UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE QUITO

CARRERA DE INGENIERÍA DE SISTEMAS

GEOLOCALIZACIÓN DE RUTAS DE BUSES EN TIEMPO REAL: CASO DE ESTUDIO CIUDAD DE OTAVALO

Trabajo de titulación previo a la obtención del

Título de Ingeniero de Sistemas

AUTOR: Auki Santiago Díaz Concha

TUTOR: Julio Ricardo Proaño Orellana

Quito – Ecuador

2022

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Auki Santiago Díaz Concha con documento de identificación N° 1004190185; manifiesto que:

Soy el autor y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir y publicar de manera total o parcial el presente trabajo de titulación.

Quito, 07 de marzo de 2022

Atentamente,

Auki Santiago Díaz Concha

1004190185

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Auki Santiago Díaz Concha con documento de identificación N° 1004190185, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Artículo Académico: "Geolocalización de Rutas de Buses en tiempo Real: Caso de estudio Ciudad de Otavalo", el cual ha sido desarrollado para optar por el título de: Ingeniero de Sistemas, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedido anteriormente.

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

Quito, 07 de marzo de 2022

Atentamente,

Auki Santiago Díaz Concha

1004190185

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Julio Ricardo Proaño Orellana con documento de identificación N° 0103909412, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: GEOLOCALIZACIÓN DE RUTAS DE BUSES EN TIEMPO REAL: CASO DE ESTUDIO CIUDAD DE OTAVALO, realizado por Auki Santiago Díaz Concha con documento de identificación N° 1004190185, obteniendo como resultado final el trabajo de titulación bajo la opción Articulo Académico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Quito, 07 de marzo de 2022

Atentamente,

Ing. Julio Ricardo Proaño Orellana, PhD

0103909412

AGRADECIMIENTO

Quiero expresar mi más grande gratitud a mi familia, por su apoyo incondicional y estar siempre presentes en todo momento. Han sido mi gran fortaleza para continuar y culminar con la carrera, y lo seguirán siendo para lograr cumplir con todos mis objetivos.

De igual me gustaría expresar mi más sincero agradecimiento al PhD. Julio Proaño, quien fue el principal colaborador en todo el proceso, y su orientación, conocimiento, enseñanza y paciencia aportaron al desarrollo de este proyecto.

Díaz Concha Auki Santiago

Artículo Científico / Scientific Paper

GEOLOCALIZACIÓN DE RUTAS DE BUSES EN TIEMPO REAL: CASO DE ESTUDIO CIUDAD DE OTAVALO

GEOLOCATION OF BUS ROUTES IN REAL TIME: CASE STUDY CITY OF OTAVALO

Auki Díaz[1], Julio Proaño[2]

Resumen

Un buen sistema de transporte público es uno de los pilares para el desarrollo económico y social de las ciudades. Las personas utilizan este servicio para trasladarse a sus trabajos y realizar varias actividades diariamente y por esta razón se requieren que el servicio y su información se encuentre disponible y al alcance de todo el público de una manera efectiva. Por otro lado, las aplicaciones móviles nos ofrecen la posibilidad de obtener información de eventos como la ubicación de autobuses en tiempo real. Sin embargo, este problema implica el uso y la coordinación de varias tecnologías. En este trabajo se plantea el desarrollo de un prototipo de aplicación para que los usuarios obtengan la posición de un autobús en tiempo real. Este prototipo fue evaluado con la empresa de bus en la ciudad de Otavalo. Los resultados muestran un desempeño satisfactorio con la utilización simultánea de varios usuarios, además presenta un bajo costo de implementación por parte de la empresa de transporte público.

Palabras clave: Aplicativo móvil, Geolocalización, Transporte Público

Abstract

A good public transport system is one of the pillars for the economic and social development of cities. People use this service to move to their jobs and perform various activities daily and for this reason it is required that the service and its information is available and available to the public in an effective way. On the other hand, mobile applications offer us the possibility of obtaining information on events such as the location of buses in real time. However, this problem involves the use and coordination of various technologies. In this work, the development of an application prototype is proposed so that users obtain the position of a bus in real time. This prototype was evaluated with the bus company in the city of Otavalo. The results show a satisfactory performance with the simultaneous use of several users, in addition to presenting a low cost of implementation by the public transport company.

Keywords: Mobile Application, Geolocation, Public Transport.

[1] Estudiante de Ingeniería de Sistemas - Universidad Politécnica Salesiana, Egresado – UPS – sede Quito. Autor para correspondencia: adiazc6@est.ups.edu.ec

[2] Doctorado en tecnologías informáticas avanzadas, cuarto nivel, Profesor de Ingeniería de Sistemas – UPS sede Quito. Email: jproanoo@ups.edu.ec

1. Introducción

El transporte público juega un papel importante en el diario vivir de las personas. La gente lo usa para diversos fines, como el trabajo, la escuela y el uso personal. Sin embargo, el transporte privado ha tenido su auge en los últimos años. Según datos de la Asociación de Empresas Automotrices del Ecuador, marzo fue el mejor mes de este período, con 9,787 autos vendidos, un aumento del 174% con respecto al mismo período en el que estalló la epidemia de Ecuador en 2020 y las ventas se desplomaron solo a 3,573 unidades.

Una de las razones por las que los usuarios prefieren el transporte privado al público es por el uso de aplicaciones que muestran el tiempo de llegada y la ubicación real del automóvil, esto brinda más confianza y seguridad a los pasajeros.

La hora de llegada del autobús es una información importante para la mayoría de los pasajeros. El largo tiempo de espera del autobús a menudo hace que los pasajeros se sientan frustrados y no estén dispuestos a tomar el autobús. Por esta razón en diferentes ciudades del país como Ibarra, Quito, Guayaquil, etc., y también a nivel Internacional en varios países como Brasil, Estados Unidos, Australia e Inglaterra se ha decidido implementar el rastreo de bus vía Sistema de Posicionamiento Global de los dispositivos móviles de los conductores de autobús tal y como se lo hace en el transporte privado como Uber, InDrive, Cabify, etc.

Otro de los problemas del transporte público es no exponer de manera eficiente sus horarios y rutas de movilización, debido a esto los pasajeros no llegan a tiempo o esperan demasiado en las paradas a la espera del autobús que los traslade a su destino.

Como alternativa de solución al problema, diferentes autores encuentran una respuesta eficiente en la tecnología móvil avanzada, los teléfonos inteligentes son uno de los medios populares para implementar la geolocalización de los buses, ya que son portátiles para los usuarios y fáciles de usar.

De acuerdo con la información otorgada hace dos años por la empresa de Marketing Digital, PubliMark (Briceño N., 2019), el número de usuarios de teléfonos inteligentes en Ecuador es del 87% del total de los habitantes y con un crecimiento al año siguiente con una tasa promedio de 6.8%. Apuntando al creciente número de usuarios de teléfonos inteligentes asociados a la tecnología de rastreo, el desarrollo de este proyecto tiene amplias perspectivas para ayudar a mejorar el transporte público.

En la ciudad de Otavalo existe una compañía y varias cooperativas que brindan el servicio de Transporte Público Interprovincial (TP-INTER) y en los últimos años se implementó un monitoreo mediante dispositivos GPS. Este servicio de transporte como el resto proporciona la información acerca de la ruta y programación de horarios. La información de la geolocalización de las unidades es operada de manera interna por las entidades mencionadas, además los horarios y rutas de los buses no se han transmitido eficientemente a todos sus usuarios. Por tal motivo los pasajeros desperdician demasiado tiempo al esperar la llegada de la unidad de TP-INTER que los lleve o acerque a su destino.

Además, la pandemia actual que estamos viviendo ha obligado a los buses de las diferentes cooperativas que están autorizadas en circular que trabajen con el 50% de sus unidades y con el 50% de su capacidad (Primera Zona, 2020). Por esta razón las autoridades de las entidades de transporte

tienen que realizar constantes modificaciones en sus rutas y horarios. Estos cambios han causado múltiples molestias a los usuarios de este servicio de transporte.

Para intentar solucionar los problemas mencionados anteriormente, se implementaron ocho paradas inteligentes con un alto costo, ubicadas en la capital de la provincia. En la actualidad, el dispositivo de parada no funciona con normalidad y su rendimiento se deteriora (Farinango Guajan, 2020).

Por lo tanto, en este trabajo, se propone el desarrollo de un prototipo de aplicativo móvil que además de brindar la información necesaria sobre los horarios y rutas de transporte, permita ver la geolocalización de las unidades TP-INTER en tiempo real. Los datos de las coordenadas de la ubicación en tiempo real serán enviadas desde un dispositivo móvil y no mediante un dispositivo GPS ubicado en el autobús. Se planteó esta manera de gestionar la geolocalización de los buses debido a que los dispositivos móviles son compatibles con los cuatro sistemas de posicionamiento. Esto significa que el celular puede implementar sistemas GPS, GLONASS, Beidou o Galileo en paralelo. De esta forma, la medición trilateral será más rápida y precisa al posicionar la ubicación geográfica en el mapa. También existe un ahorro de costes en dispositivos GPS, ya que no todos los buses incluyen este sistema, por lo que en muchos casos es necesario invertir en la compra e instalación de estas. La principal contribución de este trabajo es la arquitectura propuesta que usa una combinación de elementos entre los cuales se destaca el servicio de un base de datos en tiempo real. El artículo está organizado de la siguiente manera. En la Sección 2 se presentan los materiales y métodos implementados para el desarrollo del prototipo de aplicación. La Sección 3 expone la implementación y diferentes pruebas realizadas con su respectivo resultado.

2. Materiales y Métodos

En esta sección se explica con detalle el proceso que se siguió para la implementación de la aplicación.

La aplicación sigue una arquitectura cliente/servidor y consta de dos interfaces que son: Pasajero y Autobús. La interfaz de Pasajero muestra la ubicación de los autobuses en tiempo real y sus paradas. Además, el usuario podrá ver la información básica del autobús. Por otra parte, la interfaz de autobús transmitirá la posición del Autobús a un servidor de base de datos de tiempo real (Firebase). Todos los usuarios están previamente registrados en la base de datos por los administradores del sistema. A continuación, se describen los detalles de implementación.

2.1. Detalles de implementación

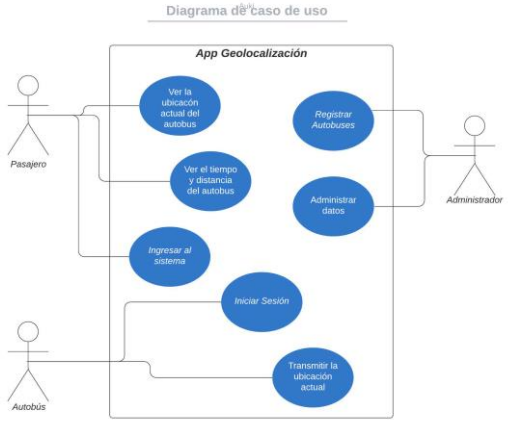

Figura 1. Diagrama de casos de uso.

Como se muestra en la Figura 1 existen tres tipos de usuario de los cuales dos son los que interactúan directamente con la aplicación y pueden navegar por sus correspondientes vistas. Ambos usuarios envían sus coordenadas geográficas a la base de datos de Firebase para almacenarlas o actualizarlas en tiempo real.

El tercer usuario corresponde al administrador que solo interactúa con el servidor de base de datos para almacenar, modificar o eliminar los datos del autobús. A continuación, se muestran detalles de las interfaces de cada usuario.

2.2. Despliegue

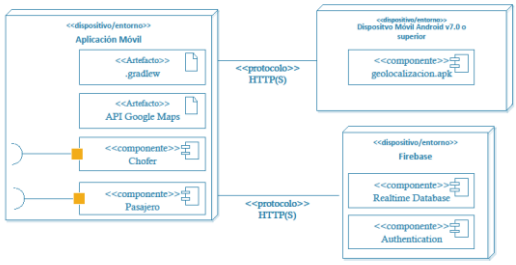

Figura 2. Diagrama de despliegue

En la Figura 2 se representa el modelo de la arquitectura en tiempo de ejecución de la aplicación. Como principales nodos se encuentran el servidor de Firebase y el aplicativo, los cuales, se comunican mediante el protocolo HTTP. Dentro del nodo del aplicativo se tiene la integración de la API de Google Maps y la configuración e importaciones que se realice mediante Gradle.

2.3. Interfaz de pasajero

Figura 3. Estructura del módulo del usuario Pasajero.

La aplicación permite al pasajero ingresar solo a la vista del mapa, la cual internamente almacena la ubicación en tiempo real del mismo, esto lo realiza solo cuando ingresa a la aplicación y abre la vista correspondiente

La interfaz permite al usuario mostrar una estimación de la distancia a la que se encuentra el autobús y un tiempo de llegada a la parada donde el usuario se encuentra. Para ello, la aplicación utiliza las clases: LatLngBus, MainFragment, MapsFragment (ver Figura 4).

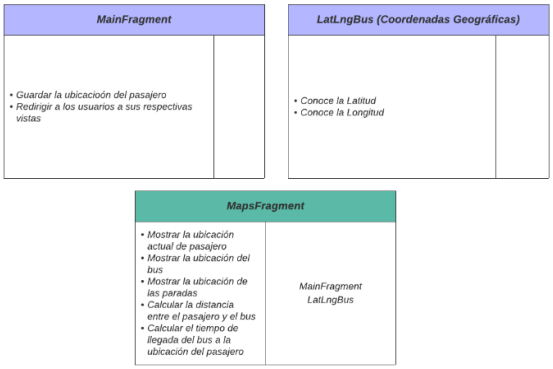

Figura 4. Tarjetas CRC para el usuario Pasajero.

2.3.1. LatLngBus

Es donde se encuentra el modelamiento de datos de las coordenadas que se almacenan y actualizan de este usuario y también del usuario autobús.

2.3.2. MainFragment

Aquí se encuentra el método que almacena o actualiza la ubicación actual del pasajero. El método primero verifica si los permisos para obtener la ubicación están habilitados, si no lo está salta un cuadro de mensaje para que el usuario puede aceptar o rechazar la solicitud, y si tiene permisos obtiene las coordenadas geográficas de su ubicación para almacenarlas en la base de datos.

2.3.3. MapsFragment

En esta clase se declaran dos constantes globales para gestionar el mapa y los cálculos que se realizaran más adelante.

Existe una constante donde se almacena el valor del radio de la Tierra en kilómetros para

realizar el cálculo de la distancia entre dos puntos (Pasajero – Autobús).

Al iniciar esta clase se ejecuta un método privado llamado *"getDeviceLocation()"* que inicializa el mapa con los parámetros, como el de zoom, para que pueda ser visible por el usuario y también se puede realizar las configuraciones necesarias.

Una vez que el mapa esté listo dentro del método *"onMapReady()"* se le agregan algunas configuraciones para su visualización y además realiza el cálculo de la distancia entre su ubicación y la del autobús.

También se le agregan los marcadores de la ubicación en tiempo real de los autobuses que se encuentren a una distancia de 200 metros del pasajero, adicionalmente se modifican las configuraciones predeterminadas de los marcadores que brinda la API de Google Maps.

Para realizar el cálculo de la distancia se utilizó la fórmula de Haversine (Ec 1).

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

(1)

En la formula la latitud inicial y final se representa por *phi*, *lambda* representa la longitud inicial y final, y la letra *r* hace referencia al radio de la Tierra. Este valor es relativo a la latitud, pues al no ser la Tierra perfectamente redonda, el valor del radio ecuatorial es de 6378 km, este valor se almacenó en una variable llamada *"RadioTierraKm"*.

Para el cálculo del tiempo de llegada se tomó como base la velocidad de 90 km/h, que es la permitida por la Agencia Nacional de Tránsito para el tránsito de los buses y de igual manera presentó una exactitud media del tiempo calculado por el aplicativo al tiempo de llegada real.

2.4. Interfaz de autobús

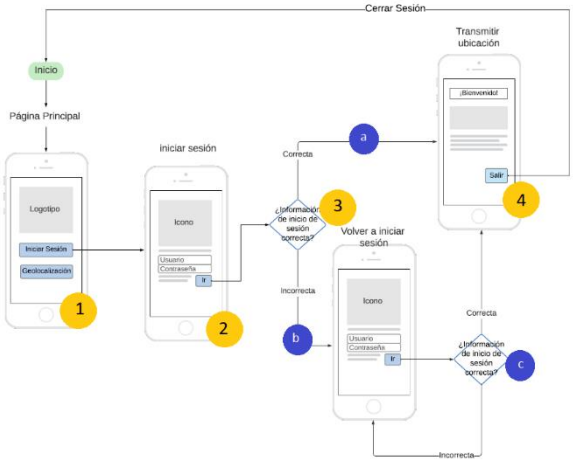

Figura 5. Estructura del módulo del usuario Autobús.

El usuario autobús solo puede enviar su ubicación en tiempo real si inicia sesión en la aplicación, las credenciales para el acceso se los da el administrador del sistema. Si ingresa correctamente sus credenciales se le redirige a la vista de su perfil donde puede ver su información básica e internamente estarían enviando sus coordenadas geográficas al servidor de base de datos.

El siguiente conjunto de pasos describen el funcionamiento de la Figura 5:

1. Cuando se abre la aplicación esta muestra una vista para *"Iniciar Sesión"*.
2. Se redirige a la vista de inicio de sesión y es donde se debe colocar las credenciales correspondientes.
3. El sistema verifica si las credenciales ingresadas son correctas.
   a) Si las credenciales fueron correctas ingresa al perfil del usuario autobús.
   b) Si no, surge un mensaje de alerta con el mensaje de *"Credenciales Incorrectas"* y se mantiene en la vista de inicio de sesión para que el usuario ingreses las credenciales correspondientes.

c) Vuelve a verificar si las credenciales son correctas y realiza la misma secuencia de pasos a partir del paso 2.

4. Si desea cerrar la sesión de su perfil debe dar clic en el botón *"Salir"* y volverá a la vista inicial.

Como muestra la Figura 6 en el proceso de autenticación intervienen las siguientes clases: *MainFragment - UserModelView, - LoginFragment,- ProfileFragment*

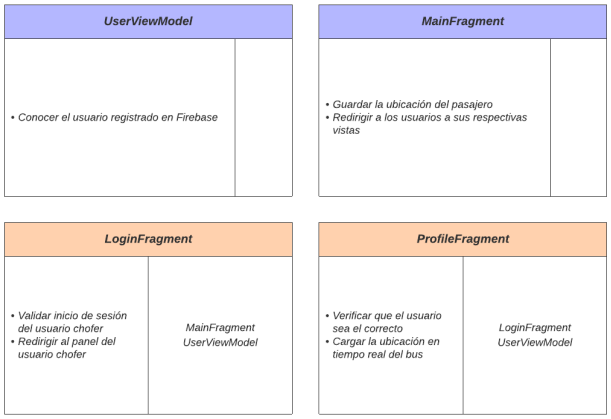

Figura 6. Tarjetas CRC para el usuario Autobús.

2.4.1. MainFragment

En esta clase está conectada con la vista principal y para este usuario solo existe un método que redirecciona a la vista de inicio de sesión del mismo.

2.4.2. UserModelView

Esta maneja el modelamiento del tipo de usuario que inicia sesión, la variable que controla esto se mantiene alerta a los cambios que exista en la sesión del usuario.

2.4.3. LoginFragment

Se ha creado un método para verificar las credenciales que ingresa el usuario, esta comprobación se lo realiza usando los recursos de Firebase Authentication, que verifica las credenciales ingresadas con los datos que se hayan almacenado en su servidor. Si las credenciales son correctas almacena al usuario usando el tipo de dato de userViewModel que está generado en la clase del mismo nombre.

2.4.4. ProfileFragment

Al iniciar esta clase se ejecuta primero el método público que ya está provisto por la superclase, dentro del método verifica si el usuario inicio sesión correctamente y si es correcto se mantiene en la vista y muestra la información del usuario, y si el usuario no inicio sesión correctamente se redirige a la vista anterior de inicio de sesión.

Una vez que haya iniciado sesión exitosamente, el método que se puede observar en la imagen almacena y actualiza la ubicación actual del autobús, enviado sus coordenadas geográficas al servidor de base de datos.

A continuación, se explica la navegación de los usuarios por las diferentes vistas.

2.5. Navegación

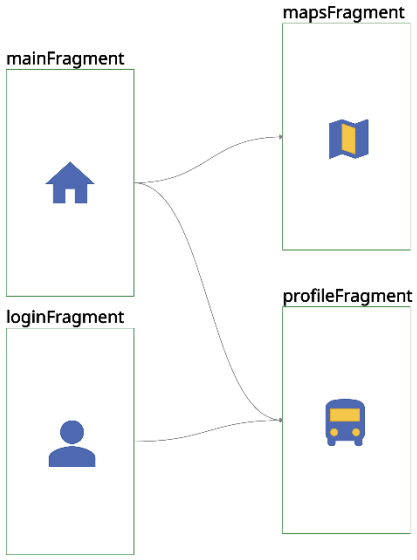

Figura 7. Modelo de navegación de la aplicación.

Para que el usuario pueda navegar a través, dentro y fuera de las diferentes vistas que contiene la aplicación se debe implementar el componente *"Navigation"* de Android Jetpack que permite implementar la navegación, desde simples clics de botones hasta patrones más complejos, como las barras de aplicaciones y los paneles laterales de navegación.

La Figura 7, representa un gráfico de navegación que, es un archivo de recursos que contiene todos los destinos y representa todas las rutas de navegación de la aplicación.

Dentro del *"Activity"* principal se agrega el componente *"NavHost"* que, es un contenedor vacío en el que se intercambian los destinos a medida que un usuario navega en la aplicación. A partir de agregar dicho componente se crearon los diferentes fragmentos donde se diseñarían las interfaces para cada usuario.

Existe un fragmento principal (mainFragment) el cual es accesible al abrir la aplicación, aquí el usuario tiene dos opciones, acceder como Autobús o Pasajero. Si accede como Pasajero navega hacia el fragmento donde puede interactuar con el mapa (mapsFragment), si accede como Autobús se dirige al fragmento de inicio de sesión (loginFragment) en el que deben ingresar sus credenciales y si son correctas se redirige hacia el fragmento de su perfil (profileFragment).

Seguidamente se expone cómo funciona y se estructura la base de datos de Firebase.

2.6. Base de datos en tiempo real

La base de datos con la que se trabajó es una no relacional debido a que, organizan grandes volúmenes de datos utilizando técnicas más flexibles como, por ejemplo, gráficos, documentos, pares de valores y columnas.

Este tipo de base de datos complementa de manera muy beneficiosa al desarrollo de la aplicación móvil, ya que, permite manejar una gran información de coordenadas de los pasajeros y autobuses que se encuentra constantemente actualizándose. Además, al ser un prototipo está sujeta a cambios frecuentes en la forma de almacenar los datos de los usuarios y sus coordenadas.

Firebase Realtime Database es el servidor de base de datos escogido para almacenar los datos como objetos JSON y los objetos principales que se guardan o actualizan son los datos de la ubicación actual del autobús como *"Autobus" y* la del pasajero como *"Pasajero"*.

Seguidamente, se presenta el esquema de base de datos no relacional utilizada en el desarrollo de la aplicación.

2.6.1. Esquema de la base de datos

Con la descripción del esquema de la base de datos se pretende dar explicación a la utilidad de cada uno de los campos pertenecientes a los perfiles de los usuarios.

2.7. Código

En esta sección se presenta y explica el código fuente creado para el cumplimiento de las funcionalidades de la aplicación.

2.7.1. Configuración de la conexión con la base de datos

Se debe tener algunos requerimientos previos a la conexión con la base de datos en tiempo real de Firebase, tales como:

- Segmentación del nivel de API 19 (KitKat) o superior.
- Usar una versión de Android 4.4 o superior.
- Implementar servicios de Google Play en el proyecto de Android.
- Acceder a Firebase con una cuenta de Google.

Al cumplir con todos los requisitos previos se creó un proyecto llamado **prueba** en la consola de Firebase para luego registrar la aplicación que se está creando en Android Studio con el proyecto creado de Firebase.

Una vez que el proyecto fue creado se debe descargar un archivo de configuración llamado **google-service.json** que contiene identificadores únicos, pero no secretos, para

el proyecto donde se está desarrollando la aplicación.

Otro de los aspectos importantes para utilizar esta base de datos es seleccionar un modo de inicio para las reglas de seguridad de Firebase. La configuración establecida está en modo prueba y permitiendo que cualquier usuario pueda leer y escribir en la base de datos creada.

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Figura 8. Reglas de seguridad de la base de datos de Firebase.

Además, se implementaron varias librerías para conectar con la base de datos en tiempo real de Firebase, los cuales son:

- 'com.google.firebase:firebase-bom:26.4.0': también conocido como lista de materiales, esta librería permite administrar todas la versiones de las librerías de Firebase implementadas en la aplicación con solo especificar la versión de BoM. Esto quiere decir que, no será necesario especificar las versiones de las librerías adicionales de Firebase que se vayan implementando en la aplicación.
- 'com.google.firebase:firebase-database': permite usar la o las base de datos creadas dentro del proyecto al cual hace referencia y también otorga todas las herramientas necesarias para realizar cualquier transacción en la base de datos.

2.7.2. Gestión de usuarios

Dentro de la consola de Firebase solo es necesario habilitar la sección de Authentication y agregar el método de autenticación que se utilizara, para este caso se optó por utilizar como credenciales un correo y contraseña.

Cada usuario tiene un identificador único generado automáticamente y el cual también se utiliza en la base de datos para almacenar los usuarios de tipo autobús.

Para utilizar esta herramienta solo es necesario agregar la librería 'com.google.firebase:firebase-auth'.

2.7.3. Proyecto en Android

2.7.3.1. Arquitectura

El desarrollo de la aplicación fue bajo la arquitectura MVVM (Model – View – ViewModel) porque permite trabajar con los datos de manera mucho más automatizada y en tiempo real que, resulta muy útil para manejar la informacion de las coordenadas geográficas requeridas por la aplicación.

La librería que se ha implementado para trabajar con este patrón es 'androidx.lifecycle:lifecycle-viewmodel:2.3.1'.

2.7.3.2. Estructura del proyecto

Como se ha explicado en una sección anterior, para gestionar dos distintos usuarios en la misma aplicación se ha optado por crear el aplicativo por fragmentos y así tener una navegación más fluida y mejor controlada para cada tipo de usuario.

Las librerías que permiten automatizar este trabajo son las siguientes:

- 'androidx.fragment:fragment:1.3.6'
- 'androidx.navigation:navigation-fragment:2.3.5'
- 'androidx.navigation:navigation-ui:2.3.5'
- ‘androidx.legacy:legacy-support-v4:1.0.0'

2.7.3.3. Geolocalización

Antes de cualquier implementación de librerías se deben colocar unas líneas de código para que soliciten la autorización al usuario del dispositivo móvil para usar el GPS y datos móviles del dispositivo. Esta configuración se debe realizar en el archivo **AndroidManifest.xml** y las líneas de código deben tener la siguiente estructura:

- <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
- <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
- <uses-permission android:name="android.permission.INTERNET" />
- <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

Ahora, para utilizar los servicios de Google Maps se debe agregar una dependencia en el archivo **build.gradle** de nivel superior la siguiente menara:

```
dependencies {
        classpath 'com.google.gms:google-services:4.3.10'
    }
```

Figura 9. Dependencia para usar los servicios de Google

Al agregar la dependencia es posible utilizar las librerías necesarias para trabajar con las herramientas de Google Maps, las librerías que se implementan son las siguientes:

- 'com.google.android.gms:play-services-maps:17.0.1'
- 'com.google.android.gms:play-services-location:18.0.0'

Al tener implementada las librerías y activa una API de Google Maps dentro de la consola de Google se debe agregar un archivo de configuración llamada **google_maps_api.xml**, en el cual se debe colocar la cadena conocida como **key** obtenida al activar dicha API. Este archivo se debe encontrar dentro del proyecto en la carpeta **/res/values.**

Para utilizar los servicios de Google Maps se creó el siguiente código que inicializa el mapa con algunas características predefinidas y también agregando los marcadores que representan al autobús y su ubicación real.

```
mMap = googleMap;

UiSettings uiSettings = mMap.getUiSettings();
uiSettings.setZoomControlsEnabled(true);
uiSettings.setCompassEnabled(true);

mMap.setMyLocationEnabled(true);
mMap.setOnMyLocationButtonClickListener(this);
mMap.setOnMyLocationClickListener(this);
mMap.setOnMarkerClickListener(marker -> {
   fusedLocationProviderClient.getLastLocation()
      .addOnSuccessListener(requireActivity(), location -> {
         if (location != null) {
            BigDecimal distanciaKm = BigDecimal
               .valueOf(calcularDistanciaHaversine(location, marker.getPosition()))
               .setScale(2, RoundingMode.HALF_UP);
            BigDecimal tiempo = distanciaKm.divide(BigDecimal.valueOf(90), 0);
            Toast.makeText(
               requireActivity(),
                  "Distancia:" + distanciaKm + " Km" +
                     "\nTiempo: " + tiempo.multiply(BigDecimal.valueOf(60)) + " min",
               Toast.LENGTH_SHORT
            ).show();
         }
      });
   return false;
});
```

Figura 10. Código para la carga inicial del mapa

También como se mencionó en una sección anterior se utiliza la fórmula de Haversine para calcular la distancia del autobús con respecto al pasajero y el método creado para realizar el cálculo es el siguiente:

```
private double calcularDistanciaHaversine(Location userLocation, LatLng busLocation) {
  double difLatitud = Math.toRadians(userLocation.getLatitude() - busLocation.latitude);
  double difLongitud = Math.toRadians(userLocation.getLongitude() - busLocation.longitude);

  double a = Math.pow(Math.sin(difLatitud/2), 2) +
        Math.cos(Math.toRadians(busLocation.latitude)) *
          Math.cos(Math.toRadians(userLocation.getLatitude())) *
          Math.pow(Math.sin(difLongitud), 2);
  double c = 2 * Math.asin(Math.sqrt(a));

  return RadioTierraKm * c;
}
```

Figura 11. Código de la implementación de la Formula de Haversine

El código previamente expuesto en la Figura 10 pertenece al diseño de la vista del usuario pasajero, para el usuario autobús no existe demasiada complejidad a excepción del envió del código creado para enviar su ubicación en tiempo real.

Cada vez que el autobús se mueve es captado por los sensores del dispositivo móvil y se ejecuta el siguiente código:

```
Map<String, Object> latlang = new HashMap<>();
latlang.put("latitud", location.getLatitude());
latlang.put("longitud", location.getLongitude());

myRef.child("Autobus")
  .child(Objects.requireNonNull(userViewModel.getUser().getValue()).getUid())
  .child("ubicacion")
  .setValue(latlang);
```

Figura 12. Código del envió de la ubicación del usuario autobús

3. Resultados y Discusión

En este apartado se evidencian los resultados de las diferentes pruebas a las que se sometió la aplicación, con el objetivo de saber su desempeño en un ambiente real.

3.1. Pruebas de estrés

Para realizar esta prueba, se enviaron múltiples solicitudes a cada una de las vistas correspondientes a cada rol de usuario para encontrar su punto de inflexión. Apache JMeter se utilizó como una herramienta que funciona independientemente del lenguaje de programación y está diseñado para este tipo de experimentos.

3.1.1. Usuario pasajero

Escenario 1

Simula el acceso simultáneo a la vista del mapa con 50 personas representando el rol de pasajero.

Tabla 1. Resultados pruebas de estrés del pasajero - Escenario 1

# Muestras	Promedio	Min	Max	% Error
50	266	181	381	0,00%

En la tabla 1 se aprecia que el tiempo promedio de carga del mapa es de 0,266 segundos y con un tiempo máximo de 0,381 segundos.

Escenario 2

Simula el acceso simultáneo a la vista del mapa con 1000 personas para encontrar el punto de inflexión de la aplicación.

Tabla 2. Resultados pruebas de estrés del pasajero - Escenario 2

# Muestras	Promedio	Min	Max	% Error
1000	5535	214	81181	1,80%

La tabla 2 presenta el punto de inflexión de la carga de la vista del mapa con un porcentaje de error de 1,8%, con un promedio de tiempo de carga de 5,535 segundos y con un tiempo máximo de carga de 81,181 segundos.

3.1.2. Usuario autobús

Escenario 1

Simula el acceso simultáneo al iniciar sesión con el rol de autobús de 50 personas.

Tabla 3. Resultados pruebas de estrés del autobús - Escenario 1

# Muestras	Promedio	Min	Max	% Error
50	116	82	271	0,00%

Como se observa en la tabla 3 el tiempo promedio de carga al iniciar sesión es de 0,116 segundos que representa un valor más bajo al rol de pasajero.

Escenario 2

También con el fin de encontrar el punto de inflexión se realiza la simulación con 1000 personas ingresando simultáneamente al perfil del autobús.

Tabla 4. Resultados pruebas de estrés del autobús - Escenario 2

# Muestras	Promedio	Min	Max	% Error
1000	2145	82	68595	0,30%

La tabla 4 presenta un porcentaje de error de 0,30% que quiere decir que también se halló el punto de inflexión del usuario autobús. El tiempo promedio de carga resulto ser 2,145 segundos y con un tiempo máximo de 68,595 segundos.

3.1.3. Análisis de resultados

Teniendo en cuenta los dos escenarios evaluados para cada usuario, el tráfico máximo evaluado fue de 1000 ingresos simultáneos por cada vista, el resultado fue satisfactorio para ser un prototipo, se puede concluir que la aplicación puede manejar adecuadamente menos de 1000 solicitudes simultáneas. Además, se debe tener en cuenta que la vista o sesión del usuario autobús no lo manejaran la misma cantidad de personas cómo con el rol de pasajero.

3.2. Pruebas de desempeño

Para conocer el valor de los recursos utilizados por la aplicación se procedió a instalarlo en diferentes dispositivos y se procede a evaluar el consumo de datos móviles.

Escenario 1

Se realizaron las pruebas con el rol de pasajero y presentó un consumo bajo en el uso progresivo de 1 hora con 20 minutos. Esta prueba se ejecutó desde una red WiFi estable y usando JMeter, la aplicación presentó un consumo aproximado de 0,05 KB recibidos por segundo y 0,13 KB enviados por segundo. Durante el tiempo usado el total de consumo fue de 3,2 MB.

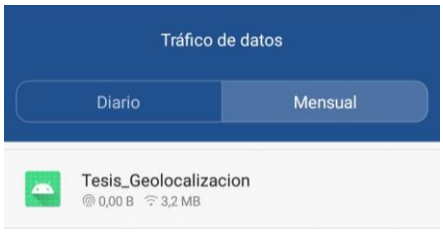

Figura 13. Consumo datos móviles – Escenario 1

Cuando el autobús presentaba interferencia en ciertos sectores de la ruta en la que transitaba, la aplicación duplica el icono del autobús hasta que vuelva a transmitir correctamente las coordenadas de su ubicación.

Figura 14. Error al perder la señal del autobús

Escenario 2

Para el rol de autobús se transmitió la señal en dos rutas, las cuales fueron: Otavalo – Cayambe y viceversa. Para transmitir las coordenadas de su ubicación el aplicativo tenía que estar funcionando en primer plano y solo era necesario iniciar sesión con las credenciales creadas previamente.

El tiempo transcurrido por cada ruta y en la que la aplicación tuvo que estar abierta fue de 40 minutos. Para la primera ruta (Otavalo - Cayambe) también se presenta un consumo aproximado de 0,05 KB recibidos por segundo y 0,10 KB enviados por segundo, con un total de datos móviles de 30,52 KB en el tiempo que se realizó la prueba (ver Figura 15). El proveedor del plan de datos móviles que se usó en este dispositivo fue CNT y transmitió correctamente en la mayor parte de la ruta en la que circuló, las interferencias que hubieron fueron por distintos tramos de la ruta donde es frecuente la perdida de señal también con otros proveedores de datos móviles.

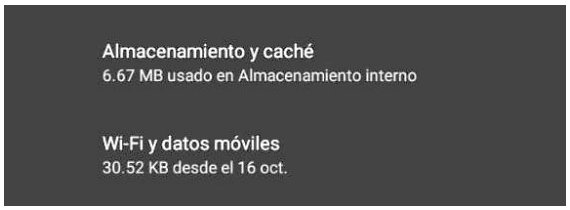

Figura 15. Consumo datos móviles – Escenario 2 - Ruta 1

En la ruta de regreso (Cayambe – Otavalo) se incrementó a más del doble del consumo anterior debido a que hubo tráfico al salir de la terminal y la aplicación como debe mantenerse aumento el consumo de los recursos. Aproximadamente los KB recibido por segundo sería 0,96 y los KB enviados por segundo sería 1,98; con un total de consumo acumulado de datos móviles de 1,04 MB, en los 40 minutos (ver Figura 16). Aunque el consumo haya incrementado no representa mayor problema porque el consumo diario por ruta estaría cubierto por el plan mensual que como mínimo brinda 2GB.

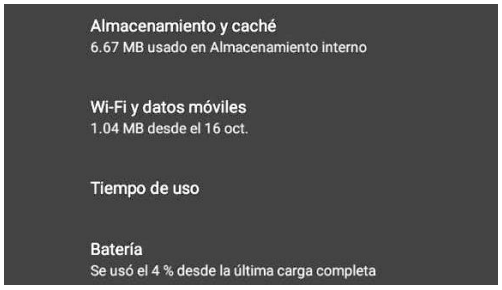

Figura 16. Consumo datos móviles – Escenario 2 - Ruta 2

Escenario 3

Con el rol de pasajero y como proveedor de datos móviles la empresa Tuenti exhibió un consumo también bajo por 1 hora de uso y el seguimiento del autobús presentó una transmisión estable. Recibió aproximadamente 0,04 KB por segundo y 0,08 KB enviados por segundo. Con un total de consumo de datos móviles de 1,26 MB (ver Figura 17).

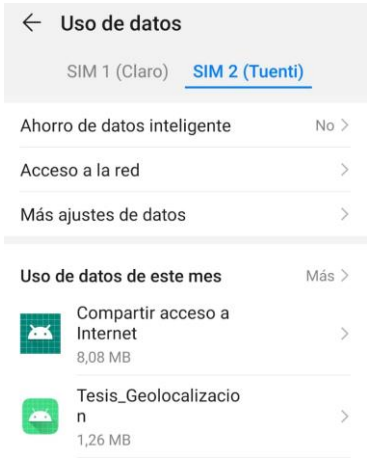

Figura 17. Consumo datos móviles – Escenario 3

Escenario 4

También con el rol de pasajero con 1 hora y 40 minutos de uso mostró un consumo relativamente bajo en el seguimiento del autobús en las dos rutas antes descritas. Los KB enviados y recibidos por segundo fueron aproximadamente iguales a los datos obtenidos en el escenario 3, con un consumo total de datos móviles de 5,6 MB.

Figura 18. Consumo de datos móviles - Escenario 4

3.2.1. Análisis de resultados

Se observó que el consumo de datos al usarlo como pasajero para rastrear el autobús y el usuario por el chofer para enviar las coordenadas geográficas en tiempo del autobús es parecido. Esto se debe a que ambos usuarios envían sus coordenadas al servidor de Firebase y solo el usuario pasajero presenta una carga un poco mayor porque debe cargar el cambio constante de la ubicación del autobús.

3.3. Pruebas de funcionalidad

Se realiza la evaluación del correcto funcionamiento y cálculo que realiza la aplicación con respecto a las coordenadas geográficas que maneja.

3.3.1. Distancia

Para evaluar el cálculo la distancia se tomarán 2 puntos en el mapa y para tener una referencia correcta se utilizará la herramienta de Google Maps que puede medir la distancia entre dos puntos geográficos seleccionados en su mapa. Se debe tener en cuenta que las coordenadas tomadas no serán exactamente iguales debido a que, la evaluación está realizada en diferentes ambientes, se colocará una latitud y longitud cercanas a los puntos evaluados por la aplicación móvil.

Tabla 5. Resultados del cálculo de la distancia

Latitud inicial	**Longitud inicial**
0,228533	-78,228569
Latitud final	**Longitud final**
0,195381	-78,233831
Ambiente	**Distancia (Km)**
Google Maps	3,63
Aplicación móvil	3,92
% Error	7,98

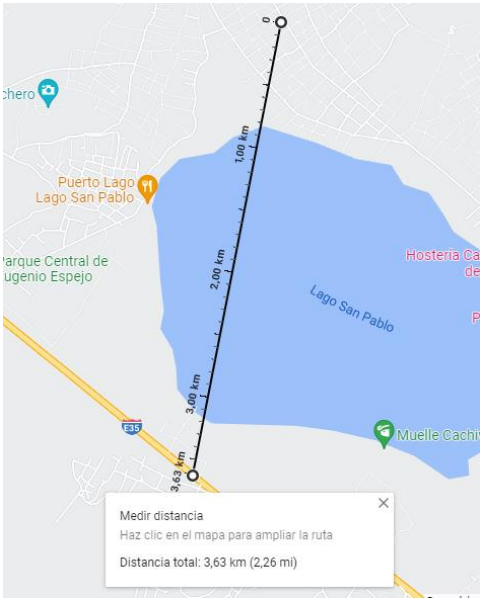

Figura 19. Cálculo de la distancia en Google Maps

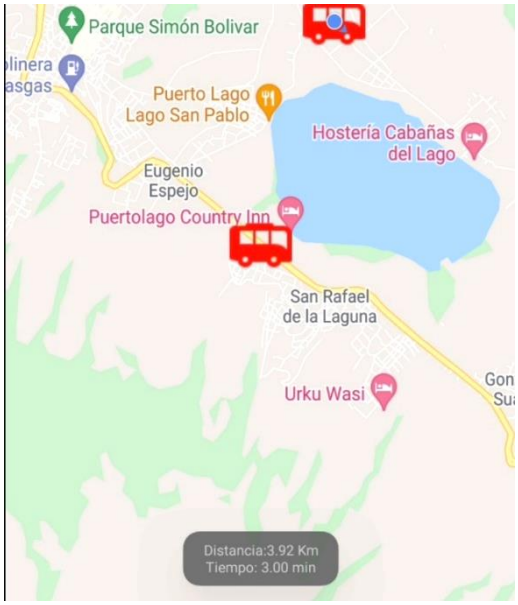

Figura 20. Cálculo de la distancia y tiempo en la aplicación móvil

El calculó del porcentaje se evaluó con la siguiente formula (Ec 2).

$$\frac{|v.aprox - v.exacto|}{|v.exacto|} * 100$$

(2)

3.3.2. Tiempo

Para obtener el cálculo del tiempo se tomaron como referencia 2 puntos distintos a los anteriores, pero que tienen la misma distancia que se calculó en la tabla 5 con la aplicación móvil, esto se realizó porque Google Maps no realiza el cálculo del tiempo entre 2 puntos geográficos en línea recta, por eso se tomó una latitud y longitud, para un inicio y final de cada punto, en una carretera en línea recta. El cálculo del tiempo obtenido por la aplicación móvil será utilizando las mismas coordenadas geográficas expuestas en la tabla 5.

Tabla 6. Resultados del cálculo del tiempo

Latitud inicial	**Longitud inicial**
0,273266	-78,239856
Latitud final	**Longitud final**
0,243411	-78,252902
Ambiente	**Tiempo (minutos)**
Google Maps	4
Aplicación móvil	3
% Error	25

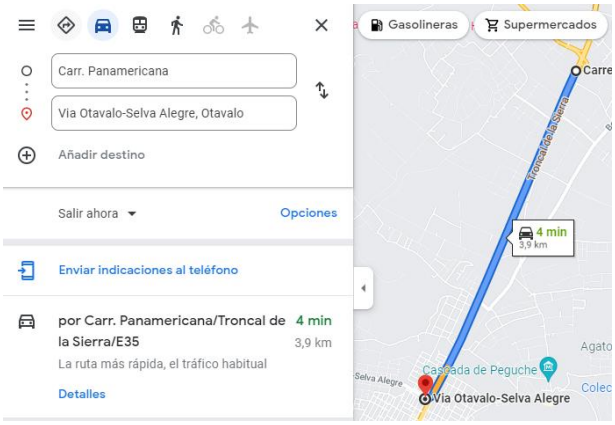

Figura 21. Cálculo del tiempo en Google Maps

3.4. Implementación

En esta sección se detalla los resultados obtenidos en el proceso de implementación de la aplicación.

3.4.1. Descripción de hardware y software

Los dispositivos en los cuales se instaló la aplicación fueron los siguientes:

Tabla 7. Limitación en el uso del aplicativo en los diferentes dispositivos

#	Modelo	Versión Android	Tiempo adquisición (año)	Red	Funcionamiento aplicación	
					Pasajero	Autobús
1	Huawei P8 Lite	5.0.1	5	3G/4G	Correcto	No envía coordenadas
2	Samsung	11	5	4G	Correcto	Correcto

3	Huawei P Smart 2020	9.0	1	4G	Correcto	Correcto
4	Xiaomi 9t	11	2	4G	Correcto	Correcto

En la tabla 5 también se puede observar si tuvo o no un correcto funcionamiento al realizar las pruebas por cada usuario.

3.4.2. Características de la aplicación

La aplicación esta desarrollada basada en código abierto lo que la vuelve escalable a futuras mejoras o cambios. Como se ha mencionado en los puntos anteriores se utilizó el IDE de Android Studio junto al lenguaje de Java y en la versión de Android 7.0 Nougat API 24, debido a que el 73,7% de los usuarios pertenecen a esta versión, (Rahman, 2021). Además, se utilizó una base de datos en tiempo real (Firebase). La principal función de las herramientas de Firebase ha sido el registro y control de los usuarios de tipo autobús y el almacenamiento de la ubicación (coordenadas de latitud y longitud) de los pasajeros y autobuses.

Una vez culminado el desarrollo se pudo conocer el tamaño de la aplicación el cual es de 14,22 MB y se procedió a ejecutar las respectivas pruebas de funcionamiento y rendimiento que presentaron los siguientes resultados.

Figura 22. Almacenamiento de la Aplicación

La aplicación desarrollada se puede instalar en cualquier dispositivo igual o superior a la versión de Android mencionada anteriormente y al ser una versión que lo tiene la mayoría de las personas el uso de la misma estaría al alcance de todos los pasajeros y choferes. El único gasto que se genera será la recarga mensual de un plan de datos que dependiendo la empresa con la que se contrate estaría como mínimo $5 mensuales por cada dispositivo móvil.

3.5. Comparación con soluciones alternativas al problema planteado

Al optar por el uso de la aplicación por parte de la empresa de transportes el gasto disminuiría a comparación del gasto que les genera usar los equipos GPS que se tienen actualmente en los autobuses que rondan en un precio de $25 a $300 y también generan gastos extra como: el pago mensual a la empresa que gestiona el rastreo que es de aproximadamente $10 a $12 y también el pago mensual por los datos móviles que necesita cada uno de los dispositivos GPS que tiene el mismo precio de recarga de los dispositivos móviles.

3.6. Discusión

Para enviar las coordenadas primero debe iniciar sesión y las credenciales ingresadas deben coincidir con los datos previamente creados en el servidor por el administrador. La validación se realiza correctamente, si las credenciales son correctas ingresa al perfil de conductor si no se enviará la alerta correspondiente al error.

Al ingresar al perfil del conductor, la aplicación procede a enviar la información hacia el servidor de forma instantánea, por lo tanto, el usuario también podrá ver la ubicación del autobús de manera inmediata.

También se realizaron pruebas en versiones diferentes de Android y existen limitación para el uso como conductor del autobús, puesto que el envío de las coordenadas no se realizaron correctamente, esto puede ser a causa de algún mal funcionamiento del GPS del dispositivo móvil o también pudo deberse a problemas con los datos móviles.

Con el rol de pasajero solo es necesario ingresar directamente al mapa que se encuentra integrado en la aplicación y podrá ver la ubicación del autobús más cercano a su posición. En esta prueba no existió mayor dificultad con el rastreo de las unidades, lo que presentó una irregularidad fue el cálculo de la distancia entre el autobús y pasajero, el cual presentó un porcentaje medio de exactitud del cálculo realizado por la aplicación con respecto a la distancia real.

4. Conclusiones

El aplicativo fue desarrollado para la plataforma Android y debido a la gran cantidad de herramientas y compatibilidad con diferentes tecnologías y servicios en la nube, ayuda a que el sistema de geolocalización sea modificable y expandible.

Al ser una aplicación donde interactúan dos usuarios en el mismo sistema es necesario crear roles para que cada uno de ellos y la información enviada, en especial por el rol de autobús, debe sincronizarse de forma automática, esta gestión es realizada con Firebase, ya que poseen diferentes herramientas para administrar a los usuarios con distintos privilegios.

El aplicativo en su primera versión presenta una correcta funcionalidad, en especial en las últimas versiones de Android, además la navegación por el sistema es bastante sencilla, por lo tanto, el usuario no podrá perderse o tener dificultades al interactuar dentro de la aplicación.

5. Referencias

[1] J. Villón Reyes, «La venta de carros aumentó 13,72% en el primer trimestre y el segmento SUV sigue creciendo,» *El Universo,* 15 Abril 2021.

[2] M. Rahman, «Android Version Distribution statistics will now only be available in Android Studio,» 10 Abril 2021. [En línea]. Available: https://www.xda-developers.com/android-version-distribution-statistics-android-studio/.

[3] S. A. Mohammed, A. H. Ariffin y N. A. Abd Aziz, «Bus Tracking App for Universities Transportation,» pp. 1-4, 2021.

[4] B. Y. Low, S. H. Dahlan y M. H. Abd Wahab, «Real-time Bus Location and Arrival Information System,» pp. 1-4, 2016.

[5] D. M. Farinango Guajan, «APLICACIÓN MÓVIL PARA RASTREO DE AUTOBUSES,»

UNIVERSIDAD TÉCNICA DEL NORTE, Diciembre 2020.

[6] S. M. Chit, L. Y. Chaw, C. L. Thong y C. Y. Lee, «A Pilot Study: Shuttle Bus Tracker App for Campus,» pp. 1-6, 2017.

[7] D. Briceño N., «Datos importantes sobre el comportamiento Digital en Ecuador 2019,» 2019. [En línea]. Available: https://publimark.ec/2019/04/11/datos-importantes-sobre-el-comportamiento-digital-en-ecuador-2019/.

[8] R. M. Alves, R. M. Alves, L. S. De Carvalho y A. G. Schäfer, «Development and implementation of a public transportations tracking system: study case,» *Revista Tecnologia e Sociedade*, pp. 1-17, Octubre 2019.

[9] IONOS, «NoSQL: la tendencia hacia un soporte de datos estructurado,» 20 Abril 2020. [En línea]. Available: https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/nosql/.

[10] Google Maps Plataform, «Descripción general del SDK de Maps para Android,» [En línea]. Available: https://developers.google.com/maps/documentation/android-sdk/overview?hl=es. [Último acceso: Agosto 2021].

[11] Primera Zona, «12 cooperativas con rutas habilitadas desde la terminal de Ibarra,» Julio 15 2020. [En línea]. Available: https://primerazona.com/2020/07/rutas-terminal-iba/.