Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Servicio seguro de rastreo mediante comunicaciones inalámbricas

Secure tracking service using wireless communications

Diego Cruz Rodríguez

La Laguna, 11 de marzo de 2022

D. **Mª Candelaria Hernández Goya**, con D.N.I. 45.441.714Q profesora Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Ricardo Aguasca Colomo**, con D.N.I. 43.644.158W profesor Titular de Universidad de Las Palmas de Gran Canaria adscrito al Instituto Universitario de Sistemas Inteligentes y Aplicaciones Numéricas en Ingeniería, como cotutor

C E R T I F I C A N

Que la presente memoria titulada:

”*Servicio seguro de rastreo mediante comunicaciones inalámbricas* ”

ha sido realizada bajo su dirección por D. **Diego Cruz Rodríguez**, con D.N.I. 51148419F.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 11 de marzo de 2022

Agradecimientos

A mi tutora Dña. María Candelaria Hernámdez Goya (ULL) y a mi cotutor D.Ricardo Aguasca Colomo (ULPGC), que me han aconsejado y guiado durante todas las etapas de este proyecto.

Al equipo técnico de ACUDROPOL por mostrarme la relevancia de este proyector para un campo como puede ser la búsqueda de personas.

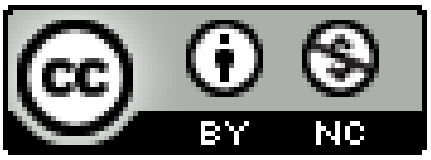
A José Iván Santos Gonzalés y Alexandra Rivero García por su asesoramiento durante el desarrollo del sistema.

Al proyecto ACTIS por prestarme un teléfono de última generación para el desarrollo del proyecto.

A mis compañeros por sus consejos durante el desarrollo.

A mi familia por apoyarme durante el grado.

Licencia



Resumen

Cada vez está más extendido emplear RPAS (Remotelly Piloted Aircraft System) para realizar tareas de vigilancia, control de perímetros, búsqueda y rastreo de personas, debido a los avances tecnológicos, convirtiéndolos en herramientas accesibles y de gran utilidad en estas labores.

Este Trabajo Final de Grado pretende aportar un nuevo sistema para realizar estas funciones. Embarcando un dispositivo móvil en el RPA, se le dotará de la capacidad de rastreo mediante balizas (Beacons) Bluetooth Low Energy y GPS, permitiéndole así realizar un seguimiento de usuarios en situaciones que por otros medios tradicionales de identificación de no sería posible.

Palabras clave: Rastreo, Bluetooth, RPA, Docker, Parse Server, Baliza, Criptografía Ligera

Abstract

The use of RPAS (Remotely Piloted Aircraft System) for surveillance, perimeter control, search and tracking of people is becoming more and more widespread due to technological advances, making them accessible and very useful tools in these tasks.

This Final Degree Project aims to provide a new system to perform these functions. Embedding a mobile device in the RPA and using Bluetooth Low Energy beacons and GPS, it will be provided with the ability to track users in situations that by other traditional means of identification would not be possible using.

Keywords: Tracking, Bluetooth, RPA, Docker, Parse Server, Beacon, Lightweight cryptography

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivación

El objetivo de este proyecto es desarrollar un servicio que permita rastrear dispositivos móviles de baja capacidad a través de un smartphone embarcado en un RPA (Remotely Piloted Aircraft) de manera que la información relativa a la posición sea transferida de manera segura a un servicio web. El principal interés para la realización del sistema presentado reside en dar complemento a otras aplicaciones de rastreo diseñadas para interiores, que requieren de dispositivos específicos en la mayoría de los casos (beacons). Se ha considerado interesante investigar la viabilidad de sistemas como el desarrollado en escenarios tales como búsqueda de personas.

1.2. Antecedentes en aplicaciones de rastreo

A la hora de realizar este proyecto, encontramos aplicaciones con ciertas similitudes. Por los acontecimientos actuales, una de estas referencias sería la aplicación Radar Covid. La aplicación genera claves aleatorias cada 10 - 20 min y se transmiten mediante Bluetooth a los móviles cercanos que tengan la aplicación instalada. Estos dispositivos recogerán los códigos de forma que ambos registran el código del otro dispositivo. Cada móvil almacena los códigos por un periodo de 14 días. Cuando alguien recibe diagnóstico positivo y lo notifica en la app, se solicitan los códigos registrados en los últimos 14 días. Estas claves se descargan diariamente y permiten identificar a la aplicación si ha habido riesgo de contagio o no [1].

Otra referencia interesante es la aplicación de Wikiloc [2], una aplicación para hacer seguimientos de rutas al aire libre mientras se realizan diversas actividades. Esta aplicación permite consultar posteriormente las actividades realizadas junto con algunas estadísticas (velocidad, tiempo, número de paradas, etc.). Dicha información puede ser consultada desde la app o desde su plataforma web, permitiendo añadir información extra como imágenes del recorrido. Estas rutas se pueden compartir en la comunidad de forma que al usar la aplicación te recomiende rutas cercanas a una zona en la que esté el usuario.

1.3. Conceptualización de la propuesta

En este Trabajo Final de Grado se ha diseñado y desarrollado un sistema de seguimiento basado en balizas Bluetooth Low Energy (BLE) para la trasmisión de datos relativos al posicionamiento de dispositivos de forma segura. En nuestro sistema encontraremos dos tipos de usuarios: rastreadores y usuarios. Los usuarios trasmiten datos relacionados con su ubicación de forma segura usando el modo baliza BLE, mientras que los rastreadores recolectan todas las balizas BLE que se encuentren a su alcance. Esta información se transmite al "backend" de la aplicación, encargándose éste de procesarla y almacenarla para su posterior visualización y explotación.

El planteamiento inicial fue utilizar dos Raspberry equipadas con GPS y Bluetooth, simulando smartphones, que emiten su identificación y los datos obtenidos del GPS siguientes: posición, dirección de movimiento y velocidad. Se optó por usar el sistema Android para desarrollar la aplicación y se probaron diferentes implantaciones de este Android sobre Raspberry. En la implantación de Android que mejor funcionaba con Rasberry, se encontraron fallos en la utilización de los módulos Bluetooth y WiFi, haciendo que las comunicaciones fueran inestables y fallaran a menudo. Para emplear GPS en las Rasberrys se tenía que usar un módulo externo a la misma, que el sistema Android no era capaz de utilizar. Para solucionar este problema se plantearon dos posibles soluciones, hacer uso directamente del módulo GPS desde la aplicación o crear un script que actualizara la señal GPS del sistema Android usando el módulo GPS. La primera opción se descartó debido a que limitaba el desarrollo, haciendo que la aplicación para estas Rasberrys funcionara únicamente con este sistema y no en cualquier Android. La segunda opción era muy compleja debido a que requería el desarrollo de un "driver" para adaptar ese módulo externo GPS al Android de la Rasberry. Finalmente, se decidió descartar estas Raspberry e implementar el sistema directamente sobre dispositivos móviles nativos en Android, para no añadir más complejidad al sistema y facilitar su despliegue.

El sistema desarrollado en este trabajo se puede desplegar, como se ha indicado, para distintas finalidades como puede ser la localización de personas desaparecidas en lugares de difícil acceso, vigilancia y control de aforo, etc.

El escenario de aplicación del sistema desarrollado para este Trabajo Final de Grado es el siguiente: Un primer smartphone, haciendo uso de la aplicación desarrollada, establece su ubicación mediante GPS, y la codifica, cifra y trasmite mediante el modo Beacon de BLE.

Un segundo smartphone embarcado en un RPA será el encargado de recoger los datos trasmitidos, mediante el protocolo Beacon BLE a través de la app desarrollada. Posteriormente, será este dispositivo móvil el encargado de remitir a un servidor la información adquirida para que sea representada y evaluada.

Durante el desarrollo se planifica y se realiza una reunión con el equipo técnico de ACUDROPOL en la que se analiza el material disponible para un operativo. Se realiza una demostración de un vuelo de rescate con un dron, para familiarizarse con el operativo humano y material. Además, en dicha reunión, se lleva a cabo una presentación de 5 minutos explicando los objetivos, desarrollo y alcance del TFG, que da pie a una discusión con el objetivo de la integración de los desarrollos realizados en las actuaciones reales de los equipos de rescate (policía, bomberos, protección civil,...).

El resultado que daría el sistema desarrollado, una vez generada la base de datos con la información recogida por el RPA de los diferentes usuarios en tierra, se puede observar en la fig. 1.1. Esta información podría representarse y explotarse con herramientas de IA

para identificar, no solo posiciones individuales con vectores que indican su sentido de movimiento y velocidad, sino dependiendo de su proximidad, dirección y velocidad podrían identificarse zonas ”calientes” o de riesgo. Así mismo se podría analizar el histórico de posiciones de un determinado usuario (resaltado en amarillo) para poder en su caso realizar un análisis de su tracking, por ejemplo, para comprobar que no ha sobrepasado ningún área prohibida. Otra utilidad interesante es poder comprobar en tiempo real posibles aglomeraciones, así como rastrear personas desaparecidas.

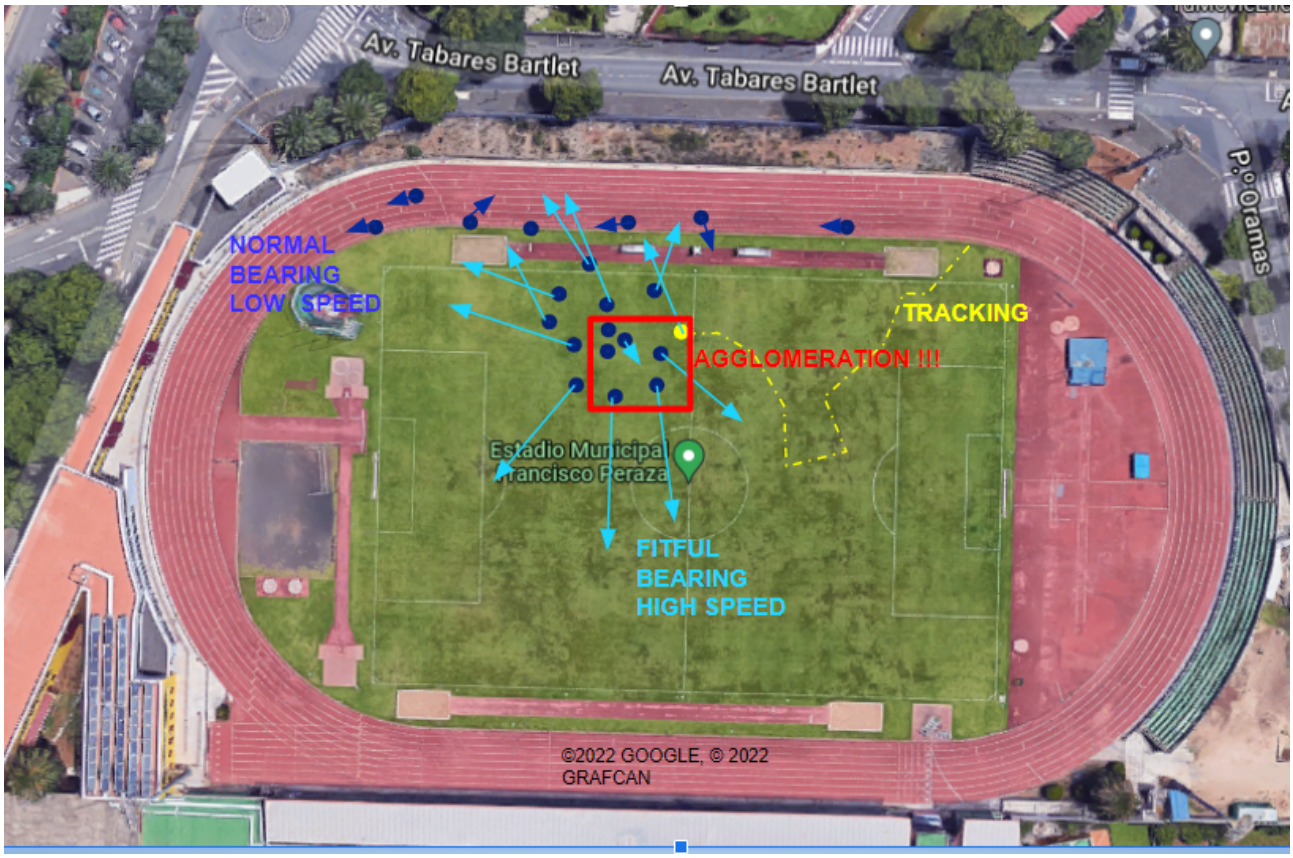


Figura 1.1: Conceptualización representación de la información el la web

1.4. Estructura de la memoria

La presente memoria consta de siete capítulos. En el primero se tratan las motivaciones que condujeron a realizar el proyecto. Un breve visión del estado actual del panorama en el que se enmarca el sistema a desarrollado. Y se establecen los objetivos a conseguir.

El segundo capítulo incluye la descripción del material y tecnologías usadas durante el desarrollo y puesta en marcha del sistema.

En el tercer capítulo muestra y explica de manera general cada uno de los componentes del sistema desarrollado, siendo explicados con mayor detalle en los capítulos siguientes: cuarto, quinto y sexto.

El séptimo capítulo está dedicado a la estimación del presupuesto del proyecto.

Por último, los capítulos octavo y noveno desarrollan las conclusiones y trabajos futuros en español e inglés respectivamente.

Capítulo 2

Tecnologías

Este capítulo recoge la descripción de las tecnologías utilizadas en el Trabajo Final de Grado.

2.1. Dispositivos

Para desarrollar este proyecto se empleó:

- Un smartphone Oppo A73 5G como usuario y rastreador de la aplicación Android
- Un smartphone Redmi Note 7 como usuario y rastreador de la aplicación Android
- Servidor de virtualización para el alojamiento del servicio ”backend” y la web
- Un ordenador para el desarrollo y configuración de los diferentes componentes del sistema.

2.2. Tecnologías del desarrollo

2.2.1. Aplicación Android

Para desarrollar la aplicación se utilizaron las herramientas oficiales ofrecidas por Google para desarrollar aplicaciones para su sistema. Como entorno de trabajo se empleó el IDE Android Studio [3] y la documentación oficial del sistema que encontramos en *developers.android.com*[4].
Como lenguaje de programación se empleó Kotlin, el lenguaje oficial recomendado por Google [5].

2.2.2. Backend de la aplicación

Para el desarrollo del backend de la aplicación encontramos diversas propuestas. Desde el desarrollo del propio backend hasta el uso de plataformas ya especializadas. Debido a la dificultad de realizar una plataforma desde cero con tecnologías sobre las que no se poseía conocimiento y dado también el tiempo limitado del que se disponía, se optó por emplear una plataforma ya existente. Hay dos plataformas con grandes sinergias con Android: Firebase y Parse. La plataforma en la nube de Google (Firebase) [6] está orientada a la creación de aplicaciones y Parse Platform [7], se puede entender como un conjunto de herramientas de código abierto para crear un backend de aplicación.

Existe gran número de recursos de documentación oficial y tutoriales para Firebase, pero se descartó por ser una plataforma ligada a un servicio web de Google que no permite realizar el alojamiento en sistemas externos, sino que requiere el pago por servicios básicos tales como adecuar los recursos del sistema y las interacciones con el mismo.

Durante el diseño e implementación del sistema se persiguió como objetivo que fuera totalmente adaptable de manera que un usuario pueda implantarlo para sus propios fines. Esto llevó a definir la estructura basada en micro-servicios, para facilitar su autogestión. Esto permitiría, a su vez poder elegir la forma de implantación que considere correcta el usuario y adaptarla a los recursos disponibles: servidores dedicados, pequeños equipos informáticos, despliegue en plataformas de contenedores basados en la nube, etc.

Este era el principal atractivo que planteaba la plataforma Parse, la posibilidad de elegir como usar su sistema, subcontratándolo como plataforma en la nube al estilo de Firebase en páginas como Back4App [8] o bien usar el soporte de los contenedores ya creados de la plataforma Parse [9], para crear un micro-servicio que mediante contenedores se pudiera desplegar en cualquier tipo de sistema. Esta última fue la opción finalmente seleccionada.

La plataforma Parse permite configurarla, estructurarla y añadir código que se ejecute en el backend para implantar funcionalidades propias de la aplicación desarrollada [10] y, en caso necesario, conectarla con servidores propios. Esto permite tener un backend completamente adaptado a medida para la aplicación. Estas funcionalidades se desarrollan en Javascript.

2.2.3. Aplicación Web

Para el desarrollo de la aplicación web se ha empleado Node.js, que permite su implantación fácilmente como un micro-servicio más de nuestro sistema, empleando los lenguajes: JavaScript, HTML y CSS.

2.3. Tecnologías de comunicación

2.3.1. Tecnología Bluetooth

Bluetooth es una tecnología de comunicaciones inalámbricas desarrollada por el grupo de trabajo IEE802.15.1 del Institute of Electronic Engineers estadounidense. Sirve para la transmisión de voz y datos punto a punto de forma inalámbrica. Con el objetivo de reemplazar conexiones por cable. A día de hoy está considerada una de las tecnologías facilitadoras del despliegue de soluciones para la Internet de las Cosas.

Está orientado a transferencias en distancias cortas (0.5m 100m) y en el establecimiento de conexiones sencillas de bajo consumo [11], [12], [13].

En concreto, se hace uso del protocolo de balizas (beacon). Empleando Bluetooth de baja energía (BLE) se radia una señal, que puede ser recogida por otro dispositivo que se encuentre en su rango.

En las especificaciones de los protocolos para este modo de comunicación se establecen la estructura y longitudes máximas de las tramas permitidas. Las especificaciones principales son iBeacon, Eddystone y AltBeacon [14]. En este proyecto se ha optado por este último dado que es de código abierto y además, la cantidad de datos disponible para transferir es superior al resto de especificaciones. De hecho, se puede contar con hasta

26 bytes. Otra ventaja consiste en la posibilidad de especificar un identificador asociado a la aplicación a la que están destinados, propiedad que simplifica algunas cuestiones relacionadas con la gestión de los dispositivos en la aplicación de rastreo diseñada.

Usaremos esta tecnología para la transmisión de información entre los dispositivos móviles de cliente y rastreador.

2.3.2. Tecnología Wifi/4G/5G

WiFi es una tecnología de transmisión de datos inalámbrica y está ampliamente estandarizado para dar conexión a internet. WiFi es una marca de WiFi Alliance, organización que promueve dicha tecnología, crear estándares y certificar las tecnologías que los cumplen, [15], [16], [17].

En este proyecto se hará uso de esta tecnología como medio de comunicación entre la aplicación móvil y el backend de la aplicación, pudiendo sustituirse por 4G o tecnologías análogas.

2.4. Tecnologías de geolocalización

El sistema de ubicación más popular a día de hoy sigue siendo GPS, que fue establecido en 1973 [18]. Empleando comunicaciones con satélites y algoritmos de triangulación es capaz de determinar la ubicación a nivel global dando, entre otros parámetros, latitud, longitud, altitud, velocidad y orientación. Contempla errores de +-3m. Encontramos un gran número de aplicaciones que emplean esta tecnología para mostrar en mapas la ubicación de un dispositivo como Google Maps[19]. También encontramos soluciones comerciales en dispositivos dedicados como pueden ser los de la marca Garmin [20], siendo esta una de las más reconocidas en el sector de localizadores GPS portátiles [21].

Esta tecnología la emplearemos en la aplicación Android para la recolección de la ubicación del dispositivo rastreado (clientes).

Capítulo 3

Descripción del sistema

Se incluye en este capítulo la descripción general del sistema desarrollado.

3.1. Esquema general

El diseño del sistema consta de tres componentes principales (Fig. 3.1):

- La aplicación móvil. Figura: (3.1: A)
- El backend de la aplicación. Figura: (3.1: B)
- La aplicación web. Figura: (3.1: C)

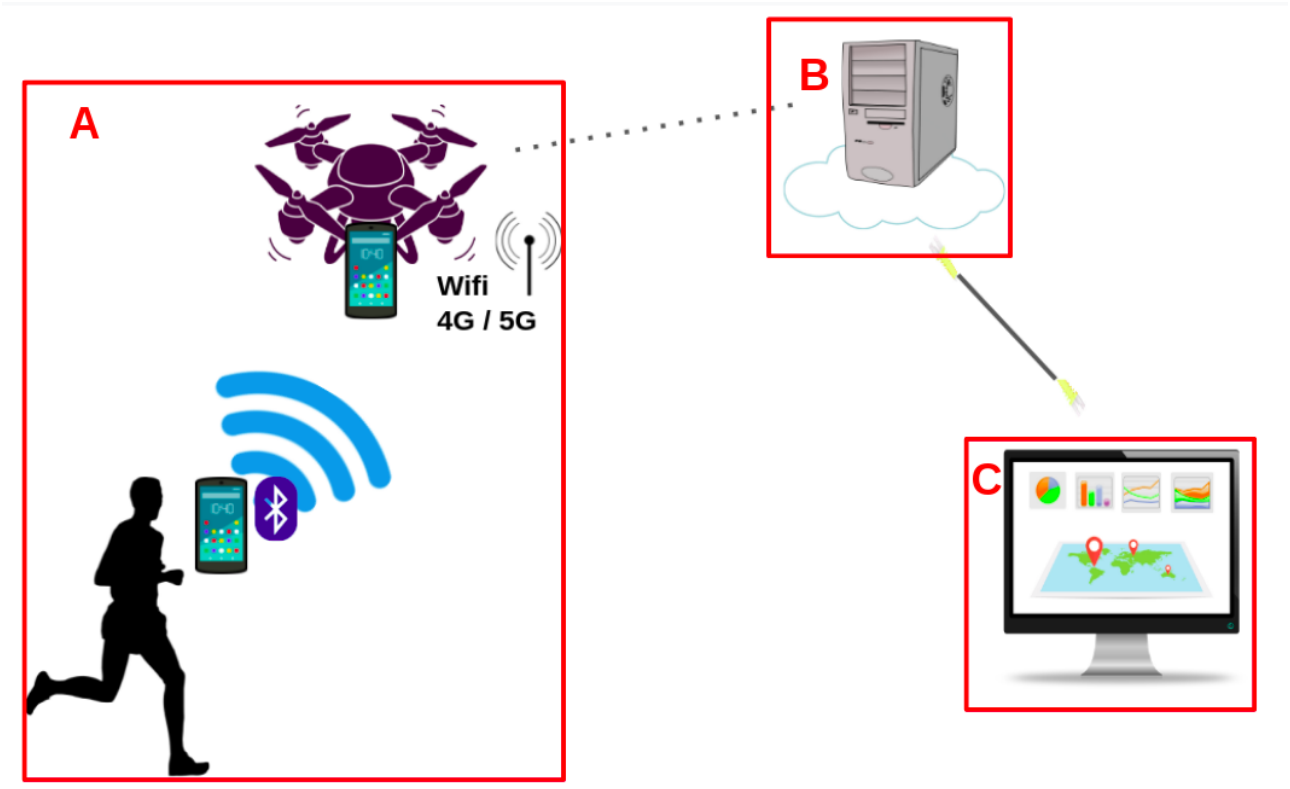


Figura 3.1: Esquema global del sistema.

3.1.1. Seguridad

Para preservar la seguridad del sistema se tiene que garantizar la seguridad de los datos generados y almacenados en cada uno de los componentes, además de las comunicaciones entre las mismas. Para ello se ha atendido principalmente a dos de los pilares fundamentales de la ciberseguridad como son la confidencialidad y la integridad.

Se han integrado diferentes soluciones dependiendo del canal de comunicaciones utilizado y las características de los dispositivos.

Tal y como se aprecia en el esquema de la figura 3.1, el canal de comunicaciones usado entre los clientes y el rastreador es BLE, usando el modo baliza, con el objetivo de evitar la necesidad de emparejar los dispositivos a conectar.

Para este canal de comunicaciones, los servicios de confidencialidad e integridad finalmente se han implementado haciendo uso de primitivas criptográficas pertenecientes a la Criptografía Ligera. Esta parte de la criptografía está especialmente recomendada para dispositivos con bajas capacidades computacionales y de comunicación, tales como redes de sensores y elementos de la Internet de las Cosas.

La decisión de utilizar primitivas de este subconjunto está motivada principalmente por las restricciones definidas en las tramas de comunicación cuando se usa el modo beacon de Bluetooth.

Teniendo en cuenta también las restricciones asociadas al escenario descrito, se ha optado por utilizar cifrado autenticado (Authenticated Encryption, AE) usando la aproximación “Encrypt-then-MAC” puesto que es una de las metodologías más seguras para proporcionar simultáneamente confidencialidad e integridad mediante criptografía simétrica [22].

En el resto de comunicaciones (aplicación móvil - backend y backend - aplicación web) se emplea el protocolo de trasferencia seguro para la capa de transporte (TLS). Esto nos permite tener un grado de seguridad adecuado al nivel de criticidad de los datos transferidos.

3.2. Aplicación móvil

La aplicación se ha desarrollado íntegramente con Android Studio. Se tomó la decisión de que fuera compatible con versiones superiores a la versión 8.0 de Android para poder usar BLE. De este modo se garantiza una compatibilidad con el 82 % de los dispositivos funcionales a día de hoy (dato extraído de Android Studio a fecha 05/03/2022). A partir de la versión 8.0 de Android se cambió la forma de trabajar con los permisos aumentando la dificultad del desarrollo [23], [24].

El código desarrollado para esta aplicación está disponible en el repositorio de GitHub siguiente: `https://github.com/DarkWayC0de/TFG_Localizacion_Android`.

3.2.1. Esquema general

El diseño de la aplicación consta de:

- Servicio en segundo plano para recolectar ubicación. Figura: (3.2: A)
- Servicio en segundo plano para gestionar balizas Bluetooth. Figura: (3.2: B)
- Servicio en segundo plano para trasmitir hacia el backend. Figura: (3.2: C)

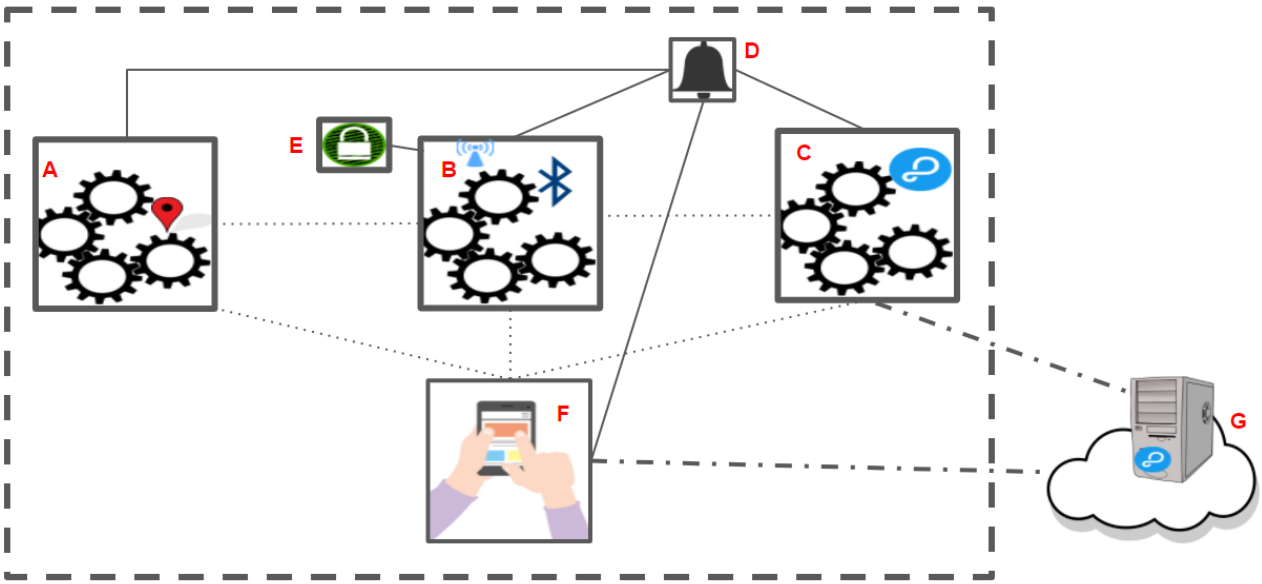


Figura 3.2: Esquema de la aplicación Android

- Notificación informativa. Figura: (3.2: D)
- Librería en C para el cifrado y MAC. Figura: (3.2: E)
- Interfaz de usuario. Figura: (3.2: F)
- Backend. Figura: (3.2: G)

3.2.2. Diseño para trabajo asíncrono

Una característica de la aplicación es la necesidad de ejecutarse en segundo plano manteniéndose activa.

Los servicios que Android proporciona para esta cuestión son completamente asíncronos entre ellos y respecto a la propia aplicación. Los servicios tienen que estar vinculados a una notificación, para que el usuario sea consciente de que está en funcionamiento. Para el correcto funcionamiento de la aplicación se emplean tres servicios en segundo plano, vinculados a una notificación informativa. Cada uno de los servicios se encarga de una única tarea, permitiendo así un mantenimiento y testeo del código simple y eficiente. Para el intercambio de información se emplean variables LiveData [25], que permiten vincular los componentes que requieran el dato de manera que lo reciban cada vez que sea modificado.

Los tres servicios desarrollados cumplen las siguientes funciones:

- Recolectar la ubicación GPS del sistema.
- Gestionar las comunicaciones Bluetooth.
- Establecer las comuniones con el backend.

Este diseño pretende evitar la pérdida de señales y datos, permitiendo que cada hilo de ejecución solo tenga que realizar su acción.

Una cuestión a tener en cuenta a la hora de iniciar y cerrar la aplicación es la necesidad de coordinar el cierre de los diferentes servicios para evitar que queden funcionando en segundo plano.

Cuando se desarrolla aplicaciones con servicios en segundo plano se debe considerar que algunas marcas de dispositivos Android suelen tender a limitar estos servicios, debido a que emplean optimizadores de batería muy agresivos. Para el correcto funcionamiento de la aplicación en esos dispositivos, aunque se recomienda para todos los dispositivos, se tendrá que excluir la aplicación del optimizador de batería del dispositivo [26].

3.2.3. Usuarios y roles

Para la gestión de los usuarios se usa como soporte los recursos aportados por la plataforma Parse incluidos en el backend y el SDK proporcionado para las comunicaciones [27]. La aplicación recoge los datos del usuario y se comunica con la plataforma para verificar que son correctos, permitiendo ingresar o no a la aplicación. En este momento, la aplicación detecta si tendrá que funcionar para el rol cliente o rastreador. En caso de ser modo cliente, descarga del backend las claves que empleará para cifrar y calcular el MAC. Estas claves se guardarán en las preferencias del propio dispositivo [28], permitiendo hacer uso de la aplicación sin conexión a Internet.

3.2.4. Recolección GPS

Para la recolección de la ubicación GPS, se emplea un servicio en segundo plano que solicita al sistema operativo actualizaciones de la localización del sistema con un intervalo de entre 2 - 5 segundos. Esta información se guarda en una variable LiveData[25] para que otros componentes de la aplicación reciban la información. También este servicio es el encargado de modificar los datos de la ubicación que se muestran en la descripción de la notificación del sistema.[29].

3.2.5. Uso de balizas Bluetooth

La gestión de las balizas Bluetooth se realiza desde un servicio en segundo plano con funcionamiento diferente dependiendo de si la aplicación se encuentra en modo cliente o en modo rastreador:

- En modo cliente, al recibir una actualización de ubicación, se comunica con la librería en C para codificar, cifrar y calcular el MAC del mensaje. Una vez realizado el empaquetado de la información generada, comienza a transmitir este mensaje en modo baliza Bluetooth.

- En modo rastreador, la aplicación se pone a la escucha de balizas Bluetooth filtrando las que contengan el identificador definido adhoc para la aplicación. Cuando detecta balizas pertenecientes al sistema, recopila todas las encontradas y las almacena en una variable LiveData [25], junto con la ubicación del propio rastreador.

Para emplear el protocolo de baliza Bluetooth y la comunicación con el controlador Bluetooth de Android, se ha empleado la librería Android Beacon Libray [30].

A la hora de usar las balizas Bluetooth nos encontramos múltiples especificaciones que definen para que se utilizara cada uno de los campos de la correspondiente trama

Bluetooth. [31]. Tras investigar y comparar las especificaciones disponibles se identificó que AltBeacon sería la especificación utilizada.

La trama definida por AltBeacon para un paquete consta de 26 bytes (figura 3.3). Los dos primeros bytes (0-1) se emplean para informar del fabricante del beacon, en este caso se usa el asociado a AltBeacon ”0x0118”. Los dos bytes siguientes (2-3) especifican la estructura que el fabricante define para el resto de la trama. Con estos primeros 4 bytes, cualquier dispositivo que reciba el paquete es capaz de identificar los campos y su longitud.

El resto del paquete está compuesto por tres campos denominados identificadores, un valor de 1 byte que representa la intensidad media de la señal recibida a 1 metro del emisor (RSSI) y otro valor de 1 byte reservado para ser utilizado para implementar características especiales.

La estructura definida para los identificadores en nuestra aplicación es la siguiente:

- Identificador 1 (Bytes 4 a 16): contiene la información de seguimiento del dispositivo cifrada con ChaCha20 en los primeros 10 bytes y en los últimos 6 bytes el MAC de los datos anteriores empleando el algoritmo Chaskey.
- Identificador 2 (Bytes 20-21): Indica que el beacon pertenece a nuestro sistema. Se ha usado el valor “0xffff”
- Identificador 3 (Bytes 22-23): lo empleamos para identificar el dispositivo que envía el mensaje, esto nos permite tener un total de 65546 dispositivos diferentes en el sistema.

El último campo, denominado MsgID, se emplea para transmitir un identificador del paquete, siendo este dato necesario para sincronizar el descifrado.

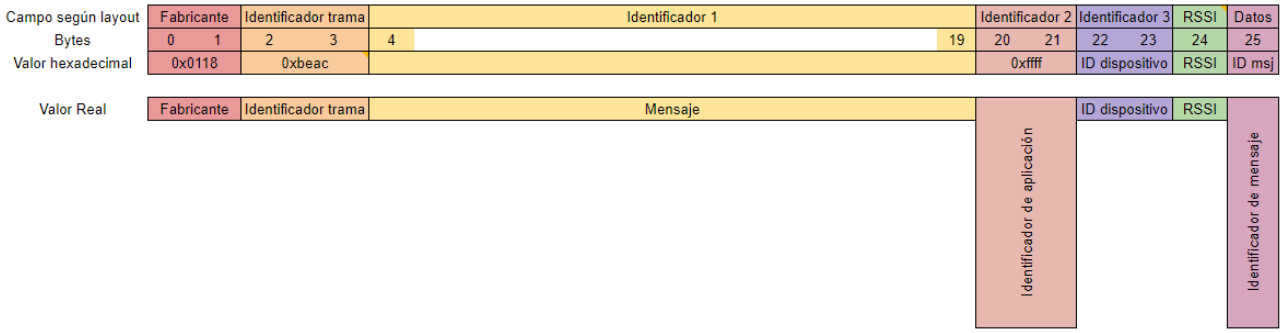


Figura 3.3: Trama AltBeacon adaptada

3.2.6. Librería C

Se desarrolló una librería en C encargada de realizar la codificación de los datos, el cifrado y el cálculo del MAC. Primero realiza la codificación en binario, como se indica en el apartado siguiente. Posteriormente, los cifra empleando la clave de cifrado del usuario, y realiza el cálculo del MAC con la clave secreta del algoritmo correspondiente. Por último, los codifica como una cadena hexadecimal que es devuelta para ser remitida por la baliza Bluetooth.

3.2.7. Codificación datos

Los datos que cada cliente envía mediante el protocolo beacon de Bluetooth son: longitud, latitud, altitud, bearing, velocidad y numero de mensaje enviado. Todos estos parámetros son recolectados por el GPS de los dispositivos clientes. Tal y como se adelantó, el primer identificador de tamaño 16 bytes (4-16) es el que contiene estos datos. Realmente, los 10 primeros bytes contienen el cifrado de dicha información, usando el cifrado en flujo ChaCha20, y los últimos 6 bytes corresponden al MAC de dicha información generado con el algoritmo Chaskey.

La codificación utilizada para empaquetar los datos obtenidos del GPS, teniendo en cuenta el rango de cada parámetro, se realizó de la siguiente forma (figura 3.4):

- Longitud: su rango va de 180,0000 a -180,0000. Para la codificación se emplea 1 bit para el signo y 21 bits para el valor de longitud, representándolo como un número entero y codificándolo en binario. Un ejemplo sería para el valor -23,5642 se codifica como 1 el bit de signo y 235642 en binario natural en 21 bits.
- Latitud: su rango es de 90,000 a -90,000. Para la codificación se emplea 1 bit para el signo y 20 bits para el valor de la latitud, representándolo como número entero y codificándolo en binario.
- Altitud: su rango es de 0 a 9000. Para la codificación en binario se emplean 14 bits.
- Bearing: su rango es de 0.0 a 360.0. Para la codificación se emplean 12 bits, representándolo como un número entero codificado en binario.
- Velocidad: debido a que la intención de este proyecto es realizar el rastreo de dispositivos que están asociados a personas que se desplazan a pie, se determinó como valor para la velocidad máxima 12m/s. Por ello el rango de velocidad contemplado va de 0.0 a 12.0. Para codificar esta variable se emplean 7 bits y representa como un número entero codificado en binario.

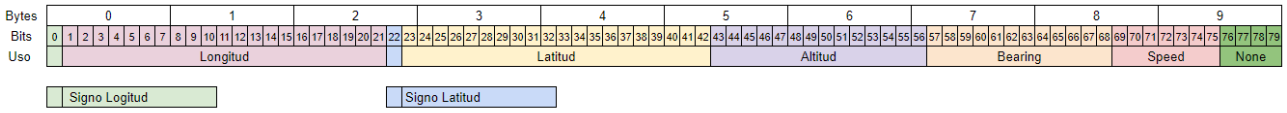


Figura 3.4: Codificación datos.

3.2.8. Integridad y confidencialidad de la información

En la implementación realizada se ha optado por utilizar Chaskey [32] como generador de Códigos de Autenticación de Mensajes (Message Authentication Code, MAC). Este algoritmo está recogido como parte del estándar ISO/IEC 29192-6:2019 (Information technology — Lightweight cryptography — Part 6: Message authentication codes).

Chaskey está definido como un algoritmo MAC basado en permutaciones que utiliza la metodología de diseño Addition-Rotation-XOR (ARX) (figura 3.5). Estas permutaciones se iteran un número determinado de veces. El uso de la aproximación ARX junto con la longitud reducida de los códigos generados lo hacen apropiado para el sistema diseñado en este Trabajo Final de Grado.

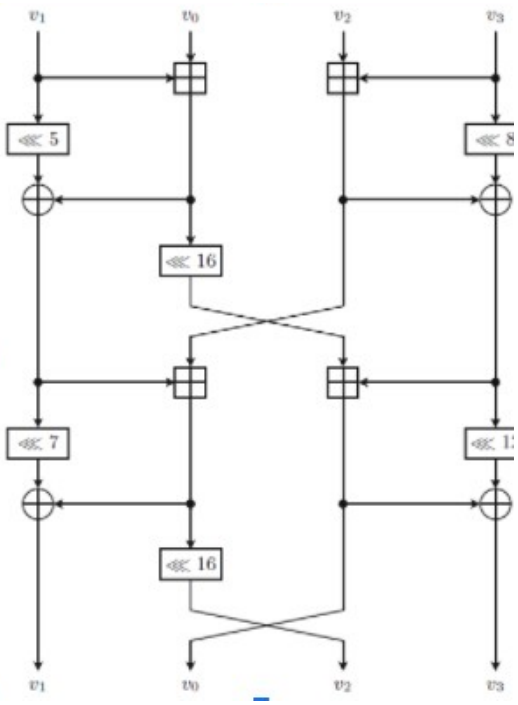


Figura 3.5: Una iteración de la permutación π de Chaskey

El algoritmo recibe como entrada una clave K de 128 bits y un mensaje m. A partir de la clave de entrada se generan dos subclaves (K_1 y k_2). El mensaje m se divide en l bloques de longitud n (128 bits en la implementación utilizada). Sobre cada uno de estos bloques se aplicará la permutación π. En la figura 3.6 se muestra cómo procede el algoritmo dependiendo de si la longitud del mensaje de entrada es múltiplo de la longitud del bloque usado (n). τ representa el MAC generado y se obtiene de truncar los t bits más a la derecha generados al aplicar el esquema. En la implementación utilizada t es 48 bits.

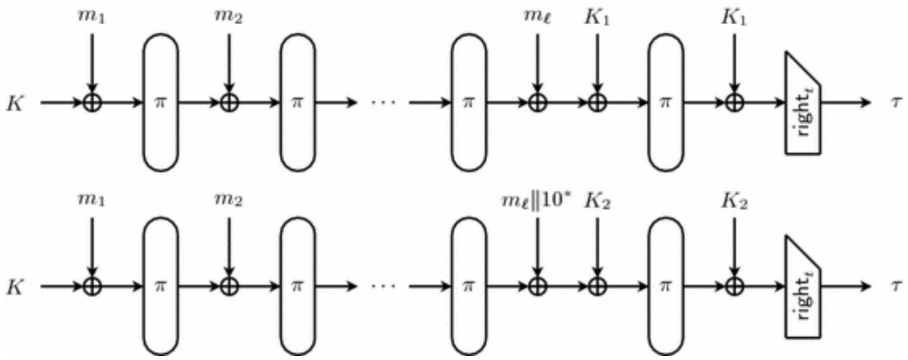


Figura 3.6: Modo de operación de Chaskey

Para que esta construcción se considere segura a día de hoy el autor de la propuesta realiza las siguientes indicaciones:

- La clave K debe ser elegida independientemente y uniformemente al azar de todo el espacio de claves.
- Aunque en el estándar se propone que la permutación contenga 8 iteraciones y la versión recogida en el estándar contiene 12, se recomienda incluir la variante de 16 rondas en las implementaciones que requieren seguridad a largo plazo. De este modo, su seguridad no se verá afectada por posibles avances criptoanalíticos.

Todas estas recomendaciones se han incluido en la implementación utilizada en este trabajo.

Otra primitiva de seguridad implementada en la aplicación móvil es el cifrado en flujo Chacha20 [33] desarrollado en 2008 a partir del cifrado Salsa20 y enviado a eSTREAM por Daniel Bernstein. Se basa también en un generador pseudoaleatorio definido sobre operaciones ARX de 32 bits: Add ($mod 2^{32}$), XOR y rotaciones. La clave es de 256 bits y se usa un contador con la finalidad de sincronizar las secuencias entre dispositivo cliente y rastreador. Suele combinarse con el MAC Poly1305, pero en esta implementación, debido a las restricciones definidas sobre la longitud de la trama de balizas Bluetooth, se ha sustituido por Chaskey.

Además de las restricciones definidas en las tramas emitidas por las balizas BLE, el usar construcciones ARX para ambos servicios de seguridad, hace que la eficiencia sea adecuada para el escenario del proyecto.

3.2.9. Transmisión de los datos al backend

La transmisión de los datos de las balizas Bluetooth de nuestro sistema, hacia el backend se realiza a través de un proceso en segundo plano.

Este proceso solo funciona cuando la aplicación está en modo rastreador. El proceso se subscribe a las modificaciones que sufre la variable de las balizas Bluetooth. Cuando recibe nuevas tramas de los clientes, extrae los datos de cada una de las balizas y los transmite al backend junto con la ubicación del rastreador y otros datos como la distancian aproximada de la baliza correspondiente. Para la trasmisión de los datos se emplea el SDK para aplicaciones Android proporcionado por Parse [27].

Figura 3.7: Aplicación Android

3.3. Aplicación web

El objetivo de la Aplicación Web es ofrecer la visualización de los datos recolectados por la aplicación. Está diseñada como un micro-servicio para un despliegue simple sobre Docker.

La aplicación se desarrolló en Visual Studio Code. Empleando NodeJS, HTML, CSS y Javascript [34], [35], [36].

El código de la aplicación web lo podemos encontrar en el siguiente repositorio de Github `https://github.com/DarkWayC0de/TFG_Localizacion_Aplicacion_Web`.

3.3.1. Comunicación con el backend

Para establecer la comunicación con el backend la aplicación web emplea la librería "Request", haciendo consultas "GET" a la API REST que nos proporciona la plataforma Parse [7], [37].

Filtrado de datos

La aplicación web dispone de campos que permiten filtrar las búsquedas de los datos de los seguimientos que se deseen visualizar. Permitiendo limitar los datos a un periodo de tiempo, un rastreador en concreto o un usuario determinado.

3.3.2. Visualización de datos

Una vez se ha establecido el filtrado de datos que se considere adecuado, se puede emplear un botón para representar los datos de la aplicación en un mapa.

Capítulo 4

Backend de la Aplicación

EL backend de la aplicación es el encargado de almacenar los datos de la aplicación, verificar que estén seguros y dar acceso a los mismos a los usuarios legítimos de la aplicación.

Para la gestión se emplea “Parse Platform” pensado para gestionar el backend de aplicaciones [7].

Se usa el contenedor “Parse Server” para gestionar el proyecto, diseñar la estructura de datos del sistema, las funciones requeridas para resolver problemáticas específicas y además permite usar un WebHoock para dotarlo de más características [9].

4.1. Estructura

El diseño del backend consta de (Fig. 4.1):

- Contenedor Proxy NGiNX.(4.1: A)
- Contenedor Let’s Encrypt NGiNX Proxy Companion.(4.1: B)
- Contenedor Parse Server.(4.1: C)
- Contenedor Mongo DB.(4.1: D)
- Contenedor WebHoock.(4.1: E)

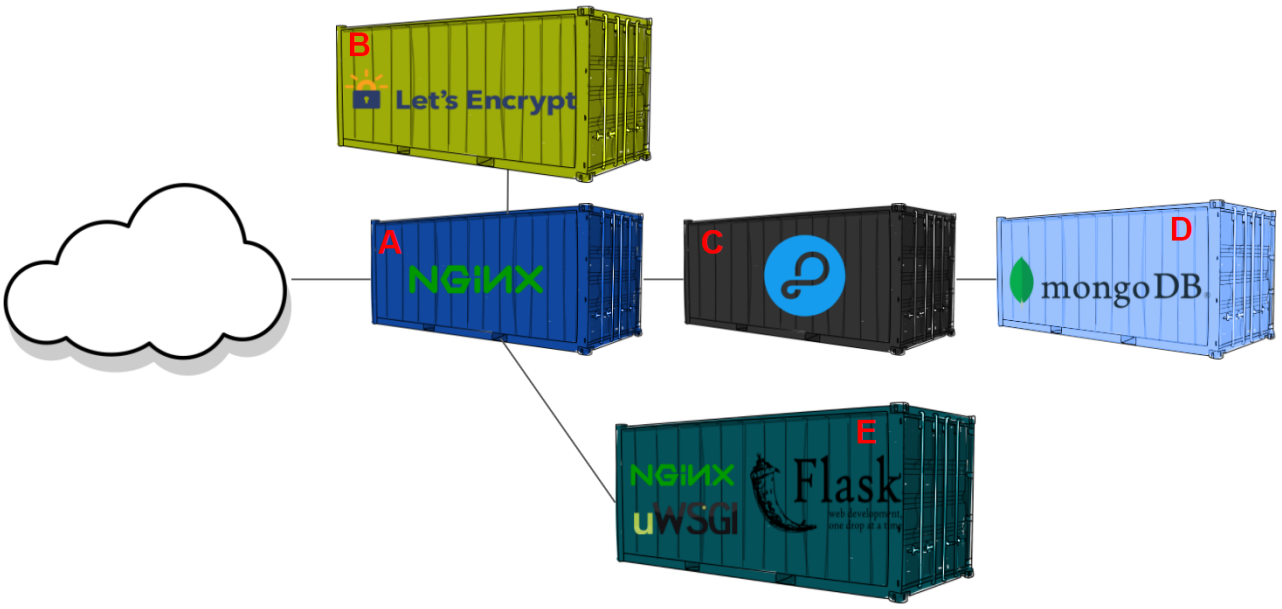


Figura 4.1: Estructura del BackEnd

4.2. Parse Server

Parse Platform es una plataforma que nos permite gestionar fácilmente una aplicación. Para ello tiene herramientas de gestión de usuarios, creación de estructuras personalizadas, y SDKs para poder trabajar con estas características desde cualquier plataforma sobre la que queramos desarrollar la aplicación. También dispone de contenedores de su sistema para que pueda ser implantado con facilidad. Por último, dispone de la posibilidad de añadir funcionalidades con códigos internos desarrollados en Javascript ("Cloud Code") y la posibilidad de conectar a una lógica externa mediante WebHoock [10].

En este desarrollo, el servidor emplea tanto Cloud Code como un WebHoock personalizado, desarrollado en Python.

4.2.1. Estructura de datos

Cuando iniciamos un servidor Parse nos genera una estructura de datos de tres clases.

- Usuario: encargado de gestionar los usuarios de nuestro sistema. Controla datos como correo electrónico, nombre de usuario, contraseña, si el correo ha sido verificado, y otros métodos de autenticación del usuario.
- Sesión: permite controlar las sesiones de cada usuario una vez que inicia sesión.
- Role: permite definir roles de usurario de cara a la interacción con el servidor Parse.

El sistema añade un campo extra en la clase usuario, el cual emplearemos para definir el rol de ese usuario dentro de nuestro sistema, pudiendo tener los valores de "Cliente" o "Rastreador".

Se añade una clase bajo el nombre "Conf", con un campo del tipo String , con una única entrada, que se genera cuando se configura el servidor. Se emplea para que las aplicaciones puedan comprobar si los usuarios introducen correctamente la IP del servidor en la aplicación Android.

Se creó una clase llamada “DatosUsuarios”, empleada para guardar la información necesaria para cifrar los mensajes de las balizas Bluetooth, con un campo que hace referencia al usuario.

Cuatro campos del tipo String guardan el ID del usuario, la clave de 32 bytes en hexadecimal para el cálculo MAC y una clave de 88 bytes en hexadecimal para el cifrado de los datos de cada usuario.

También se ha definido una clase “Ubicacion” empleada para guardar cada una de las ubicaciones de los usuarios de tipo cliente, conteniendo longitud, latitud, altitud, bearing, speed, elapsedRealtime, provider y accuracy. Provider es un parámetro GPS que informa de donde Android a obtenido la ubicación. Accuracy es un parámetro GPS que informa de la precisión de la ubicación.

En último lugar tenemos la clase “Rastreo”, que está encargada de guardar la información relativa al rastreo. En ella se guarda un puntero del usuario rastreador que realizó la adquisición, dos punteros identificar la ubicación del rastreador cuando se realiza la adquisición, y la ubicación del cliente rastreado. Encontramos el instante en la que el rastreador detectó la baliza, junto con la distancia estimada a la misma. También se encuentra el mensaje cifrado transmitido por el usuario, el identificador del usuario y el contador del mensaje para su descifrado.

4.2.2. Cloud Code

Cloud Code es una herramienta de Parse, que ofrece la capacidad de dotar de funcionalidades extras a nuestro servidor Parse. Programando funciones que pueden ser invocadas desde cualquier SDK de Parse, permite tareas periódicas y funciones “trigger” dependientes de la estructura de datos de nuestro sistema.[10].

Mediante Cloud Code, se desarrolló una función para registrar a los nuevos usuarios. Esta, aparte de registrar el usuario, define si ese usuario dentro de nuestro sistema tendrá rol de cliente o Rastreador. Si se tratara de un usuario convencional, se encarga de crear la entrada respectiva a ese usuario en la tabla “DatosUsuario”, asignándole un identificador del sistema y se definen las claves de necesarias para el cifrado y calculo MAC. Este código lo podemos encontrar en el apéndice A.

4.3. Base de datos

La plataforma de Parse puede usar Postgres o MongoDB. En este caso se empleó MongoDB. Durante el desarrollo se fue actualizando la base de datos y se detectaron problemas con a la versión 5 de MongoDB. Por esta razón nuestro sistema emplea la versión 4.4.6 desplegada como un contenedor Docker [38].

4.4. WebHoock

Finalmente, se desarrolla un WebHoock encargado de descifrar los mensajes de los usuarios rastreados por nuestro sistema.

El código del WebHoock lo podemos encontrar en el siguiente repositorio de GitHub `https://github.com/DarkWayC0de/TFG_Localizacion_WebHoock`. Cuando el rastreador registra en el backend los mensajes trasmitidos por el usuario, el backend se los trasmite

al WebHoock, encargándose este de comprobar su integridad, descifrarlos y guardar los datos en el backend. Para el desarrollo del WebHoock se empleo Visual Studio Code.

Se creó bajo la arquitectura de microservicio con el lenguaje Python 3.6 haciendo uso de las herramientas uWSGI NGiNX y Flask sobre una distribución Alpine Linux.[39][40].

Flask es un framework minimalista que nos permite crear aplicaciones web rápidamente con un mínimo número de líneas de código en Python. uWSGI permite paralelizar y escalar la aplicación para poder responder a múltiples peticiones simultáneamente. NGiNX es nuestro servidor web/proxy que nos permite publicar la Web.

Cuando el servidor Parse envía una solicitud de procesamiento, en ella envía los parámetros siguientes:

- objectId: se emplea internamente para conocer cuál es el registro del campo Rastreo perteneciente a dicha solicitud de procesamiento.
- idusuario: identifica que usuario ha cifrado el mensaje.
- Mensaje Cifrado.
- Número identificador de mensaje.

Con estos datos el WebHoock solicita las claves de cifrado y calcula el MAC correspondiente, y con esta información se comunica con la librería C. En la librería C en primer lugar se recalcula el MAC del mensaje y se revisa que concuerda con el que venía de la trasmisión. Por último descifra la información transmitida, traslada los bytes a hexadecimal y los devuelve al proceso Python. Este proceso Python es el encargado de descodificar cada uno de los parámetros trasmitido 3.2.6.

4.5. Securización del sistema

En la seguridad del backend tenemos se han tenido en cuenta dos puntos principales. En primer lugar, la seguridad del servidor Parse y la protección de las estructuras de datos y como segundo punto la seguridad de las comunicaciones.

4.5.1. Servidor Parse

Parse nos proporciona diferentes formas de asegurar la estructura de datos permitiendo decidir quién tiene acceso. Los permisos de nivel de clase permiten determinar qué acceso se le da a cada una de las clases definidas. Para la estructura de los permisos se ha intentado otorgar los mínimos permisos posibles para que cada elemento pueda realizar sus funciones [41].

4.5.2. Seguridad de las comunicaciones

Para asegurar las comunicaciones, se emplea el protocolo de trasferencia seguro para la capa de trasporte (TLS) . Es imprescindible añadir esta capa de seguridad al sistema para su correcto funcionamiento y seguridad.

Para el despliegue se han definido dos contenedores, el primero es un Proxy NGiNX que será el encargado de dar acceso desde el exterior de la red a nuestro servidor Parse. Y en segundo lugar, el contenedor “letsencrypt-nginx-proxy-companion”, encargado

de comunicarse con la autoridad certificadora gratuita Let’s Encrypt, generando el correspondiente certificado para nuestro sistema e instalándolo en el Proxy NGiNX, asegurando así el acceso a la aplicación [42].

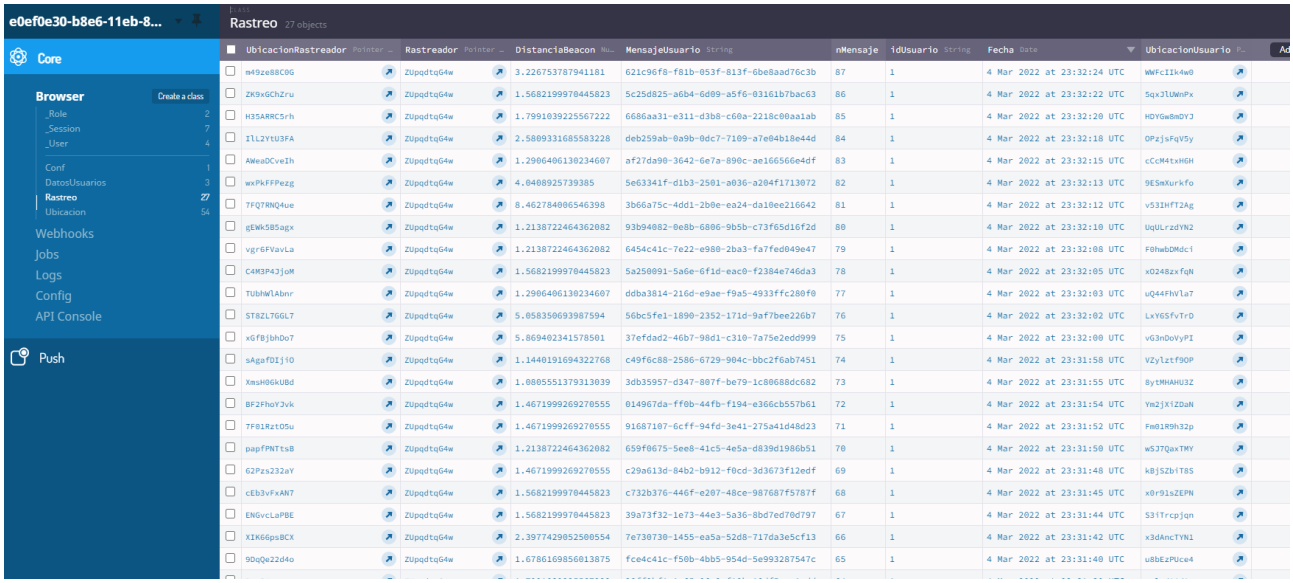


Figura 4.2: Imagen base datos backend

Capítulo 5

Presupuesto

Con este capítulo se intenta establecer un presupuesto global del desarrollo del proyecto.

5.1. Personal

Para los siguientes cálculos se toma el valor de 15€la hora para la jornada de un ingeniero informático.

Tarea	Horas	Coste €
Estudio de mercado	40	600
Replanteamiento de proyecto	60	900
Estudio y selección de herramientas	80	1200
Configuración y personalización de Parse Server	140	2100
Programación Aplicación Web	70	1050
Programación Aplicación Android	260	3900
Aplicación de seguridad en el sistema	50	750
Pruebas, configuración y depuración de errores	160	2400
Redacción de la memoria	50	750
Total	**910**	**13650**

Tabla 5.1: Costes de personal

5.2. Componentes

Componentes	Coste €
Móvil Android <8.0 x 2	380
Servidor	650
Licencia desarrollo Android Studio	0
Licencia desarrollo Visual Studio Code	0
PC para el desarrollo	650
Total	**1680**

Tabla 5.2: Costes componentes

5.3. Costes totales

Tarea	Coste €
Costes: Personal	13650
Costes: Componentes	1680
Total	**15330**

Tabla 5.3: Costes totales

Capítulo 6

Conclusiones y trabajos futuros

Tras la elaboración de este Trabajo Final de Grado, se puede afirmar que es viable realizar un seguimiento mediante ”beacons” Bluetooth, siendo una alternativa interesante para ser implantada en escenarios tales como la búsqueda de personas, el control de aforos en eventos o seguimientos en rutas de montaña.

Se pretende dejar este trabajo disponible para desarrollar nuevas líneas de trabajo como podrían ser:

- Aplicar un procesamiento de datos para obtener mucha más información de los datos recolectados por la aplicación.
- Incorporar al sistema la identificación de señales beacons y Wifi no pertenecientes a la aplicación desarrollada y aportar valores como la distancia aproximada de la señal detectada. De esta forma se puede identificar que en las proximidades de nuestro rastreador hay un dispositivo que no pertenece al sistema.
- Desarrollar la activación remota de la aplicación en modo rastreador para mejorar el muestreo.
- Aplicar un procedimiento de análisis de datos cruzándolo con la información que capte el propio RPA por medio de la cámara de abordo o del GPS que lleve incorporado.

Capítulo 7

Conclusions and future works

After the development of this Final Degree Project, it can be confirmed that it is feasible to carry out a tracking system using Bluetooth ”beacons”, providing an interesting alternative in scenarios such as people search, capacity control in events or tracking in mountain routes.

It is intended to leave this work available to develop new lines of work, such as:

- Apply data processing to obtain much more information from the data collected by the application.
- Incorporating detection of beacons and Wi-Fi signals that do not belong to the developed system and obtaining values such as the approximate distance of the signal.
- Developing remote activation of the application in tracker mode to improve sampling.
- Apply data analysis procedures, including the information captured by the on-board camera and the RPA’s built-in GPS.

Apéndice A

Parse Cloud Code: Registro de nuevos usuarios

```
/**************************************************************************************
*
* main.js
*
**************************************************************************************
*
* Diego Cruz Rodríguez
*
*
* 28/11/2021
*
*
* Código para el registro de nuevos usuarios en el sistema.
*
*
**************************************************************************************/
*
*Parse.initialize("e0ef0e30-b8e6-11eb-8529-0242ac130003", "");
*Parse.serverURL = ' /* URL DEL SERVIDOR PARSE */ '
*Parse.masterKey = " /* MASTERKEY DEL SERVIDOR PARSE */ "
*
*const validationRules = request => {
* if (request.user.role = 'Rastreador'){
* return;
* }
* throw 'No estas autorizado';
*}
*
*function getRandom(min, max) {
*   return Math.floor(Math.random() * (max - min) + min);
*}
*
*function hexRandom(){
* a = getRandom(0,16);
* return a.toString(16);
*}
*
*function hexRandomN(N){
* a = ""
* for (var i = 0; N > i; i++) {
```

```
* a += hexRandom()
* }
* return a;
*}
*
*function calcularMac(mensaje,key){
* return  message.toString();
*}
*
*function descifrar(mensaje,key){
* return  message.toString();
*}
*
*Parse.Cloud.define("Registro", async (request) => {
*    const nuser = new Parse.User();
* nuser.set("username", request.params.username);
* nuser.set("password", request.params.password);
* nuser.set("role", request.params.role);
* nuser.set("email", request.params.email);
* try {
*   await nuser.save(null, { useMasterKey: true });
*
*
*
* if (request.params.role == "Usuario"){
* const DatosUsuarios = Parse.Object.extend("DatosUsuarios");
* const obj = new DatosUsuarios();
* obj.set("user",nuser);
* obj.set("Mackey32",hexRandomN(32));
* obj.set("CifradoKey88",hexRandomN(88));
*
* const query = new Parse.Query("DatosUsuarios");
* query.descending("createdAt");
* try {
* const rquery = await query.first(null,{useMasterKey: true} )
* num = rquery.get("UserID");
* obj.set("UserID", (parseInt(num)+1).toString());
*
* try {
* await obj.save(null, {useMasterKey: true, cascadeSave: false})
* return obj;
* } catch (error) {
* throw "Error3: " + error.code + " " + error.message;
* }
* } catch (error) {
* throw "Error2: " + error.code + " " + error.message;
* }
* } else{
* return nuser;
* }
* } catch (error) {
*   // Show the error message somewhere and let the user try again.
*   throw "Error1: " + error.code + " " + error.message;
* }
*},{
* fields : {
* username : {
* required: true,
```

```
* type: String
* },
* password : {
* required: true,
* type: String
* },
* role : {
* required: true,
* type: String
* },
* email : {
* required: false,
* type: String
*  }
* }
*},validationRules);
*
*********************************************************************************/
```

Bibliografía

[1] I. Ramírez, “Radar covid: qué es y cómo funciona la app oficial de rastreo de contactos de españa,” -08-12T06:15:12+00:00 2020. [Online]. Available: https://www.xataka.com/basics/radar-covid-que-como-funciona-app-oficial-rastreo-contactos-espana

[2] “Wikiloc | rutas del mundo.” [Online]. Available: https://es.wikiloc.com

[3] “Download android studio and sdk tools | android developers.” [Online]. Available: https://developer.android.com/studio

[4] “Desarrolladores de android | android developers.” [Online]. Available: https://developer.android.com

[5] “Kotlin programming language.” [Online]. Available: https://kotlinlang.org/

[6] “Firebase.” [Online]. Available: https://firebase.google.com/?hl=es

[7] “Parse platform.” [Online]. Available: https://parseplatform.org/

[8] “Low-code backend to build modern apps | back4app.” [Online]. Available: https://www.back4app.com/

[9] “Parse server container.” [Online]. Available: https://github.com/parse-community/parse-server#docker-container

[10] “Cloud code guide | parse.” [Online]. Available: https://docs.parseplatform.org/cloudcode/guide/#more-advanced-validation

[11] K. How, “¿qué es bluetooth? toda la información sobre el estándar inalámbrico,” 20/07/20 20/07/20. [Online]. Available: https://www.ionos.es/digitalguide/servidores/know-how/que-es-bluetooth/

[12] “Bluetooth,” -03-28T03:36:13Z 2021. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Bluetooth&oldid=134330280

[13] “Bluetooth technology website | the official website of bluetooth technology.” [Online]. Available: https://www.bluetooth.com/

[14] “Baliza electrónica,” -08-16T16:32:14Z 2021. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Baliza_electr%C3%B3nica&oldid=137708431

[15] “Wifi alliance.” [Online]. Available: https://www.wi-fi.org/

[16] “Qué es el wifi, cómo funciona y qué tipos de cifrado existen.” [Online]. Available: https://www.adslzone.net/reportajes/tecnologia/que-es-wifi-como-funciona/

[17] “Wifi,” -03-29T09:55:53Z 2021. [Online]. Available: https://es.wikipedia.org/w/index.php?title=Wifi&oldid=134362761

[18] “Gps,” -03-27T23:38:41Z 2021. [Online]. Available: https://es.wikipedia.org/w/index.php?title=GPS&oldid=134325820

[19] L. Google, “Google maps - navegación y transporte público.” [Online]. Available: https://play.google.com/store/apps/details?id=com.google.android.apps.maps

[20] “Garmin.” [Online]. Available: https://www.garmin.com/es-ES/

[21] “Gps.gov: El sistema de posicionamiento global.” [Online]. Available: https://www.gps.gov/systems/gps/spanish.php

[22] M. Bellare and C.Ñamprempre, “Authenticated encryption: Relations among notions and analysis of the generic composition paradigm,” *Journal of Cryptology*, vol. 21, no. 4, pp. 469–491, Oct 2008. [Online]. Available: https://doi.org/10.1007/s00145-008-9026-x

[23] “Permisos en android .” [Online]. Available: https://developer.android.com/guide/topics/permissions/overview?hl=es-419

[24] “Inyección de dependencias en android.” [Online]. Available: https://developer.android.com/training/dependency-injection?hl=es-419

[25] “Descripción general de livedata android .” [Online]. Available: https://developer.android.com/topic/libraries/architecture/livedata?hl=es-419

[26] “Don’t kill apps, make them work!” [Online]. Available: https://dontkillmyapp.com/

[27] “Android developers guide.” [Online]. Available: https://docs.parseplatform.org/android/guide/

[28] “Preferencias en android.” [Online]. Available: https://developer.android.com/guide/topics/ui/settings?hl=es-419

[29] “Uso de localización en aplicaciones android .” [Online]. Available: https://developer.android.com/training/location?hl=es-419

[30] “Android beacon library.” [Online]. Available: https://altbeacon.github.io/android-beacon-library/

[31] “Beacon layout.” [Online]. Available: https://altbeacon.github.io/android-beacon-library/javadoc/org/altbeacon/beacon/BeaconParser.html#setBeaconLayout(String)

[32] N. Mouha, B. Mennink, A. Van Herrewege, D. Watanabe, B. Preneel, and I. Verbauwhede, “Chaskey: An efficient mac algorithm for 32-bit microcontrollers,” in *Selected Areas in Cryptography – SAC 2014*, A. Joux and A. Youssef, Eds. Cham: Springer International Publishing, 2014, pp. 306–323.

[33] Y.Ñir and A. Langley, “ChaCha20 and Poly1305 for IETF Protocols,” RFC 7539, May 2015. [Online]. Available: https://www.rfc-editor.org/info/rfc7539

[34] “Visual studio code.” [Online]. Available: https://code.visualstudio.com/

[35] “Docker.” [Online]. Available: https://www.docker.com/

[36] “Nodejs.” [Online]. Available: https://nodejs.dev/

[37] “Parse platform rest api guide.” [Online]. Available: https://docs.parseplatform.org/rest/guide/

[38] “Parse platform guide database.” [Online]. Available: https://docs.parseplatform.org/parse-server/guide/#database

[39] “Configuring uwsgi for production deployment.” [Online]. Available: ConfiguringuWSGIforProductionDeployment

[40] “How to do rapid prototyping with flask, uwsgi, nginx, and docker on openshift.” [Online]. Available: https://towardsdatascience.com/how-to-do-rapid-prototyping-with-flask-uwsgi-nginx-and-docker-on-openshift-f0ef144033cb

[41] “Class level permissions parse platform.” [Online]. Available: https://docs.parseplatform.org/parse-server/guide/#class-level-permissions

[42] “Nginx of let’s encrypt.” [Online]. Available: https://github.com/nginx-proxy/acme-companion

Glosario

BLE Bluetooth Low Energy (Bluetooth LE, coloquialmente BLE) es una tecnología de red de área personal inalámbrica, diseñada y comercializada por Bluetooth SIG destinada a aplicaciones en el cuidado de la salud, fitness y beacons, seguridad y las industrias de entretenimiento en el hogar . . 2, 6

GPS GPS (Global Positioning System) Sistema de Posicionamiento Global, originalmente Navstar GPS. . 2

MAC MAC (Message Authentication Code), código de autenticación de mensaje. . 15

RPA RPA (Remotely Piloted Aircraft) para hacer referencia a las aeronaves tripuladas por control remoto . 2

SDK SDK (Software Development Kit), conjunto de herramientas que ayudan a desarrollar aplicaciones. . 15

TLS TLS, protocolo de transferencia seguro para la capa de transporte. . 9

WebHoock WebHoock, en desarrollo web, es un método de alteración del funcionamiento de una página o aplicación web con callbacks personalizados. . 17

WiFi WiFi es una tecnología que permite la interconexión inalámbrica de dispositivos electrónicos. . 2