14. Además, se describen los módulos que va a tener y las características de cada uno.

Tabla 2.14. PBI B03: Diseño del API REST

PBI B03: Diseño del API REST	
Sprint: 03	**Prioridad:** 3
Asignado a: Andrés Baus	**Complejidad:** 3
Descripción Una API REST será la encargada de proporcionar los servicios disponibles, mediante esta se podrá realizar un CRUD de los modelos establecidos en la BDD y completar cierta funcionalidad dentro de la aplicación. El diseño del API debe de cumplir con los módulos: • **Administración de Usuarios:** Este módulo contará la capacidad para crear, leer, editar y eliminar a los usuarios del sistema. • **Autenticación**: Este módulo permitirá autenticar a los usuarios de la aplicación móvil con el sistema. • **Gestión de Rutas:** Este módulo habilita la publicación de rutas fijas y horarios. Una ruta fija es una descripción de un camino a seguir entre una dirección origen y un destino a definir por los conductores. Una ruta fija contará con horarios, los cuales son una descripción de la fecha de inicio de la ruta, fecha de finalización de la ruta y el horario de inicio del recorrido. • **Buzón de mensajes:** Este módulo permitirá el intercambio de mensajes entre los pasajeros que se encuentren suscritos a una ruta con el conductor que publicó dicha ruta. Se procederá a utilizar los diagramas de casos de uso para especificar con detalle los requerimientos funcionales y los diagramas de clase para representar la estructura en un formato de orientado a objetos. En este caso las clases van a representar únicamente a los modelos de la BDD y se utilizarán para trabajar con la información almacenada.	
Validación Diagramas de caso de uso y de clases completados.	

2.7.4. Diagramas de Casos de Uso

Los diagramas de caso de uso se utilizarán para definir las interacciones de los tipos de usuario con el prototipo.

Los tipos de usuario son:

- Nuevo
- Pasajero
- Conductor
- Administrador

Para un usuario nuevo del sistema no se tiene asignado un rol y tan solo puede registrarse, ingresando sus datos personales. Estas características se observan en el diagrama de la figura 2.3.

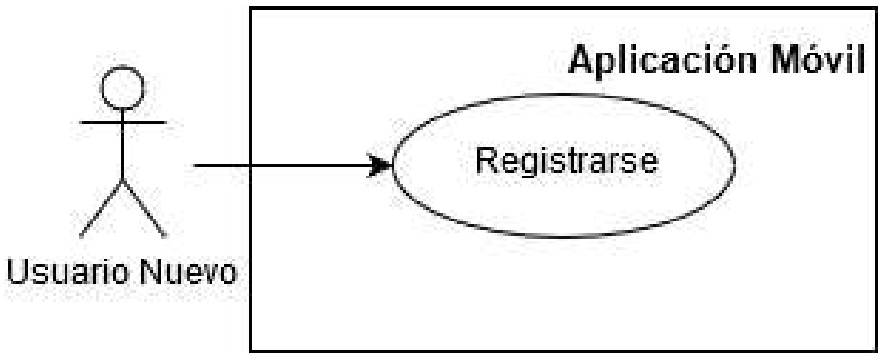


Figura 2.3. Caso de Uso: Usuario Nuevo

Un usuario con el rol de pasajero representa la base para cualquier usuario registrado y contiene la mayoría de las funcionalidades del sistema.

El rol permite la búsqueda y suscripción a rutas; recepción y emisión de mensajes con los conductores de la ruta; edición de la información de perfil como datos personales o claves de usuario; y finalmente, permite al usuario convertirse en conductor con la posibilidad de poder agregar vehículos, estas características están presentes en el diagrama de la figura 2.4.

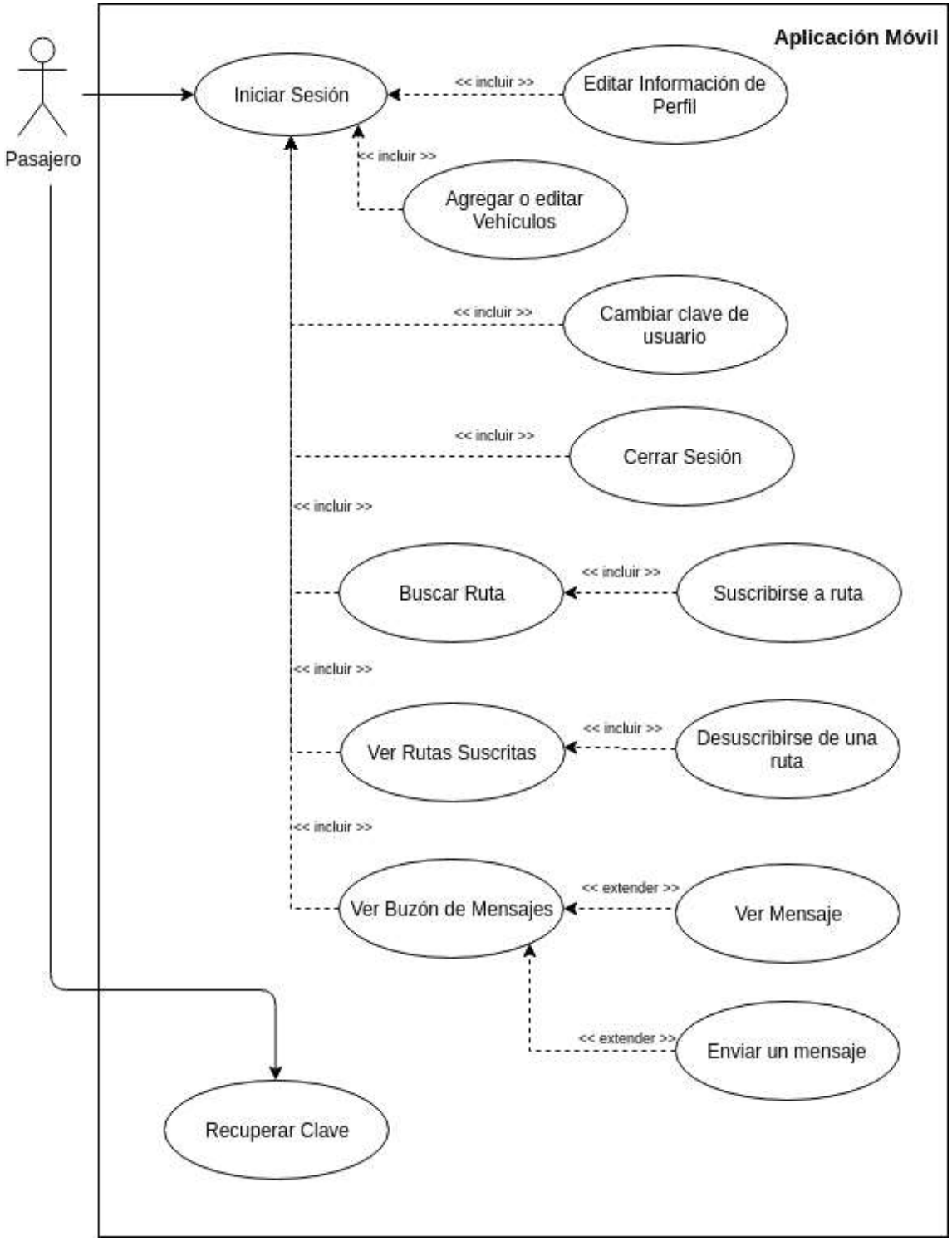


Figura 2.4. Caso de Uso: Usuario con rol de Pasajero

El usuario con rol de conductor permite al usuario publicar rutas, así también como eliminarlas cuando sea requerido. Un usuario con el rol del tipo pasajero puede o no tener el rol de conductor, estas características están representas en la figura 2.5.

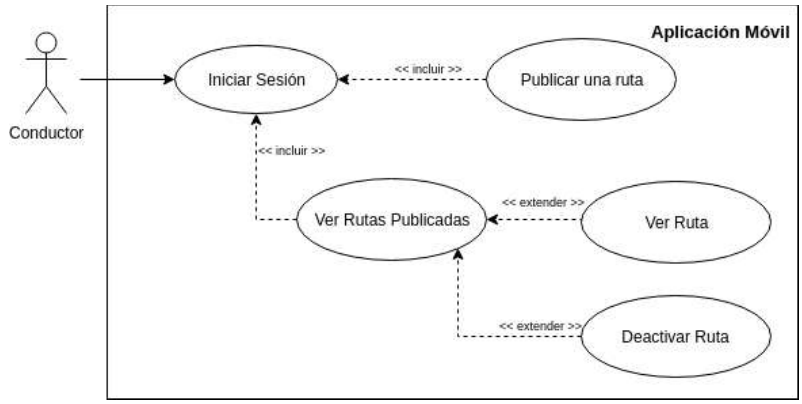


Figura 2.5. Caso de Uso: Usuario con rol de Conductor

Un usuario con un rol de administrador permite al usuario ver, editar o eliminar rutas y otros usuarios con sus vehículos, además permite agregar usuarios. Dentro del sistema existirá solo un usuario del tipo administrador, estas características están representadas en la figura 2.6.

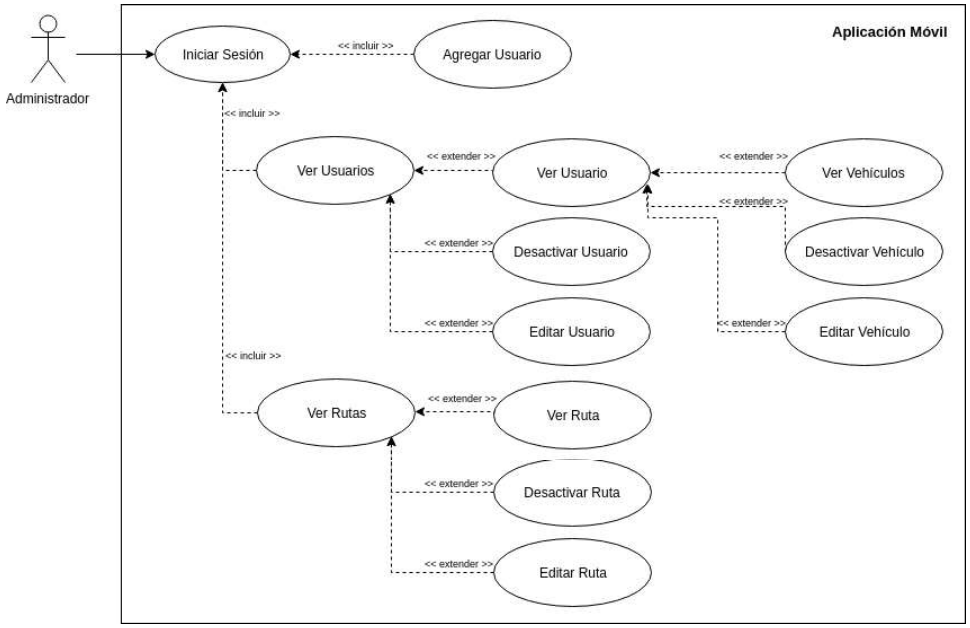


Figura 2.6. Caso de uso: Usuario con rol de Administrador

2.7.5. Diagrama de Clases

El diagrama presenta la estructura de las clases y objetos utilizados como usuario, ruta, vehículo, entre otros que se pueden observar en la figura 2.7. A excepción de la clase dbconnection, las clases son consideradas como clases estáticas, ya que no se pueden crear instancias de estas, únicamente se pueden utilizar los métodos definidos.

Estas clases serán usadas por el sistema para el intercambio de datos entre la capa de base de datos y la capa de presentación.

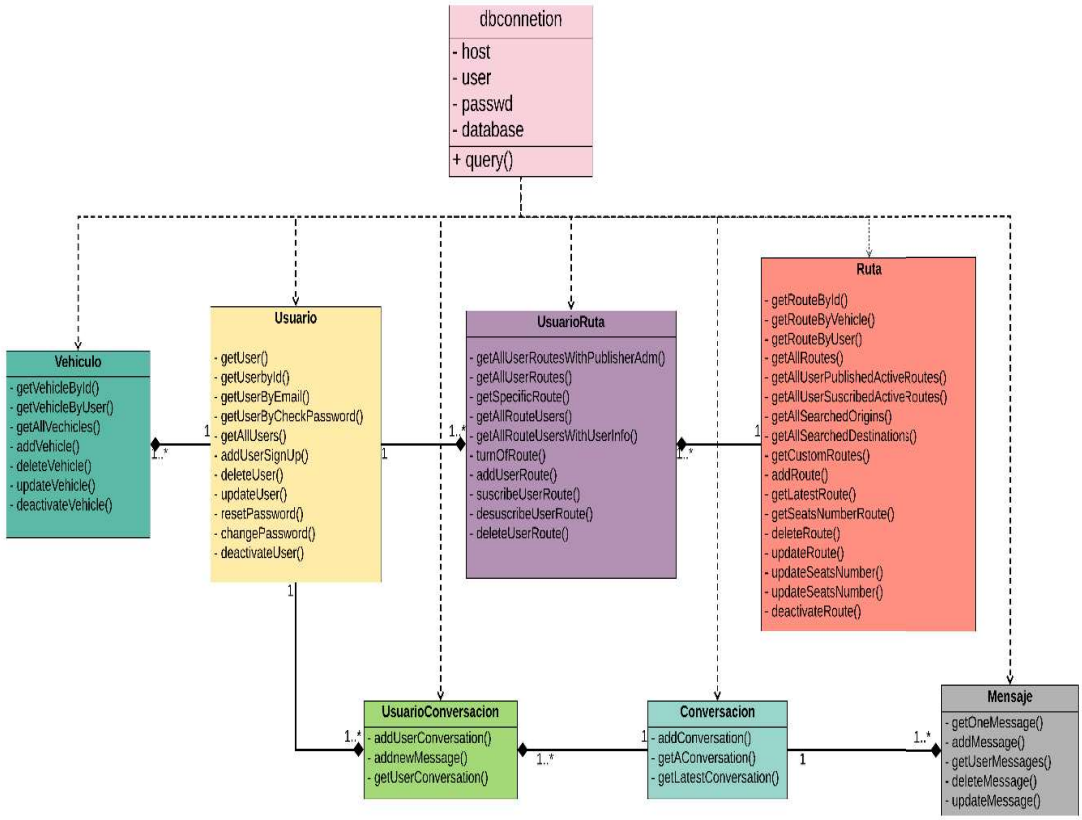


Figura 2.7. Diagrama de Clases

2.7.6. Diseño de la Aplicación móvil

El diseño de la aplicación móvil, es decir la capa de presentación, contiene la funcionalidad que debe de cubrir, así como los módulos que la componen; estas características se encuentran descritas en la tabla 2.15.

Tabla 2.15. PBI B04: Diseño de la aplicación móvil

PBI B04: Diseño de la aplicación móvil	
Sprint: 03	**Prioridad:** 3
Asignado a: Diego Llerena	**Complejidad:** 3
Descripción La capa de presentación es la encargada de mostrar la información al usuario, la aplicación móvil utilizará formularios o interfaces para capturar las interacciones del usuario y exponer los resultados a las distintas interacciones. El diseño de la aplicación móvil debe de permitir a través de la interfaz o formularios las siguientes funcionalidades expuestas a través de los siguientes módulos: • **Administración de Usuarios:** Este módulo contará la capacidad para crear, leer, editar y eliminar a los usuarios del sistema. • **Autenticación**: Este módulo permitirá autenticar a los usuarios de la aplicación móvil con el sistema. • **Gestión de Rutas:** Este módulo habilita la publicación de rutas fijas y horarios. Una ruta fija es una descripción de un camino a seguir entre una dirección origen y un destino a definir por los conductores. Una ruta fija contará con horarios, los cuales son una descripción de la fecha de inicio de la ruta, fecha de finalización de la ruta y el horario de inicio del recorrido. Todos estos indicadores permiten a los usuarios determinar si la ruta es conveniente para ellos y que puedan suscribirse a las mismas. • **Buzón de mensajes:** Este módulo permitirá el intercambio de mensajes entre los pasajeros que se encuentren suscritos a una ruta con el conductor que publicó dicha ruta. El diseño de la aplicación se lo realizará a través de sketches que mostrarán las principales interfaces y servirán de referencia para la implementación de estas.	
Validación Se realizarán sketches de las interfaces de la aplicación móvil prototipo.	

A continuación, se presentarán sketches de las interfaces de la aplicación móvil, estás permitirán completar el flujo de la aplicación desde el inicio de sesión hasta las funcionalidades expresadas en el PBI B04.

La pantalla de carga de la aplicación se observa en la figura 2.8, esta se mostrará en el inicio de la aplicación.

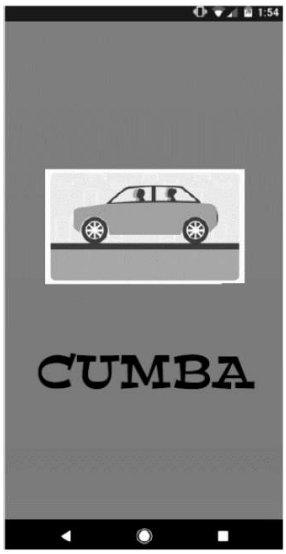


Figura 2.8. Sketch de Pantalla de Carga

El formulario de inicio de sesión se observa en la figura 2.9, a través de este los usuarios podrán ingresar autenticarse y utilizar la aplicación.

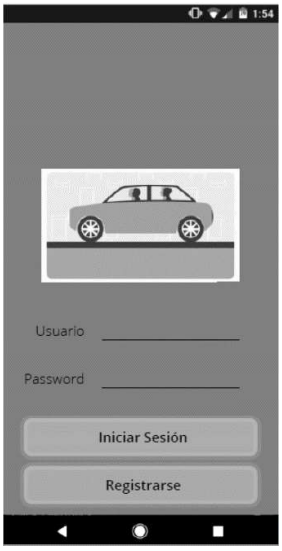


Figura 2.9. Sketch de Inicio de Sesión

La interfaz de perfil de usuario se observa en la figura 2.10, esta interfaz contiene la información personal de cada usuario, además de que permite su edición.

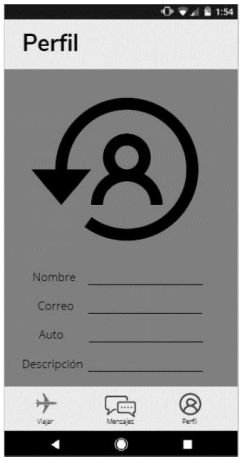


Figura 2.10. Sketch de Perfil de Usuario

La pantalla de inicio será la interfaz que se presentará al usuario cuando inicie la sesión, se encuentra en la figura 2.11. Además, permitirá a los usuarios acceder a las funcionalidades de la aplicación como la publicación y suscripción de rutas, edición de información de usuario y vehículos y finalmente los formularios para cambio de clave o de cierre de sesión.

Figura 2.11. Sketch de Pantalla de Inicio

La interfaz para buscar una ruta permite a través del ingreso de datos como fecha de salida y llegada, destino y origen obtener una lista de rutas acorde a los parámetros ingresados, estas características se pueden ver en la figura 2.12.

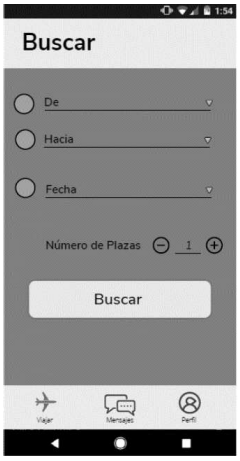


Figura 2.12. Sketch de Pantalla de búsqueda de ruta

La interfaz general para los resultados de rutas podrá ser usada para los resultados de búsqueda de rutas, lista de rutas publicadas o lista de rutas suscritas; se la puede observar en la figura 2.13. Cada ítem de resultado en la interfaz contiene información básica sobre la ruta publicada.

Figura 2.13. Sketch de Lista de Resultados de Ruta

La interfaz para el buzón de mensajes contendrá una lista de mensajes recibidos con información básica para identificar a cada mensaje, estas características se pueden ver en la figura 2.14.

Figura 2.14. Sketch de Buzón de Mensajes

La interfaz para publicar una ruta permitirá agregar información de la ruta como fecha, destino, origen y número de plazas disponibles. Además, también permitirá publicar una ruta, estas características se pueden ver en la figura 2.15.

Figura 2.15. Sketch de Información de Ruta / Publicación de Ruta

2.7.7. Selección del proveedor de IaaS

Para la selección del proveedor de IaaS se requiere comprar los aspectos técnicos de cada una de las plataformas, escalabilidad y costos. El detalle de los requerimientos se encuentra en la 2.16.

Tabla 2.16. PBI D02 Selección del Proveedor de IaaS

PBI D02: Selección del Proveedor de IaaS	
Sprint: 03	**Prioridad:** 2
Asignado a: Andrés Baus	**Complejidad:** 3
Descripción Se requiere comparar las características que brindan los servicios en la Nube que soportan IaaS para determinar restricciones de despliegue, compatibilidad, opciones de escalabilidad, y costos por uso del servicio.	
Validación Comparativa y selección del proveedor de IaaS en base a los criterios establecidos.	

Para la selección del proveedor se han investigado los siguientes proveedores de IaaS del mercado:

- Amazon Web Services (AWS) [32]
- Microsoft Azure

Se ha limitado la comparación a estos dos proveedores ya que ofrecen programas de prueba gratuita para Cloud Computing durante 12 meses con características similares de uso:

Características Comunes:

- Programa de Prueba gratuita.
- 750 horas de computación en la nube al mes.
- Soporte para sistemas operativos Linux y Windows.

Diferencias:

AWS: Amazon provee su servicio de Elastic Compute Cloud (EC2), el cual es un servicio web que provee capacidad computacional segura y redimensionable.

Además, soporta procesamiento de alto rendimiento, balanceadores de carga y una nube privada con VMWare para pruebas.

Un mayor detalle de características se puede visualizar en la tabla 2.17.

Tabla 2.17. Características de AWS para Servicios Computacionales

Proveedor	Servicios Computacionales
AWS	EC2
	Batch
	Elastic Beanstalk
	Fargate
	Auto Scaling
	Elastic Load Balancing
	Vmware Cloud on AWS

Azure: El programa de Azure llamado "Máquinas Virtuales" soporta dentro de sus características técnicas varios sistemas operativos como Windows, Linux, Oracle, IBM.

Además, ofrece características que se encuentran optimizadas para inteligencia artificial y machine learning. Las características de Azure con mayor detalle se muestran en la tabla 2.18.

Tabla 2.18. Características de Azure para Servicios Computacionales

Proveedor	Servicios Computacionales
Microsoft Azure	Virtual Machines
	Virtual Machine Scale Sets
	Azure Container Service (AKS)
	Container Instances
	Batch
	Cloud Services

Luego de examinar las características expuestas en la tabla 2.17 y la tabla 2.18 se puede concluir que:

- Ambos proveedores ofrecen servicios de máquinas virtuales lo cual es fundamental para la selección del proveedor IaaS.
- AWS provee un mayor número de características gratuitas respecto a las ofrecidas por Microsoft Azure.
- AWS provee un sistema de redimensionamiento ágil (Elastic Load Balancing) mientras que Azure no oferta una característica similar.

De acuerdo con los requerimientos del proyecto el despliegue de los servicios web puede realizarse indistintamente ya que ambos proveedores cumplen con el requerimiento funcional planteado.

Sin embargo, las opciones de ágil redimensionamiento y balanceo de carga sumadas a la posibilidad de realizar pruebas en nubes virtuales que ofrece AWS son características importantes que considerar para que el prototipo en un futuro pueda pasar a una etapa de producción masiva sin afectar las condiciones iniciales.

Por lo tanto, se selecciona AWS como el proveedor de servicios IaaS a usar en la implementación.

3. IMPLEMENTACIÓN

En este capítulo se llevará a cabo la descripción del proceso de implementación del prototipo. Se describirán los pasos a seguir durante la creación de la base de datos; los módulos del servicio web; los módulos e interfaz de la aplicación móvil; y finalmente la configuración del servicio web en la nube y la generación del archivo instalable de la aplicación móvil.

La implementación de la base de datos utilizará MySQL como sistema de BDD, el servicio web será implementado a través del framework Express que utiliza a Node JS como lenguaje de programación, y finalmente se utilizará React Native para la creación de la aplicación móvil.

3.1. Sprint 04 – Implementación de la BDD y módulos de autenticación y administración de usuarios en el servicio web y aplicación móvil.

3.1.1. Implementación de la Base de Datos

Para la implementación de la Base de datos se ha utilizado el sistema de gestión de base de datos relacional "MySQL" [6].

El criterio técnico utilizado para la codificación de la base de datos se encuentra descrito en la tabla 3.1, para esto se procedió definiendo el lenguaje de codificación, el motor de la base de datos y el esquema de sus tablas.

Tabla 3.1. PBI B05: Codificación de la Base de Datos

PBI B05: Codificación de la Base de Datos	
Sprint: 04	**Prioridad:** 3
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 1
Descripción La codificación del script de la base de datos será a través del SQL utilizando el motor de base de datos de MySQL, el código tendrá a los modelos y los elementos de cada uno, así también como las relaciones existentes entre ellos.	
Validación Script completo de la base de datos. El script se encontrará en el anexo A.	

3.1.2. Implementación del módulo de autenticación del servicio web

La autenticación de los usuarios de la aplicación móvil con el servicio web será a través de los identificadores únicos de usuario dentro de la base de datos.

Estos identificadores únicos serán provistos usando el estándar JWT (JSON Web Tokens).

El estándar JWT plantea la creación de tokens de acceso los cuales son identificadores únicos para cada usuario y gracias a estos tokens se proveerá autorización al servicio web, estas características se encuentran en la tabla 3.2.

Tabla 3.2. PBI B06: Codificación del módulo de autenticación en el servicio web

PBI B06: Codificación del módulo de autenticación en el servicio web	
Sprint: 04	**Prioridad:** 2
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 2
Descripción La codificación del módulo de autenticación para un usuario de la aplicación móvil se realizará por medio de JWT, donde con cada usuario nuevo que se registre en la plataforma se generará un token de usuario. Este token será un identificador de usuario único, debe estar cifrado para proteger los datos de autenticación del usuario y será el requisito para habilitar todas las consultas o requerimientos que la aplicación móvil deba gestionar con el servidor.	
Validación Consultas desde la aplicación móvil enviando/no enviando el token de cada usuario.	

El código del módulo de generación de tokens que permite a los usuarios iniciar sesión en el sistema se puede observar en la figura 3.1. Cuando un usuario ingresa a la aplicación el servidor recibe las credenciales de acceso y las contrasta con la base de datos. Si las credenciales son correctas el usuario recibe un token JWT único, generado a partir del nombre de usuario y una Passphrase definida para el servidor. Este token le permitirá utilizar todos los módulos de la aplicación.

```
Usuario.getUser(nombreUsuario, function(err, rows, fields) {
    if (err) {
      res.json(err);
    } else {

        if (rows.length > 0) {
            //console.log(rows);
            if (rows[0].password == req.body.password) {
            // if (rows[0].password == password) {

                token = jwt.sign(nombreUsuario, Passphrase);

                message = "Login succesfull"

                res.status(200).json({
                    message,
                    token
                });

            } else {

                message = "Nombre de usuario y password no existen";
                res.status(403).json({
                    message
                });
            }
```

Figura 3.1. Código de Generación de tokens para usuarios del Sistema.

El código para autenticación de peticiones en base a JWT, con el cual cada una de las peticiones realizadas al servidor debe contener el token del usuario que realiza la petición se puede observar en la figura 3.2. Con esto se garantiza que solo los usuarios que tienen un token de la aplicación puedan consultar la información del servidor.

Figura 3.2. Código de autenticación para peticiones al servidor.

3.1.3. Implementación del módulo de autenticación de aplicación móvil

El módulo de autenticación dentro de la aplicación móvil se encargará de almacenar y de brindar el acceso a la información recibida del servicio web. Este módulo permitirá el almacenamiento del identificador único de usuario y de su correspondiente token JWT, este detalle se puede ver en la tabla 3.3.

Tabla 3.3. PBI B07: Codificación del módulo de autenticación en la aplicación móvil

PBI B07: Codificación del módulo de autenticación en la aplicación móvil	
Sprint: 04	**Prioridad:** 2
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 2
Descripción La codificación de este módulo de autenticación será realizada usando un archivo que contendrá las funciones necesarias para este modelo. Esta clase tendrá métodos para poder almacenar el identificador único del usuario y también el JWT asignado al mismo.	
Validación Almacenamiento y consulta del identificador único y JWT asignado a usuario.	

El código para realizar una petición al servicio web y obtener las credenciales de autenticación se puede observar en la figura 3.3. Se realiza una petición http del tipo POST al servidor y se introduce en el cuerpo de la petición las credenciales del usuario.

Figura 3.3. Código para consulta de JWT del usuario desde la aplicación móvil hacia el servicio web

El código para el manejo de la respuesta del servicio web a la aplicación al momento de realizar la autenticación del usuario se puede observar en la figura 3.4.

- Entre la línea 64 y 67 se está manejando una autenticación fallida por parte del usuario. En la que se mostrará un error del mensaje.
- Entre la línea 68 y 84 se procede a almacenar el JWT recibido por el servicio web que autentica y autoriza al usuario a conectarse con el servicio web.

Figura 3.4. Código para el manejo de la respuesta del servicio web a la aplicación sobre la autenticación de un usuario

3.1.4. Implementación del módulo de administración de usuarios del servicio web

El módulo de administración de usuarios del servicio web permite a nivel del servidor realizar un CRUD de la información para los usuarios registrados en el sistema. Para conseguir esto se tiene una ruta que brinda el acceso al servicio web, para luego realizar el procesamiento de los datos en el servidor. Este detalle se encuentra en la tabla 3.4

Tabla 3.4. PBI B08: Codificación del módulo de administración de usuarios en el servicio web

PBI B08: Codificación del módulo de administración de usuarios en el servicio web	
Sprint: 03	**Prioridad:** 2
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 2
Descripción La codificación de este módulo se realizará con la creación de una ruta al servicio web, mediante la cual se recibirán los datos desde la aplicación. Las peticiones de la aplicación se recibirán como peticiones web, para luego aprovechar métodos de las clases usuario y vehículo y con esto realizar las modificaciones de la información con sentencias SQL sobre la base de datos.	
Validación Capacidad para leer, crear, eliminar y editar información de usuario en la base de datos.	

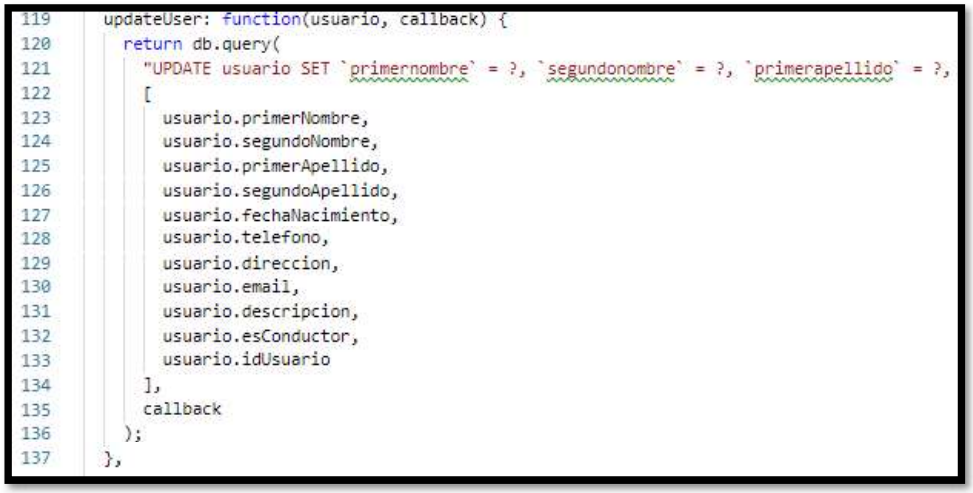
El código para la creación de un usuario en el servidor se puede observar en la figura 3.5, donde a través de una petición del tipo POST se reciben los datos del nuevo usuario y con esto se guarda la información en la base de datos.

El código para realizar la actualización de un usuario en el servidor se puede observar en la figura 3.6. Se reciben los parámetros a cambiar del usuario y se realiza una modificación de estos a nivel de base de datos.

```
addUserSignUp: function(usuario, callback) {

  return db.query(
    "INSERT INTO `usuario` (`primerNombre`, `segundoNombre`, `primerApellido`,
    [
      usuario.primerNombre,
      usuario.segundoNombre,
      usuario.primerApellido,
      usuario.segundoApellido,
      usuario.fechaNacimiento,
      usuario.telefono,
      usuario.direccion,
      usuario.email,
      usuario.nombreUsuario,
      usuario.password,
      usuario.descripcion,
      0,
      0
    ],
    callback
  );
},
```

Figura 3.5. Código para ingresar un nuevo usuario en el servidor.



```
updateUser: function(usuario, callback) {
  return db.query(
    "UPDATE usuario SET `primernombre` = ?, `segundonombre` = ?, `primerapellido` = ?,
    [
      usuario.primerNombre,
      usuario.segundoNombre,
      usuario.primerApellido,
      usuario.segundoApellido,
      usuario.fechaNacimiento,
      usuario.telefono,
      usuario.direccion,
      usuario.email,
      usuario.descripcion,
      usuario.esConductor,
      usuario.idUsuario
    ],
    callback
  );
},
```

Figura 3.6. Código para la actualización de la información de un usuario en el servidor.

El código para la desactivación de un usuario en la base de datos se puede observar en la figura 3.7. Para mantener la consistencia de datos de la aplicación a nivel del servidor se

ha optado por no eliminar la información de un usuario de la base de datos, sino que el usuario al optar por la opción eliminar desactiva su cuenta.

En el código:

- Línea 222 hasta línea 248: Envía un correo electrónico al usuario informándole de la desactivación de este.
- Línea 252 hasta línea 257: Actualizan la información del usuario en la base de datos, cambiando el estado de ese usuario a desactivado con el booleano 0.

```
220  deactivateUser: function(UserInfo, callback){
221
222    var cumbaMail = 'cumbaprojectepn@gmail.com';
223    var cumbaPass = 'Cumba12345';
224
225    var transporter = nodemailer.createTransport("SMTP", {
226      host: 'smtp.gmail.com',
227      port: 587,
228      secure: true,
229      auth: {
230        user: cumbaMail,
231        pass: cumbaPass
232      }
233    });
234
235    let mailOptions = {
236      from: cumbaMail, // sender address
237      to: UserInfo.email, // list of receivers
238      subject: 'Usuario Desactivado', // Subject line
239      text: 'El usuario: ' + UserInfo.nombreUsuario + ' ha sido desactiva con exito' // plain text body
240    };
241
       Complexity is 3 Everything is cool!
242    transporter.sendMail(mailOptions, (error, info) => {
243      if (error) {
244          return console.log(error);
245      }
246      console.log('Message sent: %s', info.messageId);
247
248    });
249
250
251    return db.query(
252      "UPDATE usuario SET `desactivarUsuario` = 1 WHERE `idUsuario` = ?",
253      [
254        UserInfo.idUsuario
255      ],
256      callback
257    );
```

Figura 3.7. Código para eliminar un usuario en el servidor.

3.1.5. Implementación del módulo de administración de usuarios en la aplicación móvil.

El módulo de administración de usuarios de la aplicación móvil se encargará de realizar el CRUD de usuarios y el intercambio de mensajes sobre usuarios con el servicio web, el detalle se encuentra en la tabla 3.5.

Tabla 3.5. PBI B09: Codificación del módulo de administración de usuarios en la aplicación móvil

PBI B09: Codificación del módulo de administración de usuarios en la aplicación móvil	
Sprint: 04	**Prioridad:** 2
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 2
Descripción La codificación de este módulo será realizada a través de un archivo que permitirá realizar el CRUD de un usuario. Además, permitirán solicitar o enviar información de usuarios al servicio web utilizando peticiones web.	
Validación Capacidad para leer, crear, eliminar y editar información de usuario.	

El código para la petición de creación de un usuario desde la aplicación móvil se puede observar en la figura 3.8. Se envía una petición http del tipo POST hacia el servicio web con la información del usuario.

```
fetch(
  constant.SERVER_IP + constant.URL_USUARIO + constant.URL_USUARIO_NEW,
  {
    method: "post",
    headers: {
      "content-Type": "application/json"
    },
    body: JSON.stringify({
      nombreUsuario: userName,
      password: password,
      primerNombre: firstName,
      segundoNombre: middleName,
      primerApellido: lastName,
      segundoApellido: lastName2,
      email: email,
      direccion: direction,
      fechaNacimiento: birthDate,
      telefono: phone
    })
  }
)
```

Figura 3.8. Código para petición de creación de un usuario desde la aplicación móvil

El código para la petición de actualización de un usuario desde la aplicación móvil se puede observar en la figura 3.9. Se envía una petición http del tipo PUT hacia el servicio web con la información del usuario.

```
68        fetch(constant.SERVER_IP + constant.URL_USUARIO, {
69          method: "put",
70          headers: {
71            "content-Type": "application/json"
72          },
73          body: JSON.stringify({
74            token: tokenStored,
75            idUsuario: this.state.idUser,
76            nombreUsuario: this.state.userName,
77            primerNombre: this.state.firstName,
78            segundoNombre: this.state.middleName,
79            primerApellido: this.state.lastName,
80            segundoApellido: this.state.lastName2,
81            email: this.state.email,
82            direccion: this.state.direction,
83            fechaNacimiento: dateToString(this.state.birthDate),
84            descripcion: this.state.description,
85            telefono: this.state.phone,
86            esConductor: this.state.isDriver
87          })
88        })
```

Figura 3.9. Código para petición de actualización de un usuario desde la aplicación móvil

El código para la petición de eliminación de un usuario desde la aplicación móvil se puede observar en la figura 3.10. Se envía una petición HTTP del tipo POST hacia el servicio web que ayude a identificar al usuario, hay que resaltar que realmente el usuario no es eliminado sino desactivado.

```
121       fetch(
122         constant.SERVER_IP +
123           constant.URL_USUARIO +
124           constant.URL_USUARIO_DEACTIVATE,
125         {
126           method: "post",
127           headers: {
128             "content-Type": "application/json"
129           },
130           body: JSON.stringify({
131             token: tokenStored,
132             usuario: {
133               idUsuario: this.state.idUser,
134               nombreUsuario: this.state.userName,
135               primerNombre: this.state.firstName,
136               segundoNombre: this.state.middleName,
137               primerApellido: this.state.lastName,
138               segundoApellido: this.state.lastName2,
139               email: this.state.email,
140               direccion: this.state.direction,
141               fechaNacimiento: dateToString(this.state.birthDate),
142               descripcion: this.state.description,
143               telefono: this.state.phone,
144               esConductor: this.state.isDriver
145             }
146           })
147         }
148       )
```

Figura 3.10. Código para petición de eliminación de un usuario desde la aplicación móvil.

3.2. Sprint 05 – Implementación de los módulos de gestión de rutas y de buzón de mensajes de usuarios en el servicio web y aplicación móvil.

3.2.1. Implementación del módulo de gestión de rutas en el servicio web.

El módulo de gestión de usuarios en el servicio web se encargará de recibir los datos de una ruta, verificar los datos e ingresarlos en la BDD, el detalle de la implementación se muestra en la tabla 3.6.

Tabla 3.6. PBI B10: Codificación del módulo de gestión de rutas de usuarios en el servicio web

PBI B10: Codificación del módulo de gestión de rutas de usuarios en el servicio web	
Sprint: 05	**Prioridad:** 2
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 2
Descripción La codificación de este módulo se realizará con la creación de una ruta mediante la cual se recibirán los datos desde la aplicación. Los datos recibidos pasan por verificaciones de formato para luego ser ingresados con sentencias SQL en la base de datos.	
Validación Capacidad para leer, crear, y eliminar información de una ruta en la base de datos.	

El código que permite crear una ruta se puede observar en la figura 3.11. El servidor recibe la información de la ruta en el cuerpo de la petición POST, luego de esto procesa la información necesaria y guarda este resultado en la base de datos.

En el código:

- Líneas 82 a 106: Procesan la información de coordenadas obtenidas desde la aplicación. Utilizan el servicio de *Google reverse geocoder* para obtener el nombre de la ubicación de ese punto del mapa y así guardarlo en la base de datos.

- Línea 113 a 119: Muestra parte de la sentencia SQL usada para guardar la información de la ruta en la base de datos.

```
82  async function busqueda (Ruta,res){
83    console.log("ingreso a la interna");
84    var latOri = Ruta.origen.split(",").shift();
85    var lonOri = Ruta.origen.split(",").pop();
86
87    var latDes = Ruta.destino.split(",").shift();
88    var lonDes = Ruta.destino.split(",").pop();
89
90    var tiempoViaje=Ruta.tiempoViaje
91
92    var resultadoOri;
93    try {
94      resultadoOri = await geocoder.reverse({lat:Number(latOri), lon:Number(lonOri)}, function(err, res) {});
95    }catch(err){
96      logger.error('Http error', err)
97      return res.status(500).send()
98    }
99
100   var resultadoDes
101   try {
102     resultadoDes = await geocoder.reverse({lat:Number(latDes), lon:Number(lonDes)}, function(err, res) {});
103   }catch(err){
104     logger.error('Http error', err)
105     return res.status(500).send()
106   }
107
108   console.log(resultadoOri[0].formattedAddress);
109   console.log(resultadoDes[0].formattedAddress);
110   console.log("Fin");
111
112
113   return db.query(
114     "INSERT INTO `ruta` (  `origen`,  `destino`,  `latitudOrigen`, `longitudOrigen`,  `latitudDestino`, `longitud
115     [
116
117       resultadoOri[0].formattedAddress,
118       resultadoDes[0].formattedAddress,
119       Ruta.origen.split(",").shift(),
```

Figura 3.11. Código para crear una nueva Ruta en el servidor.

El código para eliminar una ruta se puede observar en la figura 3.12. Para mantener la consistencia de datos, en el servidor no se elimina la información de la base, sino que se procede a desactivar la ruta.

```
//Desactivar una ruta
deactivateRoute: function(idRuta, callback) {
  return db.query(
    "UPDATE `ruta`  SET  `rutaActiva` = ?  WHERE `idRuta` = ?",
    [
      0,
      idRuta
    ],
    callback
  );
},
```

Figura 3.12. Código para la eliminación de una Ruta en el servidor.

El proceso de eliminación trae consigo una anulación automática de la suscripción de los usuarios que se encontraban suscritos. La función que realiza la anulación de la suscripción de los usuarios asociados a esa ruta se muestra en la figura 3.13. El código completo de esta operación se muestra en el anexo A.

```
function iterarFilas(rows) {
  console.log("Ingreso");
  for (i=0;i<rows.length;i++){

    console.log("Iterar filas> "+rows[i]);
    Complexity is 3 Everything is cool!
    UsuarioRuta.turnOfRoute(rows[i], function(err, rows) {
      let res = gblRes;
      if (err) {
        res.json(err);
      } else {
        var contador  = contador++;
      }

    });

  }

  return ("La ejecucion se ha completado con exito");

}
```

Figura 3.13. Código para la anulación de la suscripción de usuarios a una ruta.

3.2.2. Implementación del módulo de gestión de rutas en la aplicación móvil.

El módulo de gestión de rutas en la aplicación móvil tiene como función realizar el CRUD de las rutas almacenadas y el intercambio de mensajes entre los participantes de las rutas, todo esto usando el servicio web respectivo. El detalle de esta implementación se encuentra en la tabla 3.7.

Tabla 3.7. PBI B11: Codificación del módulo de gestión de rutas de usuarios en la aplicación móvil

PBI B11: Codificación del módulo de gestión de rutas de usuarios en la aplicación móvil	
Sprint: 05	**Prioridad:** 2
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 2
Descripción La codificación de este módulo será realizada a través de un archivo que permitirá realizar el CRUD de una ruta. Además, permitirán solicitar o enviar información al servicio web utilizando peticiones web.	
Validación Capacidad para leer, crear, y eliminar la información de ruta.	

El código para la petición de creación de una ruta desde la aplicación móvil se puede observar en la figura 3.14. Se envía una petición http del tipo POST hacia el servicio web con la información de la ruta.

```
fetch(constant.SERVER_IP + constant.URL_ROUTE + constant.URL_ROUTE_POST, {
  method: "post",
  headers: {
    "content-Type": "application/json"
  },
  body: JSON.stringify({
    token: tokenStored,
    origen: originLatitude + ", " + originLongitude,
    destino: destinyLatitude + ", " + destinyLongitude,
    fechaInicio: fechaInicio,
    fechaFin: fechaFinal,
    horaSalida: horaInicio,
    horaLlegada: horaFinal,
    descripcion: descripcion,
    titulo: titulo,
    idVehiculo: idVehiculo,
    tiempoViaje: tiempoViaje,
    idUsuario: idUsuario
  })
})
```

Figura 3.14. Código para petición de creación de una ruta desde la aplicación móvil

El código para la petición de creación de una ruta desde la aplicación móvil se puede observar en la figura 3.15, hay que destacar que realmente no se elimina la ruta, solo se la desactiva. Se envía una petición http del tipo POST hacia el servicio web con información para identificar a la ruta.

```
264     fetch(constant.SERVER_IP + constant.URL_ROUTE + constant.URL_ROUTE_DELETE, {
265       method: "post",
266       headers: {
267         "content-Type": "application/json"
268       },
269       body: JSON.stringify({
270         token: tokenStored,
271         idRuta: idRuta
272       })
273     })
```

Figura 3.15. Código para petición de eliminación de una ruta desde la aplicación móvil

3.2.3. Implementación del módulo de buzón de mensajes en el servicio web.

El módulo de buzón de mensajes en el servicio web permite el intercambio de mensajes, gestión de notificaciones y almacenamiento de conversaciones entre conductores y pasajeros. El detalle de implementación se encuentra en la tabla 3.8.

Tabla 3.8. PBI B12: Codificación del módulo de buzón de mensajes de usuarios en el servicio web

PBI B12: Codificación del módulo de buzón de mensajes de usuarios en el servicio web	
Sprint: 05	**Prioridad:** 2
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 2
Descripción La codificación de este módulo se realizará con la creación de una ruta mediante la cual se recibirán los datos desde la aplicación. Esta información será ingresada en la base de datos mediante sentencias SQL.	
Validación Capacidad para leer y crear información de mensaje en la base de datos.	

El código del servidor que permite la creación de mensajes se puede observar en la figura 3.16, los cuales serán almacenados en una tabla de mensajes en la base de datos. Cada mensaje tiene asociado identificadores como el identificador del Usuario que recibirá el mensaje y la conversación a la que pertenece.

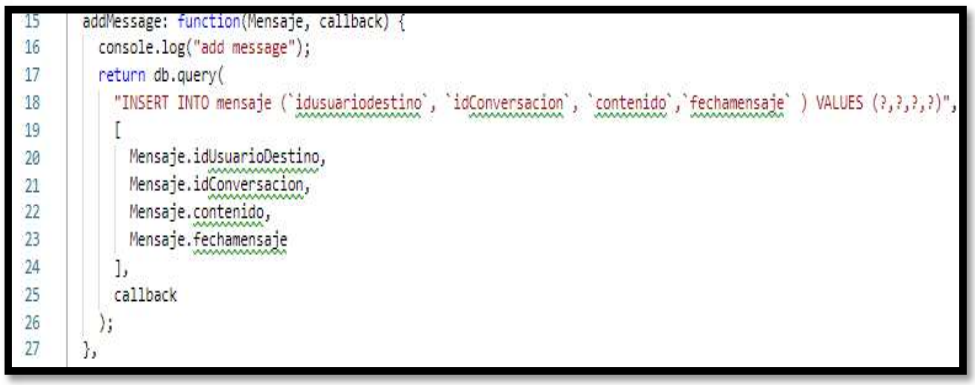


Figura 3.16. Código para guardar mensajes en el servidor.

El código para recuperar el último mensaje de una conversación se puede observar en la figura 3.17. Este método es importante para mostrar este dato al momento de recuperar las conversaciones de un usuario. Además, se ejecuta en conjunto con otros métodos para retornar a la aplicación la información de las conversaciones del usuario y el último mensaje de estas.

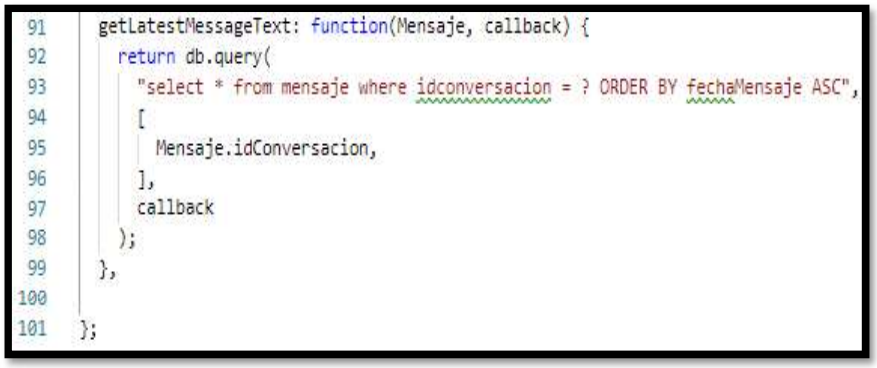


Figura 3.17. Código para recuperar el último mensaje de una conversación en el servidor.

3.2.4. Implementación del módulo de buzón de mensajes en la aplicación móvil.

El módulo de buzón de mensajes en la aplicación móvil provee características para guardar y recuperar los mensajes del usuario en la aplicación móvil, esto asociado al servicio web respectivo. El detalle de implementación se encuentra en la tabla 3.9.

Tabla 3.9. PBI B13: Codificación del módulo de buzón de mensajes de usuarios en la aplicación móvil

PBI B13: Codificación del módulo de buzón de mensajes de usuarios en la aplicación móvil	
Sprint: 05	**Prioridad:** 2
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 2
Descripción La codificación de este módulo será realizada a través de un archivo que permitirá crear y leer la información de un mensaje. Además, permitirán solicitar o enviar información al servicio web utilizando peticiones web.	
Validación Capacidad para leer, crear, eliminar y editar información de mensaje.	

El código para la petición de todas las conversaciones de un usuario desde la aplicación móvil se puede observar en la figura 3.18. Se envía una petición http del tipo POST hacia el servicio web con información para identificar del usuario.

```
145     fetch(
146       constant.SERVER_IP +
147         constant.URL_CONVERSACION +
148         constant.URL_CONVERSACION_GETALL,
149       {
150         method: "post",
151         headers: {
152           "content-Type": "application/json"
153         },
154         body: JSON.stringify({
155           token: tokenStored,
156           idUsuario: idUsuario
157         })
158       }
159     )
```

Figura 3.18. Código para petición de todas las conversaciones para un usuario desde la aplicación móvil

El código para la petición de todos los mensajes de un usuario desde la aplicación móvil se puede observar en la figura 3.19. Se envía una petición http del tipo POST hacia el servicio web con información para identificar la conversación.

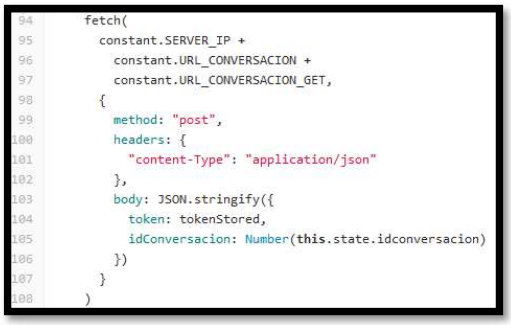


Figura 3.19. Código para petición de todos los mensajes para un usuario desde la aplicación móvil

3.3. Sprint 06 – Implementación de las interfaces de la aplicación cliente e información para pruebas en BDD

3.3.1. Implementación de las interfaces de la aplicación y conexión con los módulos

Las interfaces de la aplicación son las encargadas de presentar y capturar la información del usuario, en este punto se especifica los criterios para crear las interfaces en la aplicación móvil en base a los sketches previamente realizados en la sección 3.7.6. El detalle de esta implementación se muestra en la tabla 3.10.

Tabla 3.10. PBI B14: Codificación de las interfaces en la aplicación móvil y ensamblaje con los módulos.

PBI B14: Codificación de las interfaces en la aplicación móvil y conexión con los módulos.	
Sprint: 06	**Prioridad: 3**
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 3
Descripción La codificación de las interfaces seguirá los sketches implementados en la fase de diseño. Las interfaces deben de permitir completar el flujo planteado en el diseño y además conectarse con el servicio web para consultar y enviar información.	
Validación Aplicación móvil completa con pantallas que permitan completar el flujo	

A continuación de la figura 3.20 a la 3.60, se presentan las interfaces completas de la aplicación móvil. Se detallará el código de ciertas interfaces ya que la mayoría sigue el mismo patrón, pero algunas tienen características que son necesarias de detallar.

La interfaz de carga se puede observar en la figura 3.20, esta se presentará al iniciar la aplicación, tiene un temporizador que le permite cargar a la aplicación estilos y fuentes con las que funcionará la aplicación.

Figura 3.20. Interfaz de Carga de la Aplicación Móvil

La interfaz de inicio se puede observar en la figura 3.21, esta permite al usuario acceder a los demás menús de la aplicación.

En la esquina superior derecha se encuentra el botón que dirige al menú de opciones de perfil. El botón con la descripción “Publicar” dirige a la interfaz para publicar una ruta; el botón con la descripción “Mis Rutas Publicadas” dirige a la interfaz con las rutas publicadas por el usuario; el botón con la descripción “Buscar” dirige a la interfaz para buscar y suscribirse a rutas; el botón con la descripción “Mis Rutas Suscritas” dirige a la interfaz con las rutas suscritas por un usuario; y finalmente el botón con la descripción “Mensajes” dirige a la interfaz del buzón de mensajes.

Figura 3.21, Interfaz de Pantalla de Inicio de la Aplicación Móvil

Implementación de la Interfaz de Pantalla de Inicio de la Aplicación Móvil

La implementación de una interfaz utilizando el framework React Native es realizada utilizando el lenguaje de programación JavaScript, los archivos de código deben de contener la extensión "*.js",* en este caso se utilizará la interfaz de pantalla de inicio de la aplicación para detallar la estructura que tiene todas las interfaces de la aplicación.

Cabe destacar que para React las interfaces, botones, formularios, alertas son en realidad componentes reutilizables en cualquier parte de la aplicación [33].

El código puede ser dividido en las siguientes secciones:

1. **Archivos o Librerías importadas:** En esta sección se agregan las referencias de los módulos que sean necesarios en esta pantalla. En la figura 3.22 se puede observar el código donde:

    - `import React from 'react'`: Es una directiva en la que se utiliza la palabra clave "`import`" para importar un módulo de una librería o archivo; "`from`" es la palabra clave utilizada para definir la librería o archivo desde donde se importa.}; y finalmente "`React`" es un módulo de la librería "`react`".

En este caso se importan varios módulos de las siguientes librerías:

- **react:** Es la librería base utilizada para crear interfaces de usuario utilizando componentes reutilizables que tienen propiedades y pueden mantener un estado [34].
- **react-native:** Es la librería que contiene el framework de desarrollo de aplicaciones móviles [23]. Contiene una lista de módulos base como View, Image, AsyncStorage, entre otros; a partir de los cuales se pueden crear nuevos componentes.
- **react-native-ui-kitten:** Es una librería para react native que contiene un conjunto de componentes comunes con estilos similares [35].
- **Archivos:** Las referencias hacia las carpetas "`../../assets/icons`", "`../../utils/scale`" contienen módulos que complementan la funcionalidad de la aplicación, en este caso "`FontAwesome`" permite integrar caracteres especiales en la interfaz y "`scaleVertical`" es una función que permite escalar los componentes al tamaño de la pantalla. La referencia "`../../componentes/`" contiene archivos con componentes creados que se utilizan a través de toda la aplicación.

```
import React from "react";
import { View, Image, AsyncStorage, Alert } from "react-native";
import { RkButton, RkText, RkStyleSheet } from "react-native-ui-kitten";
import { FontAwesome } from "../../assets/icons";
import { GradientButton } from "../../components/";
import { scaleVertical } from "../../utils/scale";
```

Figura 3.22. Carpetas y Librerías importadas para la Interfaz de Pantalla de Inicio de la Aplicación Móvil

2. **Creación de clase y constructor:** En esta sección se procede a definir el nombre de la clase y de la interfaz, estas características se muestran en la figura 3.23. Todas las interfaces, en realidad son componentes reutilizables y debido a esto deben de heredar de "`React.Component`" [33].

Además, se pueden observar los siguientes elementos:

- **constructor:** Es el constructor de este componente, en esta sección se encuentra la inicialización de las propiedades y el estado del componente.
- **props:** Son las propiedades que recibe el componente desde donde sea creado.
- **state:** Es el estado interno del componente que puede cambiar a lo largo de su ciclo de vida. Permite actualizar la interfaz cuando ha habido algún cambio.

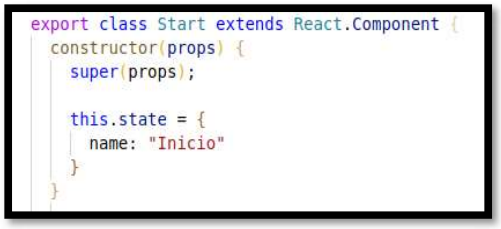


Figura 3.23. Inicialización de clase y constructor para la Interfaz de Pantalla de Inicio de la Aplicación Móvil

3. **Render:** En esta sección de código se procede a renderizar el contenido del componente y generará los elementos de la interfaz de usuario, estas características se muestran en la figura 3.24. Este método será llamado cada vez que el componente cambie su estado.

   Los elementos que se pueden ver en el código son:

   - **Componentes**: La interfaz de inicio está compuesta de componentes como View que permite agrupar otros componentes; `RkButton` es un botón con estilos preestablecidos; `RkText` es un campo de texto con estilos preestablecidos; `Image` es un contenedor para una imagen; `GradientButton` es un botón con estilos preestablecidos.
   - **Propiedades**: Cada componente tiene sus respectivas propiedades, pueden ser eventos, estilos, textos, entre otros.

Por ejemplo, se puede tomar al `RkButton` ubicado en la línea 35 hasta la 43 del código en la figura 3.25. Este componente tiene las siguientes características:

- **Nombre:** RkButton
- **Propiedades:**
    - style: Indica el estilo que se le dará al componente.
    - rkType: Es una propiedad interna del componente y define el tipo de estilo preestablecido, en este caso “social”.
    - onPress: Es un evento que permite llamar a una función cuando el componente es presionado. En este caso se llama a “`this.props.navigation.navigate("Settings")`” que llama a interfaz de opciones de perfil llamada “Settings”.
- **Hijos:** Posee un componente “`RkText`” con sus propiedades.

El resto de los componentes presentes en esta interfaz y en otras de la aplicación poseen la misma estructura y funcionan de manera similar, cada uno con su característica específica.

```
30  render() {
31    return (
32      <View style={styles.screen}>
33        <RkButton
34          style={styles.buttonProfile}
35          rkType="social"
36          onPress={() => {
37            this.props.navigation.navigate("Settings");
38          }}
39        >
40          <RkText rkType="awesome hero accentColor">{FontAwesome.bars}</RkText>
41        </RkButton>
42        <View style={styles.space} />
43        <View style={{ alignItems: "center" }}>
44          <Image
45            style={styles.image}
46            source={require("../../assets/images/logocompartir.png")}
47          />
48          <RkText rkType="h3">Inicio</RkText>
49        </View>
50        <View style={styles.space} />
51        <View style={styles.content}>
52          <View>
53            <GradientButton
54              style={styles.save}
55              rkType="large"
56              text="Publicar"
57              onPress={() => {
58                this.props.navigation.navigate("Publish");
59              }}
60              />
```

Figura 3.24. Parte 1 del método de renderizado para la Interfaz de Pantalla de Inicio de la Aplicación Móvil

```
61                <GradientButton
62                  style={styles.save}
63                  rkType="large4"
64                  text="Mis Rutas Publicadas"
65                  onPress={openPublishList}
66                />
67                <GradientButton
68                  style={styles.save}
69                  rkType="large3"
70                  text="Buscar"
71                  onPress={() => {
72                    this.props.navigation.navigate("Search");
73                  }}
74                />
75                <GradientButton
76                  style={styles.save}
77                  rkType="large5"
78                  text="Mis Rutas Suscritas"
79                  onPress={() => {
80                    this.props.navigation.navigate("SuscribeRoutes");
81                  }}
82                />
83                <GradientButton
84                  style={styles.save}
85                  rkType="large2"
86                  text="Mensajes"
87                  onPress={() => {
88                    this.props.navigation.navigate("ChatList");
89                  }}
90                />
91              </View>
92            </View>
93          </View>
```

Figura 3.25. Parte 2 del método de renderizado para la Interfaz de Pantalla de Inicio de la Aplicación Móvil

4. **Estilos:** Esta sección de código contiene los estilos de los componentes, usualmente son definidos en la parte inferior del archivo. Contienen información del tamaño, justificación, márgenes, posiciones absolutos o relativos, entre otros. El código de los estilos de la interfaz se puede observar en la figura 3.26.

   Por ejemplo, el estilo "`screen`" tiene las siguientes características:

   - **padding:** Indica una cantidad de relleno alrededor del componente, en este caso está fijado en 16.
   - **flex:** Indica el espacio que ocupará un componente como una medida relativa. En este caso está fijado en 1, indicando que ocupará todo el tamaño posible.
   - **justifyContent:** Indica el tipo de justificación que tendrá un componente, en este caso está fijado en "space-around" que signfica que estará centrado con un espacio a su alrededor.
   - **backgroundColor:** Indica el color de fondo del componente, en este caso está fijado en una variable definida en la librería kitten-ui.

El resto de los componentes presentes en esta interfaz y en otras de la aplicación tienen un estilo definido y funcionan de manera similar, cada uno con sus propiedades específicas.

```
let styles = RkStyleSheet.create(theme => ({
  screen: {
    padding: 16,
    flex: 1,
    justifyContent: "space-around",
    backgroundColor: theme.colors.screen.base
  },
  image: {
    marginBottom: 10,
    height: scaleVertical(200),
    resizeMode: "contain"
  },
  content: {
    justifyContent: "space-between"
  },
  save: {
    marginVertical: 8
  },
  buttons: {
    flex: 1,
    position: "absolute",
    flexDirection: "row",
    marginBottom: 24,
    marginHorizontal: 24,
    justifyContent: "space-around",
    alignSelf: "stretch"
  },
```

Figura 3.26. Estilos para la Interfaz de Pantalla de Inicio de la Aplicación Móvil

A través de la interfaz de opciones de perfil se puede ingresar en la interfaz de perfil de usuario, cambio de clave, administración, ayuda y finalmente cerrar sesión en la aplicación, estas características se muestran en la figura 3.27.

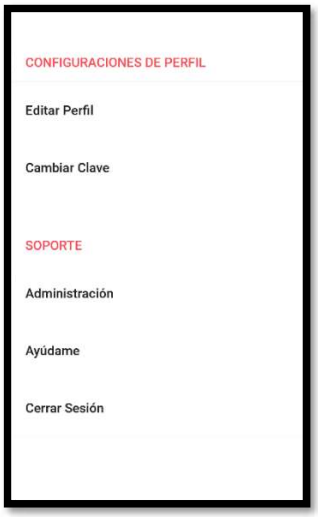


Figura 3.27. Interfaz de Opciones de Perfil de la Aplicación Móvil

La interfaz de perfil de usuario permite ver y editar los datos del usuario. Además, se puede indicar si el usuario es conductor y dirigirse a la interfaz para registrar y editar vehículos, estas características se muestran en la figura 3.28 y 3.29.

Figura 3.28. Parte 1 de la Interfaz de Perfil de Usuario de la Aplicación Móvil

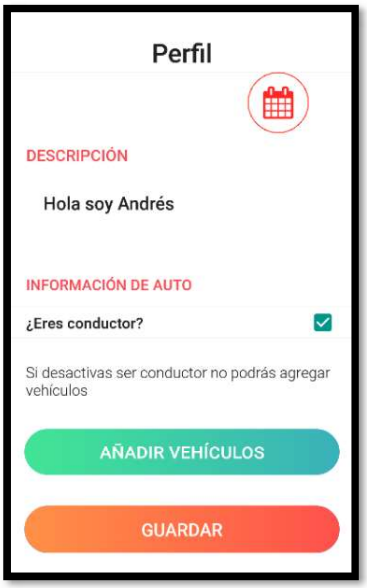


Figura 3.29. Parte 2 de la Interfaz de Perfil de Usuario de la Aplicación Móvil

Para el caso de un usuario conductor, la interfaz de lista de vehículos permite agregar y editar los vehículos uno a uno; esta interfaz se puede observar en la figura 3.30.

Figura 3.30. Interfaz de Lista de Vehículos de Usuario de la Aplicación Móvil

La interfaz de información de un vehículo permite modificar la información de un vehículo como su modelo, marca, tipo, número de asientos, año y número de placa. Además, el usuario puede eliminar el vehículo; esta interfaz se observa en la figura 3.31

Figura 3.31. Interfaz de Información de Vehículo de la Aplicación Móvil

La interfaz de cambio de clave se puede observar en la figura 3.32, a través de esta interfaz el usuario puede cambiar su clave ingresando su clave anterior y la nueva clave. Además, puede acceder a la interfaz para recuperar la clave.

La interfaz de recuperación de contraseña se puede observar en la figura 3.33, si el usuario ingresa su email con el que se registró en la aplicación recibirá un correo con una nueva clave.

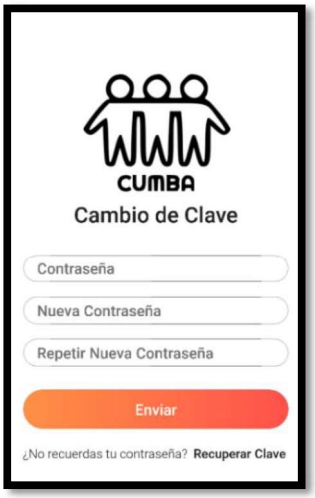


Figura 3.32. Interfaz de Cambio de Clave de la Aplicación Móvil

Figura 3.33. Interfaz de Recuperación de Password de la Aplicación Móvil

La interfaz de publicación de una ruta se puede observar en la figura 3.34 y 3.35, a través de ella se puede ingresar información de la ruta como un título, fecha de inicio, fecha final, hora de salida, hora de llegada, descripción, vehículo y la ruta seleccionando un origen y un destino a través de un mapa.

Figura 3.34. Parte 1 de la Interfaz para la Publicación de una Ruta de la Aplicación Móvil.

Figura 3.35. Parte 2 de la Interfaz para la Publicación de una Ruta de la Aplicación Móvil.

La interfaz para selección de ruta permite seleccionar en un mapa un origen y un destino a través de puntos. Además, los botones inferiores tienen mecanismos de ayuda al usuario;

- El botón con símbolo de “i” tiene información de ayuda del uso de esta interfaz.
- El botón con símbolo de retorno permite reiniciar la información ingresada en la interfaz.
- El botón con símbolo de lupa permite buscar lugares en el mapa.
- El botón con símbolo de disquete permite guardar la ruta escogida.

La interfaz para selección de ruta se puede visualizar en la figura 3.36.

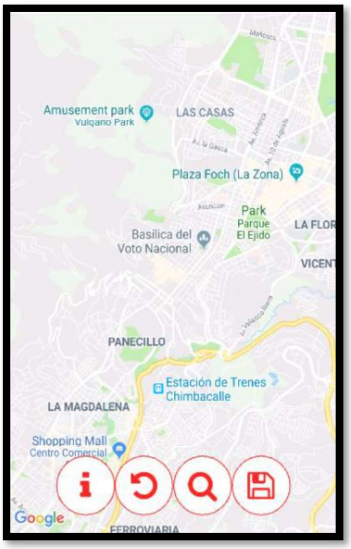


Figura 3.36. Interfaz para Selección de Ruta en Mapa de la Aplicación Móvil.

Implementación de la Interfaz de Selección de Rutan en Mapa de la Aplicación Móvil

La implementación de esta interfaz contiene un nivel mayor complejidad ya que implementa una mayor cantidad de estados a la interfaz de pantalla de inicio detallada en la sección 3.3.1.1.

Dentro de esta interfaz se maneja estado con una mayor cantidad de variables, funciones auxiliares asíncronas y síncronas y renderización de componentes individuales de la interfaz.

El código puede es dividido en las mismas secciones que la interfaz de la pantalla de inicio, pero se agregan ciertas secciones que son opcionales para otros componentes A continuación, se describe el código en sus diferentes secciones:

A. **Archivos o Librerías importadas:** En esta sección se agregan las referencias de los módulos que sean necesarios en esta pantalla. En la figura 3.37 se puede observar el código donde se importan módulos de las siguientes librerías:

    - **react:** Es la librería base utilizada para crear interfaces de usuario utilizando componentes reutilizables que tienen propiedades y pueden mantener un estado [34].
    - **react-native:** Es la librería que contiene el framework de desarrollo de aplicaciones móviles [23]. Contiene una lista de módulos base como View, Image, AsyncStorage, entre otros; a partir de los cuales se pueden crear nuevos componentes.
    - **react-native-ui-kitten:** Es una librería para react native que contiene un conjunto de componentes comunes con estilos similares [35].
    - **react-native-maps:** Es una librería para react-native que permite interactuar con mapas de Google [36]. Incluye componentes para colocar vistas de mapas y marcadores.
    - **react-native-maps-directions:** Es una librería para react-native que se integra con react-native-maps para la generación de rutas en vistas de mapas [37].
    - **react-native-google-places-autocomplete:** Es una librería para react-native [38] que permite conectarse con la API para lugares de Google [39]. Permite generar una lista de lugares a partir de parámetros como ciudad, país, latitud, longitud y otros.
    - **Archivos:** Las referencias hacia las carpetas "`../../assets/icons`", "`../../config/constants`" contienen módulos que complementan la funcionalidad de la aplicación, en este caso "`FontAwesome`" permite integrar caracteres especiales en la interfaz y "`constant`" es una objeto

que contiene variables estáticas de la aplicación. La referencia "`../../componentes/`" contiene archivos con componentes creados que se utilizan a través de toda la aplicación.

```
1  import React from "react";
2  import {
3    StyleSheet,
4    View,
5    Dimensions,
6    Alert,
7    PermissionsAndroid,
8    Modal,
9    AsyncStorage
10 } from "react-native";
11 import { RkText, RkButton, RkStyleSheet } from "react-native-ui-kitten";
12 import { FontAwesome } from "../../assets/icons";
13 import MapView, { Marker } from "react-native-maps";
14 import MapViewDirections from "react-native-maps-directions";
15 import * as constant from "../../config/constants";
16 import { GooglePlacesAutocomplete } from "react-native-google-places-autocomplete";
```

Figura 3.37. Carpetas y Librerías importadas para la Interfaz de Selección de Ruta en Mapa de la Aplicación Móvil.

B. **Creación de clase y constructor:** En esta sección se procede a definir el nombre de la clase y de la interfaz. Todas las interfaces, en realidad son componentes reutilizables y debido a esto deben de heredar de "`React.Component`" [33], esta sección se encuentra en la figura 3.38.

   Además, se pueden observar los siguientes elementos:

   - **constructor:** Es el constructor de este componente, en esta sección se encuentra la inicialización de las propiedades y el estado del componente.
   - **props:** Son las propiedades que recibe el componente desde donde sea creado.
   - **state:** Es el estado interno del componente que puede cambiar a lo largo de su ciclo de vida. Las variables describen la siguiente información:

- **region:** Almacena la información de la latitud, longitud, latitud delta y longitud delta del mapa que está mostrando la interfaz.
- **markers:** Almacenan la información de los puntos del mapa seleccionados por el usuario.
- **coordinates:** Almacenan la información de las coordenadas seleccionadas por el usuario.
- **modalVisible:** Indica si la vista modal para búsqueda de lugares es visible o no. Una vista modal es una interfaz oculta que se muestra al realizar una acción. En este caso al presionar el botón de búsqueda de la interfaz.
- **latitud y longitud:** Variables auxiliares para trabajar con los datos dentro de la interfaz.

- **Binding de métodos:** En esta sección se enlazan los métodos privados de la interfaz para que los componentes hijos puedan utilizarlos.

```
export class Maps extends React.Component {
  constructor(props) {
    super(props);

    this.state = {
      region: {
        latitude: LATITUDE,
        longitude: LONGITUDE,
        latitudeDelta: LATITUDE_DELTA,
        longitudeDelta: LONGITUDE_DELTA
      },
      markers: [],
      coordinates: [],
      modalVisible: false,
      latitude: "",
      longitude: ""
    };

    this.onMapPress = this.onMapPress.bind(this);
    this._renderDirections = this._renderDirections.bind(this);
    this.resetValues = this.resetValues.bind(this);
    this.showInfo = this.showInfo.bind(this);
    this.findLocation = this.findLocation.bind(this);
    this.setModalVisible = this.setModalVisible.bind(this);
    this.saveRoute = this.saveRoute.bind(this);
  }

```

Figura 3.38. Inicialización de clase y constructor para la Interfaz de Selección de Ruta en Mapa de la Aplicación Móvil

C. **Métodos privados:** Los métodos serán utilizados por los componentes para completar la funcionalidad de la interfaz. Entre la figura 3.39 y la 3.43 se puede ver esta sección de código. A continuación, se detalla la implementación de estos métodos:

- **setModalVisible:** El código para el método `setModalVisible`. La función `this.setState` permite actualizar el estado de la interfaz, en este caso recibe como parámetro un booleano indicando si la vista modal para la búsqueda de lugares debe de ser o no visible, este código es visible en la figura 3.39.

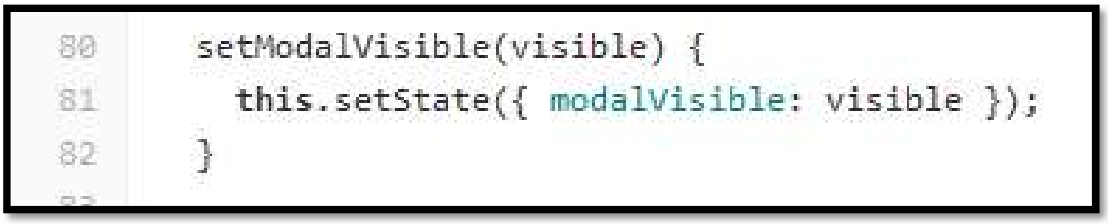
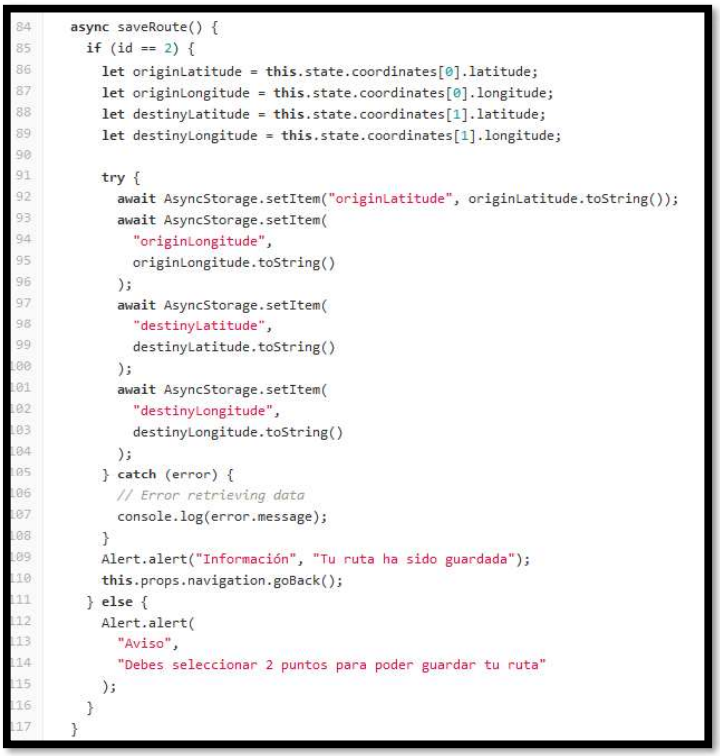


Figura 3.39. Método setModalVisible para la Interfaz de Selección de Ruta en Mapa de la Aplicación Móvil

- **saveRoute:** Este método permite guardar los datos de la ruta seleccionada en la memoria local de la aplicación. Además, es del tipo asíncrono, esto indica que la interfaz no se bloqueará mientras la aplicación realiza este proceso. La figura 3.40 se muestra el código para el método `saveRoute`.
    - **Línea 85:** En esta línea se crea un condicional, el id indica el identificador para el número de puntos seleccionados en el mapa, ya que la ruta tiene un origen y un destino está limitado a 2.

      El condicional indica que si el número de puntos seleccionado llega a 2 se procede a guardar la ruta, en caso contrario muestra indicaciones sobre el número de puntos requeridos para guardar la ruta.



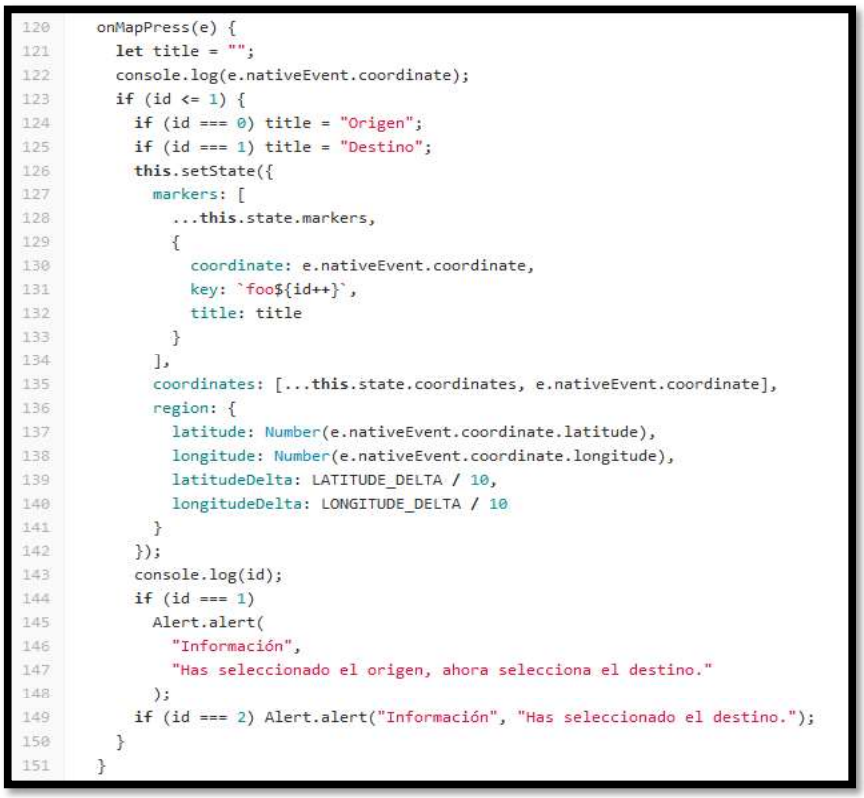
```
84   async saveRoute() {
85     if (id == 2) {
86       let originLatitude = this.state.coordinates[0].latitude;
87       let originLongitude = this.state.coordinates[0].longitude;
88       let destinyLatitude = this.state.coordinates[1].latitude;
89       let destinyLongitude = this.state.coordinates[1].longitude;
90
91       try {
92         await AsyncStorage.setItem("originLatitude", originLatitude.toString());
93         await AsyncStorage.setItem(
94           "originLongitude",
95           originLongitude.toString()
96         );
97         await AsyncStorage.setItem(
98           "destinyLatitude",
99           destinyLatitude.toString()
100        );
101        await AsyncStorage.setItem(
102          "destinyLongitude",
103          destinyLongitude.toString()
104        );
105      } catch (error) {
106        // Error retrieving data
107        console.log(error.message);
108      }
109      Alert.alert("Información", "Tu ruta ha sido guardada");
110      this.props.navigation.goBack();
111    } else {
112      Alert.alert(
113        "Aviso",
114        "Debes seleccionar 2 puntos para poder guardar tu ruta"
115      );
116    }
117  }
```

Figura 3.40. Código del método setModalVisible para la Interfaz de Selección de Ruta en Mapa de la Aplicación Móvil

- **Línea 86 a 89:** Se crean variables auxiliares con la información de la ruta guardadas en el estado de la interfaz.
- **Línea 92 a 109:** Se almacenan localmente los valores de las variables creadas para la longitud y latitud tanto origen como destino de la ruta, y se los encierra en un bloque try-catch para evitar errores. La palabra clave await es utilizada para que el método espere a que una función asíncrona retorne un valor. En este caso el módulo `AsyncStorage` es utilizado para almacenar localmente estos valores.
- **Línea 110 a 116:** Se muestran mensajes indicando el éxito o error de la operación realizada.

- **onMapPress:** Este método será lanzado cada vez que el usuario realice un touch en la pantalla. Además, contiene validaciones para verificar si se ha

seleccionado el origen o el destino de la ruta. En la figura 3.41 se observa el código para el método `onMapPress`.

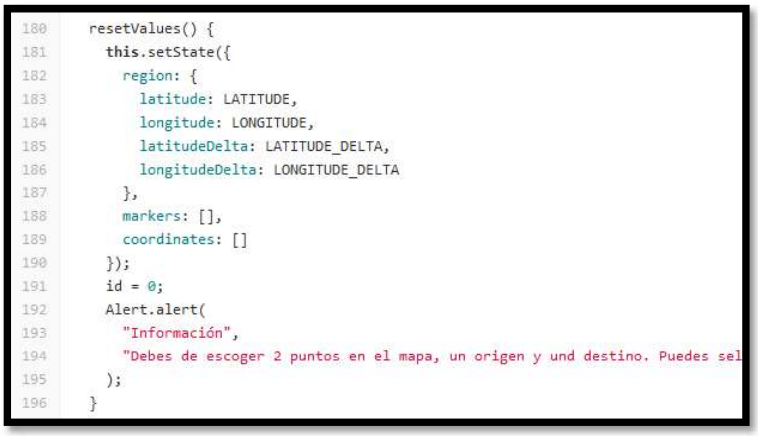


```
120   onMapPress(e) {
121     let title = "";
122     console.log(e.nativeEvent.coordinate);
123     if (id <= 1) {
124       if (id === 0) title = "Origen";
125       if (id === 1) title = "Destino";
126       this.setState({
127         markers: [
128           ...this.state.markers,
129           {
130             coordinate: e.nativeEvent.coordinate,
131             key: `foo${id++}`,
132             title: title
133           }
134         ],
135         coordinates: [...this.state.coordinates, e.nativeEvent.coordinate],
136         region: {
137           latitude: Number(e.nativeEvent.coordinate.latitude),
138           longitude: Number(e.nativeEvent.coordinate.longitude),
139           latitudeDelta: LATITUDE_DELTA / 10,
140           longitudeDelta: LONGITUDE_DELTA / 10
141         }
142       });
143       console.log(id);
144       if (id === 1)
145         Alert.alert(
146           "Información",
147           "Has seleccionado el origen, ahora selecciona el destino."
148         );
149       if (id === 2) Alert.alert("Información", "Has seleccionado el destino.");
150     }
151   }
```

Figura 3.41. Método onMapPress para la Interfaz de Selección de Ruta en Mapa de la Aplicación Móvil.

- **Línea 123 a 125:** Se utilizan condicionales para verificar si el punto seleccionado es el origen o el destino.
- **Línea 126 a 142:** Almacena la información del punto seleccionado en el mapa en el estado de la interfaz.
- **Línea 144 a 149:** Se muestran mensajes informativos sobre el punto a seleccionar en la interfaz.

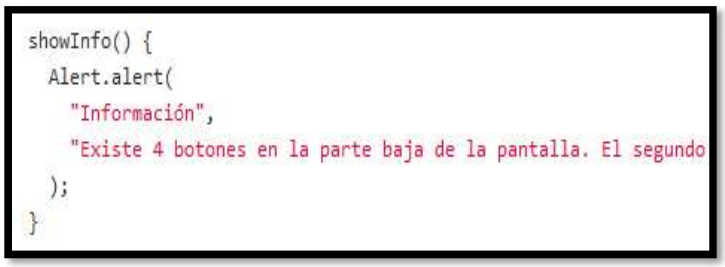
- **resetValues:** Este método será lanzado cada vez que el usuario seleccione el botón de retorno en la interfaz de la aplicación. Su función es la de restablecer a los valores del a los por defecto de la interfaz. En la figura 3.42 se observa el código para el método `resetValues`.



```
180   resetValues() {
181     this.setState({
182       region: {
183         latitude: LATITUDE,
184         longitude: LONGITUDE,
185         latitudeDelta: LATITUDE_DELTA,
186         longitudeDelta: LONGITUDE_DELTA
187       },
188       markers: [],
189       coordinates: []
190     });
191     id = 0;
192     Alert.alert(
193       "Información",
194       "Debes de escoger 2 puntos en el mapa, un origen y und destino. Puedes sel
195     );
196   }
```

Figura 3.42. Código del método resetValues para la Interfaz de Selección de Ruta en Mapa de la Aplicación Móvil.

- **Línea 181 a 191:** Se establecen el estado de la aplicación a los valores por defecto a través del método `this.setState`.
- **Línea 192 a 195:** Muestra un mensaje informativo de ayuda para el usuario.
- **showInfo**: Este método será lanzado cada vez que el usuario seleccione el botón de información en la interfaz de la aplicación. Su función es la de mostrar información de ayuda al usuario. En la figura 3.43 se muestra el código para el método showInfo.



```
showInfo() {
  Alert.alert(
    "Información",
    "Existe 4 botones en la parte baja de la pantalla. El segundo
  );
}
```

Figura 3.43. Método showInfo para la Interfaz de Selección de Ruta en Mapa de la Aplicación Móvil.

D. Render: Esta sección de código puede ser vista en las figuras 3.44 a la 3.48 donde se procederá a renderizar el contenido del componente y generará los elementos

de la interfaz de usuario. Este método será llamado cada vez que el componente cambie su estado.

Los elementos que se pueden ver en el código son:

- **Componentes**: En las figuras 3.44 a 3.48 se encuentran los componentes de la interfaz de inicio como View que permite agrupar otros componentes; `RkButton` es un botón con estilos pre establecidos; `RkText` es un campo de texto con estilos pre establecidos; `Image` es un contenedor para una imagen; `GradientButton` es un botón con estilos pre establecidos; `MapView` es un componente que permite interactuar con mapas y `Marker` es un componente que se inserta dentro de estos mapas; `Modal` es un componente permita ocultar una vista o interfaz y activarla mediante un evento, en este caso aplastar un botón; y finalmente `GooglePlacesAutocomplete` es un componente con una interfaz que permite obtener una lista de lugares o direcciones en base a una cadena de texto.
- **Propiedades**: Cada componente tiene sus respectivas propiedades, pueden ser eventos, estilos, textos, entre otros.

En la figura 3.44 y 3.45 se puede detallar este código:

- **Línea 300:** Se está renderizando las direcciones del mapa a partir de los puntos seleccionados a través del método `_renderDirections`.
- **Línea 303 a 317:** Se agrega el componente `MapView` que mostrará en la interfaz, el mapa. Además, este componente contiene los puntos seleccionados por el usuario y estos son agregados a través de los componentes `Marker`
- **Línea 318 a 357:** Se están agregando los botones de información, búsqueda, retorno y guardado; estos se encuentran en la parte baja de la interfaz que se puede ver en la figura 37.

```
299   render() {
300     let map = this._renderDirections();
301
302     return (
303       <View style={styles.container}>
304         <MapView
305           style={styles.map}
306           region={this.state.region}
307           onPress={this.onMapPress}
308         >
309           {this.state.markers.map(marker => (
310             <Marker
311               title={marker.title}
312               key={marker.key}
313               coordinate={marker.coordinate}
314             />
315           ))}
316           {map}
317         </MapView>
318         <View style={styles.buttonContainer}>
319           <RkButton
320             style={styles.buttonRight}
321             rkType="social"
322             onPress={this.showInfo}
323           >
324             <RkText rkType="awesome hero accentColor">
325               {FontAwesome.info}
326             </RkText>
327           </RkButton>
```

Figura 3.44. Parte 1 del método de renderizado de Selección de Ruta en Mapa de la Aplicación Móvil

```
337           <RkButton
338             style={styles.buttonRight}
339             rkType="social"
340             onPress={() => {
341               this.setModalVisible(!this.state.modalVisible);
342             }}
343           >
344             <RkText rkType="awesome hero accentColor">
345               {FontAwesome.search}
346             </RkText>
347           </RkButton>
348           <RkButton
349             style={styles.buttonRight}
350             rkType="social"
351             onPress={this.saveRoute}
352           >
353             <RkText rkType="awesome hero accentColor">
354               {FontAwesome.save}
355             </RkText>
356           </RkButton>
357         </View>
```

Figura 3.45. Parte 2 del método de renderizado de Selección de Ruta en Mapa de la Aplicación Móvil

En la figura 3.46 se puede detallar el siguiente código:

- **Línea 358 a 359:** Se procede a agregar el componente Modal que se encargará de ocultar y mostrar la interfaz de búsqueda de lugares o direcciones. Se declaran ciertas propiedades como:
    - **animationType:** Define el tipo de animación con la cual se mostrará la vista modal, en este caso del tipo "`slide`".
    - **transparent:** Define si la vista modal será transparente, se encuentra definido como "`false`".
    - **visible:** Define si la vista modal será visible, se encuentra asignado el valor `this.state.visible`; este parámetro por defecto está en "`false`" y cambia al apretar el botón de búsqueda en la interfaz.
    - **onRequestClose:** Es el evento que se lanza cuando la vista modal va a ser cerrada, recibe como parámetro una función que cambia el estado de la variable `modalVisible`.
- **Línea 366 a 372:** Se definen componentes para mostrar la interfaz de búsqueda de lugares y direcciones.

```
358          <Modal
359            animationType="slide"
360            transparent={false}
361            visible={this.state.modalVisible}
362            onRequestClose={() => {
363              this.setModalVisible(!this.state.modalVisible);
364            }}
365          >
366            <View style={[styles.root, { flex: 1 }]}>
367              <View style={[styles.space, { backgroundColor: "#64B5F6" }]} />
368              <View style={{ alignItems: "center", backgroundColor: "#64B5F6" }}>
369                <RkText rkType="h1" style={{ color: "#FFFFFF" }}>
370                  Buscar Sitio
371                </RkText>
372              </View>
```

Figura 3.46. Parte 3 del método de renderizado de Selección de Ruta en Mapa de la Aplicación Móvil.

La interfaz de selección de lugar o dirección en la aplicación móvil se puede observar en la figura 3.47, donde al ingresar cadenas de texto como parámetros de búsqueda se obtiene una lista de lugares o direcciones que tengan similitud. Al seleccionar uno de los resultados se puede redirigir en el mapa hacia el sitio escogido.

En la figura 3.48 se puede detallar el siguiente código:

- **Línea 374 a 381:** Declaración del componente GooglePlacesAutocomplete que se encuentra dentro de la vista modal de la figura 48. Contiene las siguientes propiedades:
    - **placeholder:** Es el texto de por defecto cuando no se ha ingresado una cadena de texto.

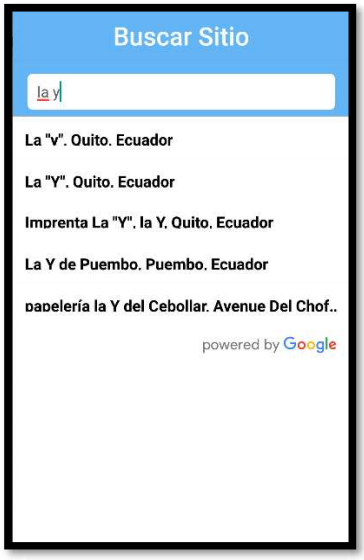


Figura 3.47. Interfaz de Selección de Lugar o Dirección en la Aplicación Móvil.

    - **minLength:** Es el tamaño mínimo de texto antes de realizar una búsqueda.
    - **autoFocus:** Esta propiedad permite ganar el foco de la pantalla cuando está se actualiza.
    - **returnKeyType:** Se define el identificador de retorno para la búsqueda.
    - **listViewDisplayed:** Define con que recurrencia se actualizará la lista.

- **fetchDetails:** Define si se obtiene toda la información de las rutas seleccionadas.
- **renderDesription:** Define qué valor se mostrará en la interfaz al obtener los resultados.

```
374  <GooglePlacesAutocomplete
375    placeholder="Buscar"
376    minLength={2} // minimum length of text to search
377    autoFocus={false}
378    returnKeyType={"search"} // Can be left out for default return key https
379    listViewDisplayed="auto" // true/false/undefined
380    fetchDetails={true}
381    renderDescription={row => row.description} // custom description render
```

Figura 3.48. Parte 4 del método de renderizado de Selección de Ruta en Mapa de la Aplicación Móvil.

En la figura 3.49 se definen las propiedades por las cuales se va a comunicar el componente entre ellas las propiedades que debe de recibir el API de lugares de Google [39]. La propiedad de `key` se encuentra oculta, ya que es una clave única a la que solo debe de tener acceso el desarrollador de la aplicación.

```
423    query={{
424      // available options: https://developers.google.com/places/web-service/autocomplete
425      key: "[redacted]",
426      language: "es", // language of the results
427      types: [
428        "locality",
429        "political",
430        "geocode",
431        "establishments",
432        "address"
433      ], // default: 'geocode'
434      location: "-0.218347, -78.512834",
435      radius: "300000&strictbounds",
436      components: "country:ec"
437    }}
```

Figura 3.49. Parte 5 del método de renderizado de Selección de Ruta en Mapa de la Aplicación Móvil.

E. Estilos: Esta sección de código contiene los estilos de los componentes, usualmente son definidos en la parte inferior del archivo. Contienen información del tamaño, justificación, márgenes, posiciones absolutos o relativos, entre otros. En la figura 3.50 se puede observar el código de los estilos de la interfaz.

```
const styles = RkStyleSheet.create(theme => ({
  container: {
    ...StyleSheet.absoluteFillObject,
    justifyContent: "flex-end",
    alignItems: "center"
  },
  map: {
    ...StyleSheet.absoluteFillObject
  },
  bubble: {
    backgroundColor: "rgba(255,255,255,0.7)",
    paddingHorizontal: 18,
    paddingVertical: 12,
    borderRadius: 20
  },
  latlng: {
    width: 200,
    alignItems: "stretch"
  },
  button: {
    width: 80,
    paddingHorizontal: 12,
    alignItems: "center",
    marginHorizontal: 10
  },
  buttonContainer: {
    flexDirection: "row",
    marginVertical: 20,
    backgroundColor: "transparent"
  },
```

Figura 3.50. Estilos para la Interfaz de Selección de Ruta en Mapa de la Aplicación Móvil.

La interfaz de rutas publicadas por un usuario permite la selección de una ruta publicada y la visualización de su información, esta interfaz puede ser observada en la figura 3.51.

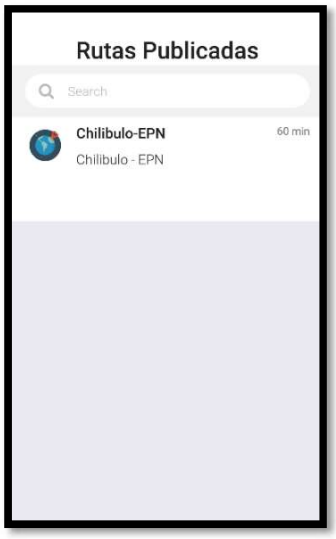


Figura 3.51. Interfaz de Lista de Rutas Publicadas de la Aplicación Móvil.

La interfaz de búsqueda de rutas brinda al usuario la posibilidad de colocar parámetros de búsqueda como fecha de la ruta, hora de salida, origen y destino de la ruta. Si se selecciona

el botón “Buscar” se obtendrá resultados de búsqueda que coincidan con los parámetros de búsqueda, en la figura 3.52 y 3.53 se puede observar esta interfaz y sus propiedades.

Figura 3.52. Parte 1 de la Interfaz de Búsqueda de Rutas de la Aplicación Móvil.

Figura 3.53. Parte 2 de la Interfaz de Búsqueda de Rutas de la Aplicación Móvil.

En la figura 3.54 se puede observar la interfaz de resultados de búsqueda de rutas, esta interfaz es el resultado después de buscar rutas bajo ciertos parámetros. En ella se

presentan los resultados que cumplan con los parámetros. En cada una de las rutas se muestra información básica sobre la ruta como el título, descripción y tiempo de viaje.

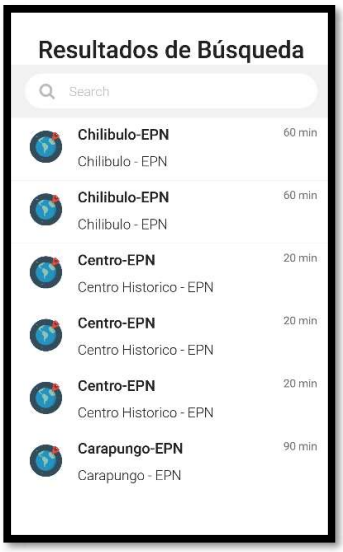


Figura 3.54. Interfaz de Resultados de Búsqueda de Rutas de la Aplicación Móvil.

En la figura 3.55 se puede observar la interfaz de lista de rutas suscritas por un usuario, esta contiene las rutas a las que un usuario se ha suscrito. En cada una de las rutas se muestra información básica sobre la ruta como el título, descripción y tiempo de viaje.

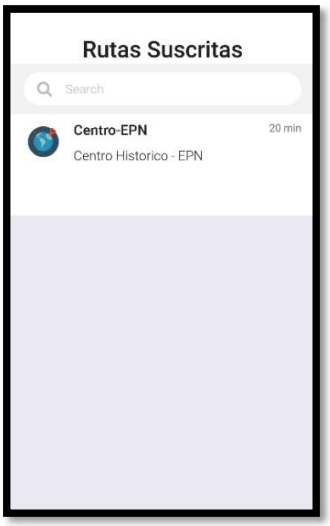


Figura 3.55. Interfaz de Lista de Rutas Suscritas del Usuario de la Aplicación Móvil.

En la figura 3.56 y 3.57 se muestra la interfaz de información de una ruta, a través de esta se puede observar la información de una ruta como su título, fecha de inicio, fecha final, hora de salida, hora de llegada, vehículo, descripción, mapa de la ruta y el perfil del conductor de la ruta.

Figura 3.56. Parte 1 de la Interfaz de Información de una Ruta de la Aplicación Móvil.

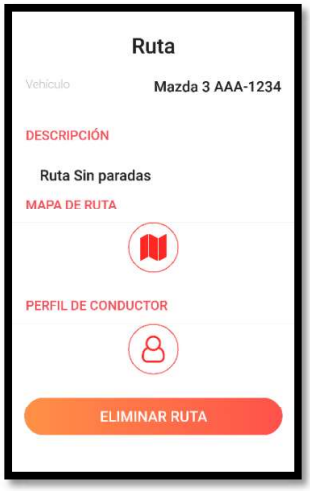


Figura 3.57. Parte 2 de la Interfaz de Información de una Ruta de la Aplicación Móvil.

En la figura 3.58 se observa la interfaz de administración, a través de interfaz se puede obtener la lista de rutas y de usuarios registrados en la base de datos. Además, se puede realizar un CRUD de los usuarios registrados y ver la información de las rutas.

Figura 3.58. Interfaz de Administración de la Aplicación Móvil.

En la figura 3.59 se puede observar la interfaz de lista de conversaciones, a través de esta interfaz un usuario podrá ver las conversaciones con otros usuarios y abrir la interfaz de lista de mensajes. Cada ítem contiene información para identificar a la conversación como el usuario con el que se tiene la conversación y la hora y el contenido del último mensaje recibido.

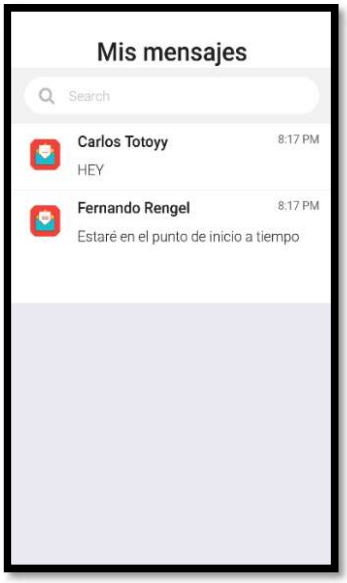


Figura 3.59. Interfaz de lista de conversaciones de la aplicación móvil.

En la figura 3.60 se puede observar la interfaz de lista de mensajes, los usuarios podrán observar todos los mensajes que han intercambiado con otros. Además de poder crear mensajes ingresando su contenido y enviarlo a través del botón en el extremo inferior izquierdo.

Figura 3.60. Interfaz de lista de mensajes de la aplicación móvil.

3.3.2. Creación de información en BDD para pruebas.

Para las pruebas funcionales es necesario la creación de información dentro de la base de datos del prototipo para probar que esta información se almacene y recupere de acuerdo con el funcionamiento descrito para la aplicación. En la tabla 3.11 se describe la creación de información en la BDD para la realización de las pruebas funcionales del prototipo.

Tabla 3.11. PBI B15: Prototipo con información en BDD para pruebas.

PBI B15: Prototipo con información en BDD para pruebas	
Sprint: 06	**Prioridad:** 2
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 1
Descripción El prototipo debe de contener información dentro de la BDD con las mismas características de un escenario real para la realización de las pruebas funcionales.	
Validación Pruebas en el prototipo completo.	

3.3.3. Despliegue de infraestructura en la nube.

Los servicios web y la base de datos del prototipo como requisito deben de estar alojados en un servidor en la nube, para esto es necesario la configuración y despliegue de la máquina virtual en la infraestructura de AWS escogida en el diseño. Las características de este despliegue se encuentran en la tabla 3.12.

Tabla 3.12. PBI B16: Despliegue de infraestructura en la nube

PBI B16: Despliegue de infraestructura en la nube	
Sprint: 06	**Prioridad:** 2
Asignado a: Andrés Baus y Diego Llerena	**Complejidad:** 1
Descripción Los servicios web y la base de datos deben de encontrarse configurados y desplegados en un servidor en la nube provisto por AWS.	
Validación Pruebas en el prototipo completo.	

A continuación de la figura 3.61 a la 3.71, se presentan los pasos realizados para el despliegue de la infraestructura en la nube. Se iniciará con la creación de una cuenta en el servicio de AWS, después se procederá a crear la instancia de máquina virtual EC2[8] y finalmente se procederá a desplegar la base de datos y los servicios web sobre la misma.

Creación de cuenta en servicio de AWS

Se procede a acceder a la página web de AWS [40] en el formulario de creación de una cuenta, se completa la información requerida como se puede observar en la figura 3.61.

Al completar la información personal se requerirá que se confirme el correo con el que se ha creado la cuenta, una vez realizado esto se obtendrá acceso a los servicios de AWS con un nivel gratuito que permite la creación de una máquina virtual EC2 por un año [32].

[8] EC2: Elastic Computing Cloud, es el nombre dado por AWS para el producto que permite la creación de máquinas virtuales de capacidad variable [32].

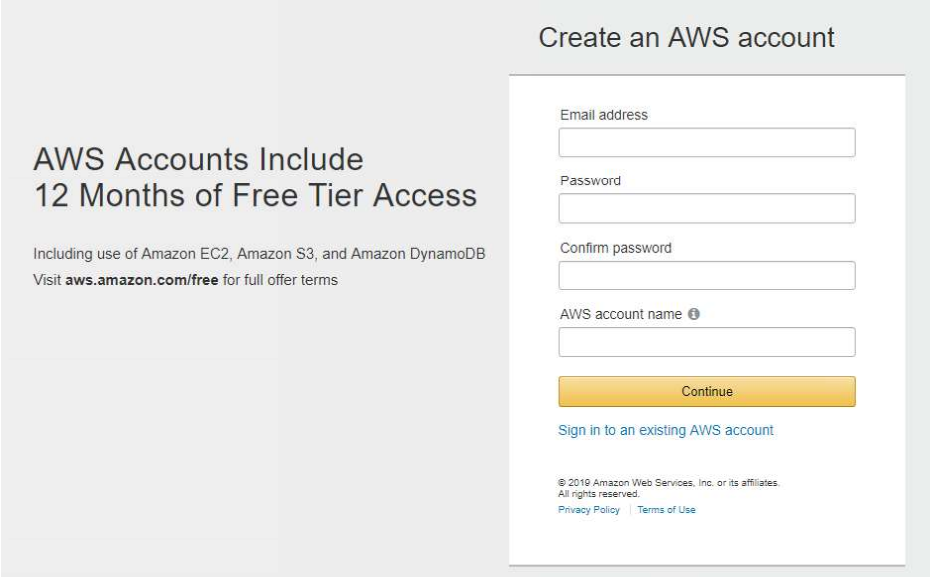


Figura 3.61. Formulario de creación de cuenta en AWS

Creación de instancia de máquina virtual EC2

Se procede a acceder a la consola de administración de AWS que se puede observar en la figura 3.62, y se procede a escoger la opción “EC2”.

Figura 3.62. Consola de administración de AWS

Esta opción abrirá el panel de creación de instancias de máquinas virtuales, que mostrará las opciones con los distintos sistemas operativos. En este caso se seleccionará la opción

del nivel gratis que permite tener una máquina virtual con Ubuntu Server 18.04 LTS [41] cargado e instalado, la información de la instancia se puede observar en la figura 3.63.

Figura 3.63. Información de instancia de máquina virtual

Después de seleccionada la opción se deberá de esperar unos minutos hasta que se concluya la creación de la máquina virtual, al finalizar se redirigirá a la consola de instancias EC2 de AWS con la instancia creada y configurada con una dirección de dominio como se puede observar en la figura 3.64

Figura 3.64. Instancia creada en consola de administración de instancias EC2 de AWS

Despliegue de base de datos

Ya que la máquina virtual se encuentra lista para su uso se procede a acceder al servidor mediante SSH[9] como se puede ver en la figura 3.65, esta conexión permitirá ejecutar comandos para la instalación de los paquetes necesarios para la base de datos y el servicio web.

[9] SSH: Secure Shell, es un protocolo de comunicaciones que permite la conexión remota a otros dispositivos de forma segura sobre redes inseguras mediante el uso de canales encriptados [47].

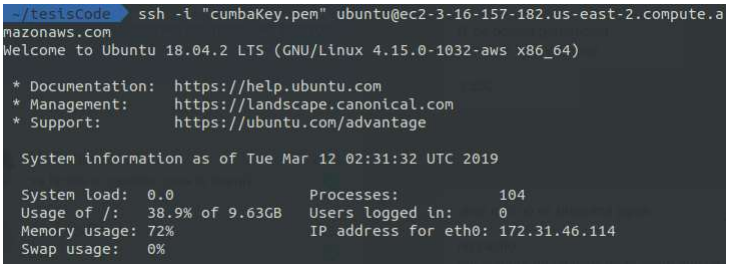


Figura 3.65. Conexión ssh a servicio en la nube

Después de conectarse al servidor se procede a instalar los paquetes necesarios para la instalación de MySQL [6] como motor de base de datos y de phpMyAdmin [42] como interfaz de administración de este servicio. Los comandos necesarios se pueden observar en la figura 3.66.

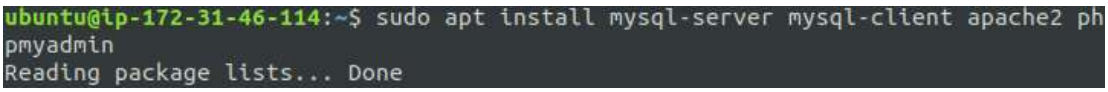


Figura 3.66. Comando de instalación de paquetes para la base de datos

Al finalizar la instalación de estos paquetes se procede a ingresar a la interfaz web de phpMyAdmin y se obtiene la interfaz de inicio de sesión como se observa en la figura 3.67; se procede a ingresar las credenciales por defecto y se ingresa en la aplicación web.

Figura 3.67. Interfaz de inicio de sesión de phpMyAdmin

Al entrar en la interfaz se procede a ejecutar el script de creación de la base de datos, que se encuentra en el anexo A. Cuando el script ha sido ejecutado se podrá observar cómo

se ha creado la estructura en la base de datos del prototipo dentro de los esquemas de phpMyAdmin, esto puede ser observado en la figura 3.68.

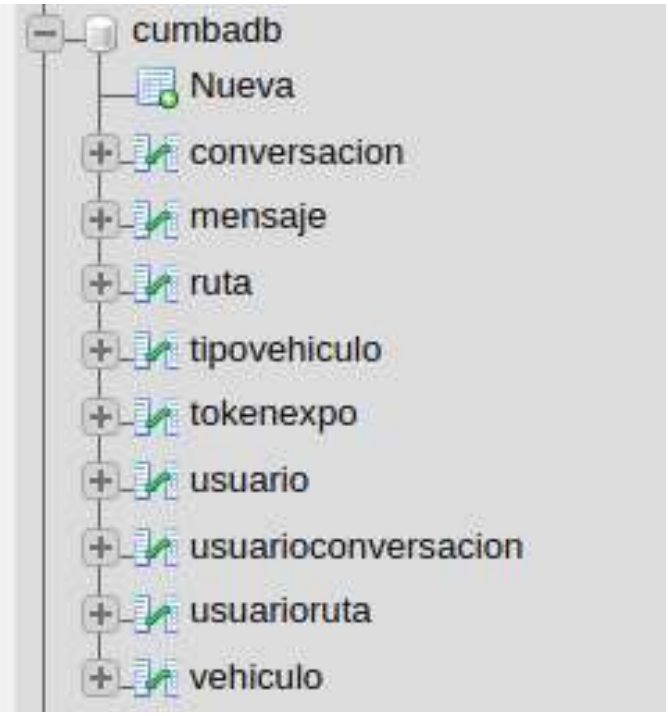


Figura 3.68. Estructura de base de datos creado en servidor en la nube

Despliegue de servicio web

Para el despliegue del servicio web se procede a crear una copia del código creado disponible en el anexo B. Para esto se procede a clonar el repositorio donde se encuentra alojado el código, el comando para realizar esta operación se puede observar en la figura 3.69.

```
ubuntu@ip-172-31-46-114:~$ git clone https://gitlab.com/tesisCumba/server.git
```

Figura 3.69. Clonación del código del servicio web en el servidor en la nube

Una vez que ha finalizado el proceso se podrá observar la estructura de carpetas dentro del servidor en la nube como se observa en la figura 3.70. Para iniciar el servicio se procede a ejecutar el comando `npm start`, que iniciará el servidor.

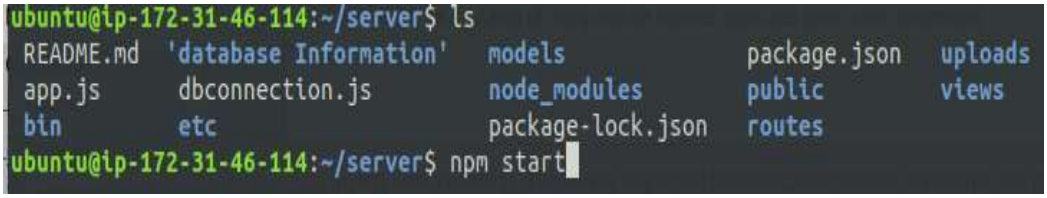


Figura 3.70. Estructura de carpetas de servicio web

Cuando el servidor ha completado su proceso de carga se mostrará un mensaje indicando que se encuentra listo para recibir peticiones como se puede observar en la figura 3.71. De esta manera el servicio web se encuentra listo para escuchar peticiones y enviar respuestas a los clientes.

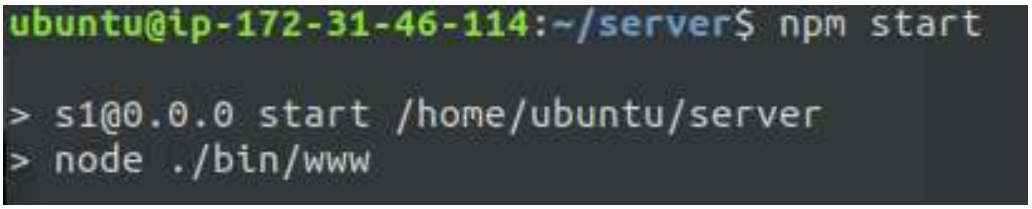


Figura 3.71. Mensaje de servidor al completar proceso de carga

4. RESULTADOS Y DISCUSIÓN

En este capítulo se llevará a cabo la descripción de las pruebas de funcionamiento del prototipo, y su funcionalidad se validará en base al Backlog.

4.1. Pruebas de funcionamiento de la Base de Datos

La funcionalidad de la base de datos será validada a través de los resultados almacenados cuando la aplicación móvil haga peticiones al servicio web y este realice operaciones sobre las tablas de la base.

4.2. Pruebas de funcionamiento del módulo de autenticación del servicio web y la aplicación móvil

La funcionalidad de este módulo será validada a través del inicio de sesión de un usuario en la que se enviará el nombre de usuario y password a través de la aplicación móvil. El servicio web las recibirá y verificará si es un usuario válido; si lo es responderá a la

aplicación con un JWT, este será almacenado en la aplicación para que pueda autenticarse con el servidor en el resto de las peticiones.

En la figura 4.1 se puede observar a el usuario “diego” colocar sus credenciales para iniciar sesión en la aplicación móvil.

Figura 4.1. Inicio de sesión de un usuario a través de la aplicación móvil.

En la figura 4.2 se puede observar el mensaje que obtiene la aplicación cuando las credenciales son validadas por el servidor. En el mensaje se indica que el usuario inició sesión correctamente y se le envía el JWT que va a usar ese usuario en el resto de las peticiones.


```
Object {
  "message": "Login succesfull",
  "token": "eyJhbGciOiJIUzI1NiJ9.ZGllZ28.eE6xOoQy9_vBn-h0Ta8xSNOIgFlauGEJZdRULH4tWu8",
}
```


Figura 4.2. Resultado de servidor al iniciar sesión.

En la figura 4.3 se puede observar el mensaje de bienvenida que recibe el usuario una vez que ha completado el proceso de inicio de sesión.

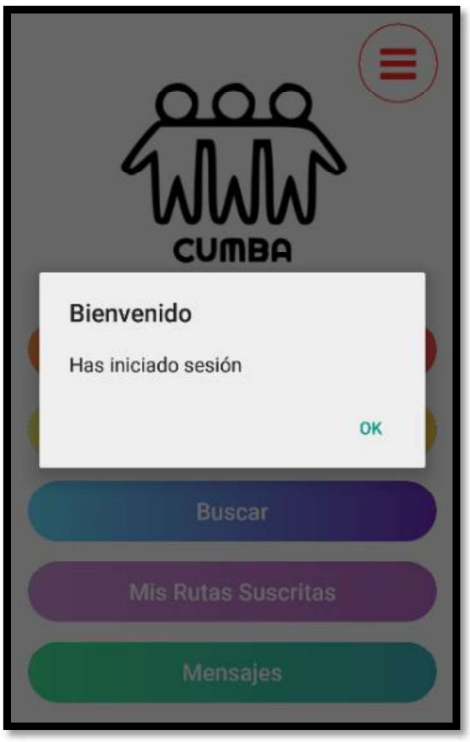


Figura 4.3. Mensaje de inicio de sesión satisfactorio en la aplicación móvil.

4.3. Pruebas de funcionamiento del módulo de administración de usuarios del servicio web y la aplicación móvil

La funcionalidad de este módulo será validada a través de dos interfaces de la aplicación móvil, en las que se podrá leer, crear, editar y eliminar usuarios.

4.3.1. Creación de un usuario

La creación de un usuario se la realiza a través de la interfaz de registro, en esta se enviará una petición conteniendo la información personal del usuario a través de la aplicación móvil. El servicio web la recibirá y agregará este nuevo usuario dentro de la base de datos y responderá con un mensaje de que se ha completado el proceso.

En la figura 4.4 y 4.5 se puede observar la interfaz de registro de usuario con la información de un usuario nuevo. Esta información es enviada en una petición al servicio web al presionar el botón “Registrarse”.

Figura 4.4. Parte 1 de registro de un nuevo usuario desde la aplicación móvil.

Figura 4.5. Parte 2 de registro de un nuevo usuario desde la aplicación móvil.

En la figura 4.6 se observa al nuevo usuario registrado almacenado en la base de datos, el resultado es obtenido a través de la consulta SQL.

```
SELECT * FROM `usuario` WHERE nombreUsuario = 'david'
```

idUsuario	primerNombre	segundoNombre	primerApellido	segundoApellido	fechaNacimiento	telefono
14	David	Luis	Quinteros	Loza	1980-11-25 00:00:00	0986854721

Figura 4.6. Consulta y resultado en la base de datos para obtener información del nuevo usuario registrado.

En la figura 4.7 se puede observar el mensaje de respuesta cuando un usuario ha sido agregado con éxito a través de una petición al servicio web.

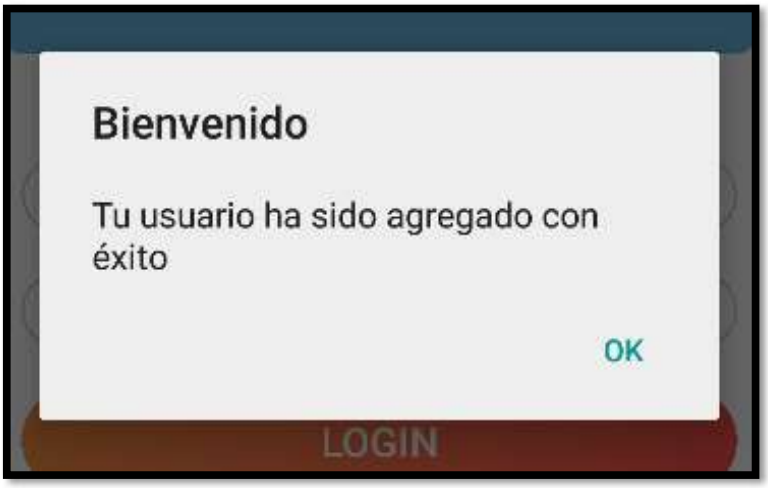


Figura 4.7. Mensaje de respuesta al registrar un usuario en la aplicación.

4.3.2. Edición y lectura de un usuario

La edición y lectura de un usuario se la puede realizar a través de la interfaz de perfil del usuario y de la de administración de usuarios, para este caso se usará la interfaz de administración de usuarios. Esta enviará una petición con el id del usuario al servicio web; este realizará una consulta a la base de datos y retornará la información del usuario a la aplicación móvil.

En la figura 4.8 se puede observar la interfaz de administración de usuarios en las que se obtiene una lista de los usuarios del sistema, para poder realizar la edición y lectura de la información del usuario se seleccionará al usuario “Fernando Rengel”.

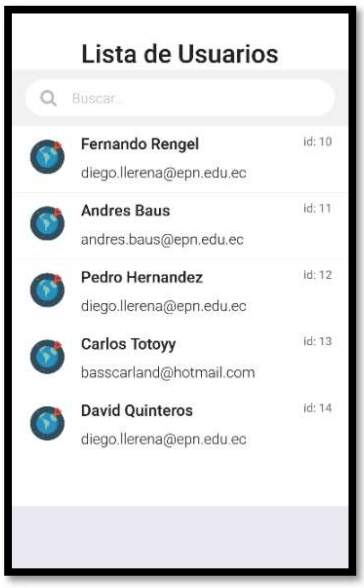


Figura 4.8. Lista de usuarios del sistema en la interfaz de administración de usuarios en la aplicación móvil.

En la figura 4.9 y 4.10 se puede observar la interfaz de administración de información del usuario, esta contiene la información del usuario y a través de ella se puede editar y desactivar el usuario. Además de ver los vehículos registrados por el usuario si existieran.

Figura 4.9. Parte 1 de la interfaz de administración de información del usuario

Figura 4.10. Parte 2 de la interfaz de administración de información del usuario.

En la figura 4.11 se realiza una edición de la información del usuario cambiando su segundo nombre por “Paúl”.

Figura 4.11. Edición de la información del usuario.

En la figura 4.12 se procede a guardar la información editada en el usuario y se recibe un mensaje de notificación diciendo que el usuario ha sido actualizado.

Figura 4.12. Respuesta al actualizar la información del usuario.

idUsuario	primerNombre	segundoNombre	primerApellido	segundoApellido	fechaNacimiento	telefono
10	Fernando	Paúl	Rengel	Llerena	1995-08-11 00:00:00	0998280693

Figura 4.13. Información del usuario editado actualizada en la base de datos.

4.3.3. Desactivación de un usuario

La desactivación de un usuario se la puede realizar a través de la interfaz de administración de usuarios Esta enviará una petición con el id del usuario al servicio web; este realizará una consulta a la base de datos colocando en 1 la columna de estado del usuario.

En la figura 4.14 se puede observar el mensaje de respuesta cuando se ha desactivado un usuario, en este caso se procederá a desactivar al usuario de la sección 4.3.2.

Figura 4.14. Mensaje de desactivación de un usuario.

En la figura 4.15 se observa la información del usuario desactivado en la base de datos, la columna “desactivarUsuario” que indica si el usuario ha sido desactivado se encuentra con el valor 1.

idUsuario	primerNombre	segundoNombre	primerApellido	segundoApellido	fechaNacimiento	telefono	desactivarUsuario
10	Fernando	Paúl	Rengel	Llerena	1995-08-11 00:00:00	0998280693	1

Figura 4.15. Información del usuario desactivado en la base de datos.

4.4. Pruebas de funcionamiento del módulo de gestión de rutas del servicio web y la aplicación móvil

La funcionalidad de este módulo será validada a través de las interfaces de creación de ruta y de información de ruta en la aplicación móvil, en las que se podrá leer, crear y eliminar rutas.

4.4.1. Creación de una ruta

La creación de una ruta se la realiza a través de la interfaz de creación de ruta, en esta se enviará una petición conteniendo la información de la ruta ingresada por el usuario. El servicio web la recibirá y agregará esta nueva ruta dentro de la base de datos y responderá con un mensaje de que se ha completado el proceso.

En la figura 4.16 se puede observar la interfaz para publicar una ruta, cierta información ha sido completada como el título, fecha de inicio y la fecha final. A través del botón debajo de la leyenda de “Selecciona ruta” se puede abrir una nueva interfaz en la que se selecciona la ruta gráficamente.

Figura 4.16. Información de ruta en interfaz de creación de ruta

En la figura 4.17 se puede observar el inicio para la interfaz de selección de mapa de la ruta, se muestra un mensaje de ayuda con cierta información de ayuda para que el usuario pueda seleccionar su ruta.

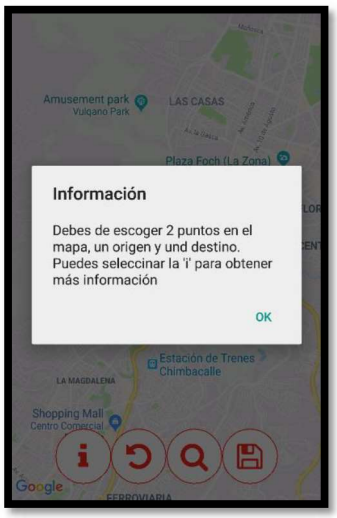


Figura 4.17. Inicio de interfaz de selección de mapa para la ruta

En la figura 4.18 se puede observar una interfaz que permite buscar un sitio y ubicarlo de en el mapa, en este caso se busca “la y” y se obtiene una lista de lugares que coinciden con el patrón de búsqueda.

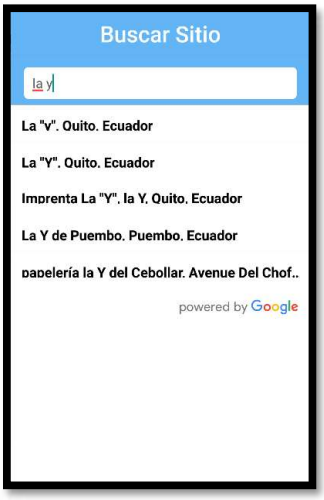


Figura 4.18. Búsqueda de sitio en la selección de mapa de la ruta

En la figura 4.19 se observa la selección del punto de origen en el mapa, la ubicación seleccionada se encuentra en la parada “La Y” del corredor central norte. Ahora se debe

de escoger el punto destino en el mapa, que en este caso es la EPN; en la figura 4.20 se puede observar que al seleccionar el punto destino se genera la ruta en el mapa.

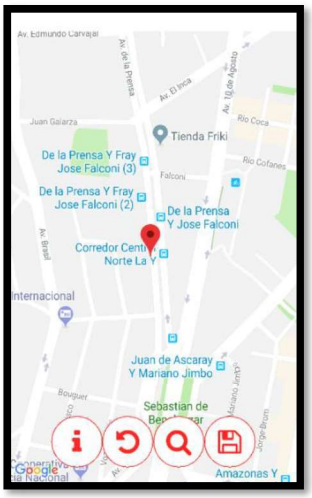


Figura 4.19. Selección de punto origen en interfaz de selección de mapa

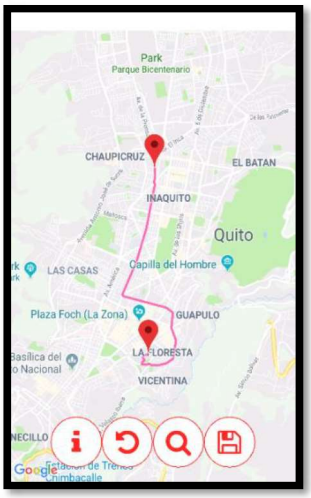


Figura 4.20. Selección de punto destino en interfaz de selección de mapa

La ruta que se ha generado en la figura 4.20 se obtiene utilizando los servicios de ubicación y de lugares de Google [39], este permite generar por defecto una ruta entre dos puntos, en este caso se ha creado la ruta más corta entre los puntos seleccionados.

Una vez completada la selección de la ruta se procede a completar la información restante y se procede a publicar, en la figura 4.21 se puede observar la información completa de la ruta.

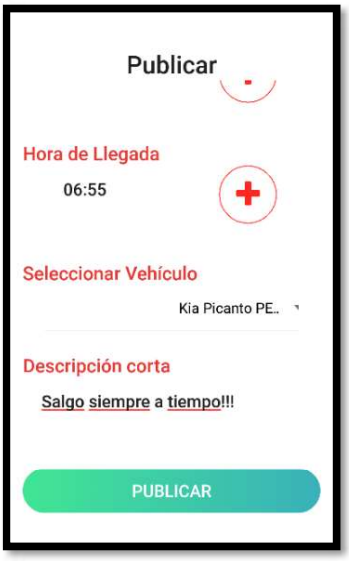


Figura 4.21. Información de ruta a publicar completa.

Al publicar la ruta se obtiene un mensaje de confirmación de que la ruta ha sido publicada, en la figura 4.22 se puede observar el mensaje.

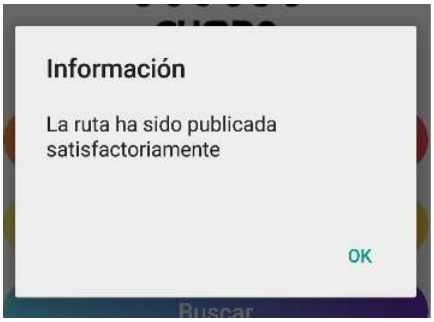


Figura 4.22. Mensaje de publicación de ruta satisfactorio.

En la figura 4.23, se puede observar a la ruta ingresada en la base de datos después de que la aplicación móvil enviara una petición al servicio web y este la enviará con una consulta a la base de datos.

idRuta	fechaInicio	fechaFin	horaSalida	horaLLegada	tiempoViaje	descripcion	titulo	numeroUsuarios
45	2019-02-01 00:00:00	2019-04-19 00:00:00	06:00	06:55	55	Salgo siempre a tiempo!!!	La Y - EPN	0

Figura 4.23. Información de ruta ingresada en base de datos.

4.4.2. Desactivación y lectura de una ruta

La edición y lectura de una ruta se la realiza a través de la interfaz de información de ruta, esta interfaz permitirá observar la información de la ruta publicada y además a través de una petición con el identificador de la ruta se la puede desactivar. El servicio web la recibirá y actualizará la información de la ruta en la base como desactivada.

En la figura 4.24 se puede observar la lista de rutas publicadas por un usuario, la información que se muestra es relevante ya que indica el título, direcciones y tiempo de viaje para cada una de las rutas listadas, además de a través de esta interfaz se va a poder acceder a la interfaz de información de ruta.

Figura 4.24. Lista de rutas publicadas por un usuario.

En la figura 4.25 y 4.26 se puede observar la información de la ruta publicada previamente en la sección 4.4.1. A través de esta interfaz se puede ver la información de la ruta y también desactivarla.

Figura 4.25. Parte 1 de información de la ruta publicada en la interfaz de información de ruta.

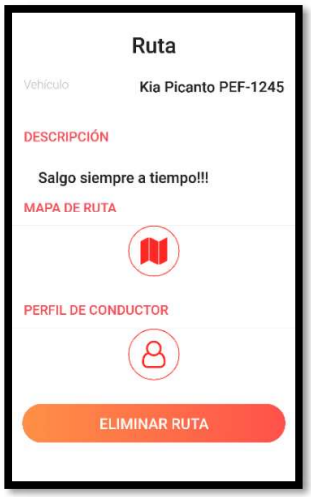


Figura 4.26. Parte 2 de información de la ruta publicada en la interfaz de información de ruta.

Si la ruta es desactivada se envía una petición al servicio web para que actualice la información en la base de datos y cambie el valor de la ruta activa a 0.

idRuta	fechaInicio	fechaFin	horaSalida	horaLLegada	tiempoViaje	descripcion	titulo	numeroUsuarios	rutaActiva
45	2019-02-01 00:00:00	2019-04-19 00:00:00	06:00	06:55	55	Salgo siempre a tiempo!!!	La Y - EPN	0	0

Figura 4.27. Información en la base de datos de la ruta desactivada.

4.5. Pruebas de funcionamiento del módulo de buzón de mensajes del servicio web y la aplicación móvil

La funcionalidad de este módulo será validada a través de las interfaces de lista de conversaciones y de mensaje en la aplicación móvil, en las que se podrá leer y enviar mensajes.

4.5.1. Lista de conversaciones

La lista de conversaciones de un usuario podrá ser observada a través de la interfaz de lista de conversaciones de la aplicación móvil. En la figura 4.28 se puede observar una lista de conversaciones en la que se muestra información útil para identificar la conversación, se muestra el usuario con el que se tiene la conversación y la hora y el contenido del último mensaje recibido. Si el usuario selecciona una conversación, se observará todos los mensajes de esa conversación.

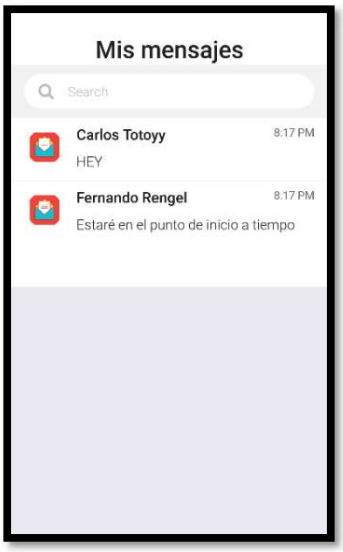


Figura 4.28. Lista de conversaciones con mensajes.

4.5.2. Creación y lectura de mensajes

La conversación de un usuario con otro podrá ser observada a través de la interfaz de lista de mensajes de la aplicación móvil. En la figura 4.29, se puede observar una lista de mensajes de la conversación entre dos usuarios de la aplicación móvil, el usuario podrá leer y crear mensajes.

Figura 4.29. Lista de mensajes.

4.6. Conclusión de Scrum

Una vez concluida la fase de pruebas y en base al Backlog definido en la tabla 2.2 se realiza el checklist respectivo de los ítems del Backlog que fueron completados. Este checklist se puede visualizar en la tabla 4.1.

Tabla 4.1. Checklist de Finalización de Backlog de Scrum

Identificador	Descripción	Estado
A01	Acceso a través de Aplicación móvil Android	Completado
A02	Usuarios con un rol de conductores	Completado
A03	Usuarios con un rol de pasajeros	Completado
A04	Usuarios con un rol de administración	Completado
A05	Rutas de usuario fijas	Completado
A06	Buzón de mensajes para los usuarios	Completado
A07	Registro de usuarios	Completado
B01	Infraestructura de servicios en la nube	Completado
D01	Diseño de diagrama de secuencia	Completado
D02	Selección del proveedor IaaS	Completado
B02	Diseño de la base de datos	Completado
B03	Diseño del API REST	Completado
B04	Diseño de la aplicación móvil	Completado
B05	Codificación de la Base de Datos	Completado
B06	Codificación del módulo de autenticación en el servicio web	Completado
B07	Codificación del módulo de autenticación en la aplicación móvil	Completado
B08	Codificación del módulo de administración de usuarios en el servicio web	Completado
B09	Codificación del módulo de administración de usuarios en la aplicación móvil	Completado
B10	Codificación del módulo de gestión de rutas de usuarios en el servicio web	Completado
B11	Codificación del módulo de gestión de rutas de usuarios en la aplicación móvil	Completado
B12	Codificación del módulo de buzón de mensajes de usuarios en el servicio web	Completado
B13	Codificación del módulo de buzón de mensajes de usuarios en la aplicación móvil	Completado
B14	Codificación de las interfaces en la aplicación móvil y conexión con los módulos.	Completado
B15	Prototipo con información en BDD para pruebas	Completado
B16	Despliegue de infraestructura en la nube	Completado

4.7. Correcciones realizadas

Las correcciones realizadas después de las pruebas de funcionamiento consistieron principalmente en mejorar la experiencia de usuario a través de la aplicación.

Para realizar esto fue necesario la implementación de consultas hacia el servidor con lo cual se recuperó información complementaria a lo ya mostrado en las interfaces de administración de usuarios, lista de rutas y lista de mensajes, estas interfaces se encuentran en las figuras 4.8, 4.24 y 4.28. Esta información complementaria permite conocer a mayor detalle los elementos mostrados en las interfaces de usuarios, rutas y mensajes; con esto mejorando la experiencia de usuario en la aplicación. No se requirió agregar nuevos métodos en el servicio, sino que se reutilizaron métodos previamente creados para otras interfaces,

Como corrección adicional se agregó la columna de *puntos* en la entidad usuario de la base de datos, esta columna se puede visualizar en la figura 2.2. Esta columna guarda un histórico de puntos para los usuarios que usen la aplicación bajo el rol de conductor y que hayan publicado una ruta. Se determinó que por cada vez que un usuario publique una ruta, este recibirá la cantidad de 5 puntos.

Las correcciones requeridas fueron resueltas, estas permitieron que se depuraran los problemas encontrados durante las pruebas de funcionamiento.

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- Al concluir el presente Proyecto de Titulación se tiene un prototipo compuesto de un servicio web y de una aplicación móvil que permite gestionar movilidad compartida. El prototipo brinda una interfaz gráfica que permite publicar y administrar rutas por parte de conductores y la suscripción de pasajeros que puedan beneficiarse de estas para llegar a sus respectivos destinos comunes. Es importante recalcar que todos los datos almacenados a través de este prototipo se encuentran disponibles en un servicio de almacenamiento en la nube.

- Como parte del presente Proyecto de Titulación se analizaron diferentes fundamentos técnicos y de desarrollo de software, lo cual propició la obtención de nuevas habilidades. Esto se debe a que se revisaron conceptos como arquitectura de capas, servicios web, computación en la nube; además, se investigaron y usaron las siguientes tecnologías: JavaScript, NodeJS, Express, MySQL y React Native. Todo esto en combinación con los conocimientos previos en el área de programación permitieron la elaboración de este Proyecto de Titulación.

- El diseño e implementación de este prototipo fue realizado utilizando como base la metodología SCRUM, donde cada una de las actividades propuestas en el cronograma de trabajo pudieron ser categorizadas en base a una prioridad y dificultad de implementación. Gracias a esta categorización, el cronograma principal fue implementado en base a agrupación de los ítems que se encontraban en la misma categoría, lo cual permitió que el prototipo sea desarrollado y ejecutado a tiempo.

- El uso de los recursos en la nube provistos por Amazon Web Services (AWS) permitieron que gratuitamente los servicios web y la base de datos implementados se encuentren disponibles 24 horas, evitando gastos operativos, brindando escalabilidad para futuros despliegues o actualizaciones, una consola de administración de los recursos consumidos por el servicio y una completa independencia de la aplicación móvil presente en los dispositivos móviles de cada usuario, pero con un límite de uso de un año en la que está disponible la capa gratuita.

- Las pruebas realizadas en la aplicación confirmaron el correcto funcionamiento de los módulos de la aplicación móvil, al estar cada módulo consumiendo uno o varios servicios web y las operaciones fueron resueltas o retornan la información esperada.

5.2. Recomendaciones

- Para el desarrollo de servicios web se recomienda que la codificación se estructure con un modelo de clases estáticas e interfaces, donde las interfaces se constituyen como las rutas de acceso al servicio y las clases por otro lado procesan la información e interactúan con la base de datos directamente.

- Si se busca mejorar el rendimiento de la aplicación para futuros prototipos, se sugiere la utilización del lenguaje de programación Java en lugar del desarrollo en React Native. Esto obedece a que java es el lenguaje nativo para el desarrollo en Android por lo que las aplicaciones muestran una mejor capacidad de respuesta.

- Al manejar llamadas secuenciales entre métodos en la capa de lógica del negocio, se debe manejar cada método como una llamada asíncrona, donde el retorno de dicho método marque el inicio del método siguiente. Se recomienda también capturar posibles errores como una respuesta o código de error en cada método ejecutado.

- Se sugiere la revisión de las características ofertadas por los proveedores de servicios en la nube, ya que algunos no soportan el despliegue de todas las tecnologías en su infraestructura. En este caso al evaluar los proveedores Amazon (AWS) y Microsoft (Microsoft Azure), que brindan servicios gratuitos para estudiantes, AWS permitía levantar servicios desarrollados en Node.JS con gran facilidad y ganar en escalabilidad respecto a lo ofertado por Azure.

- En el caso de Amazon Web Services (AWS) es importante revisar las estadísticas que proporciona la consola de administración de instancias de EC2, ya que es necesario mantener un monitoreo continuo de los límites establecidos por el proveedor de la infraestructura en la nube para el uso de máquinas virtuales, ya que, si se excede el plan utilizado, los costos de operación pueden exceder el presupuesto.

6. REFERENCIAS

[1] M. M. Belén, «Quito cuenta con 17 contraflujos viales,» Agosto 2017. [En línea]. Available: http://www.elcomercio.com/actualidad/quito-contraflujos-vias-transporte-movilidad.html. [Último acceso: 02 Septiembre 2018].

[2] A. y. P. A. y. C. S. Remache, «Análisis de la aplicación del pico y placa en la ciudad de Quito,» Junio 2017. [En línea]. Available: http://www.journaluidegye.com/magazine/index.php/innova/article/view/300/309. [Último acceso: 02 Septiembre 2018].

[3] S. Perez, «How carpooling will save the worldQuito cuenta con 17 contraflujos viales,» Septiembre 2017. [En línea]. Available: http://www.conservationmagazine.org/2014/09/how-carpooling-will-save-the-world. [Último acceso: 02 Septiembre 2018].

[4] Enciclopedia Britannica, "Britannica," [Online]. Available: https://www.britannica.com/technology/database. [Accessed 27 01 2019].

[5] Encyclopedia Britannica, «Encyclopedia Britannica,» [En línea]. Available: https://www.britannica.com/technology/SQL. [Último acceso: 28 01 2019].

[6] MySQL, «MySQL Documentation,» 15 01 2019. [En línea]. Available: https://dev.mysql.com/doc/. [Último acceso: 27 01 2019].

[7] ANSI/ISO, «ANSI,» [En línea]. Available: https://www.ansi.org/standards_activities/iso_programs/aic. [Último acceso: 28 01 2019].

[8] ANSI/ISO, «International Organization for Standardization,» [En línea]. Available: https://www.iso.org/standard/34132.html. [Último acceso: 28 01 2019].

[9] Oracle Corporation, «Oracle,» [En línea]. Available: https://www.oracle.com/corporate/. [Último acceso: 28 01 2019].

[10] Open Source, «Open Source Initiative,» [En línea]. Available: https://opensource.org/. [Último acceso: 28 01 2019].

[11] Universidad de Sevilla, «El modelo de datos relacional,» [En línea]. Available: http://www.cs.us.es/cursos/bd-2001/temas/modelo_relacional.html. [Último acceso: 19 02 2019].

[12] IBM, «Arquitecturas de tres niveles,» [En línea]. Available: https://www.ibm.com/support/knowledgecenter/es/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/covr_3-tier.html. [Último acceso: 11 03 2019].

[13] Microsoft, «Layered Application Guidelines,» [En línea]. Available: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658109(v=pandp.10). [Último acceso: 20 02 2019].

[14] IBM Cloud, «IaaS, PaaS and SaaS – IBM Cloud service models,» [En línea]. Available: https://www.ibm.com/cloud/learn/iaas-paas-saas. [Último acceso: 20 02 2019].

[15] IBM, «What is a Web Service?, IBM,» 17 02 2019. [En línea]. Available: https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.webservices.doc/concepts/dfhws_definition.html. [Último acceso: 17 02 2019].

[16] IBM, «What is WSDL?, IBM,» 17 02 2019. [En línea]. Available: https://www.ibm.com/support/knowledgecenter/en/SSTTDS_11.0.0/com.ibm.etools.mft.doc/ac34640_.htm. [Último acceso: 17 02 2019].

[17] IBM, «What is SOAP?,» 27 02 2019. [En línea]. Available: https://www.ibm.com/support/knowledgecenter/en/SSGMGV_3.1.0/com.ibm.cics.ts31.doc/dfhws/concepts/soap/dfhws_overview.htm. [Último acceso: 27 02 2019].

[18] JSON, «JSON,» 17 02 2019. [En línea]. Available: https://www.json.org/. [Último acceso: 17 02 2019].

[19] Developers Android, «Platform Architecture,» 15 01 2019. [En línea]. Available: https://developer.android.com/guide/platform/. [Último acceso: 22 01 2019].

[20] Developer.com, «Understanding the Android Platform Architecture,» 04 Marzo 2016. [En línea]. Available: https://www.developer.com/ws/android/understanding-the-android-platform-architecture.html. [Último acceso: 22 Enero 2019].

[21] O'Reilly, «Chapter 1. What Is React Native?,» 16 08 2018. [En línea]. Available: https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html. [Último acceso: 22 01 2019].

[22] Hackernoon, «Understanding the React Native bridge concept,» 22 Junio 2018. [En línea]. Available: https://hackernoon.com/understanding-react-native-bridge-concept-e9526066ddb8. [Último acceso: 22 Enero 2019].

[23] Facebook, «React Native,» 15 01 2019. [En línea]. Available: https://github.com/facebook/react-native. [Último acceso: 23 01 2019].

[24] J. Sutherland, «Scrum Handbook,» 28 Abril 2010. [En línea]. Available: https://www.researchgate.net/publication/301685699 . [Último acceso: 23 01 2019].

[25] K. S. Jeff Sutherland, «Scrum Guide,» Julio 2013. [En línea]. Available: https://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf. [Último acceso: 23 Enero 2019].

[26] Microsoft, «Why did we build Visual Studio Code?,» 15 Enero 2019. [En línea]. Available: https://code.visualstudio.com/docs/editor/whyvscode. [Último acceso: 23 Enero 2019].

[27] Genymotion, «Genymotion,» 15 Enero 2019. [En línea]. Available: https://www.genymotion.com/. [Último acceso: 01 Enero 2019].

[28] Expo, «Expo Documentation,» 15 Enero 2019. [En línea]. Available: https://docs.expo.io/versions/latest/. [Último acceso: 23 Enero 2019].

[29] Gravit Designer, «Gravit Designer,» 01 Febrero 2019. [En línea]. Available: https://www.designer.io/#interface. [Último acceso: 12 Febrero 2019].

[30] Teatro Abadía, «Diagramas del UML,» Teatro de la Abadìa, Madrid, 2015.

[31] Mozilla, «Methods,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Methods. [Último acceso: 08 04 2019].

[32] Amazon, «AWS Free Tier,» 01 Marzo 2019. [En línea]. Available: https://aws.amazon.com/free/. [Último acceso: 03 Marzo 2019].

[33] Facebook, «React Documentation,» 15 Enero 2019. [En línea]. Available: https://reactjs.org/docs/getting-started.html. [Último acceso: 29 Enero 2019].

[34] Facebook, «React,» 15 Enero 2019. [En línea]. Available: https://reactjs.org/. [Último acceso: 01 Enero 2019].

[35] Akveo, «Kitten UI,» 19 Diciembre 2017. [En línea]. Available: https://akveo.github.io/react-native-ui-kitten/. [Último acceso: 29 Enero 2019].

[36] React Community, «React Native Maps,» 01 Enero 2019. [En línea]. Available: https://github.com/react-native-community/react-native-maps. [Último acceso: 01 Febrero 2019].

[37] Bramus, «React Native Maps Directiosn,» 15 Enero 2019. [En línea]. Available: https://github.com/bramus/react-native-maps-directions. [Último acceso: 01 Febrero 2019].

[38] F. Safi, «React Native Google Places Autocomplete,» 23 Octubre 2018. [En línea]. Available: https://github.com/FaridSafi/react-native-google-places-autocomplete. [Último acceso: 01 Febrero 2019].

[39] Google, «Google Maps Platform Documentation,» 15 Enero 2019. [En línea]. Available: https://developers.google.com/maps/documentation/. [Último acceso: 01 Febrero 2019].

[40] Amazon, «Create an AWS account,» AWS, 16 Enero 2019. [En línea]. Available: https://portal.aws.amazon.com/billing/signup?nc2=h_ct&src=header_signup&redirect_url=https%3A%2F%2Faws.amazon.com%2Fregistration-confirmation#/start. [Último acceso: 11 Marzo 2019].

[41] Canonical, «Ubuntu Server,» 03 Marzo 2019. [En línea]. Available: https://www.ubuntu.com/server. [Último acceso: 11 Marzo 2019].

[42] phpmyadmin, «phpmyadmin,» 16 Enero 2019. [En línea]. Available: https://www.phpmyadmin.net/. [Último acceso: 11 Marzon 2019].

[43] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.

[44] W3Schools, «XML, W3Schools,» 17 02 2019. [En línea]. Available: https://www.w3schools.com/xml/xml_whatis.asp. [Último acceso: 17 02 2019].

[45] JavaScript, «JavaScript Resources,» [En línea]. Available: https://www.javascript.com/resources. [Último acceso: 17 02 2019].

[46] M. O. &. A. Ame, «Ride Share,» 23 Enero 2018. [En línea]. Available: http://ridesharechoices.scripts.mit.edu/home/wp-ontent/papers/APA_TPD_Webinar_Aug2010.pdf. [Último acceso: 02 Septiembre 2018].

[47] SSH, «SSH,» 04 Marzo 2019. [En línea]. Available: https://www.ssh.com/ssh/. [Último acceso: 11 Marzo 2019].

[48] J. Jorba, «Open Access,» 26 Marzo 2017. [En línea]. Available: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/61265/1/Administraci%C3%B3n%20avanzada%20del%20sistema%20operativo%20GNU_Linux_M%C3%B3dulo1_El%20n%C3%BAcleo%20Linux.pdf. [Último acceso: 08 Abril 2019].

7. ANEXOS

ANEXO I. Script de la Base de Datos.

ANEXO II. Código del Servidor.

ANEXO III. Código de la Aplicación Móvil.

ORDEN DE EMPASTADO