UNIVERSIDAD CATÓLICA DE CUENCA

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INFORMÀTICA, CIENCIAS DE LA COMPUTACIÒN E INNOVACIÒN TECNOLOGICA.

CARRERA DE INGENIERÌA DE SISTEMAS

ANALISIS DE UN PROTOTIPO PARA UN APLICATIVO MOVIL QUE PERMITA LA GEOLOCALIZACION CON LA UTILIZACION IoT PARA LAS UNIDADES DE TRANSPORTE URBANO DEL CANTON CAÑAR.

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS.

AUTOR: LUIS ENRIQUE CARVAJAL ANDRADE.

DIRECTOR: ING. LUIS PINOS CASTILLO.

CAÑAR-ECUADOR

2022

DIOS, PATRIA, CULTURA Y DESARROLLO

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INFORMÀTICA, CIENCIAS DE LA COMPUTACIÒN E INNOVACIÒN TECNOLOGÌCA.

CARRERA DE INGENIERÌA DE SISTEMAS

ANALISIS DE UN PROTOTIPO PARA UN APLICATIVO MOVIL QUE PERMITA LA GEOLOCALIZACION CON LA UTILIZACION IoT PARA LAS UNIDADES DE TRANSPORTE URBANO DEL CANTON CAÑAR

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS.

AUTOR: LUIS ENRIQUE CARVAJAL ANDRADE.

DIRECTOR: ING. LUIS PINOS CASTILLO.

CAÑAR- ECUADOR

2022

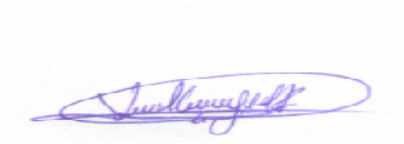
DIOS, PATRIA, CULTURA Y DESARROLLO

Universidad Católica de Cuenca	**DECLARATORIA DE AUTORIA Y RESPOSABILIDAD**	CÓDIGO: F- DB - 34 VERSION: 01 FECHA: 2021-04-15 Página **1** de **1**

Declaratoria de Autoría y Responsabilidad

Luis Enrique Carvajal Andrade portador(a) de la cedula de ciudadanía N. **0302854005**. Declaro ser el autor de la obra: "**Análisis de un Prototipo para un Aplicativo Móvil que Permita la Geolocalización con la Utilización IoT para las unidades de Transporte Urbano del Cantón Cañar**", sobre la cual me hago responsable sobre las opiniones, versiones e ideas expresadas. Declaro que la misma ha sido elaborada respetando los derechos de propiedad intelectual de terceros y eximo a la Universidad Católica de Cuenca sobre cualquier reclamación que pudiera existir al respecto. Declaro finalmente que mi obra ha sido realizada cumpliendo con todos los requisitos legales, éticos y bioéticos de investigación, que la misma no incumple con la normativa nacional e internacional en el área específica de investigación, sobre la que también me responsabilizo y eximo a la Universidad Católica de Cuenca de toda reclamación al respecto.

Cañar, **24 de febrero de 2022**



F: ……

Luis Enrique Carvajal Andrade

C.I. 0302854005

CERTIFICACIÒN

Certificó que el presente trabajo fue desarrollado por el Est. Luis Enrique Carvajal Andrade, bajo mi supervisión.

Ing. Luis Fernando Pinos Castillo

DIRECTOR DEL TRABAJO INVESTIGATIVO

UNIVERSIDAD CATÒLICA DE CUENCA.

DEDICATORIA

Este trabajo quiero dedicarlo con todo el cariño a Dios y a mis padres y a mi hijo, pilar fundamental y fuente de aliento y motivación, son quienes han estado a mi lado durante todo este proceso brindándome todo su apoyo y han hecho posible la culminación de mis estudios. De igual manera a mis hermanas y familia, que siempre han estado presentes en mi vida, impulsándome a cumplir mis ideales.

AGRADECIMIENTO:

Quiero manifestar mi gratitud a Dios por llenar mi vida con su guía y bendición que me fortalecen día a día para continuar cumpliendo mis metas, a mis padres que con sus sabias enseñanzas me han sabido llevar siempre por el camino del bien y me han alentado a alcanzar todas mis metas.

Mi profundo agradecimiento a la honorable Universidad Católica de Cuenca, a la Unidad Académica de Tecnologías de la Información y Comunicación y a los docentes que me han brindado todas las oportunidades y conocimientos necesarios para crecer como persona y desempeñarme como un profesional.

De forma especial agradezco a mi tutor el Ing. Luis Pinos, que gracias a su apoyo y basto conocimiento me instruyo con su sugerencias y apoyo para poder concluir mi presente trabajo de titulación.

APROBACIÓN DE TRIBUNAL DE GRADO

El tribunal designado por el honorable consejo directivo de la Universidad Católica de Cuenca Extensión Cañar, Facultad de Ingeniería de Sistemas instalado para receptar la sustentación del trabajo final de investigación con el tema: "ANALISIS DE UN PROTOTIPO PARA UN APLICATIVO MOIL QUE PERMITA LA GEOLOCALIZACION CON LA UTILIZACION IoT PARA LAS UNIDADES DE TRANSPORTE URBANO DEL CANTON CAÑAR", transcurrido el tiempo reglamentario procede a consignar la calificación de (_____/100).

Cañar, _____de_____ del 2022

PRESIDENTE	**DIRECTOR**
DELEGADO	**SECRETARIO**

INDICE DE CONTENIDO

Tabla de Ilustraciones

Índice de Tablas

Manual de Usuario

Este prototipo se ha desarrollado para su uso como aplicación de rastreo de las unidades de trasporte urbano. El cual emplea tecnología GSM programada en lenguaje C y comandos AT bajo la plataforma Arduino y Android Studio. Por tanto, puede implementarse en cualquier vehículo que posea una toma de voltaje de 12V, además el sistema de rastreo puede ser ejecutado desde cualquier dispositivo Android con sistema operativo 6.0 en adelante.

Para utilizar el prototipo, basta con conectar el dispositivo electrónico a una fuente de 12V, 2Amp al vehículo e instalar la app en un teléfono móvil. Como se describirá más adelante en este manual.

Ilustración 1. Pantalla Principal App Móvil. Fuente: Autor

Partes del dispositivo electrónico

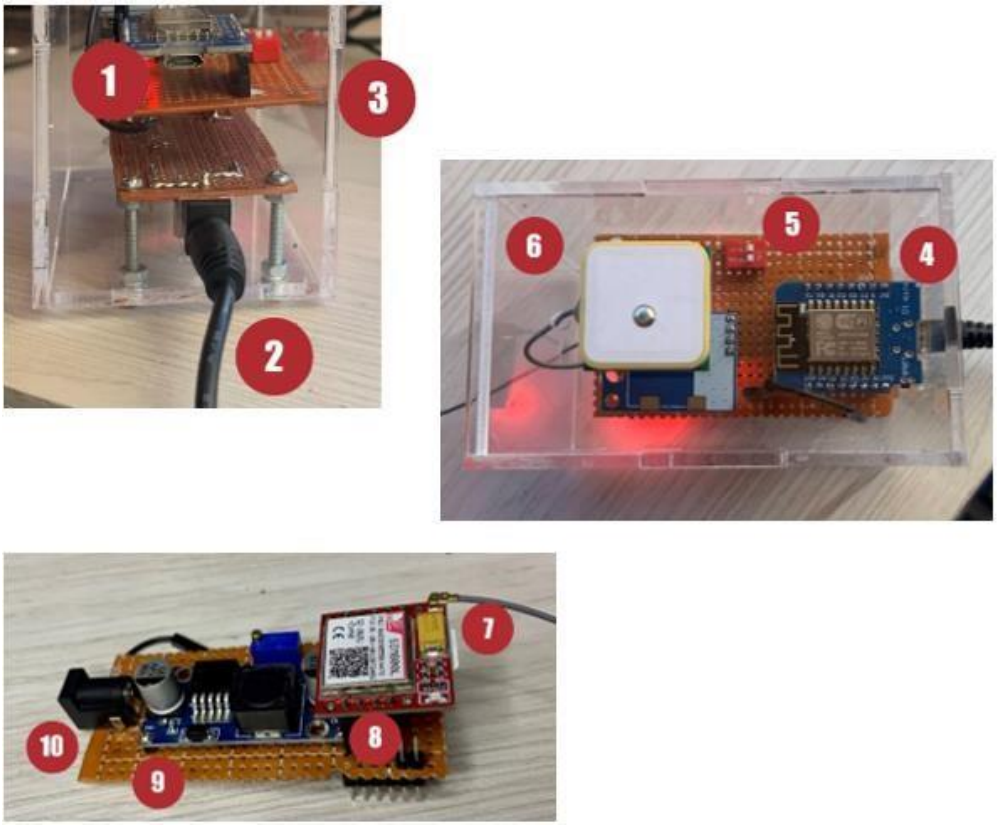


Ilustración 2: Componentes del dispositivo. Fuente: Autor

1. Conector micro USB para programación del dispositivo.
2. Cable de alimentación 12V; 2Amp.
3. Carcasa acrílica del dispositivo.
4. Tarjeta electrónica de desarrollo Esp8266.
5. Switch alimentación parcial del dispositivo.
6. Tarjeta GPS Neo-6m y antena cerámica.
7. Tarjeta sim (movistar con plan de datos).

8. Módulo GSM sim800l y antena.
9. Tarjeta reguladora de voltaje.
10. Jack entrada 12V.

Instalación del hardware

Para el correcto funcionamiento del prototipo es necesario tener en cuenta los siguientes requisitos tanto en hardware como en software:

Requisitos de hardware:

1. Toma de 12V a 2Amp.
2. Cable con conector Jack 12V
3. Tarjeta sim de cualquier operadora con datos móviles
4. Requisitos de software
5. Dispositivo móvil Android 6.0 en adelante
6. Datos móviles o wifi para conexión a internet

Una vez que se asegure de cumplir con los requisitos expuestos, puede proceder a insertar la tarjeta Sim dentro del módulo “GSM sim 800”, también debe verificar que el switch de alimentación parcial se encuentre en ON. Finalmente coloque el dispositivo sobre una base estable, y conéctelo a una fuente de alimentación, a través del Jack de 12V.

Si es la primera vez que enciende el dispositivo, espere alrededor de 4 a 7 minutos, durante este tiempo, el módulo GSM y GPS buscan los satélites y redes disponibles para establecer

la conexión, así también verifican el estado de la red y la disponibilidad de datos móviles. Todo el proceso descrito anteriormente.

Una vez establecida la conexión por primera vez con el servidor, la próxima vez que encienda el dispositivo, la conexión se establecerá inmediatamente.

Todo lo expuesto anteriormente lo puede verificar a través de los leds indicadores, que se explican a continuación.

1.2.1 Led indicador módulo GSM

- Parpadea cada segundo, mientras busca las redes disponibles e intenta establecer conexión.
- Parpadea cada dos segundos, cuando logró establecer conexión con la red GSM.
- Parpadea 2 veces por segundo aproximadamente, cunado estableció conexión con el servidor de envío de datos y logró entregar dichos datos (coordenadas GPS).
- Parpadea 2 veces cada segundo continuamente, indica error en la red, se recomienda reiniciar el dispositivo.

1.2.2 Led indicado módulo GPS

El módulo GPS cuenta con dos leds indicadores, rojo y verde. Que trabajan de la siguiente forma.

- Led rojo encendido, indica que el módulo está encendido y en operación.
- Led verde apagado, indica que el módulo está buscando los satélites para obtener las coordenadas GPS.

- Led verde encendido indica que el módulo se conectó a los satélites y está obteniendo las coordenadas GPS. (Aproximadamente después 5 o 10 segundos que se haya encendido el led verde, el módulo estará listo y calibrado para enviar las coordenadas al servidor).

Manejo app móvil

La aplicación está desarrollada en la plataforma Android Studio con lenguaje Kotlin y Java. Por tratarse de una app desarrollada para este prototipo, el archivo instalable, se lo tendrá que pasar al teléfono mediante su tarjeta sd o conexión usb.

Una vez instalada la aplicación en el terminal móvil la primera pantalla que aparecerá será la pantalla de Login, en la cual deberá ingresar sus credenciales registradas, si ya lo hizo anteriormente, si no, puede registrarse a través de la pantalla de registro de la aplicación.

Ilustración 3. Pantalla Login App Móvil. Fuente: Autor

Una vez ingresado al sistema, se encontrará con la pantalla principal en la cual podrá navegar para registrar empresas, conductores administrar su cuenta y realizar el rastreo de los vehículos ya registrados.

Dentro de la pantalla administrar, podrá personalizar su perfil, así como cambiar su contraseña y terminar de llenar sus datos como nombre completos etc.

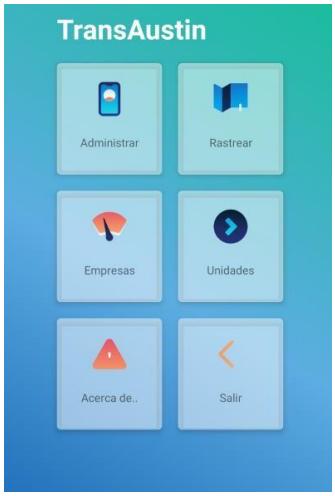


Ilustración 4. Seccion Home App Movil. Fuente: Autor

Antes de poder empezar la tarea para rastrear los vehículos, deberá registrar las empresas a las cuales pertenecerán los vehículos a ser registradas. Para lo cual deberá entrar a la pantalla de registro de empresas.

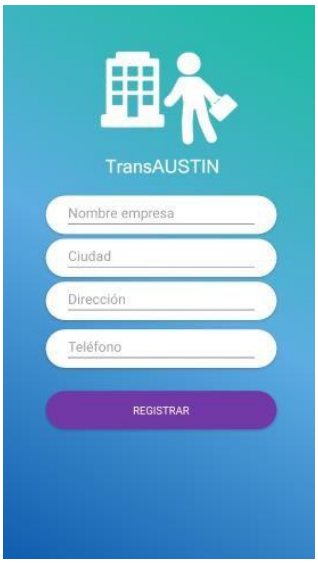


Ilustración 5. Pantalla registro empresas. Fuente: Autor

Una vez registradas las empresas, podrá registrar a los usuarios a ser rastreados, asignándole a cada una de las empresas anteriormente registradas.

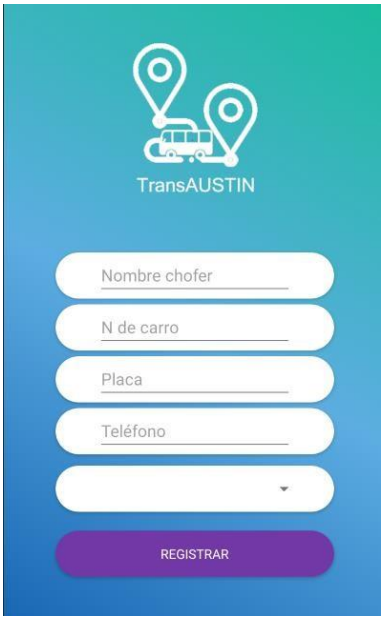


Ilustración 6. Registro de Chofer App Móvil. Fuente: Autor

Finalmente, con los usuarios a ser rastreados asignados a sus respectivas empresas, podrá iniciar la tarea para poder ubicarlos en el mapa de su localidad, a través de la pantalla de rastreo.

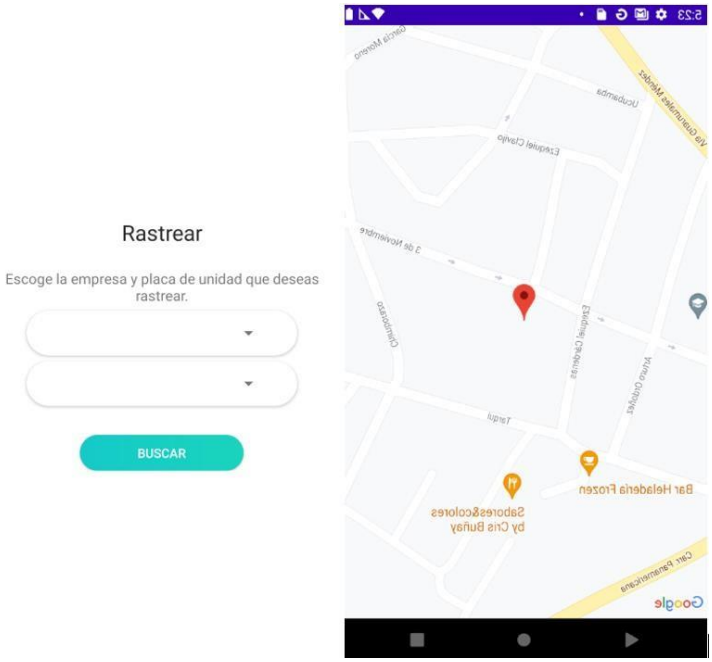


Ilustración 7 Rastreo y Mapa App Móvil. Fuente: Autor

MANUAL DEL PROGRAMADOR

El propósito de este manual del programador tiene por finalidad poner en conocimiento del lector las diferentes etapas del desarrollo del software del prototipo de GPS, así también como los diferentes lenguajes empleados y sus principales características, con el fin de que el usuario esté en la capacidad de modificar ciertos valores y parámetros a su gusto.

El presente prototipo tiene como base de programación, los lenguajes C, Kotlin y Java, desarrollados en las plataformas de programación Arduino y Android Studio, además de contar con una integración para la gestión y almacenamiento de datos con el servicio de

Google Firebase.

2.1 Requerimientos técnicos

Como requisitos mínimos del sistema para el correcto funcionamiento de las plataformas en las cuales fue desarrollado el prototipo tenemos los siguientes.

Android Studio 3.x

Windows

- Windows 7/8/10 (32 o 64 bits).
- 2 GB de RAM (8 GB de RAM recomendado).
- 2 GB de espacio libre mínimo (4 GB recomendado).
- Resolución mínima de 1.280 x 800.
- Java 8.
- 64 bits y procesador Intel (emulador).

Mac

- Mac OS X 10.8.5 o superior.
- 2 GB de RAM (8 GB de RAM recomendado).
- 2 GB de espacio libre mínimo (4 GB recomendado).
- Resolución mínima de 1.280 x 800.
- Java 6.

Linux

- GNOME o KDE Desktop.
- Ubuntu…
- 64 bits / 32 bits.

- GNU C (glibc) 2.1 o superior.
- 2 GB de RAM (8 GB de RAM recomendado).
- 2 GB de espacio libre mínimo (4 GB recomendado).
- Resolución mínima de 1.280 x 800.
- Java 8.
- 64 bits y procesador Intel (emulador).

Arduino IDE

- Windows 10 versión 14393.0 o posterior x86

2.2 Android Studio

Plug-ins utilizados

Un plug-in es un componente específico de una función que se agrega a un programa, (en este caso, Android Studio). Un complemento generalmente está diseñado para aumentar algunas características específicas con las que un programa no vino originalmente. Esto ayuda a aumentar la eficiencia de los desarrolladores. James A. 2017

- com.android.application
- kotlin-android
- com.google.gms.google-services
- kotlin-android-extensions

Dependencias

En Android Studio, las dependencias nos permiten incluir una biblioteca externa o archivos jar locales u otros módulos de biblioteca en nuestro proyecto de Android. En este

caso, se agregan los servicios de autenticación, login, vínculo para acceso y envío de datos con firebase, etc.

- implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
- implementation 'androidx.core:core-ktx:1.6.0'
- implementation 'androidx.appcompat:appcompat:1.3.1'
- implementation com.google.android.material:material:1.4.0'
- implementation androidx.constraintlayout:constraintlayout:2.1.1'
- implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.3.1'
- implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1'
- implementation 'androidx.navigation:navigation-fragment:2.3.5'
- implementation 'androidx.navigation:navigation-ui:2.3.5'
- implementation 'com.google.firebase:firebase-storage-ktx:20.0.0'
- androidtestImplementation 'junit:junit:4.+'
- androidTestImplementation 'androidx.test.ext:junit:1.1.3'
- androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
- implementation platform('com.google.firebase:firebase-bom:28.4.1')
- implementation 'com.google.firebase:firebase-analytics-ktx'
- implementation 'com.google.firebase:firebase-auth-ktx'
- implementation 'com.google.firebase:firebase-firestore-ktx'
- implementation 'com.google.firebase:firebase-database-ktx'
- implementation 'com.google.android.material:material:1.2.1'
- implementation 'androidx.cardview:cardview:1.0.0'

- implementation fileTree(dir: "libs", include: ["*.jar"])
- implementation 'com.github.bumptech.glide:glide:4.12.0'
- annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'
- implementation 'com.google.android.gms:play-services-maps:18.0.0'

Layouts y activities

Un layout define la estructura de una interfaz de usuario en su aplicación, como en una activity. Todos los elementos del diseño se crean utilizando una jerarquía de objetos View y ViewGroup. Una vista generalmente dibuja algo que el usuario puede ver e interactuar. Una actividad representa una sola pantalla con una interfaz de usuario como una ventana o un marco de Java. La actividad de Android es la subclase de la clase ContextThemeWrapper.

Dentro de los layouts declaramos los id de los elementos que utilizaremos para la recolección y envío de datos, ya sea través de botones, textiles, menús de ítems, los cuales serán los datos que utilizaremos para interactuar con los activities etc. A continuación, se detalla los Id declarados en los distintos layouts.

2.2.3.1 Activity Account Recovery

Ilustración 8. Sección Recovery. Fuente: Carvajal E.

Descripción	Encargado de recuperación de cuentas a través del email, en caso de haber olvidado la contraseña
Variables layout	emailEditText senEmailAppCompatButton
Relación layout con activity	AccountRecoveryActivity
Variables Activity	emailAddress: Recupera la información ingresada por el usuario desde el layout. intent: redirecciona al activity de login, una vez comprobado que se haya realizado correctamente el reseteo de la cuenta a travez de los servicios de Google firebase.
Funciones	onCreate

Tabla 1. Account Recovery. Fuente: Carvajal E.

2.2.3.2 Activity Administrar

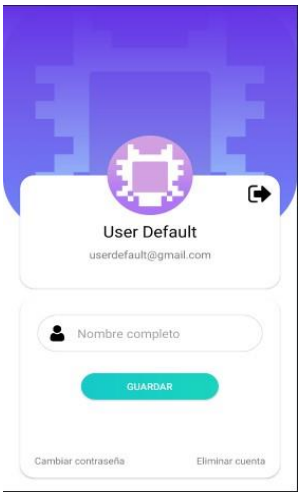


Ilustración 9.Activity Administrar. Fuente: Autor

Descripción	Encargado de la administración de la cuenta y datos del administrador de la aplicación.

Variables layout	signOutImageView nameTextView emailTextView profileImageView
	nameEditText updateProfileAppCompatButton updatePasswordTextView deleteAccountTextView
Relación layout con activity	AdministrarActivity
Variables Activity	name: Recupera la información ingresada en el textbox nameEditText del layout. intent: Variable utilizada para llamar a las diferentes activities de acuerdo a como lo necesite el programa. auth: Variable encargada de traer a la propiedad de autenticación de la plataforma firebase a través de las librerías agregadas anteriormente user: Extrae la información del usuario actualmente logueado en el sistema. profileUpdates: Actualiza el nombre del usuario a través de la función “userProfileChangeRequest” uri: Recibe la imagen subida por el usuario en su perfil folder: Almacena la imagen almacenada en la variable uri a través de la función FirebaseStorage

Funciones	onCreate: Encargada de mostrar pon pantalla el layout administración fileManager: Encargada de gestionar los datos almacenados del usuario como su nombre o imagen de perfil Volver: Encargada de volver al activity del menú principal. updateProfile: Función encargada de actualizar el nombre del usuario. updateUI: Función encargada de actualizar la imagen de perfil. imageUpload: Función encargada de subir la imagen proporcionada por el usuario para su perfil

Tabla 2. Administración. Fuente: Carvajal E.

2.2.3.3 Activity Autenticación

Ilustración 10. Login. Fuente: Autor

Descripción	Encargado de la autenticación del usuario, y las opciones para direccionamiento a las opciones de registro.

Variables layout	passwordEditText emailEditText signInButton recoveryAccountTextView registerHere
Relación layout con activity	AuthActivity
Variables Activity	Auth: Variable encargada de traer a la propiedad de autenticación de la plataforma firebase a través de las librerías agregadas anteriormente. mEmail: Recibe el email ingresado por el usuario a través del auth layout. mPassword: Recibe el password ingresado por el usuario a través del auth layout. intent: Variable utilizada para llamar a las diferentes activities de acuerdo a como lo necesite el programa.
Funciones	onCreate: Encargada de mostrar pon pantalla el layout administración, recibir las variables de password e email del usuario y enviarlas a la función signIn. OnStart: Encargada de verificar si existe un usuario loggeado anteriormente y enviarlo directamente al menú principal, caso contrario se abrirá la pantalla de login. singIn: Encarga de recibir la información del usuario y enviarla a la función “signInWithEmailAndPassword” proporcinada por la biblioteca firebase, encargada de comprobar las credenciales y dar acceso al sistema showDashboard: Encargada de iniciar la activity que muestra el menú principal.

Tabla 3. Actividad de Autenticación Usuario. Fuente: Carvajal E.

2.2.3.4 Activity check

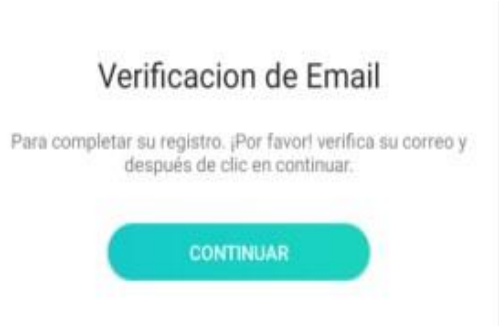


Ilustración 11. Verificación de email. Fuente: Carvajal E.

Descripción	Encargado de enviar un email de verificación al momento de realizar el registro de un nuevo usuario en el sistema.
Variables layout	veficateEmailAppCompatButton
Relación layout con activity	CheckActivity
Variables Activity	Auth: Variable encargada de traer a la propiedad de autenticación de la plataforma firebase a través de las librerías agregadas anteriormente.
	user: Recibe los datos del usuario que intenta loggearse en el sistema, a través de la función currentUser de firebase. intent: Variable utilizada para llamar a las diferentes activities de acuerdo a como lo necesite el programa.

Funciones	onCreate: Encargada de mostrar pon pantalla el layout checkActivity. OnStart: Encargada de verificar si el usuario que intenta ingresar al sistema ha confirmado su email, caso contrario lo dirigirá a la pantalla de verificación. sendEmailVerification: a través de la función "sendEmailVerification" de firebase envía un email al correo proporcionado por el usuario en el momento de realizar su registro. signOut: Carga la activity de loggin a través de un intent. showDashboard: Encargada de iniciar la activity que muestra el menú principal.

Tabla 4. Actividad Verificación de Email. Fuente: Carvajal E.

2.2.3.5 Activity Delete Account

Ilustración 12. Eliminar Cuenta. Fuente: Carvajal E.

Descripción	Encargado eliminar la cuenta de un usuario en caso de requerirlo a través de su contraseña.
Variables layout	passwordEditText
	deleteAccountAppCompatButton
Relación layout con activity	Activity_delete_account

Variables Activity	Auth: Variable encargada de traer a la propiedad de autenticación de la plataforma firebase a través de las librerías agregadas anteriormente. Password: Recibe el password ingresado por el usuario para iniciar el proceso para eliminar la cuenta. User: Recupera los datos del usuario loggeado en el sistema a traves de la función auth de firebase auth, current user. Email: Recupera el email del usuario loggeado en el sistema. intent: Variable utilizada para llamar a las diferentes activities de acuerdo a como lo necesite el programa.
Funciones	onCreate: Encargada de mostrar pon pantalla el layout checkActivity. deleteAccount: Recibe el password ingresado por el usuario y atraves de la función user.delete, elimina la cuenta correspondiente al usuario loggeado. signOut: Una vez eliminada la cuenta dirige a la pantalla de login.

Tabla 5. Actividad Eliminar Cuenta. Fuente: Autor

2.2.3.6 Activity main

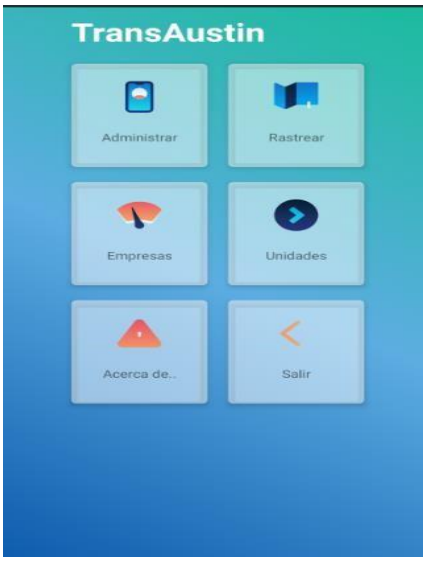


Ilustración 13. Menú Principal. Fuente: Autor

Descripción	Menú principal de la aplicación
Variables layout	cv_administrar cv_rastrear cv_empresas cv_unidades cv_about cv_salir
Relación layout con activity	Activity_main
Variables Activity	Auth: Variable encargada de traer a la propiedad de autenticación de la plataforma firebase a través de las librerías agregadas anteriormente. intent: Variable utilizada para llamar a las diferentes activities de acuerdo a como lo necesite el programa.
Funciones	onCreate: Encargada de mostrar pon pantalla el layout MainActivity. signOut: Una vez eliminada la cuenta dirige a la pantalla de login.

Tabla 6. Main Activity. Fuente: Autor

2.2.3.7 Activity Empresas

Ilustración 14. Menú Empresas. Fuente: Autor

Descripción	Encargada de registrar las empresas a las cuales se le anexará los choferes a ser rastreados.
Variables layout	nameEmpresaText cityEmpresaText addressEmpresaText phoneEmpresaText registerREmpresa
Relación layout con activity	Empresas Activity
Variables Activity	Auth: Variable encargada de traer a la propiedad de autenticación de la plataforma firebase a través de las librerías agregadas anteriormente. Database: Encargada de realizar la referencia a la base de datos Firebase Realtime Database Db: Encargada de crear las isntalacias necesarias para la base de datos FireStore Variables que reciben la información ingresada por el usuario: eName, eCity, eAddress, ePhone Empresa: Crea un HashMap de los datos ingresados por el usuario para eniarlas a FireStore.

Funciones	onCreate: Encargada de mostrar pon pantalla el layout EmpresasActivity.

Tabla 7. Activity Empresas

2.2.3.8 Activity Unidades

Ilustración 15. Menú Unidades. Fuente: Autor

Descripción	Encargada de registrar a los choferes a ser rastreados en dependencia a cada empresa ya registrada.
Variables layout	uName carNumber carPlate uPhone idSppiner registrarRVButton
Relación layout con activity	Activity_Unidades

Variables Activity	Auth: Variable encargada de traer a la propiedad de autenticación de la plataforma firebase a través de las librerías agregadas anteriormente. Database: Encargada de realizar la referencia a la base de datos Firebase Realtime Database Db: Encargada de crear las isntalacias necesarias para la base de datos FireStore Variables que reciben la información ingresada por el usuario: uName, uPlaca, uNumber, uPhone. Name: Recive los nombres de las empresas ya registradas para poder asignarlas a los choferes. Spinner: Item menú en el cual se visualizan los nombres de las empresas ya registradas Unidad: Crea un hasmap con los datos del chofer ingresados por el usuario, para enviarlos a FireStore. Latitud, longitud y velocidad: Variables creadas inicialmente con datos nulos, las cuales serán enviadas a Realtime database junto con la variable que contiene la placa del vehículo.:
Funciones	onCreate: Encargada de mostrar pon pantalla el layout UnidadesActivity. writeNewCar: Función encargada de recibir la información de la placa del vehículo de cada chofer y almacenarla en Realtime DataBase, junto con los valores nulos inicales de latitud, longitud y velocidad.

Tabla 8. Activity Unidades. Fuente: Autor

2.2.3.9 Activity Buscar

Ilustración 16. Menú Buscar Empresa. Fuente: Autor

Descripción	Encargada de realizar la selección entre las empresas y choferes para realizar el rastreo.
Variables layout	idSpinnerEmpresa idSppinerUnidad searchButtom
Relación layout con activity	SearchActivity

Variables Activity	Auth: Variable encargada de traer a la propiedad de autenticación de la plataforma firebase a través de las librerías agregadas anteriormente. Database: Encargada de realizar la referencia a la base de datos Firebase Realtime Database Db: Encargada de crear las isntalacias necesarias para la base de datos FireStore. idUser: Recupera el id del usuario actualmente loggeado a través de la función auth.currentUser.uid uId: Convierte la variable IdUser a string para utilizarla en consultas y comparación más adelante. list: Crea un array tipo cadena para almacenar los ítems de los spinners. name: Realiza la consulta a la base de datos de las empresas disponibles y extrae el nombre de cada una de ellas. adaptador: Convierte el objeto tipo arraylist a un objeti tipo spinner.
	SpinnerEmpresas: Variable encargada de almacenar el valor de la variable adaptador convirtiéndola de una variable de datos a una variable visible desde la UI SpinnerPlaca: Variable que espera a recibir el valor del item seleccionado en el spinner empresas a través de un objeto tipo escucha para cargar su información de acuerdo a eso.

Funciones	onCreate: Encargada de mostrar pon pantalla el layout fillplate: Función que recibe el nombre de la empresa seleccionada en el spninner empresas para crear otro objeto tipo arraylist almacenada en la variable "listu" la cual será llenada a través de la consulta a la base de datos con la información de las unidades disponibles a ser rastreadas. Finalmente almacena la placa del vehículo seleccionado en el spinnerPlaca y la pasa al activity localización.

Tabla 9. Activity Buscar

2.2.3.10 Activity Maps

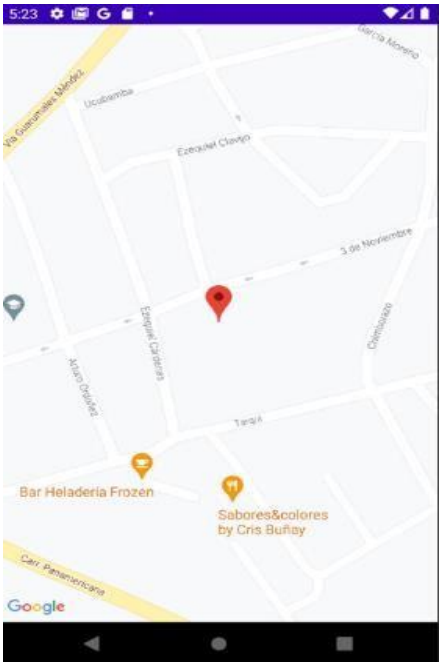


Ilustración 17. Menú Rastreo. Fuente: Autor

Descripción	Encargado eliminar la cuenta de un usuario en caso de requerirlo a través de su contraseña.
Variables layout	----------
Relación layout con activity	MapsActivity

Variables Activity	nMap: Encargada de cargar la relación con el api de Google mapas proporcionada por Google firebase. Database: Encargada de realizar la referencia a la base de datos Firebase Realtime Database Db: Encargada de crear las isntalacias necesarias para la base de datos FireStore mpaFragment: Crea un fragment que contendrá el mapa durante el rastreo. Data: Variable encargada de recibir los valores de latitud y longitud actual en base a la placa seleccionada en la activity de búsqueda, las convierte a tipo double y pasa esta información al activity Rastreo.
Funciones	onCreate: Encargada de mostrar y crear el fragment que contendrá el mapa. onMapReady: Función encargada de recibir periódicamente la información en tiempo real de la latitud y longitud, desde Realtime Database, y pasa dicha información al activity rastreo.
Funciones activity rastreo	OnDataChange: cada vez que la activiy mapas envía los datos del vehículo a ser rastreado, OnDataChange Posiciona los datos recibidos dentro del mapa de Google y moviendo el indicador de posicionamiento en tiempo real, a través de la función (tmpRealTimemarket).

Tabla 10. Activity Rastreo - Mapa. Fuente: Autor

2.3 Arduino

El prototipo está desarrollado en Arduino, complementado con ciertas librerías que se encargan del manejo de los módulos GSM y GPS, además de que el código total está

desarrollado para el módulo esp8266, el cual debe ser agregado, desde un repositorio y el gestor de tarjetas como se explica a continuación.

2.3.1 Gestor de tarjetas

Este es el método elegido para los usuarios finales.

Pre requisitos

- Arduino 1.6.8, descárgalo de la página web de Arduino:
- Conexión a internet

Instrucciones

- Inicia Arduino y abre la ventana Preferencias.
- Introduce http://arduino.esp8266.com/stable/package_esp8266com_index.json en la casilla Gestor de URLs Adicionales de Tarjetas. Puedes añadir múltiples URLs separándolas con comas.
- Abre en gestor de tarjetas desde Herramientas > «Última tarjeta seleccionada» > Gestor de tarjetas y busca la plataforma esp8266.
- Seleccione la versión que desee de la lista.
- Click en el botón Instalar.
- No olvides seleccionar tu tarjeta ESP8266 desde Herramientas > Menú de tarjetas tras la instalación.

2.3.2 Librerías

- TINY_GSM_MODEM_SIM800: Librería encargada de simular la operación de la tarjeta esp8266 como una terminal móvil, conjuntamente con el módulo sim 800l
- TinyGPS++.h: Librería encargada de gestionar la información recibida del módulo GPS Neo 6M.
- TinyGsmClient.h: Librería encargada de poner al moódulo esp8266 en modo cliente con el módulo gsm para la recepción y envío de comandos AT.
- ArduinoHttpClient.h: Librería encargada de poner a nuestro circuito en modo cliente http, para el envío de datos.
- SoftwareSerial.h: Librería utilizada para la gestión de datos a través del puerto serial del Arduino, en este caso servirá para el monitoreo del sistema durante el proceso de compilación.

2.3.3 Variables

2.3.3.1 Variables de conexión con firebase y comunicación software

FIREBASE_HOST []: Dirección del host que almacena la base de datos, en este caso se refiere a la “RealTime database de firebase”.

FIREBASE_AUTH: Clave de Api de firebase.

FIREBASE_PATH: Ruta a la cual será subida la información. (Placa del automóvil en la cual irá instalado el dispositivo) SSL_PORT: Puerto SSL de firebase.

apn[]: Apn de la operadora telefónica a la cual pertenece el chip. user[]: Usuario de la operadora telefónica a la cual pertenece el chip. pass[]: Contraseña de la operadora telefónica a la cual pertenece el chip.

2.3.3.2 Variables definidas para comunicación de hardware

Sim800Serial (): Inicialización de la comunicación serial para el módulo GSM.

Rxpin: Declaración del pin receptor de datos GSM en la tarjeta esp8266.

Txpin: Declaración del pin de transmisión de datos GSM en la tarjeta esp8266.

Modem (Sim800Serial): Variable encargada de iniciar la librería Tiny Gsm con la información almacenada en la variable sim800Serial.

SoftwareSerial ss(): Inicialización de la comunicación serial para el módulo GPS

rxd2: Declaración del pin receptor de datos GPS en la tarjeta esp8266. txd2: Declaración del pin receptor de datos GPS en la tarjeta esp8266.

TinyGPSPlus gps: Variable encargada de iniciar la librería Tiny GPS plus con la información almacenada en la variable Software serial ss.

2.3.3.3 Variables de programación.

http_client: Variable encargada de almacenar las variables de autenticación con firebase y enviarlas a la librería HttpClient. previousMillis: Variable para conteo de en milisegundos, para la toma de datos del GPS.

Interval: Intervalo de tiempo en los cuales serán tomados los datos de posicionamiento.

Response: Variable encargada de recibir la respuesta del servidor, del estado del envío de datos.

url: Variable tipo string, en la cual se almacenará la cadena tipo json para el envío de datos.

contentType: Declara el tipo de dato que se enviará al servidor.

Latitude: Recibe los datos del gps y los convierte en coordenadas a través de la función “gps.location.lat()”.

Longitude: Recibe los datos del gps y los convierte en coordenadas a través de la función “gps.location.lng()”.

Velocidad: Recibe los datos del gps y los convierte en datos de velocidad a través de la función “gps.speed.kmph()”.

gpsData: Cadena tipo string en la cual se arma la cadena de texto con los datos obtenidos del GPS para posteriormente ser enviados por json.

2.3.4 Funciones

Void setup():

Función encargada de inicializar las variables de comunicación serial con el hardware y poner a la tarjeta esp8266 en modo modem.

Void loop ():

Una vez inicializada la comunicación serial, void loop, se encarga de inicializar las credenciales de la operadora, perteneciente al chip insertado en el módulo sim 800l, verifica la conexión correcta con la red móvil y en caso de que sea exitosa dicha conexión, envía la ejecución del programa al loop gps.

Void gps_loop ():

Función encargada de comprobar que el módulo GPS está obteniendo datos de algún satélite, obtener los datos y armar la cadena de texto con todos estos datos,

además de las credenciales para la conexión con firebase. Finalmente envía todos estos datos a la función PostToFirebase

Void PostToFirebase ():

Recibe los datos desde la función void gps loop, abre la conexión con el servidor, la mantiene abierta mientras arma la cadena json para el envío de todos los datos, espera la respuesta del servidor, e imprime por pantalla dicha respuesta, finalmente cierra la conexión con el servidor y vuelve a iniciar todo el proceso.

2.4FireBase

2.4.1 Google Autentication

Google firebase proporciona una herramienta ponte para la gestión de usuario, ya sea a través de diferentes servicios como Google, Facebook correos electrónicos, etc.
En este proyecto se ha utilizado la opción de correo electrónico, parala gestión de los usuarios, la cual se habilita a través de la consola de firebase.

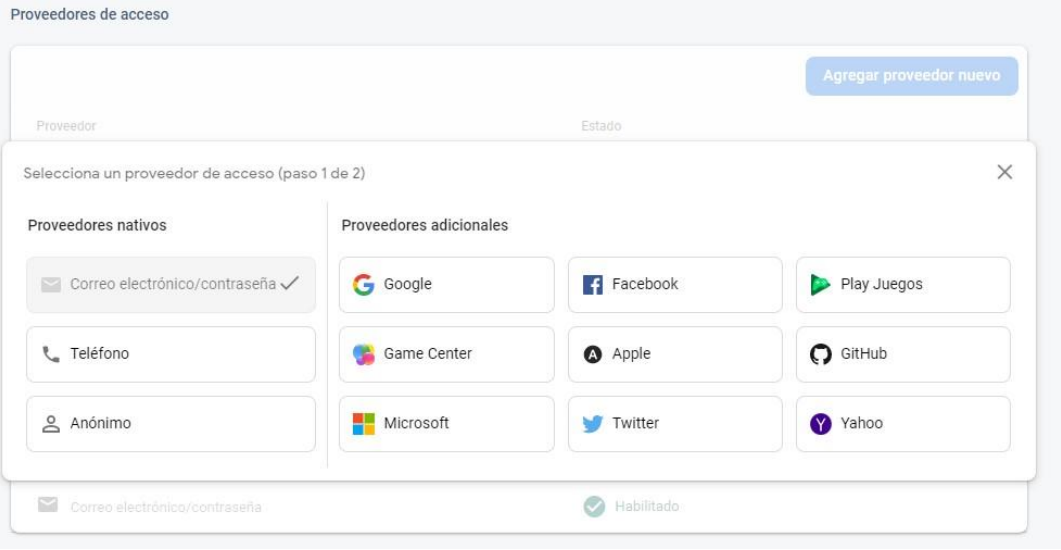


Ilustración 18. Servicios de autenticación proporcionados por Firebase. Fuente: console.firebase.google.com/project/transaustin-9e76e/authentication/providers

2.4.2 Firebase firestore

Cloud Firestore es una base de datos flexible y escalable para el desarrollo en servidores, dispositivos móviles y la Web desde Firebase y Google Cloud. Al igual que Firebase Realtime Database, mantiene tus datos sincronizados entre apps cliente a través de objetos de escucha en tiempo real y ofrece soporte sin conexión para dispositivos móviles y la Web, por lo que puedes compilar apps con capacidad de respuesta que funcionan sin importar la latencia de la red ni la conectividad a Internet.

El modelo de datos de Cloud Firestore admite estructuras de datos flexibles y jerárquicas. Es una base de datos no relacional (NoSQL) que almacena tus datos en documentos, organizados en colecciones. Los documentos pueden contener objetos anidados complejos, además de sub colecciones.

En este caso tenemos dos colecciones padres estructurados de la siguiente manera:

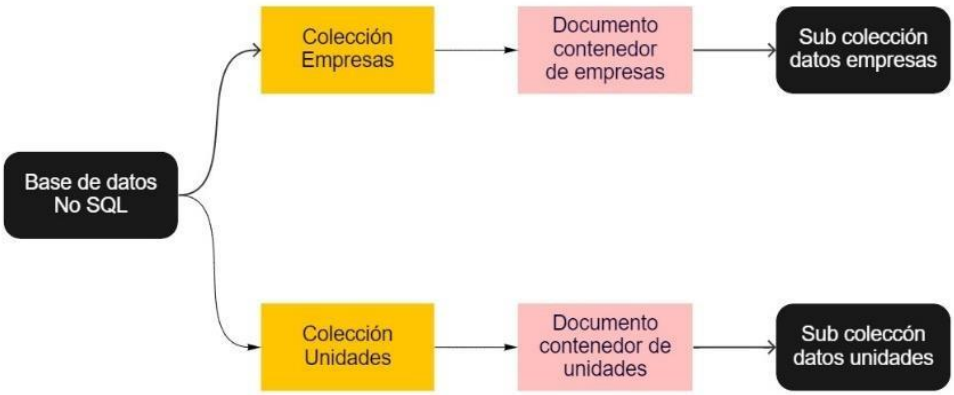


Ilustración 19. Estructura BBDD Firestore. Fuente: Autor

2.4.3 Base de Datos Tiempo Real

Firebase Realtime Database es una base de datos alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. Cuando compilas apps multiplataforma con nuestros SDK de plataformas de Apple, Android y JavaScript, todos tus clientes comparten una instancia de Realtime Database y reciben actualizaciones automáticamente con los datos más recientes.

En lugar de solicitudes HTTP típicas, Firebase Realtime Database usa la sincronización de datos (cada vez que cambian los datos, los dispositivos conectados reciben esa actualización en milisegundos). Proporciona experiencias colaborativas y envolventes sin pensar en el código de red.

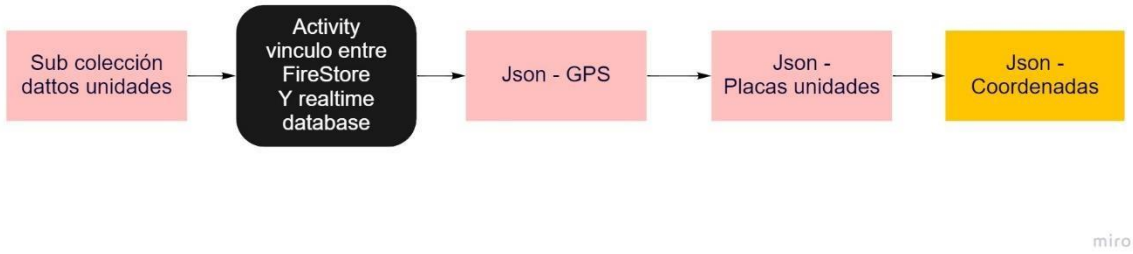


Ilustración 20. Estructura de Base de Datos en Tiempo Real. Fuente: Autor

ANEXOS

Anexo 1: Protocolo de Tesis

Anexo: Formato del Anteproyecto.

A. TÍTULO
Análisis de un prototipo para un aplicativo móvil que permita la geolocalización con la utilización IoT para las unidades de transporte urbano del cantón Cañar.

B. DOMINIO, LÍNEA Y ÁMBITOS DE INVESTIGACIÓN			
Energía eléctrica y tecnologías de la información para la innovación y el desarrollo sostenible	**Ciencia de los ordenadores, Analítica de datos y Algoritmos computacionales**	Analítica de Datos	
		Ingeniería de Software	x
		Algoritmos computacionales	
		Inteligencia de negocios	
		Gobierno de TI	
		Auditoria y seguridad informática	
		Simulación	

C. PLANTEAMIENTO DEL PROBLEMA

Hoy en día en el Ecuador existe una erudición establecida de parte de los choferes de transportación publica de no tener responsabilidad al momento de conducir los buses que radica en el incumplimiento de rutas, horarios y reglas de circulación ha provocado que los buses excedan sus límites y se generen accidentes.

Existen regulaciones establecidas por las autoridades competentes para normalizar que los transportistas sigan el recorrido determinado; sin embargo, no garantiza que se cumpla con la ruta, inclusive ha existido quejas que en muchas ocasiones los choferes optan por caminos no determinados y exceden la velocidad del vehículo.

Esto ha causado disgusto e inseguridad por parte de la ciudadanía, pues, no pueden predecir si una unidad va a cumplir con las reglas de circulación. En consecuencia, se propone implementar un prototipo de sistema que brinde el control y seguimiento de la unidad de transporte urbano. Por otra parte, permita obtener una gestión optimizada.

D. OBJETIVO GENERAL

Diseñar un prototipo para un aplicativo móvil que permita la geolocalización con la utilización IoT para las unidades de transporte urbano del cantón Cañar.

E. OBJETIVOS ESPECÍFICOS

1. Definir las variables de estimación de geolocalización con la utilización IoT para las unidades de transporte urbano del cantón Cañar.
2. Diseñar el módulo eléctrico que será implementado en el bus de transporte urbano para lograr obtener las coordenadas para las unidades de transporte urbano del cantón Cañar.
3. Desarrollar la aplicación que permita visualizar la geolocalización con la utilización IoT para las unidades de transporte urbano del cantón Cañar

F. JUSTIFICACIÓN

El elevado costo de los sistemas de seguridad vehicular existentes en el mercado ecuatoriano hace que los propietarios de los vehículos no opten por dispositivos inteligentes para proteger el mismo, lo cual origina que sus propietarios estén expuestos a riesgos.

El prototipo que se desarrolló pretende combinar la tecnología GPS, junto con red móvil celular y de esta manera lograr una automatización del servicio de transporte urbano. Esto se logra a través de registrar la ubicación de las unidades de transporte urbano y de esta manera lograr determinar que cumpla con la ruta establecida y los reglamentos de circulación. Para que se pueda mejorar la calidad del servicio que brinda el transporte público.

G. ALCANCE

- Este sistema se desarrollará una aplicación móvil Android para el control de la ubicación de la unidad de transporte.
- Desarrollo de prototipo funcional en un microcontrolador Arduino que se colocara en el vehículo para obtener la localización de este.
- Se implementará módulos GPS, GSM que permitirá obtener la ubicación y velocidad del vehículo.
- Base de datos alojada en un servidor gratuito.

H. CONCEPTOS RELACIONADOS

GPS:

El GPS se compone de tres elementos: los satélites en órbita alrededor de la Tierra, las estaciones terrestres de seguimiento y control, y los receptores del GPS propiedad de los usuarios. Desde el espacio, los satélites del GPS transmiten señales que reciben e identifican los receptores del GPS; ellos, a su vez, proporcionan por separado sus coordenadas tridimensionales de latitud, longitud y altitud, así como la hora local precisa.

Hoy están al alcance de todos en el mercado los pequeños receptores del GPS portátiles. Con esos receptores, el usuario puede determinar con exactitud su ubicación y desplazarse fácilmente al lugar a donde desea trasladarse, ya sea andando, conduciendo, volando o navegando. El GPS es indispensable en todos los sistemas de transporte del mundo ya que sirve de apoyo a la navegación aérea, terrestre y marítima. Los servicios de emergencia y socorro en casos de desastre dependen del GPS para la localización y coordinación horaria de misiones para salvar vidas. Actividades cotidianas como operaciones bancarias, de telefonía móvil e incluso de las redes de distribución eléctrica, ganan en eficiencia gracias a de la exactitud cronométrica que proporciona el GPS. Agricultores, topógrafos, geólogos e innumerables usuarios trabajan de forma más eficiente, segura, económica y precisa gracias a las señales accesibles y gratuitas del GPS. [1]

Geolocalización:

La Geolocalización es el posicionamiento en el que se define la localización de un objeto espacial, representado mediante punto, vector, polígono, área, volumen en un sistema de coordenadas. Este proceso es utilizado frecuentemente en los Sistemas de Información Geográfica.

Posee una definición tecno científica, aplicada a la existencia de las cosas en un espacio físico, mediante el establecimiento de relaciones entre las imágenes de ráster o vector sobre una proyección geográfica o sistemas de coordenadas. Por ellos la

Georreferenciación adquiere una gran importancia para los modelos de datos realizados por los SIG.

El proceso de georreferenciación se basa en la introducción de puntos de imagen sin referencia espacial, y de forma similar determina los puntos que debe tener el archivo en condiciones reales. Gracias a ellos, se establece una relación de equivalencia entre el punto de no referencia y el punto de referencia, de manera que el archivo ráster puede obtener la ubicación geográfica que le corresponde en circunstancias normales. [2]

Arduino:

Arduino es una gama de circuitos electrónicos open source, basados la mayor parte en un microcontrolador del fabricante Atmel. Estos circuitos integran los componentes necesarios para permitir un uso rápido y sencillo del microcontrolador. Esta simplificación esta orientada a hacer accesible a todos la creación y la programación de objetos o dispositivos interactivos. Estos objetos pueden contener todo tipo de captadores, indicadores luminosos o interruptores que queramos.

Entre otros, las tarjetas Arduino están equipadas con conectores estandarizados para conectar módulos compatibles, llamados shields. Estos últimos son circuitos de un tamaño mas o menos parecidos al de Arduino y que se apilan sobre estos conectores. Ofrecen extensiones de hardware que permiten añadir funcionalidades originales a su proyecto. Además de estos conectores, las tarjetas tienen conectividad USB, que permitan programar fácilmente el microcontrolador que incorporan. [3]

Tecnología GPRS/GSM:

La tecnología GSM (o Global System for Mobile communications por sus siglas en inglés) es un sistema estándar referente a la telefonía móvil digital, y nos permite conectar dispositivos que cuenten con esta tecnología a nivel mundial. GSM se considera un estándar de segunda generación, debido a su velocidad de transmisión, ya que su extensión de tercera generación denominada UMTS ofrece mayor velocidad de transmisión, lo que hace posible la aplicación de internet de banda ancha (mejor conocido como internet 3G), además de emplear diferentes protocolos de radio como el W-CDMA.

Una de las características mas notables de esta tecnología es la utilización de tarjetas SIM, la cual es una tarjeta desmontable donde se almacenan datos referentes a la

suscripción del usuario con la compañía telefónica, parámetros de la red y directorio telefónico. Por este medio se puede obtener el número de teléfono del usuario, y la compañía a la cual el usuario esta activado. [4]

Android:

Android es un sistema operativo y una plataforma software, basado en Linux para teléfonos móviles. Además, también usan este sistema operativo (aunque no es muy habitual), tablets, netbooks, reproductores de música e incluso PC's. Android permite programar en un entorno de trabajo (framework) de Java, aplicaciones sobre una máquina virtual Dalvik (una variación de la máquina de Java con compilación en tiempo de ejecución). Además, lo que le diferencia de otros sistemas operativos, es que cualquier persona que sepa programar puede crear nuevas aplicaciones, widgets, o incluso, modificar el propio sistema operativo, dado que Android es de código libre, por lo que sabiendo programar en lenguaje Java, va a ser muy fácil comenzar a programar en esta plataforma. Dado que Android está basado en el núcleo de Linux, tiene acceso a sus recursos, pudiendo gestionarlo, gracias a que se encuentra en una capa por encima del Kernel, accediendo así a recursos como los controladores de pantalla, cámara, memoria flash, etc. [5]

IoT (Internet of Things):

IoT visualiza un mundo totalmente conectado, donde las cosas están capaz de comunicar datos medidos e interactuar con los demás. Esto hace posible una representación digital del mundo real, a través del cual muchas aplicaciones inteligentes en una variedad de industrias. Pueden ser desarrollado. Estos incluyen: Smarthomes, Weareables, Smart ciudades, salud, automoción, medio ambiente, agua inteligente, etc. Las soluciones de IoT se están implementando en muchas áreas, optimizando las industrias de la producción. Características muy específicas, grandes volúmenes de datos y necesidades grandes y potencia durante largos períodos. Se ha aplicado a muchas otras áreas, y la información inmutabilidad está garantizada en aplicaciones que va más allá de las criptomonedas. [6]

GIS:

Se podría definir a los Sistemas de Información Geográfica como herramientas informáticas, capaces de gestionar y analizar la información georreferenciada, con vistas a la resolución de problemas de base territorial y medioambiental. El primero de los términos pondría el acento en el carácter computarizado del tratamiento de la información. Se trataría, por tanto, de la realización de operaciones automáticas a través de los ordenadores. En segundo lugar, merece destacarse que se dirige a la gestión, análisis y modelización de información geográfica, se refiere a información que se distinguen por la existencia de una doble componente: temática y espacial- en este sentido, los SIG se hallan dirigidos al conocimiento y estudio de las estructuras espaciales, donde la posición relativa que ocupan los elementos geográficos relativo a un determinado fenómeno resulta esencial. Finalmente, los SIG aparecen como herramientas multipropósito, dirigidas a la solución de problemas en campos tan dispares como la planificación territorial, la gestión catastral, la prevención de riesgos natrales o el análisis de mercados. [7]

I. TRABAJOS RELACIONADOS

Para el presente proyecto se toma como referencia los siguientes trabajos y se puntualizara los temas que nos servirán.

Dentro de estos trabajos previos [8] se desarrolló un artículo sobre el **"Monitoreo y localización de personas extraviadas utilizando Arduino y GSM/GPS"** en México, el cual nos imparte información sobre la variedad que se puede implementar IoT, en esta investigación está encaminado a la localización de personas extraviadas; desarrollaron un sistema basado en Arduino y GSM/GPRS para la monitorización de personas seguro y de bajo costo, en el cual se implementó una shield GSM Icomsat v 1.1, que es una placa compatible con Arduino, capaz de enviar mensajes de texto a partir de un chip de alguna compañía telefónica celular; trabaja con 4.5V hasta 5.5V suministrado por Arduino y es necesario programarse para determinar los envíos de mensajes y el procesamiento de los datos entrantes.

Del presente trabajo fue de gran ayuda la información impartida sobre los procedimientos y técnicas a seguir en la administración de la energía, las conversiones realizadas en ello que apoyara sumamente en el desarrollo de la habilitación y el funcionamiento correcto de la alimentación del módulo eléctrico.

Otro trabajo proyecto desarrollado en la misma área [9], se desarrolló una investigación sobre un **"Sistema de Geolocalización de vehículos a través de la red GSM/GPRS y tecnología Arduino"** en Colombia. Este sistema fue diseñado para la recepción y almacenamiento de las variables de posicionamiento en tiempo real, utilizando el estándar TCP/ IP. Los dispositivos que hacen posible el funcionamiento del sistema primeramente, el proceso de obtención de datos se realiza por medio de un módulo GPS L80, cuyo propósito es obtener la información sobre la ubicación del vehículo, por medio de la triangulación con los satélites a los cuales se conecta el dispositivo; una tarjeta Arduino UNO, encargada de tratar la información proporcionada por el GPS para transmitirla a través de Internet y un módulo GPRS SIM900, que a través de datos móviles e instrucciones programadas mediante comandos AT es usado para enviar las coordenadas geográficas a través del protocolo TCP/IP por medio de una petición GET utilizando HTTP.

La información transmitida se recibe en un servidor web, el cual funciona bajo Apache 2.4, que es un servidor web libre, se utilizó el sistema Raspberry Pi 3 como host, éste aloja una base de datos y un aplicativo en PHP encargado de la inserción de los valores en la base de datos.

El tiempo entre cada uno de los envíos de datos fue configurado dentro del algoritmo desarrollado en Arduino. Respecto al servidor, se obtuvo la información de la posición del prototipo en formato JSON a partir de la tabla GPSTracker, que hizo más sencilla la consulta de los datos a través de PHP para después mostrarlos en un sitio web.

Este trabajo me beneficio en adquirir el conocimiento sobre las herramientas y el procedimiento seguido de cada uno de los dispositivos y sistema embebidos con ayuda de los respectivos comandos del cual se componen el Arduino para la transmisión de los datos mediante los protocolos TCP/IP respecto al servidor que se utilizó en la inserción de los datos para las consultas que se visualizaron en el sitio web el cual es la diferencia respecto a mi proyecto considerado ya que se realizara en una aplicación móvil basado en Android.

El presente proyecto desarrollado en la misma área [10] de la Universidad Politécnica Salesiana de Guayaquil presenta un sistema de **"Control y Monitorización del recorrido de los buses de**

transporte público mediante tecnología GPS y GSM" el cual brinda un esquema del sistema que se divide en dos partes: equipos remotos y estación base, la cual recepta la información de los dos equipos remotos que es enviada por medio de SMS, por motivo de logística de la operadora celular local es la opción más adecuada para enviar los datos.

Los equipos remotos se encuentran estructurados por: dispositivo GPS, Microcontrolador y Modem Celular. Los cuales se encuentran comunicados de forma serial, teniendo en el microcontrolador el elemento principal del equipo remoto, en el cual se programa las interfaces de los otros dos periféricos mencionados. La estación base de encuentra estructura por un Dspic y un Modem GSM, la elección de este tipo de microcontrolador, se basa por el motivo que se quiera expandir las unidades se necesita de un dispositivo que recepte la información y la procese correctamente a mayor velocidad que los microcontroladores de gama media. Los datos son receptados por el Modem Celular y mediante un Dspic se envía una trama serial, la cual hace que la información que reciba el Modem Celular automáticamente sea enviada por el Dspic hacia el computador Base, en la cual se encuentra la base de datos del Sistema de Monitorización.

De este trabajo me resulto sumamente servicial a entender de mejor manera sobre el funcionamiento de la tecnología GSM que resulta ser un sistema de gran importancia a nivel mundial, porque permite la movilidad de una red, además es parte importante de la infraestructura de la sociedad, ya que se utiliza para las comunicaciones, red celular e internet. Al utilizar dicha red inalámbrica móvil, no se requiere de cables ya que es alimentada por las baterías mencionadas del antecedente anterior.

J. METODOLOGÍA

La metodología que se utiliza para el desarrollo del presente proyecto de investigación se utilizara el método Mixto.

La aplicación de este método, para adquirir las diferentes técnicas para georreferenciación, tomando como referencia las distintas tecnologías existentes en el mercado.

De igual manera se implementará en este proyecto la metodología ágil como Programación extrema para la creación de un sistema embebido, además el Modelo Cascada para el desarrollo de aplicativos móviles y la integración de placas para el rastreo vehicular, en el cual debe ser un proceso lineal estableciendo con un plan paso a

paso, completando un requerimiento tras otro sin volver a los anteriores, por lo cual esta propuesta se compone de las siguientes etapas:

- Requisitos del proyecto.
- Diseño
- Implementación
- Verificación
- Mantenimiento

K. CRONOGRAMA DE ACTIVIDADES

N°	ACTIVIDAD	MES						MEDIOS DE VERIFICACIÓN
		I	II	III	IV	V	VI	
1	**Definir las variables de estimación de geolocalización con la utilización IoT para las unidades de transporte urbano del cantón Cañar.**							
1.1	Explorar proyectos de investigación relacionados al tema	x	x					Antecedentes
1.2	Recolectar datos mediante una exploración de campo		x					Documentación
1.3	Realizar la comparación entre las diferentes técnicas		x					Documentación
2	**Diseñar el módulo eléctrico que será implementado en el bus de transporte urbano para lograr obtener las coordenadas para las unidades de transporte urbano del cantón Cañar.**							
2.1	Análisis de Requerimientos Herramientas y Componentes		X					Hardware
2.2	Diseño de Diagrama de los Componentes y Procesos			X				Documentación
2.3	Selección y Diseño de la Base de Datos			X				Base de Datos
2.4	Construcción del módulo eléctrico y programación de software del circuito controlador			X	X			Sistema Electrónico

3	**Desarrollar la aplicación que permita visualizar la geolocalización con la utilización IoT para las unidades de transporte urbano del cantón Cañar**							
3.2	Construcción de la aplicación móvil					X	X	Sistema Informático
3.3	Implementación de la aplicación móvil						X	Pruebas del Prototipo
3.4	Pruebas del sistema							
3.5	Manual del sistema							Manual

L. DECLARACIÓN FINAL

Los abajo firmantes declaramos bajo juramento que el proyecto descrito en este documento no ha sido presentado a otra institución nacional o internacional para su financiamiento, no causa perjuicio al ambiente, es de nuestra autoría y no transgrede norma ética alguna.

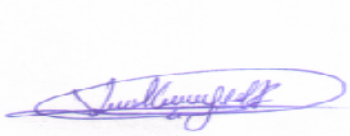
M. PARTICIPANTES

DIRECTOR:	ING. LUIS PINOS
ESTUDIANTE 1	LUIS ENRIQUE CARVAJAL ANDRADE

N. FIRMAS DE RESPONSABILIDAD

Lugar:	CAÑAR
Fecha:	03/03/2021

Firmas:



Nombre: Ing. Luis Pinos	Nombre: Luis Enrique Carvajal A.
CC: 0301829255	C.C.: 0302854005
Director del Proyecto	**Estudiante / Egresado**

O. APROBACIÓN	
Firmas:	
Nombre: CC: **Primer Par Revisor**	Nombre: C.C.: **Segundo Par Revisor**

Matriz de Congruencia					
TEMA: Análisis de un prototipo para un aplicativo móvil que permita la geolocalización con la utilización IoT para las unidades de transporte urbano del cantón Cañar.					
PROBLEMA	**OBJETIVOS**	**HIPÒTESIS**	**OPERACIONALIZACIÓN**		
			VARIABLES Y SUBVARIABLES	**INDICADORES**	**METODOLOGÍA TÉCNICAS E INSTRUMENTOS**
4.2.2Problema Principal: ¿Cómo beneficiaria la implementación de un prototipo de geolocalización de una unidad del transporte público urbano? **4.2.3Problemas Secundarios:** **P1.-** ¿Cómo construiría el módulo y el sistema móvil para obtener los datos de rastreo del vehículo? **P2.-** ¿Cómo se implementaría el prototipo para la geolocalización de la unidad de transporte urbano?	**5.1 Objetivo General:** Desarrollo de un sistema móvil de geolocalización vía GPS, con consultas mediante GSM, para la unidad de transporte público urbano, brinda apoyo a controlar mejor la ruta y velocidad del vehículo. **5.2Objetivos Específicos:** **O1.-** Diseñar un módulo electrónico y un sistema móvil que obtenga los datos de localización del vehículo. **O2.-** Implementar un prototipo para la geolocalización en una unidad de transporte urbano.	**6.1.1 Hipótesis General:** ¿El desarrollo del prototipo para supervisar la ruta y movilidad del vehículo permitirá la automatización y control? **.6.1.2Hipótesis Especificas: Primera Hipótesis** **H1**.- ¿El diseño del módulo y la aplicación móvil permitirá obtener los datos localización del vehículo? **H2.-** ¿La implementación del prototipo permitirá ubicar la unidad de transporte urbano?	**6.2 Variables de la Investigación.** **Variable Independiente:** Ubicación **Variable Dependiente:** Coordenadas	**Para la Variable Independiente:** Sistema de geolocalización **Para la variable Dependiente** Datos generados por la ubicación.	**Tipo y nivel de la Investigación:** **Tipo de la Investigación:** La metodología que se utiliza para el desarrollo del presente proyecto de investigación se utilizara el método Mixto. La aplicación de este método, Para comparar las diferentes técnicas de posicionar la ubicación del vehículo, tomando como referencia la ubicación. **Nivel de la investigación:** **a)** Investigación Exploratorio-Explicativo Con la investigación de exploratoria, reconocer e identificar los problemas de una deficiente calidad en el control y cumplimiento de la movilidad del transporte. **b)** Investigación Bibliográfica Permitirá apoyar la investigación que se desea realizar mediante una dilatada búsqueda de la información a través de la documentación de primera línea con fundamentación científica.

					Método y diseño de la Investigación: **Método:** Mixto **Diseño:** La investigación que se plantea en el presente proyecto se basa en la observación e indagación de bases teóricas, investigación no experimental, la arquitectura general del prototipo se construirá sobre el Arduino Uno, correspondiente al hardware se encuentra la representación de los módulos GSM/GPS y de software la representación del módulo eléctrico.

REFERENCIAS

[1] GPS.gov, «GPS.GOV,» NOAA, 25 Marzo 2015. [En línea]. Available: https://www.gps.gov/spanish.php. [Último acceso: 22 Febrero 2021].

[2] S. R. Juarez, Caracterizacion historica de la evolucion de la desembocadura del rio Guadalfo en la Costa Tropical, Granada: Almeria, 2014.

[3] N. GOILAV, Arduino Aprender a desarrollar para crear objetos inteligentes, Barcelona: ENI, 2016.

[4] R. Padilla, V. Quintero Rosas y A. Diaz Ramirez, «Monitoreo y localización de personas extraviadas utilizando arduino y GSM/GPS,» *Industrial Data,* vol. 18, nº 128, p. 129, 2016.

[5] M. Baez, A. Borrego, J. Cordero, L. Cruz, M. Gonzalez, F. Hernadez, D. Palomero, J. Rodrigues de Llera, D. Sanz, M. Saucedo , P. Torralbo y A. Zapata, Introducción a Android, Madrid: E.M.E, 2019.

[6] B. R. J. A. .. Dr.N.SATHISH KUMAR, «IOT Based Smart Garbage alert system using Arduino UNO,» *EL SEIVER,* vol. 18, nº 10, pp. 173-190, 2018.

[7] J. M. S. Preciado, Sistemas de Informacion Geografica, Madrid: UNED, 2020.

[8] R. Padilla, V. Quintero Rosas y A. Diaz Ramirez, «Monitoreo y localización de personas extraviadas utilizando arduino y GSM/GPS,» *Industrial Data,* vol. 18, nº 128, pp. 129-134, 2016.

[9] J. A. Castro Correa, S. B. Sepúlveda Mora, B. Medina Delgado, D. Guevara Ibarra y O. A. López Bustamante, «SISTEMA DE GEOLOCALIZACIÓN DE VEHICULOS A TRAVES DE LA RED GSM/GPRS Y TECNOLOGIA ARDUINO,» *EIA,* vol. 16, nº 31, pp. 5-10, 2019.

[10] P. f. A. Ramirez, «Dspace,» 10 Enero 2016. [En línea]. Available: https://dspace.ups.edu.ec/bitstream/123456789/2356/14/UPS-GT000128.pdf. [Último acceso: 2021 Febrero 16].

[11] M. Campoverde-Molina y L. Valverde, «Accessibility analysis of the web portals of the educational institutions in Cuenca, Ecuador,» *Revista Cátedra,* vol. 2, nº 2, pp. 55-75, 2019.

[12] V. Simbaña-Gallardo y S. Luján-Mora, «Instructions about the manuscript structureof Revista Cátedra,» *Revista Cátedra,* vol. 1, nº 1, pp. 36-52, 2018.

[13] Universidad Católica de Cuenca, «Directrices para autores/as,» 2020. [En línea]. Available: https://killkana.ucacue.edu.ec/index.php/killkana_tecnico/about/submissions.

Anexo 2: Código Fuente

Código fuente Aplicativo Móvil

- **AccountRecoveryActivity**

```
package com.example.transaustin
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import
com.example.transaustin.databinding.ActivityAccountRecoveryBindi
ng
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase

class AccountRecoveryActivity : AppCompatActivity() {
    private lateinit var binding: ActivityAccountRecoveryBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding =
ActivityAccountRecoveryBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.senEmailAppCompatButton.setOnClickListener {
            val emailAddress =
binding.emailEditText.text.toString()

Firebase.auth.sendPasswordResetEmail(emailAddress).addOnComplete
Listener { task ->
                if(task.isSuccessful) {
                    val intent = Intent(this,
AuthActivity::class.java)
                    this.startActivity(intent)
                } else {
                    Toast.makeText(this, "Ingrese un email de
una cuenta valida.",
                        Toast.LENGTH_SHORT).show()
                }
            }
        }
    }
```

- **AdministrarActivity**

```
package com.example.transaustin
import android.content.Intent
import android.net.Uri
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import com.bumptech.glide.Glide
import
com.example.transaustin.databinding.ActivityAdministrarBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
```

```
import com.google.firebase.auth.ktx.userProfileChangeRequest
import com.google.firebase.ktx.Firebase
import com.google.firebase.storage.FirebaseStorage
import com.google.firebase.storage.StorageReference

class AdministrarActivity : AppCompatActivity() {
    private lateinit var auth: FirebaseAuth
    private lateinit var binding: ActivityAdministrarBinding
    private val fileResult = 1
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding =
ActivityAdministrarBinding.inflate(layoutInflater)
        setContentView(binding.root)
        auth = Firebase.auth


        binding.signOutImageView.setOnClickListener {
            volver()
        }

        binding.updateProfileAppCompatButton.setOnClickListener
{
            val name = binding.nameEditText.text.toString()
            updateProfile(name)
        }

        binding.profileImageView.setOnClickListener {
            fileManager()
        }

        binding.updatePasswordTextView.setOnClickListener {
            val intent = Intent(this,
UpdatePasswordActivity::class.java)
            startActivity(intent)
        }

        binding.deleteAccountTextView.setOnClickListener {
            val intent = Intent(this,
DeleteAccountActivity::class.java)
            startActivity(intent)
        }

        updateUI()
    }
    private fun fileManager() {
        val intent = Intent(Intent.ACTION_GET_CONTENT)
        intent.type = "image/*"
        startActivityForResult(intent, fileResult)
    }
    private fun volver() {

        val intent = Intent(this, MainActivity::class.java)
        startActivity(intent)
    }
    private fun updateProfile(name: String) {

        val user = auth.currentUser

        val profileUpdates = userProfileChangeRequest {
            displayName = name
```

```
            }
            user!!.updateProfile(profileUpdates)
                .addOnCompleteListener { task ->
                    if (task.isSuccessful) {
                        Toast.makeText(baseContext, "Se realizaron
los cambios correctamente.",
                            Toast.LENGTH_SHORT).show()
                        updateUI()
                    }
                }
        }
        private fun updateUI() {
            val user = auth.currentUser

            if(user != null) {
                binding.emailTextView.text = user.email
                binding.nameTextView.text = user.displayName
                binding.nameEditText.setText(user.displayName)
                Glide
                    .with(this)
                    .load(user.photoUrl)
                    .centerCrop()
                    .placeholder(R.drawable.profile_photo)
                    .into(binding.profileImageView)
                Glide
                    .with(this)
                    .load(user.photoUrl)
                    .centerCrop()
                    .placeholder(R.drawable.profile_photo)
                    .into(binding.bgProfileImageView)
            }
        }
        override fun onActivityResult(requestCode: Int, resultCode:
Int, data: Intent?) {
            super.onActivityResult(requestCode, resultCode, data)
            if (requestCode == fileResult) {
                if (resultCode == RESULT_OK && data != null) {
                    val uri = data.data

                    uri?.let { imageUpload(it) }

                }
            }
        }
        private fun imageUpload(mUri: Uri) {

            val user = auth.currentUser
            val folder: StorageReference =
FirebaseStorage.getInstance().reference.child("Users")
            val fileName: StorageReference =
folder.child("img"+user!!.uid)

            fileName.putFile(mUri).addOnSuccessListener {
                fileName.downloadUrl.addOnSuccessListener { uri ->

                    val profileUpdates = userProfileChangeRequest {
                        photoUri = Uri.parse(uri.toString())
                    }

                    user.updateProfile(profileUpdates)
                        .addOnCompleteListener { task ->
```

```
                        if (task.isSuccessful) {
                            Toast.makeText(this, "Se realizaron
los cambios correctamente.",
                                Toast.LENGTH_SHORT).show()
                            updateUI()
                        }
                    }
                }
            }.addOnFailureListener {
                Log.i("TAG", "file upload error")
            }
        }
```

- **AuthActivity**

```
package com.example.transaustin
import android.app.AlertDialog
import android.content.Intent
import android.os.Binder
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.provider.ContactsContract
import android.util.Log
import android.widget.Toast
import com.example.transaustin.databinding.ActivityAuthBinding
import
com.google.android.gms.auth.api.signin.GoogleSignInOptions
import com.google.android.gms.common.api.ApiException
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import kotlinx.android.synthetic.main.activity_auth.*

class AuthActivity : AppCompatActivity() {
    private val GOOGLE_SIGN_IN = 100
    private lateinit var auth: FirebaseAuth
    private lateinit var binding: ActivityAuthBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityAuthBinding.inflate(layoutInflater)
        setContentView(binding.root)
        auth= Firebase.auth

        binding.signInButton.setOnClickListener{
            val mEmail = binding.emailEditText.text.toString()
            val mPassword =
binding.passwordEditText.text.toString()
            when {
                mEmail.isEmpty() || mPassword.isEmpty() -> {
                    Toast.makeText(baseContext, "Authentication
failed.",
                        Toast.LENGTH_SHORT).show()
                } else ->
            {
                signIn(mEmail, mPassword)
            }
            }

        }
     binding.registerHere.setOnClickListener{
         val intent = Intent(this, RegisterActivity::class.java)
```

```
            startActivity(intent)
        }

        binding.recoveryAccountTextView.setOnClickListener {
            val intent = Intent(this,
AccountRecoveryActivity::class.java)
            startActivity(intent)
        }
    }
    public override fun onStart() {
        super.onStart()
        val currentUser = auth.currentUser
        if(currentUser != null){
            if(currentUser.isEmailVerified) {
                showDashboard()
            } else {
                val intent = Intent(this,
CheckActivity::class.java)
                startActivity(intent)
            }
        }
    }
    private fun signIn(email: String, password: String){

        auth.signInWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {

                    Log.d("TAG", "signInWithEmail:success")
                    val user = auth.currentUser
                    showDashboard()

                } else {

                    Log.w("TAG", "signInWithEmail:failure",
task.exception)
                    Toast.makeText(baseContext, "Authentication
failed.",
                        Toast.LENGTH_SHORT).show()

                }
            }
    }
    private fun showDashboard(){
        val intent =Intent(this,MainActivity::class.java)
        this.startActivity(intent)
    }
```

- **CheckActivity**

```
package com.example.transaustin
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import com.example.transaustin.databinding.ActivityCheckBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.auth.ktx.userProfileChangeRequest
```

```
import com.google.firebase.ktx.Firebase

class CheckActivity : AppCompatActivity() {
    private lateinit var auth: FirebaseAuth
    private lateinit var binding: ActivityCheckBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityCheckBinding.inflate(layoutInflater)
        setContentView(binding.root)
        auth= Firebase.auth
        val user = auth.currentUser
        binding.veficateEmailAppCompatButton.setOnClickListener
{
            val profileUpdates = userProfileChangeRequest {  }

user!!.updateProfile(profileUpdates).addOnCompleteListener {
task ->
                if(task.isSuccessful) {
                    if(user.isEmailVerified) {
                        showDashboard()
                    } else {
                        Toast.makeText(this, "Por favor verifica
tu correo",
                            Toast.LENGTH_SHORT).show()
                    }
                }
            }
        }
        binding.signOutImageView.setOnClickListener {
            signOut()
        }
    }
    private fun sendEmailVerificacion(){
        val user = auth.currentUser

user!!.sendEmailVerification().addOnCompleteListener(this){
            task ->
            if (task.isSuccessful){
                Toast.makeText(this, "Se envio un correo de
verificaion",
                    Toast.LENGTH_SHORT).show()
            }
        }
    }
    public override fun onStart() {
        super.onStart()
        val currentUser = auth.currentUser
        if(currentUser != null){
            if(currentUser.isEmailVerified){

                showDashboard()
            } else {
                sendEmailVerificacion()
            }
        }
    }
    private fun showDashboard(){
        val intent = Intent(this,MainActivity::class.java)
        this.startActivity(intent)
    }
    private fun signOut() {
```

```
        Firebase.auth.signOut()
        val intent = Intent(this, AuthActivity::class.java)
        startActivity(intent)
    }
}
```

- **DeleteAccount Activity**

```
package com.example.transaustin
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import
com.example.transaustin.databinding.ActivityDeleteAccountBinding
import com.google.firebase.auth.EmailAuthProvider
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase

class DeleteAccountActivity : AppCompatActivity() {

    private lateinit var binding: ActivityDeleteAccountBinding
    private lateinit var auth: FirebaseAuth
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding =
ActivityDeleteAccountBinding.inflate(layoutInflater)
        setContentView(binding.root)
        auth = Firebase.auth
        binding.deleteAccountAppCompatButton.setOnClickListener
{
            val password =
binding.passwordEditText.text.toString()
            deleteAccount (password)
        }
    }
    private  fun deleteAccount(password : String) {
        val user = auth.currentUser

        if (user != null){
            val email = user.email
            val credential = EmailAuthProvider
                .getCredential(email!!, password)

            user.reauthenticate(credential)
                .addOnCompleteListener { task ->
                    if(task.isSuccessful) {

                        user.delete()
                            .addOnCompleteListener {
taskDeleteAcount ->

if (taskDeleteAcount.isSuccessful)
{
                                    Toast.makeText(this, "Se
elminó tu cuenta.",

Toast.LENGTH_SHORT).show()
                                    signOut()
                                }
```

```
                                }
                            }
else {
                                Toast.makeText(this, "La contraseña
ingresada es incorrecta.",
                                    Toast.LENGTH_SHORT).show()
                            }
                        }
                }
    }
    private  fun signOut(){
        auth.signOut()
        val intent = Intent(this, AuthActivity::class.java)
        this.startActivity(intent)
    }

}
```

- **Empresas Activity**

```
package com.example.transaustin
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import
com.example.transaustin.databinding.ActivityEmpresasBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.ktx.database
import com.google.firebase.firestore.FirebaseFirestore
import com.google.firebase.ktx.Firebase
import kotlinx.android.synthetic.main.activity_empresas.*

class EmpresasActivity : AppCompatActivity() {
    private lateinit var database: DatabaseReference
    private lateinit var auth: FirebaseAuth
    private lateinit var binding: ActivityEmpresasBinding
    private val db = FirebaseFirestore.getInstance()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        database = Firebase.database.reference
        binding =
ActivityEmpresasBinding.inflate(layoutInflater)
        setContentView(binding.root)
        auth= Firebase.auth
        binding.registerREmpresa.setOnClickListener{
            val eName = binding.nameEmpresaText.text.toString()
            val eCity = binding.cityEmpresaText.text.toString()
            val eAddress =
binding.addressEmpresaText.text.toString()
            val ePhone =
binding.phoneEmpresaText.text.toString()
            when {
                eName.isEmpty() || eCity.isEmpty() ||
eAddress.isEmpty() || ePhone.isEmpty()-> {
                    Toast.makeText(baseContext, "Authentication
failed.",
                        Toast.LENGTH_SHORT).show()
                } else ->
            {
```

```
                val idUser = auth.currentUser?.uid
                val uId = idUser.toString()
                val empresa = hashMapOf(
                    "Name" to eName,
                    "admin" to uId,
                    "city" to eCity,
                    "address" to eAddress,
                    "phone" to ePhone
                )
                db.collection("Empresas").document(eName)
                    .set(empresa)
                    .addOnSuccessListener { Log.d( "TAG",
"DocumentSnapshot successfully written!") }
                    .addOnFailureListener { e -> Log.w("TAG",
"Error writing document", e) }
                val unidad = hashMapOf(
                    "name" to "Seleccionar",
                    "admin" to uId,
                    "empresa" to eName,
                    "placa" to "Seleccionar",
                    "numero" to "Seleccionar",
                    "phone" to "Seleccionar"
                )
                db.collection("Unidades").document("Seleccionar")
                    .set(unidad)
                    .addOnSuccessListener { Log.d( "TAG",
"DocumentSnapshot successfully written!") }
                    .addOnFailureListener { e -> Log.w("TAG",
"Error writing document", e) }

            }
            }
            nameEmpresaText.setText("")
            cityEmpresaText.setText("")
            addressEmpresaText.setText("")
            phoneEmpresaText.setText("")
        }

            }
```

- **MainActivity**

```
package com.example.transaustin
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import com.example.transaustin.databinding.ActivityMainBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {
    private lateinit var auth: FirebaseAuth
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
        auth = Firebase.auth
```

```
        cv_administrar.setOnClickListener{

                val intent = Intent(this,
AdministrarActivity::class.java)
                startActivity(intent)
        }
        cv_rastrear.setOnClickListener{
            val intent = Intent(this,
SearchActivity::class.java)
            startActivity(intent)
        }
        cv_about.setOnClickListener{
            Toast.makeText(applicationContext,"Acerca de...",
Toast.LENGTH_SHORT).show()
        }
        cv_unidades.setOnClickListener{
            val intent = Intent(this,
UnidadesActivity::class.java)
            startActivity(intent)
        }
        cv_salir.setOnClickListener{
            Toast.makeText(applicationContext,"Salir",
Toast.LENGTH_SHORT).show()
            signOut()
        }
        cv_empresas.setOnClickListener{
            val intent = Intent(this,
EmpresasActivity::class.java)
            startActivity(intent)
        }
    }
    private fun signOut(){

        Firebase.auth.signOut()
        val intent =Intent(this,AuthActivity::class.java)
        startActivity(intent)
    }
}
```

- **MapsActivity**

```
package com.example.transaustin
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback
import com.google.android.gms.maps.SupportMapFragment
import com.google.android.gms.maps.model.LatLng
import com.google.android.gms.maps.model.MarkerOptions
import com.example.transaustin.databinding.ActivityMapsBinding
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.ValueEventListener
import com.google.firebase.database.ktx.database
import com.google.firebase.firestore.auth.User
import com.google.firebase.ktx.Firebase
import kotlinx.android.synthetic.main.activity_update_password.*

class MapsActivity : AppCompatActivity(), OnMapReadyCallback {
```

```
    private lateinit var mMap: GoogleMap
    private lateinit var binding: ActivityMapsBinding
    private lateinit var database: DatabaseReference
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMapsBinding.inflate(layoutInflater)
        database = Firebase.database.reference
        setContentView(binding.root)
        val mapFragment = supportFragmentManager
            .findFragmentById(R.id.map) as SupportMapFragment
        mapFragment.getMapAsync(this)
    }
        override fun onMapReady(googleMap: GoogleMap) {
        mMap = googleMap
        val datos = object:ValueEventListener{
            override fun onDataChange(dataSnapshot:
DataSnapshot) {
                                    var data =
dataSnapshot.getValue(MapsPojo::class.java)
                var latitud: Double? = data?.latitud
                var longitud: Double? = data?.latitud
                var velocidad: Double? = data?.latitud
            }
            override fun onCancelled(databaseError:
DatabaseError) {

                Log.w("TAG", "loadPost:onCancelled",
databaseError.toException())
            }
        }
    }
```

- **Localización**

```
package com.example.transaustin
import com.google.firebase.database.Exclude
import com.google.firebase.firestore.IgnoreExtraProperties
@IgnoreExtraProperties
data class Localizacion (
    var latitud: Double? = 0.0,
    var longitud: Double? = 0.0,
    var velocidad: Double? = 0.0
    ) {
    @Exclude
    fun toMap(): Map<String, Any?> {
        return mapOf(
            "latitud" to latitud,
            "longitud" to longitud,
            "velocidad" to velocidad
        )
    }
}
```

- **Data**

```
package com.example.transaustin;
public class plateData {
    private Double Longitud;
    private Double Latitud;
    private Double Velocidad;
    public plateData() {

    }
```

```
    public Double getLongitud() {
        return Longitud;
    }
    public Double getLatitud() {
        return Latitud;
    }
    public Double getVelocidad() {
        return Velocidad;
    }
}
```

- **Rastreo**

```
package com.example.transaustin;
import androidx.annotation.NonNull;
import androidx.fragment.app.FragmentActivity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import
com.example.transaustin.databinding.ActivityRastreoBinding;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import org.jetbrains.annotations.NotNull;
import java.util.ArrayList;

public class Rastreo extends FragmentActivity implements
OnMapReadyCallback {
    private GoogleMap mMap;
    private ActivityRastreoBinding binding;
    private DatabaseReference mDatabase;
    private ArrayList<Marker> tmpRealTimeMarker = new
ArrayList<>();
    private ArrayList<Marker> RealTimeMarker = new
ArrayList<>();
    Bundle datos;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding =
ActivityRastreoBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
        mDatabase =
FirebaseDatabase.getInstance().getReference().child("GPS");
```

```
    }
    @Override
    public void onMapReady(GoogleMap googleMap) {
        datos = getIntent().getExtras();
        String datosObtenidos = datos.getString("placa");
        Log.d("La placa es:", datosObtenidos);

        mMap = googleMap;

mDatabase.child(datosObtenidos).addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull @NotNull
DataSnapshot dataSnapshot) {
                for (Marker marker:RealTimeMarker){
                    marker.remove();
                }
                    Double latitud = (Double)
dataSnapshot.child("Latitud").getValue();
                    Double longitud = (Double)
dataSnapshot.child("Longitud").getValue();
                    Double velocidad = (Double)
dataSnapshot.child("Velocidad").getValue();
                    String vel= velocidad.toString();
                    MarkerOptions markerOptions = new
MarkerOptions();
                    markerOptions.position(new
LatLng(latitud,longitud));
                    markerOptions.title(vel+"km/h");
                    LatLng latLng = new
LatLng(latitud,longitud);

tmpRealTimeMarker.add(mMap.addMarker(markerOptions));

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 18f));
                RealTimeMarker.clear();
                RealTimeMarker.addAll(tmpRealTimeMarker);
            }
            @Override
            public void onCancelled(@NonNull @NotNull
DatabaseError error) {

            }
        });
}
```

- **RegisterActivity**

```
package com.example.transaustin
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.util.Patterns
import android.widget.Toast
import
com.example.transaustin.databinding.ActivityRegisterBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.ktx.database
```

```
import com.google.firebase.ktx.Firebase
import java.util.regex.Pattern
class RegisterActivity : AppCompatActivity() {
    private lateinit var auth: FirebaseAuth
    private lateinit var binding: ActivityRegisterBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding =
ActivityRegisterBinding.inflate(layoutInflater)
        setContentView(binding.root)
        auth= Firebase.auth
        binding.signUpRButton.setOnClickListener{
            val mEmail = binding.emailREditText.text.toString()
            val mPassword =
binding.passwordREditText.text.toString()
            val mRepeatPassword =
binding.passwordREditText2.text.toString()
            val passwordRegex = Pattern.compile("^"+
            "(?=.*[-@#$%^&+=])"+
                    ".{6,}"+
                    "$")

            if (mEmail.isEmpty() ||
!Patterns.EMAIL_ADDRESS.matcher(mEmail).matches()){

                Toast.makeText(baseContext, "Ingrese un email
valido",
                    Toast.LENGTH_SHORT).show()

            } else if (mPassword.isEmpty() ||
!passwordRegex.matcher(mPassword).matches()){

                Toast.makeText(baseContext, "La contrasena es
debil",
                    Toast.LENGTH_SHORT).show()
            } else if (mPassword != mRepeatPassword) {
                Toast.makeText(baseContext, "Las contrasenas no
coninciden",
                    Toast.LENGTH_SHORT).show()
            } else {
                signUp(mEmail,mPassword)
            }
        }
    }
    private fun signUp(email: String, password: String){
        auth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {

                    val intent = Intent(this,
CheckActivity::class.java)
                    startActivity(intent)

                } else {
                                        Log.w("TAG",
"createUserWithEmail:failure", task.exception)
                    Toast.makeText(baseContext, "Authentication
failed.",
                        Toast.LENGTH_SHORT).show()

                }
```

```
            }
        }
        public override fun onStart() {
            super.onStart()
            val currentUser = auth.currentUser
            if(currentUser != null){
                if(currentUser.isEmailVerified) {
                    showDashboard()
                } else {
                    val intent = Intent(this,
CheckActivity::class.java)
                    startActivity(intent)
                }
            }
        }
        private fun showDashboard(){
            val intent =Intent(this,MainActivity::class.java)
            this.startActivity(intent)
        }
}
```

- **SearchActivity**

```
package com.example.transaustin
import android.R
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.Spinner
import com.example.transaustin.databinding.ActivitySearchBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.ktx.database
import com.google.firebase.firestore.FirebaseFirestore
import com.google.firebase.ktx.Firebase
import kotlinx.android.synthetic.main.activity_main.*
import kotlinx.android.synthetic.main.activity_search.*

class SearchActivity : AppCompatActivity() {
    private lateinit var database: DatabaseReference
    private lateinit var auth: FirebaseAuth
    private lateinit var binding: ActivitySearchBinding
    private val db = FirebaseFirestore.getInstance()
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        database = Firebase.database.reference
        binding = ActivitySearchBinding.inflate(layoutInflater)
        setContentView(binding.root)
        auth= Firebase.auth
        val idUser = auth.currentUser?.uid
        val uId = idUser.toString()
        val spinnerEmpresas =
findViewById<Spinner>(com.example.transaustin.R.id.idSpinnerEmpr
esa)

        val list = ArrayList<String>()
```

```
        var item:String = ""
        db.collection("Empresas").whereEqualTo("admin", uId)
            .get().addOnSuccessListener { documents ->
                for (document in documents) {

                    val name = document.data["Name"].toString()
                    list.add(name)

                }
                val adaptador = ArrayAdapter(this,
R.layout.simple_spinner_item,list)
                spinnerEmpresas.adapter = adaptador
            }
            .addOnFailureListener { exception ->
                Log.w("TAG", "Error getting documents: ",
exception)
            }
        spinnerEmpresas.onItemSelectedListener = object:
            AdapterView.OnItemSelectedListener{
            override fun onItemSelected(
                parent: AdapterView<*>?,
                view: View?,
                position: Int,
                id: Long
            ) {
                item = list[position]
                fillplate(item)

            }
            override fun onNothingSelected(parent:
AdapterView<*>?) {
                TODO("Not yet implemented")
            }
        }
    }
    private fun fillplate(empresa: String ){
        val empres = empresa
        println(empres)
        val spinnerPlaca =
findViewById<Spinner>(com.example.transaustin.R.id.idSppinerUnid
ad)
        val listu = ArrayList<String>()
        var itemu = "String"
        db.collection("Unidades").whereEqualTo("empresa",
empres)
            .get().addOnSuccessListener { documents ->
                for (document in documents) {
                    val nameu =
document.data["placa"].toString()
                    listu.add(nameu)

                }
                val adaptadoru = ArrayAdapter(this,
R.layout.simple_spinner_item,listu)
                spinnerPlaca.adapter = adaptadoru
            }
            .addOnFailureListener { exception ->
                Log.w("TAG", "Error getting documents: ",
exception)
            }
```

```
        spinnerPlaca.onItemSelectedListener = object:
            AdapterView.OnItemSelectedListener{
            override fun onItemSelected(
                parent: AdapterView<*>?,
                view: View?,
                position: Int,
                id: Long
            ) {
                itemu = listu[position]
                localizacion(itemu)
            }
            override fun onNothingSelected(parent:
AdapterView<*>?) {
                TODO("Not yet implemented")
            }
        }
    }
    private fun localizacion (placas:String) {
        if (placas == "Seleccionar")
        {
            searchButtom.isClickable = false
            searchButtom.isEnabled = false
            println(placas)
            }
            else
            {
                searchButtom.isClickable = true
                searchButtom.isEnabled = true
                searchButtom.setOnClickListener{
                    val intent = Intent(this,
Rastreo::class.java)
                    intent.putExtra("placa", placas);
                    startActivity(intent)
            }
        }

    }
}
```

- **UnidadesActivity**

```
package com.example.transaustin
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.AdapterView
import android.widget.ArrayAdapter
import android.widget.Spinner
import android.widget.Toast
import
com.example.transaustin.databinding.ActivityUnidadesBinding
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.database.DatabaseReference
import com.google.firebase.database.ktx.database
import com.google.firebase.firestore.FirebaseFirestore
import com.google.firebase.ktx.Firebase

class UnidadesActivity : AppCompatActivity() {
```

```
    private lateinit var database: DatabaseReference
    private lateinit var auth: FirebaseAuth
    private lateinit var binding: ActivityUnidadesBinding
    private val db = FirebaseFirestore.getInstance()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        database = Firebase.database.reference
        binding =
ActivityUnidadesBinding.inflate(layoutInflater)
        setContentView(binding.root)
        auth= Firebase.auth

        val idUser = auth.currentUser?.uid
        val uId = idUser.toString()
        val spinner = findViewById<Spinner>(R.id.idSppiner)
        val list = ArrayList<String>()
        var item = "String"
        db.collection("Empresas").whereEqualTo("admin", uId)
            .get().addOnSuccessListener { documents ->
                for (document in documents) {

                    val name = document.data["Name"].toString()
                    list.add(name)
                                    }
                val adaptador = ArrayAdapter(this,
android.R.layout.simple_spinner_item,list)
                spinner.adapter = adaptador
            }
            .addOnFailureListener { exception ->
                Log.w("TAG", "Error getting documents: ",
exception)
            }
        spinner.onItemSelectedListener = object:
            AdapterView.OnItemSelectedListener{
            override fun onItemSelected(
                parent: AdapterView<*>?,
                view: View?,
                position: Int,
                id: Long
            ) {
                item = list[position]
                println(item)
            }
            override fun onNothingSelected(parent:
AdapterView<*>?) {
                TODO("Not yet implemented")
            }
        }
        binding.registrarRVButton.setOnClickListener{

            val uName = binding.uName.text.toString()
            val uPlaca = binding.carPlate.text.toString()
            val uNumber = binding.carNumber.text.toString()
            val uPhone = binding.uPhone.text.toString()
            when {
                uName.isEmpty() || uPlaca.isEmpty() ||
uNumber.isEmpty() || uPhone.isEmpty()  -> {
                    Toast.makeText(baseContext, "Authentication
failed.",
                        Toast.LENGTH_SHORT).show()
```

```
                } else ->
                 {

                    val unidad = hashMapOf(
                    "name" to uName,
                    "admin" to uId,
                    "empresa" to item,
                    "placa" to uPlaca,
                    "numero" to uNumber,
                    "phone" to uPhone
                )

                db.collection("Unidades").document(uName)
                    .set(unidad)
                    .addOnSuccessListener { Log.d( "TAG",
"DocumentSnapshot successfully written!") }
                    .addOnFailureListener { e -> Log.w("TAG",
"Error writing document", e) }

                     binding.uName.setText("")
                     binding.carPlate.setText("")
                     binding.carNumber.setText("")
                     binding.uPhone.setText("")

                     writeNewCar(uPlaca,0.0,0.0,0.0)

            }
            }
        }
    }
    data class GPS(val placa: String? = null, val latitud:
Double? = null, val longitud: Double? = null, val velocidad:
Double? = null) {

    }
    fun writeNewCar(placa: String, latitud: Double, longitud:
Double, velocidad: Double) {
        val geolocalizacion =
GPS(placa,latitud,longitud,velocidad)

database.child("GPS").child(placa).setValue(geolocalizacion)

    }
```

- **UpdatePasswordActivity**

```
package com.example.transaustin
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import
com.example.transaustin.databinding.ActivityUpdatePasswordBindin
g
import com.google.firebase.auth.EmailAuthProvider
import com.google.firebase.auth.FirebaseAuth
import com.google.firebase.auth.ktx.auth
import com.google.firebase.ktx.Firebase
import java.util.regex.Pattern

class UpdatePasswordActivity : AppCompatActivity() {
```

```
    private lateinit var binding: ActivityUpdatePasswordBinding
    private lateinit var auth: FirebaseAuth
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding =
ActivityUpdatePasswordBinding.inflate(layoutInflater)
        setContentView(binding.root)
        auth = Firebase.auth

        val passwordRegex = Pattern.compile(
            "^" +
                    "(?=.*[-@#$%^/&+=])" +      // Al menos 1
carácter especial
                    ".{6,}" +                  // Al menos 6
caracteres
                    "$"
        )

        binding.changePasswordAppCompatButton.setOnClickListener
{
            val currentPassword =
binding.currentPasswordEditText.text.toString()
            val newPassword =
binding.newPasswordEditText.text.toString()
            val repeatPassword =
binding.repeatPasswordEditText.text.toString()

            if (newPassword.isEmpty() ||
!passwordRegex.matcher(newPassword).matches()) {
                Toast.makeText(
                    this, "La contraseña es debil.",
                    Toast.LENGTH_SHORT
                ).show()
            } else if (newPassword != repeatPassword) {
                Toast.makeText(
                    this, "Confirma la contraseña.",
                    Toast.LENGTH_SHORT
                ).show()
            } else {
                chagePassword(currentPassword, newPassword)
            }
        }
    }

    private  fun chagePassword(current : String, password :
String){
        val user = auth.currentUser

        if (user != null){
            val email = user.email
            val credential = EmailAuthProvider
                .getCredential(email!!, current)

            user.reauthenticate(credential)
                .addOnCompleteListener { task ->
                    if(task.isSuccessful) {

                        user.updatePassword(password)
                            .addOnCompleteListener {
taskUpdatePassword ->
```

```
                                    if
(taskUpdatePassword.isSuccessful) {
                                        Toast.makeText(this, "Se
cambio la contraseña.",

Toast.LENGTH_SHORT).show()
                                        val intent = Intent(this,

MainActivity::class.java)
                                        this.startActivity(intent)
                                    }
                                }

                    } else {
                        Toast.makeText(this, "La contraseña
actual es incorrecta.",
                            Toast.LENGTH_SHORT).show()
                    }
                }
            }
        }
```

Código fuente circuito controlador

```
//Select your modem

//SSL/TLS is currently supported only with SIM8xx series

#define TINY_GSM_MODEM_SIM800

//Increase RX buffer

#define TINY_GSM_RX_BUFFER 256

//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

#include <TinyGPS++.h> //https://github.com/mikalhart/TinyGPSPlus

#include <TinyGsmClient.h> //https://github.com/vshymanskyy/TinyGSM

#include <ArduinoHttpClient.h> //https://github.com/arduino-libraries/ArduinoHttpClient

#include <SoftwareSerial.h>

#include <String.h>

//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

const char FIREBASE_HOST[]  = "transaustin-9e76e-default-rtdb.firebaseio.com";

const String FIREBASE_AUTH  = "xiVUEB0CGn0DnVfPc5Z7RQXtJ0UCtPWyqNJVznIs";

const String FIREBASE_PATH  = "GPS/BJB-909";

const int SSL_PORT          = 443;

//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

```
//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
// Your GPRS credentials
// Leave empty, if missing user or pass
char apn[]  = "internet.movistar.com.ec";
char user[] = "movistar";
char pass[] = "movistar";
//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
//GSM Module RX pin to ESP32 2
//GSM Module TX pin to ESP32 4
SoftwareSerial Sim800Serial(4, 5);
#define rxpin 4
#define txpin 5
//HardwareSerial sim800(1);
TinyGsm modem(Sim800Serial);
//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
//GPS Module RX pin to ESP32 17
//GPS Module TX pin to ESP32 16
SoftwareSerial ss(13, 12);
#define rxd2 13
#define txd2 12
//HardwareSerial neogps(2);
TinyGPSPlus gps;
//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN


//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
TinyGsmClientSecure gsm_client_secure_modem(modem, 0);
HttpClient http_client = HttpClient(gsm_client_secure_modem, FIREBASE_HOST,
SSL_PORT);
//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

```
unsigned long previousMillis = 0;
long interval = 10000;
//*************************************************************************
****************************
void setup() {
 Serial.begin(115200);
 Serial.println("esp32 serial initialize");
  Sim800Serial.begin(19200);
 Serial.println("SIM800L serial initialize");
 ss.begin(9600);
 Serial.println("neogps serial initialize");
 delay(3000);
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 //Restart takes quite some time
 //To skip it, call init() instead of restart()
 Serial.println("Initializing modem...");
 modem.restart();
 String modemInfo = modem.getModemInfo();
 Serial.print("Modem: ");
 Serial.println(modemInfo);
 //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  // Unlock your SIM card with a PIN
 //modem.simUnlock("1234");
http_client.setHttpResponseTimeout(90 * 1000); //^0 secs timeout
}
//*************************************************************************
****************************
void loop() {
 //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 //Restart takes quite some time
 //To skip it, call init() instead of restart()
```

```
 //Serial.println("Initializing modem...");
 //modem.init();
 //String modemInfo = modem.getModemInfo();
 //Serial.print("Modem: ");
 //Serial.println(modemInfo);
 //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 //Serial.print(F("Waiting for network..."));
 //if (!modem.waitForNetwork()) {
  //Serial.println(" fail");
  //delay(1000);
  //return;
 //}
 //Serial.println(" OK");
 //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 Serial.print(F("Connecting to "));
 Serial.print(apn);
 if (!modem.gprsConnect(apn, user, pass)) {
  Serial.println(" fail");
  delay(1000);
  return;
 }
 Serial.println(" OK");
 //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
  http_client.connect(FIREBASE_HOST, SSL_PORT);

//MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
MMM
 while (true) {
  if (!http_client.connected()) {
   Serial.println();
```

```
    http_client.stop();// Shutdown
    Serial.println("HTTP  not connect");
    break;
   }
   else{
    gps_loop();
   }
  }
  }
//********************************************************************
***************************
void PostToFirebase(const char* method, const String & path , const String & data, HttpClient* http) {
  String response;
  int statusCode = 0;
  http->connectionKeepAlive(); // Currently, this is needed for HTTPS

//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNN
  String url;
  if (path[0] != '/') {
    url = "/";
  }
  url += path + ".json";
  url += "?auth=" + FIREBASE_AUTH;
  Serial.print("POST:");
  Serial.println(url);
  Serial.print("Data:");
  Serial.println(data);

//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNN
```

```
 String contentType = "application/json";
 http->put(url, contentType, data);
  //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 // read the status code and body of the response
 //statusCode-200 (OK) | statusCode -3 (TimeOut)
 statusCode = http->responseStatusCode();
 Serial.print("Status code: ");
 Serial.println(statusCode);
 response = http->responseBody();
 Serial.print("Response: ");
 Serial.println(response);
 //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 if (!http->connected()) {
  Serial.println();
  http->stop();// Shutdown
  Serial.println("HTTP POST disconnected");
 }
 //NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
}
//**********************************************************************
****************************
void gps_loop()
{
//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNN
 //Can take up to 60 seconds
 boolean newData = false;
 for (unsigned long start = millis(); millis() - start < 2000;){
  while (ss.available()){
   if (gps.encode(ss.read())){
    newData = true;
    break;
```

```
    }
  }
}

//NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
 //If newData is true
 if(true){
 newData = false;
   String latitude, longitude, velocidad;
 //float altitude;
 //unsigned long date, time, speed, satellites;
 unsigned long speed;
   latitude = String(gps.location.lat(), 6); // Latitude in degrees (double)
 longitude = String(gps.location.lng(), 6); // Longitude in degrees (double)
  //altitude = gps.altitude.meters(); // Altitude in meters (double)
 //date = gps.date.value(); // Raw date in DDMMYY format (u32)
 //time = gps.time.value(); // Raw time in HHMMSSCC format (u32)
 velocidad =  String(gps.speed.kmph());
   Serial.print("Latitude= ");
 Serial.print(latitude);
 Serial.print(" Longitude= ");
 Serial.println(longitude);
 Serial.print(" Speed= ");
 Serial.println(velocidad);
 String gpsData = "{";
 gpsData += "\"Latitud\":" + latitude + ",";
 gpsData += "\"Longitud\":" + longitude + ",";
 gpsData += "\"Velocidad\":" + velocidad + "";
 gpsData += "}";
 //PUT   Write or replace data to a defined path, like messages/users/user1/<data>
```

```
  //PATCH   Update some of the keys for a defined path without replacing all of the data.

  //POST  Add to a list of data in our Firebase database. Every time we send a POST request, the Firebase client generates a unique key, like messages/users/<unique-id>/<data>

  //https://firebase.google.com/docs/database/rest/save-data

    PostToFirebase("PATCH", FIREBASE_PATH, gpsData, &http_client);

    }

/NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN

}
```

Universidad Católica de Cuenca	**AUTORIZACION DE PUBLICACION EN EL REPOSITORIO INSITUCIONAL**	CÓDIGO: F- DB - 30 VERSION: 01 FECHA: 2021-04-15 Página **1** de **1**

Luis Enrique Carvajal Andrade portador(a) de la cedula de ciudadanía N. **0302854005**. En calidad de autor/a y titular de los derechos patrimoniales del trabajo de titulación "**Análisis de un Prototipo para un Aplicativo Móvil que Permita la Geolocalización con la Utilización IoT para las unidades de Transporte Urbano del Cantón Cañar**", de conformidad con lo establecido en el artículo 114 Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación, reconozco a favor de la Universidad Católica de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos y no comerciales. Autorizo además a la Universidad Católica de Cuenca para que realice a publicación de este trabajo de titulación en el Repositorio Institucional de conformidad a lo dispuesto en el artículo 144 de la Ley Orgánica de Educación Superior.

Cañar, **23 de marzo de 2022**

F: ……..

Luis Enrique Carvajal Andrade

C.I. 0302854005