Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Aplicación Guía Bus

Bus Guide Application

Ana Beatriz Gil González

La Laguna, 10 de junio de 2019

D. **Pino Caballero Gil**, con N.I.F. 45534310-Z Catedrática de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora.

D. **José Angél Concepción Sánchez**, con N.I.F. 42234897-C Personal Investigador adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"*Bus Guide Application*"

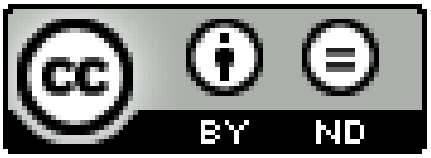
ha sido realizada bajo su dirección por D. **Ana Beatriz Gil González**, con N.I.F. 54114132-P.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 10 de junio de 2019

Agradecimientos

En primer lugar, la gran ayuda recibida de mi tutora Pino Caballero Gil y de mi cotutor José Ángel Concepción Sánchez, que permitió que pudiera realizar esta aplicación con más facilidad.
Quiero agradecer también las reuniones en el grupo CryptULL con otros tutores de los Trabajos de Fin de Grado de mis compañeros, y en especial a Alexandra Rivero y a Iván Santos, que siempre han estado apoyándome para solucionar problemas encontrados por el camino.
Agradezco también las reuniones con los compañeros con los cuales comparto puntos en común de Trabajos de Fin de Grado, permitiendo que aprendiera cuestiones útiles para mi aplicación. Gracias a mis compañeros, el trayecto no ha sido tan duro, ya que ellos saben lo mucho que se sufre hasta llegar al objetivo. Por tanto agradezco su colaboración y compañía. En especial dar las gracias a Mireia Schloz, ya que al tener ambas un Trabajo de Fin de Grado basado en una aplicación Android hemos podido ayudarnos mutuamente.
Finalmente, pero no por ello menos importante, doy las gracias a mi familia por todos sus ánimos diarios. Siempre han hecho sacrificios, y han invertido tiempo y dinero para que yo consiguiera mi felicidad y un buen futuro. Siempre les estaré eternamente agradecida ya que son mi máximo apoyo de vida.

Licencia



Resumen

En los últimos años, el uso de los sistemas de geolocalización ha aumentado debido al crecimiento en la utilización de los dispositivos móviles. Debido a ello, en este campo ha aparecido una nueva metodología de localización que no depende del GPS gracias a las balizas o beacons de Bluetooth Low Energy (BLE).

En el caso de este Trabajo de Fin de Grado, se aprovecha el uso de esta tecnología para el desarrollo de una aplicación móvil Android que, mediante el uso de estos dispositivos, permitirá guiar a las personas durante sus viajes en el transporte público sin la necesidad de utilizar el Sistema de Posicionamiento Global GPS.

La aplicación proporciona muchas ventajas al usuario como es la comodidad y la sencillez. Una de las ventajas respecto al mundo tecnológico es que usa las últimas tecnologías como son los beacons y la tecnología BLE.

Palabras clave: Guiado en transporte público, Android, Bluetooth, Beacons, Localización, Firebase.

Abstract

In recent years, the use of geolocation systems has increased due to the growth in the use of mobile devices. Due to this, a new localization tool has appeared in this field that does not depend on GPS thanks to the Bluetooth Low Energy (BLE) beacons.

In the case of this Final Degree Project, the use of this technology is exploited for the development of an Android mobile application that, through the use of these devices, will allow guiding people during trips in public transport without the need to use the Global Positioning System GPS.

The application provides many advantages to the user as is the comfort and simplicity. One of the advantages with respect to the technological world is that it uses the latest technologies such as beacons and BLE technology.

Keywords: Guided on public transport, Android, Bluetooth, Beacons, Location, Firebase.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivación

Actualmente, los **beacons** son una nueva tecnología que nada más entrar en el mercado tuvo muchísimo éxito, llamando la atención de muchas empresas. Estos dispositivos están muy **presente en nuestra sociedad** pero aún no han llegado a su máximo auge. Los grandes avances realizados a favor de estas tecnologías han ayudado a resolver muchos problemas que antes se veían complicados de solucionar de manera sencilla para el/la usuario/a.

La incorporación de los beacons se ha visto como una gran inversión para el presente y el futuro, destacando en varios sectores, como el turístico, el médico, el logístico o el militar. Por ejemplo, en el sector turístico los beacons nos **ofrecen mucha comodidad** ya que podrían evitar el uso de mapas y el GPS.

Gracias a esa gran evolución, ese gran uso que le espera en el futuro y las facilidades que da a cualquier usuario/a, nace este Trabajo de Fin de Grado (**TFG**). Esta aplicación se centra específicamente en ayudar a los/as usuarios/as que usan el transporte para llegar a su destino de forma cómoda, es decir, sin tener que mirar el teléfono constantemente para ver si va todo bien.

En este TFG se ha diseñado y desarrollado un sistema de guiado en el transporte público para todas las edades. La única herramienta que el/la usuario/a debe tener a su disposición es un dispositivo que funcione con la tecnología **Bluetooth** para detectar dispositivos.

En cuanto al diseño de la interfaz, se ha intentado que la interacción con el/la usuario/a sea lo más intuitiva posible.

La redacción del presente trabajo se ha realizado intentando hacer un uso inclusivo del lenguaje. Por ello se han utilizando términos colectivos, siempre que fuera posible. Sin embargo, tal como se observa en esta sección, términos como usuarios/as, no tienen sinónimo colectivo y por eso, sintiéndolo mucho, utilizaremos en este trabajo a partir de ahora la palabra usuarios para referirnos a todas las personas que usen el sistema.

1.2. Objetivos

Se han definido una serie de objetivos para el desarrollo del sistema:

- **Sistema sencillo** Al ser una aplicación que será usada por personas de diferentes edades, tiene que ser sencilla y fácil de entender. Además, tiene que ser una aplicación clara y concisa, es decir, al final el usuario lo que quiere es llegar a su

destino y es lo que la aplicación desarrollada les proporciona.

- **Facilidad de guiado** El guiado de transporte público depende de muchas variables como ¿Qué ruta elegir? ¿Qué autobús coger? Si cambio de ruta, ¿qué ruta nueva seguir?, etc. Estas variables deben simplificarse al máximo así que el sistema debe proporcionar una ruta adecuada que no lleve a ninguna confusión.
- **Interfaz amigable** Otro de los objetivos es minimizar los requisitos de interacción con el usuario, lo que le permite una mayor comprensión de la aplicación. Utilizar sonidos o vibraciones cuando deba bajarse en una parada es un ejemplo de esto.
- **Base de datos óptima.** Conseguir un diseño de la base de datos lo más óptimo posible, dada la complejidad que podría llegar a tener.

1.3. Aportación del proyecto

Hoy en día el transporte público es muy utilizado por muchas personas, y con el paso de los años sigue aumentando su uso. El transporte urbano aumenta un **1,3 %** en tasa anual y el interurbano un **2,5 %**. Concretamente en Canarias donde sería un buen comienzo para la aplicación ha aumentado un **3,9 %** [16] Por ello, son necesarias más aplicaciones dirigidas a este mundo, y sobretodo utilizando las nuevas tecnologías como es en este caso, los beacons.

Un beacon es un dispositivo pequeño que emite una señal en la onda corta de la tecnología Bluetooth. Su alcance máximo es en torno a los 100 metros teóricos y su señal puede ser localizada por otros dispositivos. Cierto es que los beacons no son una tecnología que pueda sustituir las funciones del GPS, pero sí que realiza otras funciones las cuales no son idóneas para él. Además, esta tecnología, al igual que la de GPS, es independiente de las redes de telefonía con lo cual se evita pagar por su uso.

Para aplicaciones como FourSquare, que pretende servir de guía dentro de las ciudades en lugares entre los que se encuentran museos o locales de ocio, la tecnología GPS es ineficiente. Por ejemplo, en el caso de museos, esta tecnología no podría detectar cuándo un usuario se acerca a una obra de arte, mientras que con el uso de los beacons sí que se puede. Otro ejemplo del uso de esta tecnología son empresas como Paypal, que están desarrollando sistemas de micropagos digitales para poder realizar transacciones desde el móvil sin tener que pasar por caja a través del uso de beacons.

Este proyecto se centra principalmente en **incrementar** la comodidad del trayecto a las personas que utilizan el transporte público. Les proporciona facilidad para entender dónde se encuentran y cómo llegar a su destino. Es una aplicación concreta que aporta únicamente lo que el usuario desea en cada momento.

Como aportación a destacar, este sistema se basa en la detección de dispositivos **BLE** permitiendo avisar al usuario de dónde se encuentra exactamente. Otra aportación a destacar es que el usuario no tiene que estar pendiente del dispositivo para saber cómo va su trayecto ya que este produce una vibración cada vez que detecta una parada, a diferencia del GPS que implica estar pendiente todo el rato de la aplicación para saber dónde te encuentras.

La herramienta desarrollada pretende dar todas las facilidades existentes al usuario y permitir que con poca información y **sencillez** pueda cumplir los deseos del usuario.

1.4. Fases

Las fases que se han seguido para el desarrollo de la aplicación son:

1. **Estudio y análisis de plataformas y tecnologías aplicables al proyecto.** Aprender a usar la plataforma Android Studio y las tecnologías que pueden ser usadas en esa plataforma.

2. **Primer desarrollo de la aplicación móvil.** Desarrollar una primera aplicación que realice los principales puntos solicitados.

3. **Implementación de la base de datos por medio de Firebase.** Una vez desarrollada una primera versión de la aplicación, diseñar y usar una base de datos en la nube por medio de Firebase.

4. **Realizar un desarrollo final con más funcionalidades.** Terminar de desarrollar la aplicación completando el resto de funcionalidades solicitadas e incluyendo algunas adicionales.

5. **Implementación de mecanismos de seguridad de los datos.** Dar mayor seguridad a la información proporcionada desde la base de datos mediante la función hash estándar SHA-3.

6. **Redacción de la memoria de Trabajo Fin de Grado de alta calidad.** Redactar una memoria donde se describa y exponga todo el diseño, desarrollo y resultados obtenidos. Esta memoria ha sido desarrollada en LaTeX [7] [14]

1.5. Estructura de la memoria

La estructura de la memoria está compuesta de los siguientes capítulos:

Capítulo 1. Introducción. Se introduce la temática.

Capítulo 2. Preliminares. Se definen los preliminares, como algunas aplicaciones similares y diferentes tecnologías utilizadas en el proyecto, incluyendo Android.

Capítulo 3. Aplicación desarrollada. Se explican todos los principales detalles de la aplicación desarrollada, incluyendo el sistema Bluetooth utilizado.

Capítulo 4. Base de datos utilizada. Se expone la base de datos Firebase, así como su diseño y funcionamiento en el proyecto.

Capítulo 5. Seguridad. Se explica la seguridad del proyecto.

Capítulo 6. Presupuesto. Se muestra un presupuesto general del coste del diseño y desarrollo de la aplicación.

Capítulo 7. Conclusiones y mejoras futuras. Se extraen algunas conclusiones del proyecto y futuras mejoras.

Capítulo 8. Conclusions and future improvements. Conclusions of the project and future improvements are presented.

Capítulo 2

Preliminares

2.1. Definición del problema

Hoy en día casi no existen aplicaciones enfocadas concretamente en el guiado de personas durante su trayecto en el transporte público, y en la mayoría de casos, estas aplicaciones son demasiado generales o no cumplen con los objetivos concretos de este proyecto.

El guiado en el transporte público comenzó con un simple **mapa** por el cual los usuarios a través de este se ayudaban para llegar al destino deseado. Para guiarse con estos, todas las funciones debía realizarlas el usuario (mirar qué ruta coger, qué transporte, dónde bajarse, las conexiones, y otros muchos factores).

Posteriormente apareció la tecnología GPS y aplicaciones como **Google Maps**[15], que permiten calcular trayectos, pero que no es solo utilizada para transporte público, sino también para coches, para ir andando y/o en bicicleta. Para guiarse por medio del GPS, el usuario debe poner la ubicación en la que se encuentra o especificar de donde va a salir. Posteriormente debe especificar el destino al que desea llegar y la aplicación le mostrará la ruta más óptima posible. El usuario seleccionará esta u otra de las disponibles y la aplicación le dirá los pasos a seguir. Una de las desventajas es que el usuario debe estar siempre pendiente de la aplicación para saber si ha llegado a tu destino, si tiene que bajarse en una parada y muchos otros datos. Por tanto, esta aplicación pretende solucionar esos pequeños problemas relacionados con la comodidad del usuario.

La aplicación desarrollada en este proyecto pretende solucionar esos pequeños problemas relacionados con la comodidad del usuario.

2.2. Aplicaciones similares

Existen hoy en día muy pocas aplicaciones relacionadas con el transporte público [1]. Aparte de la más conocida que es **Google Maps**, existen aplicaciones más específicas. Dos de ellas son:

Moovit[12]

Es una app que realiza actualizaciones constantes de cualquier cambio que se produzca en la ruta. Por ejemplo, cuando una parada deja de estar disponible, el usuario no viajará hasta ella, o si un autobús está fuera de servicio, no esperará por ella y otros muchos casos. Moovit muestra cuánto dura la espera hasta que llegue el transporte, la duración del trayecto, cuántas paradas quedan al usuario. y todo ello en tiempo real. También contiene una pestaña de favoritos donde el usuario puede guardar las paradas

que realice más frecuentemente.

Urban step[13]

Esta aplicación muestra un horario interactivo para autobuses, trenes, metro y estaciones bike sharing. Con solo un toque el usuario puede saber cuánto queda para que llegue el transporte deseado. Si usa esta aplicación con la ubicación, le permite ver todas las paradas cercanas y añadir alguna de ellas a favoritos. Por otro lado, si hace clic en una de esas paradas le aparecerá el nombre, que transporte pasa por ahí y su frecuencia. Buscando esa parada también podrá ver toda esta información.

2.3. Conceptualización de la propuesta

Gracias al avance en los diez últimos años, el desarrollo tecnológico móvil ha permitido a la sociedad cambiar de mentalidad sobre su uso y todas sus capacidades. Un ejemplo muy claro fue con la incorporación de Internet, la apuesta de futuro de grandes compañías y el desarrollo de aplicaciones por parte de la comunidad de programadores. Todo ello ha llevado a la creación de lo que actualmente se conoce como dispositivos móviles inteligentes.

Gracias a estos dispositivos se interconectan millones de personas y se ofrecen multitud de tareas: redes sociales, noticias, navegación, contenido multimedia, y otras muchas. La herramienta propuesta en este TFG es un proyecto que nace de la idea de facilitarle aún más a los usuarios su experiencia en el transporte público. Este proyecto a diferencias de otras aplicaciones, como las nombradas anteriormente, utilizan la tecnología GPS,en cambio, esta aplicación no utiliza esa tecnología sino que usa la tecnología BLE por medio de los beacons.

La pregunta es ¿qué aporta un sistema como el propuesto? Los sistemas de navegación generalmente están adaptados al guiado a través de un mapa hacia un destino usando los mecanismos de transporte convencionales (andando, en bici o en coche por ejemplo). Por tanto, pocos de estos sistemas se centran realmente en el guiado de transporte público y aún menos hacen uso de otra tecnología que no sea el GPS.

Hoy en día el transporte público es muy utilizado tanto por personas del lugar como por turistas. En el caso de los ciudadanos, la aplicación propuesta les permitirá ir a cualquier sitio sin que tengan que preocuparse de cómo llegar, ya que directamente pueden introducir la salida y el destino, y ya la aplicación les mostraría los pasos a seguir. En el caso de los turistas, cuando se encuentran de vacaciones lo que quieren es ver, apreciar y disfrutar de lo que encuentra a su alrededor, y no estar pendiente del dispositivo o del mapa para no pasarse la parada durante su trayecto en el transporte público. Gracias a la aplicación propuesta podrán cumplir este propósito, ya que no es necesario que el usuario esté pendiente de qué hacer porque serían notificados automáticamente de cada acción a realizar.

2.4. Tecnologías

2.4.1. Justinmind

Es una herramienta de creación de prototipos de aplicaciones web y móviles. En este proyecto ha sido utilizado para crear los prototipos de la aplicación móvil, prototipos que únicamente cumplían la idea principal de proyecto.

Respecto a estos mockups creados han cambiado mucho con respecto al diseño

actual de la aplicación.

2.4.2. Bluetooth

Bluetooth es una tecnología para redes inalámbricas que permite la transmisión de voz y datos entre distintos dispositivos mediante una radiofrecuencia segura (2.4 GHz). Esta tecnología por lo tanto, permite las comunicaciones sin cables ni conectores y la posibilidad de crear redes inalámbricas domésticas para sincronizar y compartir información que se encuentra almacenada en diversos equipos.

Existen tres clases de Bluetooth:
Clase 1. Con una alcance de 100 metros aproximadamente (En este proyecto es esta clase de Bluetooth).
Clase 2. Con un alcance de 10 metros.
Clase 3. Con un alcance de 1 metro.

Se puede decir que el hardware que forma un sistema Bluetooth cuenta con dos partes, el dispositivo de radio que modula y transmite la señal y el controlador digital.

Bluetooth Low Energy (BLE) es una tecnología que se introdujo como parte de la especificación de Bluetooth 4.0. Lo que hace que esta tecnología sea tan interesante es lo sencillo que es implementar la comunicación entre pequeños dispositivos y una aplicación en cualquier plataforma móvil actual (iOS, Android, Windows Phone, etc). Esta tecnología a diferencia de Bluetooth permite ahorrar energía del dispositivo, y las búsqueda no para hasta que el usuario lo desee, por ello es una ventaja para la aplicación ya que es necesario que esté en continuo funcionamiento para detectar las paradas con beacons.

2.4.3. Github

Github es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones de Git. Se utiliza principalmente para la creación de código fuente de programas.

El código de los proyectos Github se almacena típicamente de forma pública, aunque utilizando una cuenta de pago, también puede generar repositorios privados.

El proyecto desarrollado para este TFG ha sido alojado en el enlace [11], donde se pueden consultar todos los cambios que se han ido realizando durante el desarrollo.

2.4.4. Android

Es un sistema operativo creado en 2006. Es uno de los sistemas para dispositivos móviles, por lo general en pantallas táctiles, más usados en el mundo. Android puede adaptarse a múltiples resoluciones de pantalla y soporta conexiones Wifi, Bluetooth, etc. También permite el envío de mensajes **SMS** y **MMS**, cuentan con navegadores web, posibilita el desarrollo de streaming y está capacitado para trabajar con archivos como **MP3**, **GIF** y otros formatos **multimedia**.

El sistema operativo Android desarrollado por Google está basado en Linux, siendo una ventaja para el desarrollo en este entorno dado que es de carácter libre, personalizable y fácil de utilizar. La seguridad es otro de los puntos fuertes de este sistema dado que las actualizaciones en este ámbito son más periódicas y aumentan con el paso del tiempo. Y por último, en el tema de conectividad, permite total acceso a su

sistema y de esta forma aumentar la velocidad del proceso de transferencia de archivos o información.

Actualmente, como todo programa, Android y sus APIs van renovando versiones a medida que la tecnología avanza.

Al crear nuevas versiones consiguen evolucionar el sistema operativo y presentar, gracias a ello, un impulso comercial, ya que cuando te mantienes en una versión antigua con el paso del tiempo deja de ser atractivo y de ahí la necesidad de evolucionar. Las nuevas versiones suelen venir cargadas de nuevas APIs o librerías de gran utilidad para los desarrolladores de aplicaciones, de tal forma que aumenta el atractivo de cara al consumidor, otras veces también con las nuevas versiones vienen correcciones de defectos apreciados o mejoras.

La versión más actual de Android es **Pie 9.0 API 28**, esta versión reserva energía para los procesos y aplicaciones que se utilizan realmente, proporciona mayor rapidez al dispositivo, más eficiencia, navegaciones más intuitivas y mayor uso de gestos y mayor precisión

Android Studio es el entorno de desarrollo de la aplicación.Hoy en día este entorno ya dispone de herramientas de emulación de aplicaciones y de gestor de paquetes de Android para cargar e instalar las APIs.

Agentes implicados

Android Studio es un entorno de desarrollo de aplicaciones **Android** y familiarizarse con él era uno de los primeros objetivos del presente **TFG**.

Este proyecto se ha implementado con Java y XML, también existía la posibilidad de desarrollarlo en Kotlin pero Java y XML es un lenguaje más familiar y a parte es la pareja de lenguaje recomendada por Google para el desarrollo de aplicaciones.

Este proyecto se ha implementado finalmente con Java y XML ya que es un lenguaje más familiar y a parte es la pareja de lenguaje recomendada por Google para el desarrollo de aplicaciones.

Java

Es un lenguaje que tiene como una de sus principales características la posibilidad de ejecutar programas en casi cualquier plataforma. Además, una de las ventajas de usar este lenguaje para desarrollar aplicaciones Android es que Google proporciona muchas herramientas de ayuda.

XML

Es un lenguaje de marca basado en las etiquetas. Gracias a este lenguaje se puede almacenar información y datos de forma legible para los humanos y para los ordenadores. Con él se crean los elementos que forman parte de la aplicación, es decir, con XML se crea y organiza lo que aparecerá en la pantalla del proyecto.

Funcionamiento

Instalación

Lo primero que se debe hacer es instalar Android Studio [6], la plataforma para el desarrollo de la aplicación.

Primeros pasos

Para comenzar a desarrollar un proyecto en Android Studio[5] se tienen seguir los siguientes pasos:

1. **Crear un nuevo proyecto**
2. **Generar la actividad principal y su diseño.** En este caso, y que suele ser el más recomendado, se seleccionó una actividad vacía para así empezar desde cero. Esta actividad vacía posee solamente un TextView a modo de ejemplo.
3. **Dar un nombre al proyecto.**
4. **comenzar a desarrollar el nuevo proyecto.**

Preparación entorno de trabajo

Antes de dar paso al comienzo del desarrollo del proyecto, cabe mencionar que el proyecto se subió a Github2.4.3 para trabajar con un control de versiones.

Con Github

Para poder trabajar con Github en Android Studio, es necesario tener una cuenta. Una vez que se tiene la cuenta, se ha de crear un repositorio donde se guardará el proyecto. En este caso se llama TFG.

Posteriormente hay que acceder a *File >Settings >Version Control >Github* e indicar los datos de la cuenta de usuario. A continuación, ha de pulsarse el botón Test para comprobar que los datos de conexión son correctos.

Una vez hecho esto hay que obtener el fichero git.exe, cuando el fichero ha sido descargado se accederá a File >Version Control >Git y se busca el fichero previamente descargado.

Cuando ya el proyecto se encuentra configurado con Github de forma remota, a la hora de realizar un **commit** y un **push** con los cambios del mismo, se debe acceder a *VCS >Commit*. Además, para hacer el push hay una opción que permite hacerlo junto con el commit directamente.

Comienzo proyecto

Para comenzar el proyecto hay que tener clara la manera en que se generan actividades en Android Studio. Una actividad es una pantalla formada por diferentes layouts que forma parte de la aplicación.

Actividad

Toda actividad que se genere en la aplicación siempre debe tener su archivo **Java** y su correspondiente **layout**.

Sección Java de una actividad

En las clases Java se generan los métodos y las variables a utilizar para realizar las funcionalidades deseadas. Y como toda clase es necesario generar su constructor y su destructor. Cada constructor será individual para el objeto, es decir, no puede ser accedido desde otra clase.

Listing 2.1: Constructor y Destructor

```
1     @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.fin);
    }

    @Override
    public void onDestroy(){
         super.onDestroy();
    }
```

La función *onCreate()* será la ejecutada en primera instancia nada más abrir la actividad. En esta función es donde se inicializan las *variables locales*, que se usarán en toda la clase. Y también es donde se le asignará el layout correspondiente, es decir, su **interfaz gráfica**.

Se puede navegar entre distintas actividades. Para ello se utilizan los disparadores pasándolo como parámetro a la función *startActivity(intent)*.

Listing 2.2: Disparador

```
    Intent cambiar = new Intent(Ruta.this, Beacon.class);
    cambiar.putExtra("Datos", mDeviceList);
    finish();
    startActivity(cambiar);
```

Se utiliza la función *finish()* antes de cambiar de actividad porque permite destruir la actividad actual. A estos lanzadores, en el caso de que queramos mover información de una actividad a otra, se hará mediante la función *putExtra(nombre_acceso, variable)*.

Para poder acceder a esas variables transmitidas de una actividad a otra se usan las siguientes líneas de código:

Listing 2.3: Bundle

```
    Bundle datos,valor,check; //Declaración de variables de la clase Bundle
        que es el mapa de todos los extras agregados
    datos = getIntent().getExtras(); //Obtiene un mapa de datos extendidos
    datos_obt= datos.getString("Datos"); //Obtiene el valor de la variable
        pasada con el identificador Datos
```

Sección XML de una actividad

En los ficheros XML se introducen las descripciones gráficas que formarán una actividad. Cada recurso o atributo añadido tendrá sus propias características. Algunos ejemplos de recursos son:

- *TextView*. Elemento gráfico que servirá para mostrar fragmentos de texto estático definido por el programador o texto dinámico que puede definirse en el código Java.
- *Button*. Elemento gráfico que la mayoría de las veces se usa para una acción concreta, como cuando el usuario pincha en él, por ejemplo, para ir a otra actividad. La consecuencia de pulsar dicho botón de definirá en el programa java con una función denominada onCLickListener(), que como indica, se lanzará cuando se realice el click.

- *EditText:* Se trata de un elemento gráfico en el cual el usuario puede insertar caracteres o números que pueden almacenarse en una variable interna del programa.
- *ListView:* Elemento gráfico que, al igual que el TextView, puede ser estático o variable en función que como se defina. Se usa para listar elementos.

Además, algunas de las características que se le pueden añadir a estos recursos son las siguientes:

- *Android:id* . Con este atributo se define el nombre identificador de cada uno de los elementos gráficos para, de esta manera, poder acceder a ellos.
- *Android:layout_width y Android:layout_height*. Estos atributos sirven para establecer las dimensiones gráficas del elemento. Las dimensiones se definirán con un número real acompañado de los caracteres dp (densidad de píxeles, dimensión fija), sp (scale pixeles, dimensión variable), in(pulgadas), mm(milímetros).
- *Android:text.* Atributo que permite fijar un texto dentro del elemento gráfico.
- *Android:textColor.* Atributo que permite definir el color del texto contenido en el elemento gráfico.

Este fichero puede ser rellenado o modificado gracias a dos ayudas, una mediante código primera imagen y otra por diseño, arrastrando los componentes necesarios segunda imagen. Ver imagen 2.1

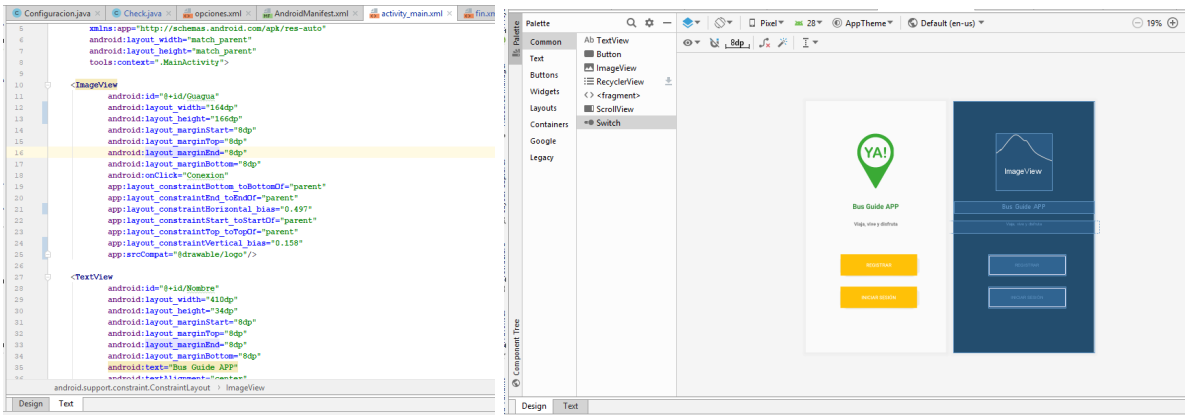


Figura 2.1: Diseño.

Ficheros básicos

AndroidManifest

El fichero **AndroidManifest.xml** es un archivo de configuración donde podemos aplicar las configuraciones básicas de la aplicación. Puede ser modificado a través de la interfaz gráfica. En este fichero se incluyen todos los permisos a los que puede acceder al usuario y la propia aplicación. El fichero está situado en la raíz de cada aplicación.

BuildGradle [3]

Gradle es un sistema de compilación que reúne en uno las mejores prestaciones de otros sistemas de compilación. Se basa en Java Virtual Machine (JVM), lo que significa que se puede escribir en el script en Java y Android lo entenderá y lo usará.

En definitiva, gradle es un plugin que facilita la actualización y la exportación de un proyecto a otro. Esto significa que un desarrollado puede tener su propio lenguaje de programación y automatizar el proceso de compilación en un solo paquete, pudiendo así distribuirlo.

Cada vez que se modifica cualquiera de los dos ficheros nombrados a continuación, es necesario sincronizar el proyecto para ver si todo va correcto. Si no, no se podrá ejecutar la aplicación.

A nivel de app

Es el archivo específico para el módulo de la aplicación. Estos ficheros contienen configuración de Android (CompileSdkVersion y BuildToolsVersion). También contiene DefaultConfig y ProductFlavors (Propiedades de manifiesto como minSdkVersion, targetSdkVersion, información de prueba y applicationId). Contiene los BuildTypes que construyen propiedades como debuggable, habilitar ProGuard, firmar depuración, sufijo de nombre de versión y testinformation. Y por último, contiene las dependencias.

A nivel de proyecto

Es el archivo de construcción del proyecto. Este archivo de generación de proyecto contiene buildscript, que define repositorios y dependencias, y también contiene la versión del complemento gradle.

Capítulo 3

Aplicación desarrollada

3.1. Propuesta

En este TFG se ha diseñado e implementado un sistema basado en una aplicación móvil que permite el guiado en el transporte público para todas las personas que lo utilizan. En concreto, y dadas las virtudes que ofrecen las tecnologías móviles disponibles en el mercado, se ha hecho uso de dispositivos basados en Android para crear dicho sistema. Este sistema complementa las aplicaciones ya existentes relacionados con el tema tratado, proporcionando una aplicación más concreta y basada en el uso de beacons mediante BLE.

3.2. Requisitos

A continuación se muestra una enumeración y breve descripción de los requisitos establecidos para el diseño y desarrollo de la aplicación.

1. **Salidas y Destinos.** Después de que el usuario entre en la aplicación a través de su cuenta, la pantalla que pide una parada desde la cual realizar la salida del trayecto, o también puede detectar, si el usuario se encuentra en una parada, ésta como la salida. Esta pantalla debe mostrar una lista de destinos a los cuales el usuario puede llegar y una lista de salidas desde el usuario puede salir.

2. **Instrucciones de ruta.** La aplicación debe presentar paso a paso las instrucciones de forma sencilla para que el usuario siempre sepa qué hacer. Cada instrucción tendrá un checkbox que no podrá ser manejado por el usuario, sino que será manejado por la aplicación. Así el usuario sabrá que ha hecho y que le falta por hacer.

3. **Rutas favoritas.** La aplicación debe permitir al usuario de manera sencilla añadir o eliminar rutas favoritas y mostrarle cómo hacerlo y cómo acceder a estas.

3.3. Objetivo del proyecto

El objetivo de este proyecto consiste en la localización de beacons por medio de la tecnología Bluetooth. Estos beacons se encuentran en el exterior, concretamente en las paradas del transporte público.

Se pretende que cuando un usuario quiera realizar una trayectoria, especificando su salida y su destino, se le muestre la ruta a realizar. En medio del trayecto, cuando sea detectado un beacon, la aplicación lo interpretará y le notificará al usuario si tiene que bajarse en esa parada y coger otra línea, o simplemente no bajarse. En cualquiera de los casos, en medio de la trayectoria el usuario podría cambiar su trayecto y la aplicación

le mostraría esa nueva ruta. Cuando el usuario llegue a su destino será avisado de que ha llegado.

Para el desarrollo de un sistema capaz de realizar todas las interacciones nombradas por medio de comunicaciones Bluetooth, se ha desarrollado una aplicación para el sistema operativo Android, la cual dispone de las herramientas necesarias para realizar todos los objetivos mencionados a continuación:

- **Familiarizarse con todo el entorno de trabajo.** Android Studio y Firebase.
- **Realizar una pequeña aplicación** que permita detectar dispositivos, por medio de BLE.
- **Diseñar un estructura de datos** para almacenar la información de las paradas y los vehículos de transporte en Firebase Database.
- **Diseñar una estructura de datos** para almacenar la información del usuario que usa la aplicación en Firebase Auth.
- Cuando el usuario ya se encuentre en una parada, que la parada sea detectada y le **muestre los destinos a los que puede llegar.**
- **Mostrar la ruta** a realizar al usuario.
- **Avisar a los usuarios** cada vez que se detecte un beacon de una parada.

3.4. Aplicaciones

Esta aplicación, a parte de ser útil para la función principal, también tiene otras funciones secundarias que serán nombradas a continuación.

- **Localización de la parada en la que se encuentra en usuario.** Cuando el usuario no sabe en qué parada se encuentra la aplicación la detecta y le ofrece al realizar su ruta desde ahí o desde una distinta, pero así el usuario tendría una idea de donde se encuentra.
- **Agregar rutas a favoritos.** Cuando el usuario realiza una ruta de manera cotidiana tiene la posibilidad de agregarla a favoritos y así acceder a ellas de manera más rápida.

3.5. Diseño

El diseño utilizado es un **Diseño Centrado en el Usuario (UCD)**. Este tipo de diseño tiene como objetivo la creación de un producto que resuelva las necesidades de los usuarios, consiguiendo la mayor satisfacción y mejor experiencia de uso posible con el mínimo esfuerzo por parte de estos. Cada decisión tomada se basa en las necesidades, objetivos, expectativas, motivaciones y capacidades de los usuarios.

Este diseño en concreto cumple las siguientes etapas.

- **Diseño de un prototipo que resuelva las necesidades del usuario.** Cómo mostrarle al individuo la información de manera que le parezca lo más sencilla posible.
- **Poner a prueba el diseño.** Para ello, usando un test de usuarios con personas de diferentes edades.

Durante el desarrollo de este diseño UCD, se ha pasado por distintos ciclos de vida hasta llegar a la maqueta final. En primer lugar se han desarrollado las maquetas por medio de la aplicación JustinMind 2.4.1.

Dentro de la estructura del proyecto se mostrará cómo se encuentra diseñada la aplicación actualmente y cómo era el diseño preliminar. Se podrá apreciar que en todos los diseños actuales respecto al diseño preliminar se han realizado bastantes cambios.

3.5.1. Diseño preliminar

Al comienzo de la implementación el diseño de la aplicación iba a ser muy sencillo, formado por 5 actividades. Únicamente el usuario tenía que registrarse o iniciar sesión para poder usar la aplicación. Una vez que el usuario entrara, tenía que indicarle a la aplicación de donde quería salir y a donde quería llegar. Una vez proporcionado, la aplicación le daría al usuario una ruta, la cual podría modificar en cualquier momento del trayecto.

Cuando el trayecto ha sido mostrado al usuario, con la información de que debe hacer en cada parada, comienza el trabajo con los beacons. Este trabajo consiste en que cada vez que detecta una parada, va a avisar al usuario de donde se encuentra hasta llegar a su destino.

3.5.2. Diseño final

Al final el diseño ha sido terminado siendo más complejo. Se han añadido nuevas pantallas, teniendo un total de once. En la versión actual, el usuario una vez entra a la aplicación, puede configurar su cuenta, cambiar la contraseña, poner una foto de perfil y poner un nombre de usuario. También podrá añadir o eliminar rutas, es decir, si un usuario habitualmente realiza un trayecto puede añadirlo a favoritos y así acceder a él directamente.

3.5.3. Funciones que controla

- Una parada que ha sido detectada no se volverá a detectar durante el trayecto.
- La aplicación únicamente detecta paradas que están en la ruta correspondiente.
- Se detectan las paradas en orden, es decir, si pasa antes por la parada número dos que por la parada número uno, no detectará la dos sino que esperará a detectar la número uno.
- Se usan las notificaciones de sonido y vibración cada vez que se desea llamar la atención al usuario.
- Se comprueba que el dispositivo tenga activado el Bluetooth y la ubicación para que la aplicación pueda funcionar correctamente.
- Si el usuario se pasa la parada, se le avisará de que debe bajarse e iniciar la nueva ruta desde la nueva parada.
- Si el usuario no está pendiente del móvil y se detecta la parada número uno y el usuario no para la información y se detecta la parada siguiente, avisará al usuario de esa parada número dos detectada.
- El usuario en medio de la ruta puede encontrarse una parada que no pertenece a su ruta, pero le pilla de camino a la autobús, en ese caso la aplicación no lo

tomará como que se ha salido de ruta, sino como una simple parada que pillaba de camino. Esa parada no será detectada.

3.6. Mapa

La aplicación se basa en el mapa que puede observarse en la imagen 3.1. Este mapa muestra las diferentes paradas que la forman, todas ellas pudiendo ser la salida o el destino elegido por el usuario.

Cada parada tiene una dirección MAC. Por ejemplo, para la parada Intercambiador La Laguna, el beacon que se encuentra situado en esta parada tiene la dirección CC:F7:38:83:39:83. Esa información se encuentra guardada en la base de datos Real-time de Firebase.

En conclusión, cada parada tiene asignado únicamente un beacon con la dirección correspondiente y cada beacon tiene asignado únicamente una parada del mapa. Ver imagen 3.1

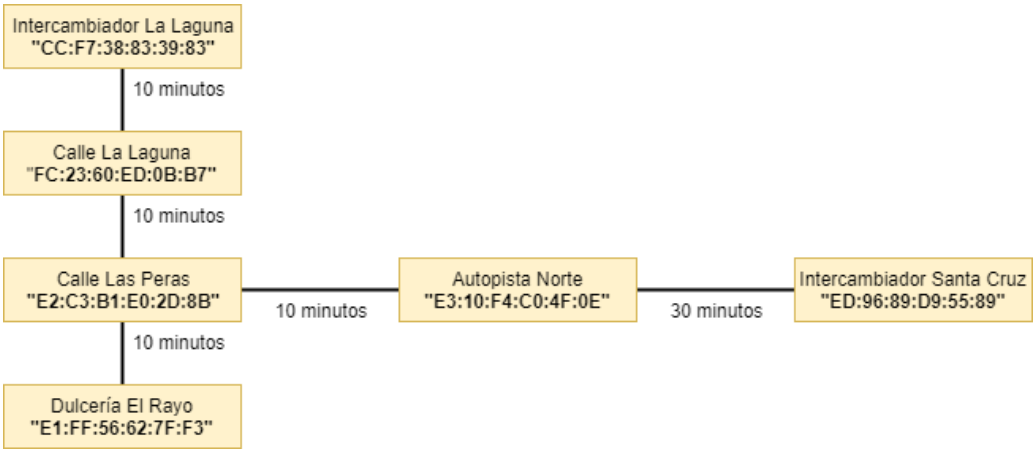


Figura 3.1: Mapa

3.7. Estructura

Esta aplicación consta de once actividades que forman las pantallas. Cada una de ella con funciones completamente distintas. En el Apéndice A de esta memoria se puede encontrar el código de cada una de ellas.

3.7.1. MainActivity

Esta es la actividad que forma la pantalla principal de la aplicación. Esta actividad le permite al usuario es entrar a la aplicación, bien sea registrándose pinchando el botón *Registrar* que le conduciría a la actividad *Registro* 4.1 o iniciando sesión pinchando el botón *Iniciar sesión* que le conducirá a la actividad *Iniciar* 3.5. También es donde se controla si el usuario tiene activo o no el Bluetooth y/o la ubicación, en caso de no ser así le pregunta al usuario que si quiere activarlos, en caso negativo la aplicación se cierra.

Bluetooth

Listing 3.1: Comprobación Bluetooth

```
1  BluetoothAdapter mBluetoothAdapter=BluetoothAdapter.getDefaultAdapter();
       //Obtiene el adaptador del dispositivo
2
```

```
3      if (mBluetoothAdapter == null) {
4          AlertDialog.Builder builder= new AlertDialog.Builder(this);
5          builder.setMessage(R.string.fundir);
6          builder.setNegativeButton(R.string.aceptar,null);
7          Dialog dialog=builder.create();
8          dialog.show();
9      }else {
10         if (!mBluetoothAdapter.isEnabled()) {
11             Intent enableBtIntent = new Intent(BluetoothAdapter.
                   ACTION_REQUEST_ENABLE);
12             startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
13             if (!mBluetoothAdapter.isEnabled()) {
14                 finish();
15             }
16         }
17     }
```

Primero comprueba si el Bluetooth es compatible con la aplicación, en caso de no serlo por medio de *buildersetMessage* construye un mensaje de aviso al usuario. Una vez comprobado eso se comprueba si el Bluetooth está activo *mBluetoothAdapter.isEnabled* en caso de no estarlo le pregunta al usuario si quiere activarlo si el usuario deniega se cierra la aplicación, en caso de que acepte se activa automáticamente.

Ubicación

Listing 3.2: Comprobación Ubicación

```
1      LocationManager lm = (LocationManager)getSystemService(Context.
           LOCATION_SERVICE);
2      if((!lm.isProviderEnabled(LocationManager.GPS_PROVIDER)) && (!lm.
           isProviderEnabled(LocationManager.NETWORK_PROVIDER))){
3          AlertDialog.Builder builder = new AlertDialog.Builder(this);
4          builder.setMessage("El sistema GPS está desactivado, ¿Desea activarlo
               ?")
5              .setPositiveButton("Si", new DialogInterface.OnClickListener() {
6                  public void onClick(@SuppressWarnings("unused") final
                       DialogInterface dialog, @SuppressWarnings("unused") final
                       int id) {
7                      startActivity(new Intent(android.provider.Settings.
                           ACTION_LOCATION_SOURCE_SETTINGS));
8                  }
9              })
10             .setNegativeButton("No", new DialogInterface.OnClickListener() {
11                 public void onClick(final DialogInterface dialog,
                       @SuppressWarnings("unused") final int id) {
12                     finish();
13                 }
14             });
15     Dialog dialog=builder.create();
16     dialog.show();
17 }
```

Primero obtiene la ubicación del dispositivo y comprueba si está activo, en caso contrario le pregunta al usuario que si quiere activarlo si el usuario deniega se cierra la

aplicación, en caso de afirmación se cambia a la configuración del dispositivo para activar la ubicación.

Por último, hay veces que el BLE no funciona correctamente por el uso en exceso de este. Por ello se ha creado una actividad adicional llamada *Conexion* 3.3 que permite al usuario comprobar si todo funciona correctamente. Se podrá acceder a esa actividad a través de la imagen del logo. Ver imagen 3.7.1

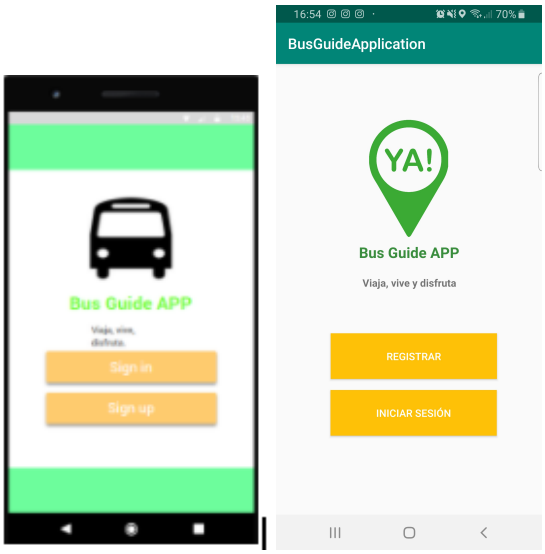


Figura 3.2: MainActivity

En la primera imagen puede verse el diseño preliminar y en la segunda el diseño final. Respecto a las funciones, la interfaz actual no ha variado.

3.7.2. Conexion_2

Esta actividad es para realizar las pruebas del funcionamiento de BLE. Ver imagen 3.3

Esta actividad en un comienzo no estaba pensada realizarla. Pero al ver que existía un problema cuando se usaba en exceso BLE, se decidió implementar.

3.7.3. Registro

En esta actividad el usuario puede crearse una cuenta para poder utilizar la aplicación. Para ello, tiene que introducir un correo en el formato correcto y una contraseña. Ambas contraseñas tienen que ser iguales. Una vez que el usuario ha dado toda esta información, puede hacer clic en el botón *Registrar* y, si esta todo correcto, se moverá a la pantalla principal 3.6. Además, también se comprueba si ya existe una sesión sin cerrar previamente, para que vaya directamente a la pantalla principal sin pasar por esta actividad.

Y la otra opción existente en esta actividad es hacer clic en el botón *Sign in with Google* para entrar con una de las cuentas Google del dispositivo. Si está todo correcto, accederá a la pantalla principal de la aplicación. Ver imagen 3.4

En la primera imagen puede observarse el diseño preliminar de la aplicación y en la segunda imagen el diseño inicial. Respecto a las funciones, se puede apreciar que al usuario, actualmente, no se le pide el nombre para registrarse ya que con el correo es suficiente. Otra de las funciones añadidas es la posibilidad de registrarse con una

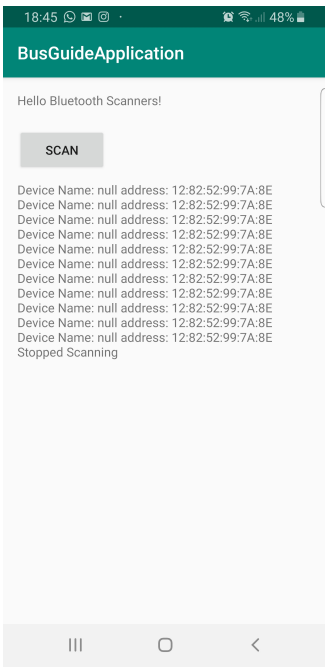


Figura 3.3: Conexion

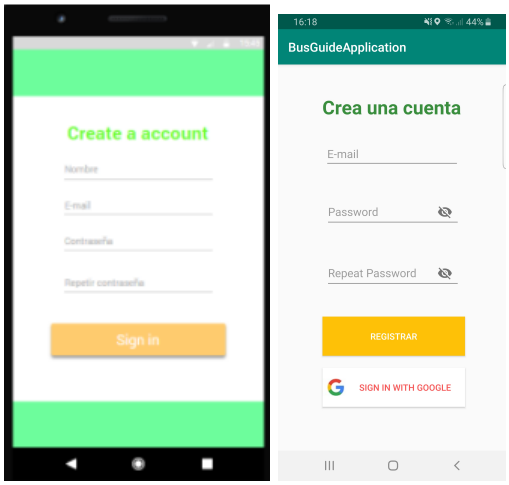


Figura 3.4: Registrar.

cuenta Google.

3.7.4. Iniciar

Desde esta pantalla, el usuario puede entrar a la aplicación con su correo electrónico. En cuando el usuario hace clic en el botón *Iniciar sesión*, si todo está correcto, pasará a la pantalla principal 3.6. Ver imagen 3.5

En la primera imagen se puede apreciar el diseño actual y en la segunda el diseño preliminar. Respecto a las funciones, se puede apreciar que no ha cambiado nada. Esta actividad, junto a *MainActivity* 3.7.1, son las que más se acercan al diseño preliminar.

3.7.5. Inicio

En esta pantalla el usuario especifica en la aplicación de donde quiere salir y a dónde quiere llegar. Una vez que el usuario elija una salida y un destino válido, le podrá dar al botón *Buscar* y pasar a la pantalla *Ruta* 3.9. La actividad contiene dos diales selectores

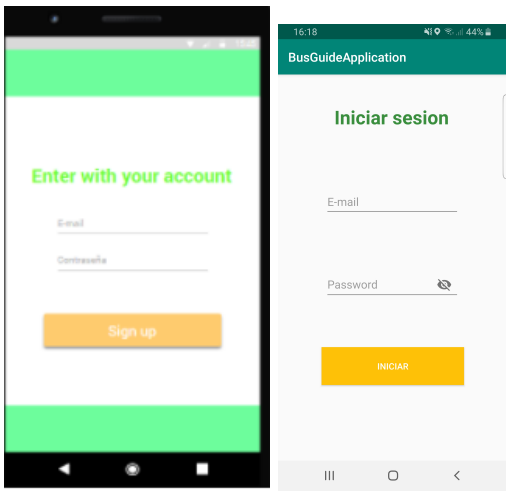


Figura 3.5: Iniciar.

para que el usuario seleccione la parada a la que quiere llegar y de donde quiere salir. Estos selectores acceden a la base de datos para darle al usuario todas las rutas posibles.

Esta actividad contiene la búsqueda de dispositivos, por si el usuario se encuentra en una parada, detectar el beacon y preguntar al usuario si desea salir de ahí o de otra parada. En caso afirmativo, solo tendrá que proporcionar el destino.

Desde esta pantalla, el usuario también puede acceder a la pantalla de *Configuración* 3.8 o salir de su cuenta. Cuando cierra la sesión, la aplicación lo desplazará a la actividad *Mainactivity* 3.7.1.

Por último, desde esta pantalla el usuario también puede acceder a sus rutas favoritas. Si el usuario hace clic en el botón asignado para ello, se desplazará a la actividad *Favorito* 3.7. Ver imagen 3.6

En el diseño preliminar esta actividad no existía. Más bien era una actividad mezcla entre la actual y Ruta 3.9. En el caso del diseño actual se han separado porque es más sencillo para el usuario manejarse con dos pantallas bien diferenciadas que con una solamente.

3.7.6. Favorito

En esta actividad el usuario dispone de todas las rutas que tiene agregadas a favoritos. Todas ellas contienen un check seleccionable por el usuario. Cuando el usuario elige únicamente una de las rutas y le da al botón *Buscar*, accederá a la actividad Ruta 3.9. El usuario también podrá eliminar una o varias rutas de las que tiene agregadas. Esto lo consigue seleccionándola o los checkbox y pulsando el botón *Eliminar*.

También podrá volver a la pantalla Inicio 3.6 pinchando el botón *Salir*. Ver imagen 3.7

Esta actividad en un comienzo no estaba planteada, pero se vio útil para el usuario ya que así puede acceder a sus rutas más comunes desde esta pantalla.

3.7.7. Configuracion

En esta actividad el usuario puede añadir o cambiar su nombre, cambiar la contraseña o añadir o cambiar la foto de perfil.

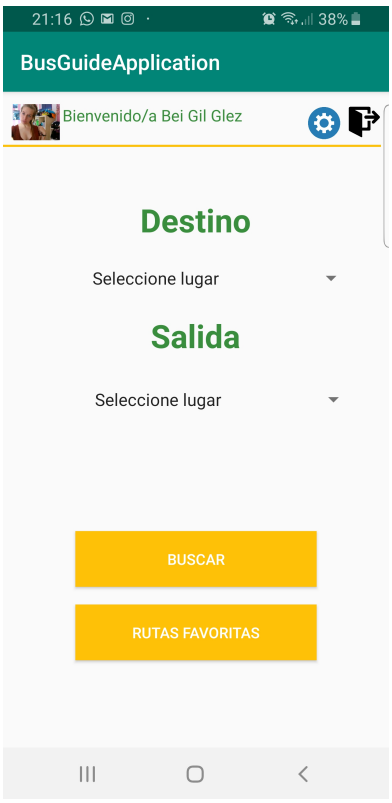


Figura 3.6: Inicio

Dispone de dos botones: el botón *Cambiar*, que le permite cambiar cualquiera de los datos rellenados, y el que le permite volver a la pantalla principal *Inicio* 3.6. Ver imagen 3.8

3.7.8. Ruta

En esta actividad son mostradas las diferentes acciones a realizar para llegar al destino deseado por el usuario (el número de paradas y el tiempo, que se van actualizando según se va llegando a las paradas). También se hace uso aquí de la búsqueda de dispositivos para la detección de paradas, paradas que se encuentran en ruta y que no han sido detectadas anteriormente. Cuando sea una parada detectada se moverá a la actividad *Beacon* 3.10

El usuario, gracias a esta actividad, puede agregar o quitar rutas de favoritos mediante los botones habilitados para ello. Por último, el usuario también puede cambiar la ruta por medio del botón *Cambiar la ruta*, volviendo en este caso a la actividad *Inicio* 3.6. Ver imagen 3.9

En la primera imagen se puede apreciar el diseño preliminar y en la segunda el diseño inicial. Como se detalla anteriormente, el diseño preliminar es una mezcla de la actividad *Inicio* 3.6 junto a esta. Al comienzo se pensó en hacerlo en una donde el usuario debía escribir el destino al que quería llegar y automáticamente se le mostrasen los datos. La salida siempre iba a ser detectada, pero ¿Que pasaría si el usuario no quiere salir de esa parada si no de otra? o ¿Si el usuario se encuentra en dos paradas y únicamente le detecta una? Por ello, el usuario actualmente tiene la posibilidad de elegir. Este fue uno de los muchos motivos por el que se pensó que dividirla en dos actividades sería más útil.

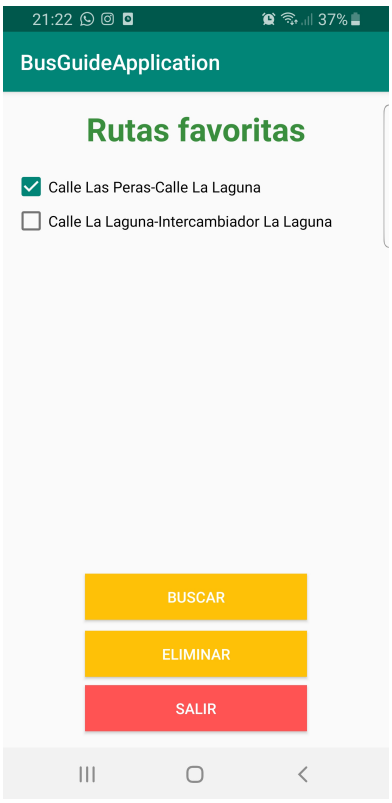


Figura 3.7: Favorito

3.7.9. Beacon

Esta actividad es usada cuando la mencionada anteriormente detecta un Beacon. En esta pantalla se muestra el nombre de la parada y la acción a realizar. En esta actividad se pueden encontrar dos botones *Cambiar Ruta* que le permite al usuario cambiar la ruta actual por una nueva y el botón *Parar información* que permite moverse a una de las dos actividades, Fin o Ruta. Este último botón puede volver a la actividad *Ruta* o a la actividad *Fin* esto depende de que parada sea, si es la parada destino va a la actividad *Fin* 3.11 y en caso contrario, se va a actividad *Ruta* 3.9.

Esta actividad también realiza búsqueda de dispositivos, ya que si el usuario no pulsa ninguno de los dos botones y el transporte en el que va llega a la parada siguiente, tendrá que avisar al usuario de ello. En caso de que el usuario no se baje en la parada destino porque no se dio cuenta, en cuanto detecte una parada fuera de ruta, la aplicación avisará al usuario de que se ha salido de ruta.

Otra característica de esta pantalla es que, cada vez que se detecta una parada distinta a la destino, se emite una vibración en el dispositivo y, en caso de que sea la parada destino, se reproduce un sonido haciendo que sea más fácil llamar la atención del usuario.

Vibración

Listing 3.3: Constructor y Destructor

```
1     Vibrator vibrator;
2     vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
3     if (vibrator.hasVibrator()) {
4        tiempo = 800;
5        vibrator.vibrate(tiempo);
```

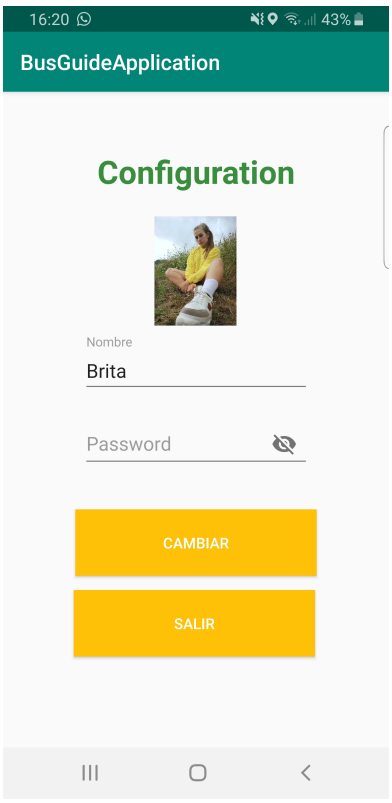


Figura 3.8: Configuracion

```
6       }
```

Sonido

R.raw.sonido contiene el sonido elegida. Esta carpeta se encuentra dentro de res, donde están todos los recursos de la aplicación. *R.raw.sonido* accede a la carpeta recursos donde se encuentra la canción descargada y se accede a ella por medio de esa línea. Ver figura 3.10

Listing 3.4: Constructor y Destructor

```
1       MediaPlayer mp;
2       mp=MediaPlayer.create(this, R.raw.sonido);
3       mp.start();
```

En la primera imagen se puede apreciar el diseño preliminar de la aplicación y en la imagen segunda el diseño inicial. Respecto a funciones, se puede ver que no ha variado. Únicamente se ha añadido otro layout con la acción a realizar en esa parada. Esto es debido a que si el usuario va distraído y se detecta una parada, éste no tenga que volver a la actividad anterior para saber qué hacer.

3.7.10. Fin

Esta actividad únicamente avisa al usuario de que ha llegado a la parada destino. Posee un botón Salir que le permite volver a la actividad *Inicio* 3.6. Ver imagen 3.11

Esta actividad es otra de las añadidas en el diseño final de la aplicación, la cual es útil porque si el usuario llega a su destino debe ser notificado de ello de forma clara.

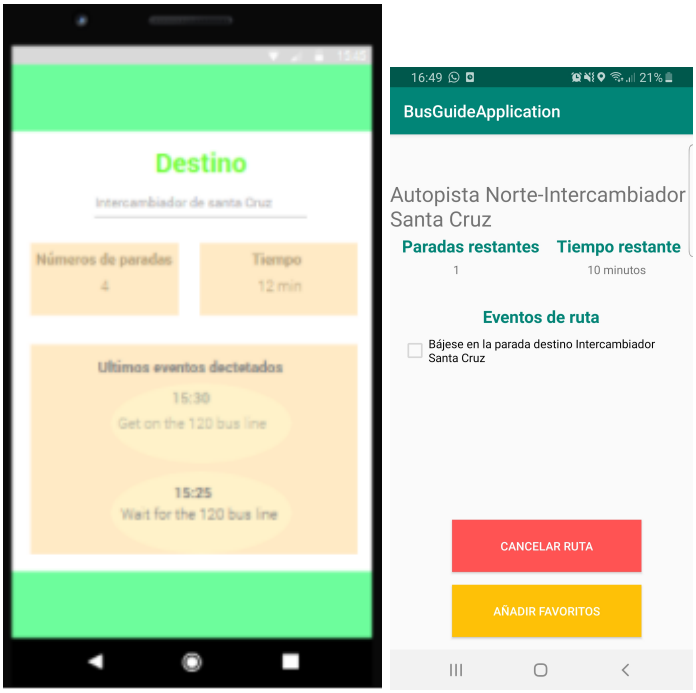


Figura 3.9: Ruta.

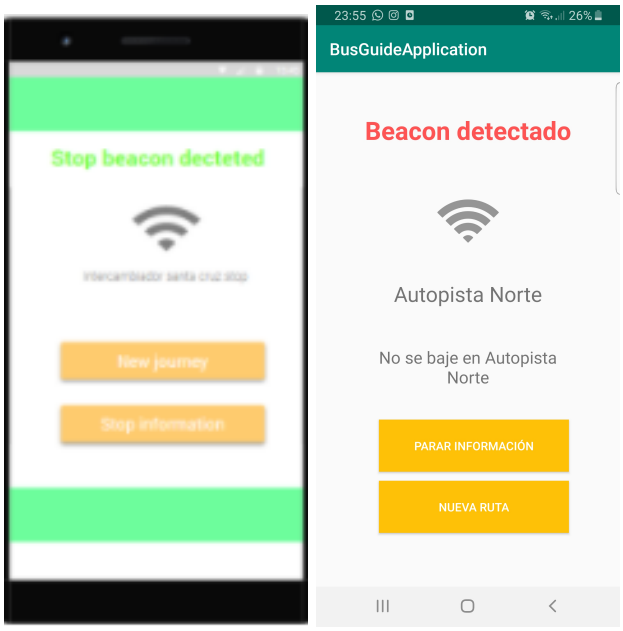


Figura 3.10: Beacon.

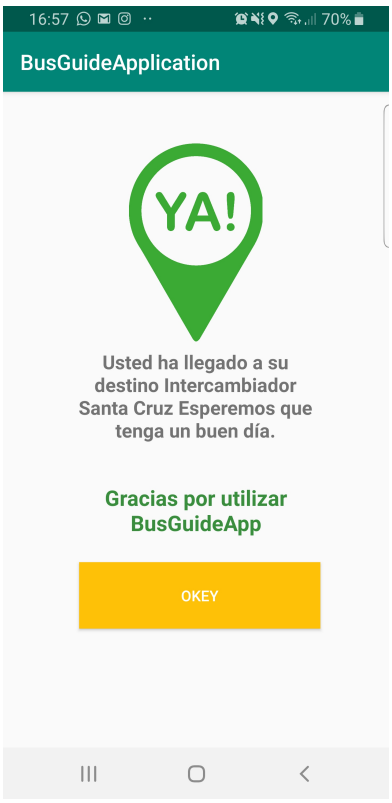


Figura 3.11: Fin

3.7.11. Check

Esta estructura, a diferencia de las otras, no es una actividad sino una clase y sirve para generar los checkbox que usan algunas de las actividades.

Únicamente tiene un constructor, el cual guarda el id del check (Ver imagen 3.12), su nombre/texto e inicializa su valor a *false*.

```
1   package com.example.busguideapplication;
2
3   public class Check {
4
5       public int cod;
6       public String nombre;
7
8       public Check(int cod, String nombre) {
9           this.cod = cod;
10          this.nombre = nombre;
11      }
12
13  }
14
```

Figura 3.12: Check

3.8. Sistema Bluetooth

3.8.1. Agentes implicados

Para realizar un correcta detección de beacons, los cuales simulan paradas de autobuses, es necesario utilizar Bluetooth. En el caso de este proyecto se necesita que la búsqueda sea constante, es decir, una búsqueda que comience cuando se requiera y que pare cuando se le indique.

Al comienzo del desarrollo del proyecto se comenzó realizando una búsqueda Bluetooth como la que encontramos en cualquier dispositivo móvil, es decir, una búsqueda que únicamente duraba entre 1-2 minutos. Sin embargo, la aplicación necesitaba una duración más duradera ya que entre diferentes paradas de autobuses puede existir más de 1 o 2 minutos de diferencia.

Por ello, finalmente, utilizamos Bluetooth Low Energy (BLE), que es una tecnología inalámbrica como Bluetooth pero con la diferencia de que ha sido diseñado para tener una duración de búsqueda más larga, tanto tiempo como el usuario desee. Proporciona un consumo de energía y un costo considerablemente reducidos, manteniendo un rango de alcance de comunicación similar. Este tipo de Bluetooth es compatible con los dispositivos móviles Android utilizados. Y lo más importante, esta tecnología ha sido pensada también para aplicaciones que utilizan beacons, como es en este proyecto.

3.8.2. Funcionamiento

Para poner BLE en funcionamiento, se ha generado una actividad dedicada únicamente para la búsqueda de dispositivos.3.3

Antes que nada es necesario agregarle permisos al fichero Manifest de acceso a Bluetooth y ubicación del dispositivo.

Listing 3.5: Permisos

```
1  <uses-permission android:name="android.permission.BLUETOOTH"/>
2  <uses-permission android:name="android.permission.BLUETOOTH\_ADMIN"/>
3  <uses-permission android:name="android.permission.ACCESS\_COARSE\
       _LOCATION"/>
4  <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION
       "/>
```

Una vez añadidas estas líneas de código y el usuario le dé los permisos a la aplicación, se continuará con las líneas de código de la propia actividad. Será necesario inicializar tres variables de clase:

Listing 3.6: Variables Bluetooth

```
1  BluetoothManager btManager;
2  BluetoothAdapter btAdapter;
3  BluetoothLeScanner btScanner;
4
5  btManager = (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE
       );
6  btAdapter = btManager.getAdapter();
7  btScanner = btAdapter.getBluetoothLeScanner();
```

La primera variable de la clase *BluetoothManager* representa el adaptador de Bluetooth. Utiliza la función *getSystemService(Context.BLUETOOTH_SERVICE)* para crear

una instancia de BluetoothManger. Es necesario utilizar ese contexto para poder utilizar bluetooth.

A continuación se inicializa una variable de la clase *BluetoothManager*, que es utilizada para obtener una instancia del *BluetoothAdapter* y llevar a cabo la administración del Bluetooth.Se llama a la función *getAdapter* para obtener el BluetoothAdapter y se guarda en *btAdapter*.

Y, por último, una variables de la clase *BluetoothLeScanner*. Esta clase proporciona métodos para realizar operaciones relacionadas con la exploración de dispositivos BLE. Que se usa para la función *getBluetoothLeScanner* que permite obtener la instancia de BluetoothLeScanner, es decir, esa función devuelve un objeto *BluetoothLeScanner* para realizar operaciones de búsqueda de BLE.

Una vez inicializada todas las variables que se van a utilizar, se podrá comenzar con la búsqueda de dispositivos. Para ello se debe llamar a la Función *startScan* y se le pasa como parámetro el objeto *leScanCallback* inicializado previamente.

Listing 3.7: StartScan()

```
1     btScanner.startScan(leScanCallback);
```

Para detener la búsqueda es de la misma forma pero llamando a la función stopScan

Listing 3.8: StopScan()

```
1    btScanner.stopScan(leScanCallback);
```

Por último,se dispone del objeto leScanCallback de la clase ScanCallback.

Listing 3.9: Clase ScanCallback()

```
1    private ScanCallback leScanCallback = new ScanCallback() {
2    @Override
3    public void onScanResult(int callbackType, ScanResult result) {
4        mDeviceList.add(result.getDevice().getAddress());
5    }
6
7    @Override
8    public void onBatchScanResults(List<ScanResult> results) {
9        super.onBatchScanResults(results);
10   }
11
12   @Override
13   public void onScanFailed(int errorCode) {
14       super.onScanFailed(errorCode);
15   }
16};
```

La clase ScanCallback es utilizada para obtener los resultados de la búsqueda y contiene cada dispositivo que detecta. Ese dispositivo detectado se guarda en el ArrayList mDeviceList. Cabe mencionar que es necesario que para que todo funcione correctamente se deben tener los 3 métodos inicializados. El primero se llama cada vez que se detecta algo, el segundo es la devolución de la llamada cuando se entregan los resultados del lote y el tercero se produce en caso de que de error. Además, no se debe olvidar tampoco tener activado el bluetooth y la ubicación para su correcto funcionamiento.

Capítulo 4

Base de datos utilizada

4.1. Agentes implicados

Para obtener la información del usuario o de las rutas es necesario acceder a una base de datos. Este proyecto se basa en el uso de Firebase, tanto para la autenticación de usuarios como para el acceso a las rutas. El sistema se aloja en la nube y la aplicación móvil accede a él.

Al comienzo del proyecto, únicamente era necesario utilizar Firebase Authentication para la autenticación de usuarios, pero una vez que se iba a desarrollando más la aplicación se vio necesario el uso de Database RealTime para poder guardar la información de la ruta y sus paradas.

Firebase nos proporciona dos tipos de base de datos: Cloud Firestore y RealTime Database (la usada en este proyecto). [10]

Realtime Database almacena y sincroniza datos con una base de datos NoSQL alojada en la nube. Esos datos se sincronizan con todos los clientes en tiempo real y se mantiene disponible cuando la app no tiene conexión. Esta base de datos almacena los datos en formato **JSON**. Permite la actualización automática de los datos. Si es información sencilla **es muy fácil de almacenar**. Las consultas son directas con funciones de ordenamiento y filtrado limitado. Tiene operaciones de escritura y transacción básicas. Sobre la seguridad de estos sistemas sigue reglas en cascada que se deben validar por separado. Las reglas de Firebase Database es la única opción disponible de seguridad, y debes validar los datos por separado por medio de la regla validate. En el caso de la aplicación solo se puede acceder a la base de datos si el usuario está identificado.

Por último, se ha elegido esta base de datos porque al ser información sencilla con la que se trabaja, el formato JSON es escalable y fácil de procesar.

4.2. Funcionamiento

Antes que nada hay que crear un nuevo proyecto por medio de la `consola firebase`[4], el cual queda asignado a la cuenta con la que se inicie sesión. Los proyectos de Firebase abarcan las aplicaciones de diferentes plataformas como Android o iOS. Una vez que se crea el proyecto, hay que añadir Firebase a la aplicación Android.

4.2.1. Autenticación

En la aplicación es necesario el registro e inicio de sesión de los usuarios para poder utilizarla. Para el registro, el usuario podrá hacerlo por medio del correo o de una cuenta Google. Véase [8] y [2]. A continuación, son explicados ambos casos.

Autenticación e-mail

Lo primero que se debe hacer para que Firebase permita crear o iniciar sesión con una cuenta y contraseña es ir al panel de control y en *Authentication >Métodos de acceso*, activar la opción *correo electrónico/contraseña*.

Registro

Cuando el usuario pulsa sobre el botón registrar, se llama a una función que en primer lugar revisa que los datos introducidos son correctos. Para ello se comprueba lo siguiente:

1. No exista ningún espacio vacío.
2. Tamaño de la contraseña superior a 6.
3. Las contraseñas escritas sean iguales.
4. Formato del correo sea correcto.
5. Si ese correo no tiene una cuenta ya existente

Listing 4.1: Registro

```
1  FirebaseAuth.getInstance().signInWithEmailAndPassword(email,pass).
       addOnCompleteListener(new OnCompleteListener<AuthResult>() {
2      @Override
3      public void onComplete(@NonNull Task<AuthResult> task) {
4          if(task.isSuccessful()){
5              Toast.makeText(Registro.this, "Este e-mail ya esta registrado"
                   , Toast.LENGTH_LONG).show();
6          }else{
7              FirebaseAuth.getInstance().createUserWithEmailAndPassword(
                   email, pass).addOnCompleteListener(new OnCompleteListener<
                   AuthResult>() {
8                  @Override
9                      public void onComplete(@NonNull Task<AuthResult> task
                           ) {
10                         if(task.isSuccessful()){
11                             Toast.makeText(Registro.this, "Usuario
                                   registrado", Toast.LENGTH_LONG).show();
12                         }
13                     }
14             });
15         }
16     }
17 });
```

En el código 4.1 primero comprueba si el usuario está registrado. Si ese correo ya existe se avisa al usuario de que ese correo ya tiene una cuenta. En caso contrario se llama a la función createUserWithEmailAndPassword(email,contraseñ) y se registra el usuario.

Iniciar sesión

Una vez comprobado, se llama a otra función que se encarga de ir a Firebase Authentication y comprobar si el correo con esa contraseña estan registrado y si son correctos los datos.

Listing 4.2: Iniciar sesión

```
1     FirebaseAuth.getInstance().signInWithEmailAndPassword(email,pass).
          addOnCompleteListener(new OnCompleteListener<AuthResult>() {
2         @Override
3         public void onComplete(@NonNull Task<AuthResult> task) {
4             if(task.isSuccessful()){
5                     Toast.makeText(Iniciar.this, "Usuario entrando", Toast.
                          LENGTH_LONG).show();
6             }else{
7                      Toast.makeText(Iniciar.this, "Usuario o contraseña
                           incorrecto", Toast.LENGTH_LONG).show();
8             }
9         }
10    });
```

Mediante la función *signInWithEmailPassword(email, contraseña)* se pasan los datos a Firebase para que realice la comprobación correspondiente.

Autenticación con google

Para poder trabajar con **cuentas Google** es necesario ir a *Authentication >Métodos de acceso* y activar la opción *correo Google*.

Listing 4.3: Autenticar Google

```
1     private GoogleApiClient googleApiClient;
2
3     GoogleSignInOptions gso = new GoogleSignInOptions.Builder(
          GoogleSignInOptions.DEFAULT_SIGN_IN)
4             .requestIdToken(getString(R.string.default_web_client_id))
5             .requestEmail().build();
6
7     googleApiClient = new GoogleApiClient.Builder(this).enableAutoManage(this
          ,this).addApi(Auth.GOOGLE_SIGN_IN_API,gso).build();
8
9     Intent intent = Auth.GoogleSignInApi.getSignInIntent(googleApiClient);
10 startActivityForResult(intent,777);
```

Las líneas del código 4.3 se obtiene toda la información necesaria para poder acceder a los datos de la cuenta google seleccionada, es decir, poder iniciar sesión. La función startActivityforResult se sobreescribirá para realizar las funciones necesarias para la aplicación.

Listing 4.4: StartActivityForResult

```
1     //Esta función si recibe el código 777 genera una variable perteneciente
          a la clase GoogleSigninResult que guardatodos los datos obtenidos
          sobre la cuenta.
2     @Override
3     protected  void onActivityResult(int requestCode, int resultCode, Intent
          data){
4         super.onActivityResult(requestCode, resultCode, data);
5         if(requestCode==777){
6             GoogleSignInResult result=Auth.GoogleSignInApi.
                  getSignInResultFromIntent(data);
7             handleSignInResult(result);
```

```
        }
    }

    //Esta función revisa si todo está correcto, en caso contrario avisa al
       usuario de que ha habido un problema con los datos obtenidos
    protected void handleSignInResult(GoogleSignInResult result){
        if(result.isSuccess()){
            firebaseAuthWithGoogle(result.getSignInAccount());
        }else{
            Toast.makeText(this, "No es posible entrar con ese correo", Toast
               .LENGTH_SHORT).show();
        }
    }
    //Y por último esta función obtiene las credenciales y revisa si son
       correctas en caso contrario avisa al usuario.
    private void firebaseAuthWithGoogle(GoogleSignInAccount signInAccount){
        AuthCredential credential = GoogleAuthProvider.getCredential(
           signInAccount.getIdToken(),null);
        firebaseAuth.signInWithCredential(credential).addOnCompleteListener(
           this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if(!task.isSuccessful()){
                Toast.makeText(getApplicationContext(),"No se puede autenticar
                   con Firebase", Toast.LENGTH_LONG).show();
            }
        }
    });
}
```

Una vez que se comprueba que todo está correcto, se podrá cambiar a la actividad *Inicio*.

Listing 4.5: Lanzador con indicadores

```
    Intent intent = new Intent(Registro.this,Inicio.class);
    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.
       FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(intent);
```

Comprobación usuario

El usuario cuando entra a la aplicación puede tener una **cuenta activa** con la cual accedería directamente a la aplicación. Esto se comprueba con las siguientes líneas de código.

Listing 4.6: Comprobación cuenta activa

```
    mAuthListener = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        FirebaseUser user= firebaseAuth.getCurrentUser();
        if(user!=null){
            goMainScreen();
        }
```

```
8     }
9 };
```

Esto se hace gracias a la función *getcurrentUser()* que comprueba si existe una cuenta activa.

Cerrar sesión

Como toda aplicación existente en el día de hoy, cuando un usuario inicia sesión con una cuenta puede salir de esta cuando quiera. Dependiendo de si la cuenta ha sido inicializada con una cuenta creada o con google se realizará de una manera u otra.

Correo

Listing 4.7: Cerrar cuenta correo

```
1     FirebaseAuth.getInstance().signOut();
```

Google

Te permite cerrar sesión como una cuenta creada, pero también mediante las siguientes líneas de código

Listing 4.8: Cerrar cuenta google

```
1         firebaseAuth.signOut();
```

Listing 4.9: Cerrar cuenta google

```
1     Auth.GoogleSignInApi.signOut(googleApiClient).setResultCallback(new
          ResultCallback<Status>() {
2         @Override
3         public void onResult(@NonNull Status status) {
4             if(status.isSuccess()){
5                 goLogInscreen();
6             }else{
7                 Toast.makeText(getApplicationContext(), "No se puede cerrar
                      sesión", Toast.LENGTH_LONG).show();
8             }
9         }
10    });
```

4.2.2. Configuración

La aplicación le permite al usuario **cambiar o poner datos** de su cuenta, como ponerle o modificar nombre, cambiar la contraseña y poner o modificar la foto de perfil.

Poner o modificar nombre

Listing 4.10: Poner cambiar nombre usuario

```
1     String nombre_nuevo = Nombre.getText().toString(); //Obtienes el nombre
          nuevo introducido por el usuario
2     UserProfileChangeRequest profileUpdates = new UserProfileChangeRequest.
          Builder().setDisplayName(nombre_nuevo).build(); //Le asignamos a esa
          cuenta el nuevo nombre
3     user.updateProfile(profileUpdates); //Actualizamos
```

Cambiar contraseña

Listing 4.11: Cambiar contraseña usuario

```
1    user.updatePassword(contraseña_nueva);
```

Poner o modificar foto perfil

Para poder poner una foto de nuestro dispositivo necesitamos **permisos de acceso a galería** por ello hay que añadir el siguiente permiso al fichero Manifest.

Listing 4.12: Permisos

```
1    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE
         "/>
```

Una vez dado los permisos, tanto con esta línea de código como dando permisos a la aplicación de acceso a la galería ya podemos acceder a la galería del dispositivo.

Listing 4.13: Poner o modificar foto de perfil usuario

```
1    Intent gallery= new Intent(Intent.ACTION_PICK, MediaStore.Images.Media.
         INTERNAL_CONTENT_URI);
2    startActivityForResult(gallery,CHOOSE_IMAGE);
3
4    @Override
5    protected void onActivityResult(int requestCode, int resultCode, Intent
         data){
6        if(resultCode==RESULT_OK && requestCode==CHOOSE_IMAGE){
7            FirebaseAuth firebaseAuth= FirebaseAuth.getInstance();
8            FirebaseUser user=firebaseAuth.getCurrentUser();
9            uriprofile=data.getData();
10           camarafotos.setImageURI(uriprofile);
11           UserProfileChangeRequest profileUpdates = new
                 UserProfileChangeRequest.Builder()
12               .setPhotoUri(uriprofile).build();
13           user.updateProfile(profileUpdates);
14   }
```

Las líneas de código añadidas anteriormente son las que nos permitirán elegir la foto de perfil deseada. Al comienzo se crea un disparador que nos permitirá coger una foto de la galería. Se le pasa toda la información obtenida a la función *onActivityResult* la cual tenemos que sobreescribir para que realice las funciones que deseamos. Dentro de esa función revisa si todo está correcto y asigna esa foto de perfil a la cuenta.

4.2.3. Database Realtime

En este proyecto es necesario el uso de una base de datos que almacene toda la información de los **trayectos** y los **usuarios**. Esta información se encuentra guardada en formato **JSON** y tiene las siguientes **reglas de seguridad** (Ver figura 4.1) para que cada usuario solo pueda acceder a su información una vez estén autenticados en la aplicación.[9]

Para acceder a la información de la base de datos desde el código de la aplicación es necesario obtener la referencia de la base de datos.

Listing 4.14: Crear e inicializar variables

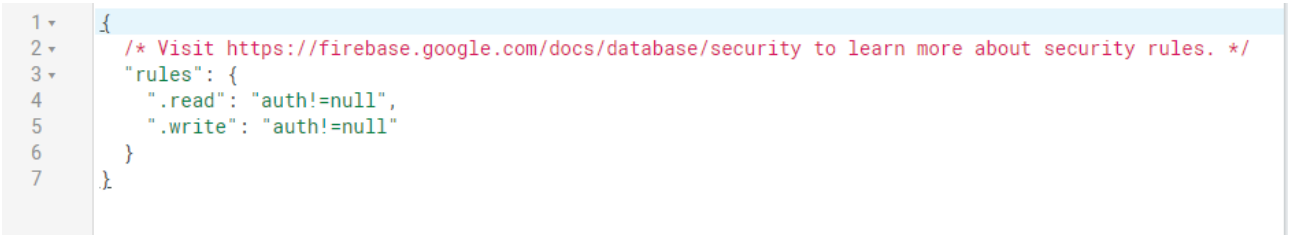


Figura 4.1: Reglas

```
private DatabaseReference mDatabase;
private FirebaseDatabase mFirebasedata;

mFirebasedata = FirebaseDatabase.getInstance();
mDatabase = mFirebasedata.getReference();
```

Consultar Base de Datos

Listing 4.15: Consultar base de datos

```
mDatabase.child(nombre_clave).addListenerForSingleValueEvent(new
    ValueEventListener() { //La función addListenerforsingleValueEvent
    significa que solo va a llamar una vez a la base de datos para
    comprobar los datos
@Override
    public void onDataChange(@NonNull final DataSnapshot dataSnapshot) {
        //Aqui se producen las consultas de la base de datos
            datasanpshot.getValue();
        }
    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {  //En
        caso de que de error el acceso a los datos
    }
});
```

Añadir o actualizar datos de la base de datos

Listing 4.16: Añadir o actualizar datos de la base de datos

```
mDatabase.child(nombre_clave).setValue(Valor_nuevo);
```

Borrar un dato de la base de datos

Listing 4.17: Borrar un dato de la base de datos

```
mDatabase.child(nombre_clave).setValue(null);
```

La rama principal de la base de datos del proyecto se denomina *Dispositivos*. De esta rama cuelgan otras cinco que son las siguientes:

- *Array_dest*. Se encuentran todas las paradas de las que dispone la aplicación. Todas ellas forman un mapa 3.1, más un valor inicial *"Seleccione lugar"*, el cual aparece por defecto antes de que el usuario elija una salida y un destino. Ver imagen 4.2

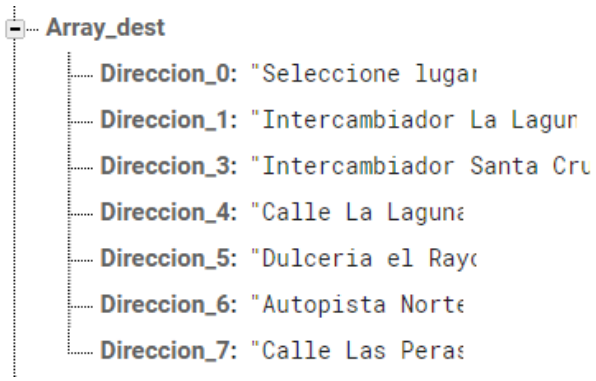


Figura 4.2: Destinos

- *Beacons*. Aquí se encuentran todos los beacons de los que dispone el proyecto. Cada uno de esos beacons contiene la dirección MAC, el nombre de la parada que representa y su firma digital. Ver imagen 4.2

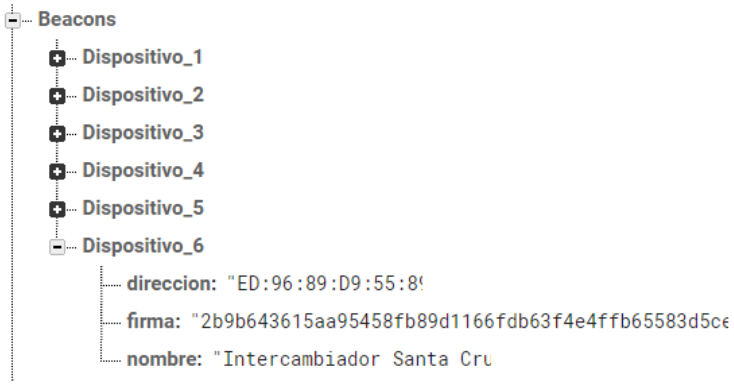


Figura 4.3: Beacons

- *Disp_Nombre*. Esta rama se encarga de asignar a cada parada su dispositivo correspondiente. Ver imagen 4.4

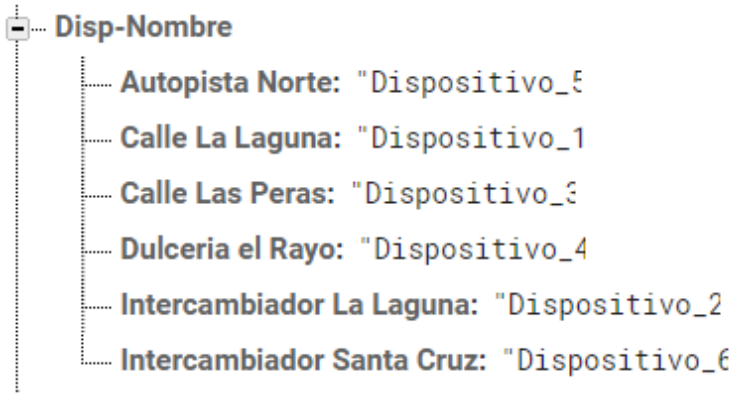


Figura 4.4: Dispositivos con nombre

- *Rutas*. Esta rama se podría considerar la más compleja y con más información. Está formada por todas las paradas del mapa. Dentro de cada parada se encuentran todas las paradas a las que puede llegar y dentro de esta los beacons que se

encuentran en las paradas que lo forman y la acción a realizar en ella. También contiene el número de paradas y el tiempo hasta el destino, que se va actualizando según se van pasando paradas. Ver imagen 4.5

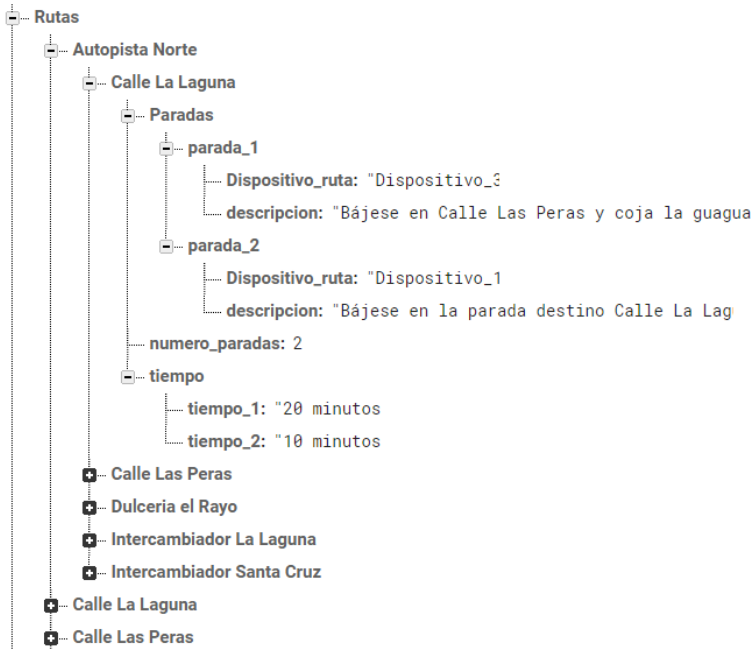


Figura 4.5: Rutas

La rama *Paradas* contiene todas las paradas que forman la ruta con la acción a realizar en esta y el dispositivo que se encuentra ubicado en ella.

- *Usuarios.* Para que puedan acceder varios usuarios a la aplicación es necesario mantener la información de cada usuario (Salida, Destino, paradas de su ruta etc) por separado. Para esto sirve esta rama. Ver imagen 4.6

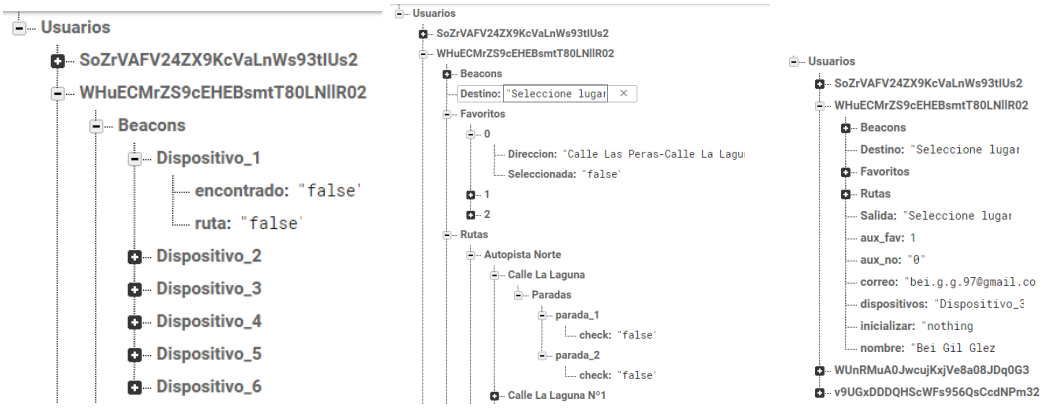


Figura 4.6: Usuarios.

Dentro de cada usuario se crea la rama *Beacons* con cada uno de los dispositivos existentes. Lo que contiene cada dispositivo es una rama *ruta* que está a true en caso de que la parada se encuentre en la ruta (así sólo detectará las paradas que están en ruta) y una clave *encontrado* que si es true significa que esa parada ya ha sido detectada y así si la vuelve a detectar no vuelve a avisar al usuario.

Luego encontramos la rama *Favoritos* que guarda las rutas que el usuario tiene agregadas a Favoritos. Esta contiene un rama *Dirección* que es la ruta que el

usuario ha agregado a favoritos y una rama seleccionar para controlar cuál ha sido seleccionada por el usuario para borrarla o para buscarla.

También contiene otras variables: la clave *usuario* que son la *salida* y el *destino* de la ruta, *aux_no* que controla si el usuario a querido o no salir de una parada detectada (0 es que no lo desea y 1 es que si), el *correo* y el *nombre del usuario*, *inicializar* que representa variable de ayuda que guarda el nombre de la parada que se ha detectado cuando el usuario quiere buscar una ruta y, por último, *dispositivo* que guarda cada uno de los dispositivos que forman su ruta. Ver imagen 4.7

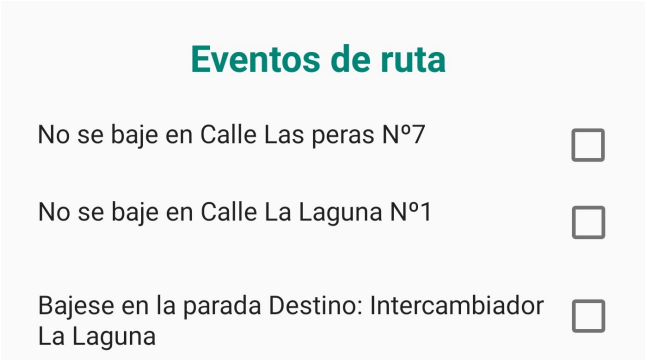


Figura 4.7: Checks

Otras variables son la *salida* y el *destino* de la ruta, *aux_no* que controla si el usuario quiere o no salir de la parada detectada, 0 no lo desea y 1 si lo desea, el *correo* y el *nombre del usuario*, *inicializar* que es una variable de ayuda que guarda el nombre de la parada que se ha detectado cuando el usuario quiere buscar una ruta y por último *dispositivo* que guarda cada uno de los dispositivos que forman su ruta.

Capítulo 5

Seguridad

La **seguridad** es necesaria para la protección de infraestructuras computacionales, concretamente protegiendo la información contenida en ella.

En este proyecto se ha decidido trabajar con el estándar **SHA** (Secure Hash Algorithm). SHA consiste en una familia de funciones hash. Existen 4 versiones SHA0, SHA1, SHA2 y SHA3 (el usado en el proyecto).

Para realizar la firma hash se ha realizado por medio de código, utilizando la librería **BouncyCastle**. Esta librería contiene una colección de APIS utilizadas en criptografía, teniendo versiones para Java, utilizado en Android.

5.1. Función Hash

Las funciones hash son funciones computables mediante un algoritmo que tiene como entrada un conjunto de elementos, que suelen ser cadenas, y los convierte en un rango de salida de tamaño concreto.

5.2. Funcionamiento SHA-3

El algoritmo no tiene un esquema como su familia pasada. Se trata de un sistema conocido como esponja basado en una amplia función o permutación aleatoria. La entrada consistirá en una cadena de bits de cualquier longitud que son absorbidos por esa esponja y se obtiene una salida de los bits deseados por el usuario simplemente exprimiendo dicha esponja.

El algoritmo es complejo y funciona internamente como un array de tres dimensiones con movimientos de permutación similares a los realizados sobre un cubo de rubik.

La salida puede obtener cuatro tamaños distintos, 224,256,386 y 512 (el realizado en este caso), dependiendo de ese parámetro se genera lo demás. Al comienzo de la etapa de absorción, para que el mensaje entre en bloques de r bits se rellena con un patrón mínimo de 10x1 que consiste en un bit, cero o más de un 0, máximo(r-1) y un bit final 1. Ver imagen 5.1.

La función se itera 24 veces, número máximo de iteraciones. Esta función implica 5 pasos:

- **Theta**(θ) La cual se realiza por medio de una operación XOR de cada bit con una de las columnas del mismo slice pero adyacente, luego se realiza otra vez la operación XOR pero con otra columna adyacente pero no del mismo slice.

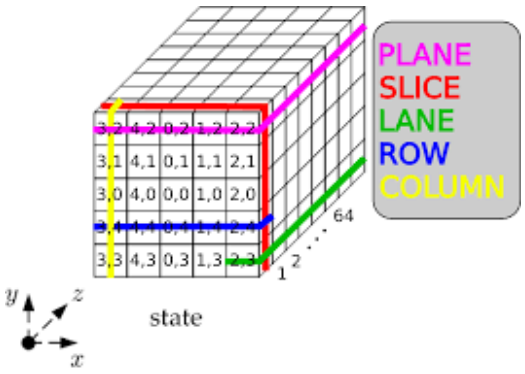


Figura 5.1: RSA

- **RHO** (ρ) Los bits se desplazan en su plane por un número bits dado
- **PI**(π) Es una permutación de filas de columnas
- **Chi** (χ) Es la única operación no lineal, está implica la operación XOR de una columna en particular con la operación AND de la negación de la columna adyacente y la columna de al lado
- **Iota** (ι) Es la operación XOR de una constante redonda y de lane [0,0] del estado.

5.3. Aplicación de SHA3-512 en aplicación

En este proyecto se generan unas firmas por medio de los hashes. Esa firma se encontrará registrada en la base de datos. Para la generación se usa la concatenación de la dirección del beacon de la parada y su nombre.

Las firmas se guardan en la base de datos debido a que si uno de los beacons que se encuentra en la parada ha sido corrompido, la firma obtenida por parte del código no será igual a la obtenida desde la base de datos, lo que significará que ese beacon ha sido corrompido y no será accedido desde la aplicación.

Cuando la aplicación detecta un beacon, obtiene su dirección MAC. Mediante esa dirección MAC, le pide a la base de datos que le dé el nombre de la parada y su firma. Para ver si no ha sido modificado ningún dato de los obtenidos de la base de datos, se genera por medio del código y gracias a la librería **BouncyCastle** la firma hash usando los mismos parámetros la dirección detectada más el nombre de la parada (obtenido gracias a Firebase). Si la firma coincide se avisa al usuario de que ha llegado correctamente.

Para poder añadir esta librería al proyecto se tuvieron que añadir las siguientes líneas a nivel de app en el fichero gradle.

Listing 5.1: Implementaciones **SpongyCastle**

```
1        implementation 'com.madgag.spongycastle:core:1.54.0.0'
2        implementation 'com.madgag.spongycastle:prov:1.54.0.0'
3        implementation 'com.madgag.spongycastle:pkix:1.54.0.0'
4        implementation 'com.madgag.spongycastle:pg:1.54.0.0'
```

Una vez añadidas, y sincronizado el proyecto, se podrá trabajar con la librería mencionada. Dentro del fichero donde se va a realizar la comprobación de firmas, la clase *Ruta* habrá que añadir las siguientes líneas, que realizarán la firma a través de código.

Listing 5.2: Código de generación firma

```
1        for(int i=0;i<firma.size();i++){   //Generará tantas firmas como
             firmas se encuentran en la base de datos
2            String direccion = direcciones.get(i);
3            String nombre dirección = nombre_direcciones.get(i);
4            String concat = direccion.concat(nombre_direccion); //
                 Concatenamos la dirección MAC con el nombre de la parada
5            byte[] input, output5;
6            StringBuffer hexString = new StringBuffer();
7            final SHA3.DigestSHA3 md = new SHA3.DigestSHA3(512); //Se llama
                 al constructor de SHA3 indicandoles el número de bits que
                 queremos de salida
8            input = concat.getBytes(); //Obtenemos los bits de nuestra
                 concatenación
9            md.reset(); //Reseteo por si existe algo en el objeto
10           md.update(input); //Actualizamos con nuestra entrada
11           output5=md.digest(); //Obtenemos el digest
12           for(int j=0;j<output5.length;j++){
13               hexString.append(Integer.toHexString(0xFF & output5[j])); //
                     Se obtiene la cadena del hash en valores hexadecimales
14           }
15           firma_comprobacion.add(hexString.toString());
16       }
```

Arrays utilizados:

- *Firma*. Contiene las firmas que se han obtenido previamente desde la base de datos.
- *Direcciones*. Contiene todas las direcciones que existen en la base de datos.
- *Nombre_direcciones*. Contiene todos los nombres de las paradas disponibles, obtenidos previamente desde la base de datos
- *Firma_comprobacion*. Contiene las firmas generadas mediante el código

Se realiza en la clase Ruta porque es en ella donde se detectan las paradas que están en la ruta, para poder así, si la información es correcta, ir a la clase *Beacon* y avisar al usuario mediante la vibración o sonido de que ha llegado a una de las paradas de la ruta.

Capítulo 6

Presupuesto

6.1. Personal

El presupuesto establecido para el desarrollo de la aplicación se muestra en la siguiente tabla, en la cual se detallan todas las tareas y pasos necesarios para el personal. Se toma como sueldo aproximado doce euros la hora, trabajando cuatro horas aproximadamente, cinco días de la semana. Esto equivale a un total de cuarenta y ocho euros por jornada laboral.

Funciones	**Jornadas**	**Total**
Investigación, documentación y análisis de de las herramientas existentes	3	144
Formación en el desarrollo de la aplicación Android	12	576
Familiarizarse con Firebase	5	240
Estudio y análisis de las bases de datos firebase	8	384
Desarrollo de la aplicación con Firebase Database	20	960
Desarrollo de la autenticación de usuario	8	384
Crear una aplicación que funcione con Bluetooth y posteriormente Bluetooth BLE	20	960
Desarrollo de las actividades que forman la aplicación	30	1140
Capa de seguridad en la obtención de datos de ruta	3	144
Total	109	5.102

Cuadro 6.1: Personal.

6.2. Componentes

A continuación se muestra en detalle los costes de los componentes utilizados en este proyecto

Componentes	**Coste**
Licencia para desarrollar Android	25
IDE Android Studio	0
Móvil Android Samsung s8	250
Firebase Realtime Database	25/mes
Total	275 euros + 25 euros/por mes

Cuadro 6.2: Componenetes.

6.3. Coste Total

Personal	5.102
Componentes	275 euros + 25 euros/por mes
Total	5.377 + 25 euros/-mes

Cuadro 6.3: Coste Total.

Capítulo 7

Conclusiones y mejoras futuras

7.1. Conclusiones

El Trabajo de Fin de Grado presentado partió con el objetivo de facilitar al usuario su experiencia en el transporte público y proporcionarle toda la información que necesitara durante el mismo. Es por eso que se ha desarrollado una aplicación con un sistema de localización de dispositivos beacons ubicados en las paradas del transporte público, utilizando los servicios de Bluetooth Low Energy.

El sistema propuesto se basa en una aplicación móvil que, gracias a diversas peticiones a Firebase, procesa las rutas disponibles y le da al usuario la información sobre ese trayecto. El sistema está protegido por medio de SHA3-512, el último miembro de la familia de estándares Secure Hash Algorithm. Además se accede a todos los datos por medio de Firebase, por lo que cualquier cambio en la base de datos se verá plasmado automáticamente en la aplicación.

Las pruebas realizadas hasta el momento determinan que el sistema ha logrado cumplir todos los objetivos que se han propuesto desde un comienzo, las cuales han sido realizadas con personas de diferentes edades, obteniendo unos resultados que han sido muy prometedores.

Finalmente, este trabajo mejora las expectativas deseadas desde un comienzo, y da pie a muchas otras mejoras ya que se podría mejorar la experiencia de los usuarios en el transporte público. Además, es una aplicación muy útil ya que en vez de usar GPS como el resto de aplicación utiliza la tecnología BLE para detectar Beacons.

Por último, cabe mencionar que este proyecto será presentado en TLP Innova 2019.

7.2. Mejoras futuras

Algunos de los trabajos futuros que se plantean para el proyecto desarrollado son:

- **Compartir trayectoria con otros usuarios.** Poder compartir la trayectoria que el usuario está realizando con otros que tengan la aplicación. De esta manera, ellos podrán saber dónde se encuentra y cuánto le queda para llegar a su destino.
- **Poner el tiempo de espera.** Añadir el tiempo de espera para que llegue el transporte que el usuario debe coger.
- **Mandar correo de verificación.** Para aumentar la seguridad de los datos del usuario y de la aplicación, mandar un correo de verificación para comprobar que el email es real, y si es así registrar al usuario.

- **Puntos.** Mediante un control de uso de la aplicación, se podría dar puntos canjeables a los usuarios, que pueden ser cambiados por viajes gratis o descuentos en bonos, entre otras posibilidades.
- **Optimizar la base de datos.** Optimizar la base de datos utilizada en este proyecto ya que hay algunas ramas y claves que se podrían eliminar.
- **Ofrecer varias rutas para llegar al destino deseado.** Mostrarle al usuario varias posibilidades para llegar al destino que desea para que el usuario elija cual es la que más le conviene

Capítulo 8

Conclusions and future improvements

8.1. Conclusions

The Final Degree Project was presented with the objectives of facilitating the use of the company in public transport and provide all the information you need during the same. That's why an application has been developed with a system for locating beacons that are located at public transport stops using Bluetooth LE services.

The system is based on a mobile application that, thanks to the different bases of a fire base, processes the available routes and gives the user the information about that route. The system is protected by SHA3-512, the latest member of the Secure Hash Algorithm family. In addition, you can access all data through Firebase, so any changes to the database will be automatically seen in the application.

The tests carried out so far determine that the system has managed to meet the objectives that we have improved from the beginning, those that have been made with people of different ages, obtaining results that have been very promising.

In general, this work improves the desired expectations from the beginning, and leads to many other improvements as it could improve the user experience in public transport. Vspace 3mm. In addition, it is a very useful application that at the moment of using gps like the rest of the application uses BLE technology to detect beacons. Finally, it can be said that this project will be presented in TLP Inovance 2019.

8.2. Future improvements

Some of the future works that are proposed for the developed project are:

- **Share trajectory with other users.** Be able to share the trajectory that the user is doing with others who have the application. In this way, they will be able to know where they are and how much they have left to reach their destination.
- **Set the waiting time.** Add the waiting time for the transport that the user must reach.
- **Send verification mail.** To increase the security of the user and application data, send a verification email to verify that the email is real, and if so, register the user.
- **Points.** By means of a control of use of the application, you could give points that can be exchanged to users, which can be exchanged for free trips or discounts on bonuses, among other possibilities.

- **Optimize the database.** Optimize the database used in this project since there are some branches and keys that could be deleted.
- **Offer several routes to reach the desired destination.** Show the user several possibilities to reach the destination you want for the user to choose which one is best for him

Apéndice A

Apendice A. Código

Todo el código puede ser encontrado en github [11]

A.1. Manifest

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/AndroidManifest.xml

A.2. MainActivity

Actividad

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/res/layout/activity_main.xml

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/MainActivity.java

A.3. Registro

Actividad

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/res/layout/registro.xml

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/Registro.java

A.4. Iniciar

Actividad

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/res/layout/iniciar.xml

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/Iniciar.java

A.5. Inicio

Actividad

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/res/layout/inicio.xml

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/Inicio.java

A.6. Configuracion

Actividad

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/res/layout/configuracion.xml

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/Configuracion.java

A.7. Rutas

Actividad

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/res/layout/ruta.xml

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/Ruta.java

A.8. Beacon

Actividad

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/res/layout/beacon.xml

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/Beacon.java

A.9. Conexion_2

Actividad

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/res/layout/connexion_2.xml

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/Connexion.java

A.10. Favorito

Actividad

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/res/layout/favorito.xml

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/Favorito.java

A.11. Fin

Actividad

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/res/layout/fin.xml

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/Fin.java

A.12. Check

Clase

https://github.com/AnaBeatrizGilGlez/TFG/blob/master/app/src/main/java/com/example/busguideapplication/Check.java

Bibliografía

[1] *https://bit.ly/2EXwwyg*.

[2] *https://bit.ly/2I8QSGR*.

[3] *https://bit.ly/2ZlYfAp*.

[4] *https://console.firebase.google.com/u/0/*.

[5] *https://developer.android.com/*.

[6] *https://developer.android.com/studio/install?hl=ES*.

[7] *https://en.wikibooks.org/wiki/LaTeX/Floats,_Figures_and_Captions*.

[8] *https://firebase.google.com/docs/auth?hl=es-419*.

[9] *https://firebase.google.com/docs/database/admin/retrieve-data?hl=es-419*.

[10] *https://firebase.google.com/docs/database/rtdb-vs-firestore?hl=es-419*.

[11] *https://github.com/AnaBeatrizGilGlez/TFG*.

[12] *https://moovitapp.com/*.

[13] *https://soundcloud.com/urbanstep*.

[14] *https://www.fing.edu.uy/~canale/latex.pdf*.

[15] *https://www.google.com/maps*.

[16] *https://www.ine.es/daco/daco42/daco4210/tv0518.pdf*.