Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Sistema móvil para la gestión de grupos

Mobile system for group management

Juan Pablo Claros Romero

La Laguna, 10 de junio de 2019

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

D. **José Iván Santos González**, con N.I.F. 78.637.989-T investigador FPI del Gobierno de Canarias 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:

"*Sistema móvil para la gestión de grupo*"

ha sido realizada bajo su dirección por D. **Juan Pablo Claros Romero**, con N.I.F. 42.279.940-Y.

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

Especialmente a mi tutora Dña. María Candelaria Hernández Goya y a mi cotutor D. José Iván Santos González por sus consejos y continua ayuda para resolver la gran cantidad problemas que me han surgido durante el desarrollo del trabajo, y por su motivación e interés para llevar a cabo todo este trabajo juntos desde el primer día que nos reunimos.

A mi familia y amigos por su continuo apoyo y ánimo que me han brindado desde que empecé a trabajar en el proyecto.

Licencia

Resumen

En los grandes eventos donde se producen acumulaciones masivas de personas (como podría ser la Bajada de la Virgen o Los Indianos en la isla de La Palma), hay momentos en los que podemos perder de vista a nuestros amigos o familiares y tener dificultades para localizarlos. Muchas veces recurrir a las llamadas telefónicas no es la mejor solución por la gran cantidad de ruido que se produce en dichos eventos.

Hoy en día, el uso de los teléfonos móviles o smartphones está creciendo tanto que las personas lo usan para innumerables cuestiones, como medir distancias, localizar sitios o personas, realizar un seguimiento de rutinas para algún deporte, comunicarnos mediante mensajes, etc.

Este TFG propone una aplicación para la localización de personas a través de su ubicación y, además, facilita la comunicación entre ellas mediante un servicio de mensajería, de modo que no es necesario recurrir a las llamadas. El escenario planteado se centra en obtener la localización de aquellos amigos/as o familiares que están agregados en la aplicación en eventos con gran multitud de personas.
La solución desarrollada está basada en la gestión de grupos implementada en la aplicación, permitiendo que la comunicación y la localización no sea global, sino restringida a un conjunto específico de personas.

Para llevar a cabo esto, se va a hacer uso de las siguientes tecnologías de comunicación: Bluetooth Low Energy, Wifi Direct, LTE Direct y se utilizará además el "Sistema de Posicionamiento Global (GPS)". Para asegurar la confidencialidad de los mensajes intercambiados en la aplicación se usará el cifrado AES.

Palabras clave: eventos, smartphones, localización, gestión de grupos, BLE, WiFi Direct, LTE Direct, GPS, confidencialidad.

Abstract

In large events where there are massive accumulations of people (such as the Bajada de la Virgen or Los Indianos on the island of La Palma), there are times when we may lose sight of our friends or family and have difficulty locating them. Many times using telephone calls is not the best solution because of the large amount of noise that occurs in such events.

Nowadays, the use of mobile phones or smartphones is growing so much that people use them for countless issues, such as measuring distances, locating sites or people, tracking routines for a sport, communicating through messages, etc..

This TFG proposes an application for locating people through their location and also facilitates communication between them through a messaging service, so there is no need to use calls. The proposed scenario focuses on obtaining the location of those friends or relatives who are added in the application in events with large crowds of people. The solution developed is based on group management implemented in the application, allowing communication and location is not global, but restricted to a specific group of people.

To do this, the following communication technologies will be used: Bluetooth Low Energy, Wifi Direct, LTE Direct and the "Global Positioning System (GPS)"will also be used. In order to ensure the confidentiality of the messages exchanged in the application, AES encryption will be used.

Keywords: events, smartphones, location, group management, BLE, WiFi Direct, LTE Direct, GPS, confidenciality.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivación

Desde su creación hasta lo que son actualmente, los teléfono móviles han pasado de ser usados únicamente para realizar llamadas y poder comunicar a unas personas con otras que estén lejos, a convertirse en smartphones y de esta manera cubrir gran cantidad de las necesidades de las mismas. Ya sea llamar, como lo hacían los primeros teléfonos, o usar aplicaciones que te permitan medir distancias, realizar videollamadas, comunicar a las personas mediante mensajes de texto, tener las redes sociales en tu mano, sacar fotos mediante una aplicación y convertirlo en un fichero pdf, usar servicios de geolocalización ... y una gran cantidad de cosas más que se amplían gracias a la posible conexión a Internet.
Podríamos decir que son como unos pequeños ordenadores que nos permiten realizar multitud de funcionalidades, sustituyendo a otros dispositivos como podría ser una cámara fotográfica, reproductores de música, navegadores GPS, consolas de videojuegos, gestores de mensajería instantánea y muchos otros más.

Si nos fijamos en esta evolución, podemos afirmar que los teléfonos móviles o smartphones tienen una gran importancia en la actualidad. Esta importancia se debe, no solo por las funcionalidades que nombramos anteriormente, sino porque cuentan con diferentes tecnologías que le permiten desempeñar todas esas tareas. Algunas de las tecnologías de comunicación con las que trabajan son BLE (Bluetooth Low Energy), WiFi Direct, LTE Direct. Dichas tecnologías permiten comunicación inalámbrica, permitiendo el intercambio de datos entre diferentes dispositivos de una forma rápida y eficaz. Uno de los parámetros que diferencian estas tecnologías es la distancia máxima a la que llegan (siendo hasta 60 metros de distancia para BLE, 200 metros en el caso de WiFi Direct y 500 metros para LTE). Además de estas tecnologías de comunicación, el GPS (Sistema de Posicionamiento Global) proporciona servicios de geolocalización asociando estos datos a una representación en un mapa.

Actualmente hay grandes eventos en los que se producen grandes aglomeraciones de personas (por ejemplo Los Indianos en la Isla de La Palma) y a lo que se suele recurrir cuando las personas pierden de vista a sus amigos o familiares es a llamarles para localizarles, pero tenemos dos inconvenientes:

- El primer inconveniente es el **excesivo ruido** debido a la gran multitud de personas que se forma.
- El otro inconveniente que se nos presenta es que puedan colapsar las **infraestructuras de red** ya que puede que no seamos los únicos que recurramos a llamar en ese momento.

Es por todo esto que en este proyecto se va a hacer uso de estas tecnologías nombradas anteriormente, de manera que la aplicación que se desarrolla proporcione una solución a este problema que nos encontramos, permitiendo la localización de las personas mediante el uso de un mapa y, además, posibilitando el que se puedan comunicar mediante un chat, en lugar de realizar llamadas. Todo esto a través de la gestión de grupos definidos por el usuario.

1.2. Estado del arte

El principal objetivo que hay que conseguir es agregar a las personas para así, una vez agregadas, poder llevar a cabo la conexión entre ellas y tener la posibilidad de localizarlas haciendo uso de las tecnologías ya nombradas. Hay que tener en cuenta que existen ciertas restricciones asociadas a dichas tecnologías, como podría ser la distancia de alcance, que podrían limitar su aplicación. A pesar de esto, las distancias proporcionadas son adecuadas para llevar a cabo esta aplicación (distancias de hasta 500 metros). A continuación se describen con un poco más de detalle las tecnologías nombradas anteriormente:

- **Bluetooth Low Energy (BLE):** Comparado con el Bluetooth clásico, BLE [4] está diseñado para proporcionar un consumo de energía y un costo considerablemente reducidos, manteniendo un rango de alcance de comunicación similar **(entre 1 y 60 metros)**. Permite la comunicación entre dispositivos de pila de botón y dispositivos Bluetooth, que opera en 2.4 GHz (una de las bandas ISM[1]), con una tasa de transferencia de 1 Mbps en la capa física. Está basado en un microchip de bajo coste con opciones más amplias para su empleo en la industria; Tiene soporte para seguridad, ya que emplea el sistema de cifrado AES y esquemas de seguridad configurables.

- **Wifi Direct:** es un protocolo que permite a los dispositivos intercambiar archivos o, por ejemplo, enviar a imprimir un archivo sin necesidad de conexiones cableadas. Wi-Fi Direct [2] se puede ver como un Bluetooth actualizado y más rápido, ya que hace uso de la red Wifi. Es un tipo de conexión limitada en espacio (no es para enviar fotos a nuestros familiares que viven en otra ciudad) y pensada para compartir cosas sencillas. Es útil cuando quieres recibir o enviar alguna información a alguien pero no quieres dar datos personales. Es muy seguro, ya que está protegido con WPA2 [21] (AES de 256 bits), mientras que el Bluetooth 4.0 cuenta con cifrado AES 128 con bits. Tiene un alcance máximo de **200 metros** [5].

- **LTE Direct:** LTE Direct [1] es una modificación de las actuales conexiones LTE que permite conectar directamente distintos dispositivos de forma similar a como lo hacen las conexiones Wi-Fi Direct, permitiendo a los usuarios controlar los dispositivos inteligentes de manera sencilla. LTE Direct también puede ser utilizado para mostrar información a la gente según pasea por la calle. Gracias a una serie de sensores es posible que se transmita información directamente desde dichos sensores al smartphone con promociones, anuncio de conciertos o simplemente la temperatura real al pasar cerca de un termómetro. Esta tecnología consumirá menos energía que el Wi-Fi Direct ya que su chip tiene un gasto de batería similar al del Bluetooth de bajo consumo. LTE Direct aún está en desarrollo y podría tardar algún tiempo en llegar a nosotros. También es posible que en España llegue con otro nombre igual que el LTE ha llegado como 4G. Tiene un alcance de hasta **500 metros**.

[1] ISM es el acrónimo del nombre que se le da a las bandas de frecuencias para uso sin licencia reservadas internacionalmente para uso no comercial de radiofrecuencia electromagnética en áreas industrial, científica y médica. Estas bandas son utilizadas por los teléfonos inalámbricos, los microondas, o los dispositivos Bluetooth, WiFi en redes LAN, entre otras tecnologías.

En cuanto al uso de los teléfonos móviles para permitir la comunicación entre personas haciendo uso de alguna de estas tecnologías, podemos poner como ejemplo el trabajo publicado en [23]. Este trabajo presenta un sistema que se basa en el uso de salas de chat con la posibilidad de transmisión real, definiendo dos posibilidades de chat: uno público con acceso directo para todos los usuarios registrados y autenticados, o uno privado que puede ser usado por aquellos grupos de personas que estén autorizados a entrar.

También podemos ver aplicaciones de localización de personas como "Localizador Familiar y Móvil" [24] de Life360. Esta aplicación permite localizar a tu familia de una manera precisa, haciendo uso de Círculos Life360 (grupos para amigos, familia, compañeros de clase, etc.) y los Lugares Life360. Esta aplicación te permite:

- Ver la ubicación de los miembros de los Círculos en un mapa.
- Elegir cuando compartir tu ubicación con cada Círculo.
- Chatear de forma individual o con todos los integrantes de un Círculo.
- Recibir alertas de cuando un miembro de un Círculo llega a un lugar.
- Rastrear un teléfono perdido o robado.

En cuanto al proyecto propuesto, el objetivo es implementar una aplicación móvil que permita localizar a personas que pertenezcan a uno de los grupos en los que se encuentra el usuario. Lo que se busca es una aplicación que, a diferencia de la nombrada anteriormente, esté solo enfocado en distancias que no superen los 500 metros, ya que está orientado a localizar a personas en eventos gran multitud, con las herramientas necesarias para completar esa tarea (ubicación en el mapa, chat para comunicarse y notificaciones para de cercanía para indicar cuando el usuario se acerque a un amigo o familiar e informar de a que distancia está). Aunque la aplicación anterior pueda ubicar a las personas en el mapa y pueda comunicarlas con chat, no interesa conocer ubicaciones a grandes distancias.

Todo esto estará desarrollado en el entorno Android Studio, usando como lenguaje de programación Java para llevar a cabo la implementación, y XML para la parte gráfica de la aplicación.

1.3. Objetivos y competencias del proyecto

Como ya se ha comentado con anterioridad, el objetivo de este proyecto es el diseño e implementación de una aplicación para dispositivos **Android** que permita localizar a las personas que pertenezcan a un grupo X y que esté a una distancia máxima de 500 metros del usuario haciendo uso de las tecnologías **BLE, WiFi Direct, LTE o GPS** para ubicarlos en el mapa, apoyándose en la base de datos para guardar todos estos datos. Teniendo en cuenta esta idea, se describen a continuación algunas de las tareas realizadas:

- **Registro y Login:** Para poder usar la aplicación, los usuarios deberán iniciar sesión con su email y contraseña. Si no tienen una cuenta, podrán registrarse indicando su nombre, número de teléfono, email y la contraseña que elijan.
- **Creación de grupos de amigos:** Se pueden crear grupos para añadir a aquellos amigos que tengas agregados en la aplicación, y poder diferenciarlos según el grupo que se cree (grupo amigos tenerife, grupo familia, grupo fiestas, etc).

- **Detección basada en posicionamiento GPS (detectar amigos en rango de X metros):** Se pueden localizar a los amigos mediante el uso del GPS del teléfono móvil cuando están a X metros.
- **Detección basada en rango de tecnología Wireless (Bluetooth, Wi-Fi):** Se pueden localizar a los amigos mediante el uso del Bluetooth, WiFi o LTE, a parte del GPS del teléfono móvil.
- **Alarma mediante audio y vibración en el móvil mostrando aviso de amigo cercano junto con sus datos:** Cuando un amigo está a X metros cerca, saldrá una notificación en el teléfono móvil indicando los datos de la persona (Nombre, Apellidos, distancia, etc).
- **Posibilidad de solicitud de unión de personas a grupos:** Se pueden enviar solicitudes de unión a grupo para agregar a las personas a los grupos que se tengan creados.
- **Posibilidad de comunicación mediante mensajes:** Los usuarios podrán comunicarse, mediante mensajes, con las personas que tengan agregadas en la aplicación.
- **Medida de protección de la privacidad y confidencialidad:** Los datos de los usuarios que usen la aplicación estarán protegidos para que no sean distribuidos sin el consentimiento del usuario. Ningún usuario podrá ver los datos de otro si no son amigos.

1.4. Planificación del proyecto

En esta parte se describe la planificación que se ha llevado a cabo para la creación del proyecto. Se han definido diferentes fases para cubrir los diferentes aspectos, desde su planteamiento hasta su finalización. Estas fases son las siguientes:

- **Análisis de tecnologías y antecedentes:** Se ha realizado un análisis de las diferentes tecnologías nombradas con anterioridad, además de investigar antecedentes actuales que hagan referencia a la aplicación a desarrollar.
- **Formación en el uso de Android Studio y el lenguaje de programación Java:** Se han implementado las diferentes tareas que la aplicación realiza pero en un formato sencillo, sin control de autenticación ni localización precisa del usuario, simplemente para que a medida que se fue implementando la aplicación, se pudiera ir aprendiendo como trabaja este entorno.
- **Generación de grupos:** El usuario puede crear grupos. Estos grupos se guardarán dentro del nodo del usuario.
- **Detección basada en posicionamiento GPS:** El usuario puede visualizar en el mapa haciendo uso del GPS del dispositivo android, guardando la posición en la base de datos.
- **Detección basada en rango de tecnología Wireless (Bluetooth, WiFi o LTE):** Al igual que el GPS, el usuario puede visualizar en el mapa haciendo uso de estas tecnologías.
- **Notificaciones:** Se han implementado las diferentes notificaciones para cada una de las distintas acciones que se realizan en la aplicación (añadir un usuario a un grupo, notificar si un usuario está a X metros, notificar cuando un usuario manda un mensaje por un grupo y cuando un usuario acepta o rechaza la invitación a un grupo). Para llevar a cabo este desarrollo, se han utilizado las Cloud Functions (que posteriormente hablaremos mas en detalle), programadas en lenguaje JavaScript.

- **Gestión de incorporaciones a grupos:** Se añade el usuario al grupo cuando éste acepta la invitación para unirse al mismo. Cuando se añade, se incorpora al nodo de dicho usuario y además se incorpora ese usuario dentro del nodo del grupo en la base de datos.
- **Sistema de comunicación entre miembros del grupo:** Se proporciona a los usuarios un chat de grupos para que puedan comunicarse entre ellos. Este chat será un nodo a parte que contiene toda la información del mensaje (quien envía el mensaje, la fecha y hora de envío, el mensaje, el grupo en el que se envía y el nombre del usuario que lo remite).
- **Medidas de protección de la privacidad, la confidencialidad y la autenticación:** Estas medidas de autenticación ya se están llevando a cabo gracias a que Firebase proporciona la seguridad. Además de esto, se añadirán reglas a la base de datos de Firebase para controlar que los usuarios lean y escriban en ella sólo si están autenticados, en que nodos puede escribir el usuario que es autenticado, etc.

1.5. Estructura de la memoria

La memoria se ha estructurado de la siguiente manera:

- **Capítulo 2.** Se habla de las tecnologías que se han usado para el desarrollo de la aplicación. En este caso se habla de Firebase y de los seleccionados de entre todos los que proporcionas, además del servicio que se ha cogido de las API de Google.
- **Capítulo 3.** Se explicará en profundidad como se ha desarrollado e implementado la aplicación. Se hablará de cómo está desarrollado cada una de las funcionalidades de la app y se enseñará el aspecto de la misma en cada una de ellas.
- **Capítulo 4.** En este capítulo se hablará de la seguridad que se ha implementado en la aplicación, como el cifrado de los mensajes y las reglas Firebase aplicadas.
- **Capítulo 5 y 6.** En estos capítulos se encuentran las conclusiones de todo el trabajo realizado en este proyecto y la visión de futuro que se tiene del mismo.
- **Capítulo 7.** En este capítulo se hará un análisis sobre el coste total del proyecto.

Capítulo 2

Tecnologías

2.1. Firebase

Para llevar a cabo el desarrollo de esta aplicación, se ha usado una plataforma móvil comprada por Google en 2014 y mejorada por el equipo Divshot[1], llamada **Firebase** [6], que permite desarrollar apps de una forma más rápida y fácil de entender. Esta plataforma está desplegada en la nube y está disponible para diferentes entornos como **IOS, Android y web**. Contiene diversas funciones para que cualquier desarrollador pueda combinar y adaptar la plataforma a medida de las necesidades que tenga.

En este capítulo se describirán algunas funcionalidades proporcionadas por **Firebase**, que se utilizan en este proyecto. Estas son:

- Realtime Database.
- Autenticación.
- Cloud Functions.

2.1.1. Realtime Database

Firebase Realtime Database [19] es una base de datos proporcionada por Google que está alojada en la nube. Los datos se almacenan en formato JSON y se sincronizan en tiempo real con cada cliente conectado. Cuando se compilan apps multiplataforma con sus SDKs[2] de IOS, Android y Javascript, todos los clientes comparten una instancia de Realtime Database y reciben actualizaciones automáticamente con los datos más recientes.

A continuación se pueden observar las principales funciones que esta base de datos proporciona a los desarrolladores:

- **Tiempo real:** En lugar de solicitudes HTTP típicas, Firebase Realtime Database usa la sincronización de datos (cada vez que cambian los datos, los dispositivos conectados reciben esa actualización en milisegundos). Proporciona experiencias colaborativas y envolventes sin pensar en el código de red.
- **Sin conexión:** Las apps de Firebase continúan respondiendo, incluso sin conexión, dado que el SDK de Firebase Realtime Database hace que los datos persistan en el disco. Cuando se restablece la conexión, el dispositivo cliente recibe los cambios que faltaban y los sincroniza con el estado actual del servidor.

[1] Alojamiento web para desarrolladores y diseñadores

[2] **SDK** es el acrónimo de "Software Development Kit" (Kit de desarrollo de software). El SDK reúne un grupo de herramientas que permiten la programación de aplicaciones móviles.

- **Acceso desde dispositivos cliente:** Se puede acceder a Firebase Realtime Database directamente desde un dispositivo móvil o un navegador web; no se necesita un servidor de aplicaciones. La seguridad y la validación de datos están disponibles a través de las reglas de seguridad de Firebase Realtime Database: reglas basadas en expresiones que se ejecutan cuando se leen o se escriben datos.
- **Escalable con varias bases de datos:** Con Firebase Realtime Database y el plan de precios Blaze, se pueden satisfacer las necesidades de datos de la app a gran escala: se puede dividir la información en diversas instancias de bases de datos dentro del mismo proyecto de Firebase.

Realtime Database es una base de datos **NoSQL** y, como tal, tiene diferentes optimizaciones y funcionalidades en comparación con una base de datos relacional. La API de Realtime Database está diseñada para permitir solo operaciones que se pueden ejecutar rápidamente. Esto permite crear una muy buena experiencia en tiempo real que puede servir a una gran cantidad de usuarios sin afectar la capacidad de respuesta

2.1.2. Autenticación

Para la mayoría de Apps es necesario autenticar a los usuarios para que la app guarde sus datos en la nube de forma segura y proporcione la misma experiencia personalizada en todos los dispositivos del usuario.

Firebase Authentication [18] proporciona servicios de backend, SDK fáciles de usar y bibliotecas de Interfaz de Usuario (IU) ya elaboradas para autenticar a los usuarios en un app. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad federada populares, como Google, Facebook, Twitter y mucho más.

Firebase Authentication se integra estrechamente en otros servicios de Firebase y aprovecha los estándares de la industria como **OAuth 2.0** [25] y **OpenID Connect** [8], por lo que se puede incorporar fácilmente al el backend de la app.

Se puede usar Firebase Authentication para optimizar el proceso de autenticación en el proyecto, y no solo en una, sino en varias instancias de la base de datos del servicio Realtime Database de Firebase. Para conseguir esto, se usan las reglas personalizadas de Firebase Realtime Database en cada una de las instancias de la base de datos.

A continuación se pueden observar las principales funciones que el servicio Firebase Authentication proporciona a los desarrolladores. Se pueden encontrar dos tipos:

FirebaseUI Auth:

- **Solución de autenticación directa:** La forma recomendada de agregar un sistema de acceso completo a la app, **FirebaseUI**, proporciona una solución de autenticación directa que controla los flujos de IU para los usuarios que acceden con direcciones de correo electrónico y contraseñas, números de teléfono y con proveedores de identidad federada populares, que incluyen el Acceso con Google y el Acceso con Facebook.

Autenticación del SDK de Firebase: Si la autenticación se hace de esta manera, el desarrollador deberá controlar de forma manual como quiere que se realice teniendo disponible las siguientes alternativas.

- **Autenticación basada en correo electrónico y contraseña:** Autentica a los usuarios con sus direcciones de correo electrónico y contraseñas. El SDK de Firebase Authentication proporciona métodos a fin de crear y administrar usuarios que utilizan sus direcciones de correo electrónico y contraseñas para acceder. Firebase Authentication también maneja el envío de correos electrónicos para restablecer la contraseña.
- **Integración con proveedores de identidad federada:** Autentica usuarios mediante la integración con proveedores de identidad federada. El SDK de Firebase Authentication proporciona métodos que permiten a los usuarios acceder con sus cuentas de Google, Facebook, Twitter y GitHub.
- **Autenticación con número de teléfono:** Envía mensajes SMS a los teléfonos de los usuarios para autenticarlos.
- **Integración con sistemas de autenticación personalizados:** El desarrollador conecta el sistema de acceso existente de la app al SDK de Firebase Authentication y obtiene acceso a Firebase Realtime Database y otros servicios de Firebase.
- **Autenticación anónima:** Permite crear cuentas anónimas temporales para permitir el uso de funciones que requieran autenticación sin exigir que los usuarios accedan primero. Si posteriormente el usuario opta por registrarse, se puede actualizar la cuenta anónima y convertirla en una normal, de manera que el usuario pueda reanudar la actividad donde la interrumpió.

2.1.3. Cloud Functions

Cloud Functions para Firebase [17] permite ejecutar de forma automática el código de backend en respuesta a eventos activados por las funciones de Firebase y las solicitudes HTTPS. El código de estas funciones se almacena en la nube de Google y se ejecuta en un entorno administrado, para que así, de esta manera, no sea necesario que el desarrollador administre ni escale sus propios servidores siguiendo una filosofía de aplicación serverless muy común en la actualidad.

A continuación se pueden observar las principales funciones de las **Cloud Funtions**:

- **Integra la plataforma de Firebase:** Las funciones que se escriban pueden responder a eventos generados por otras funciones de Firebase y Google Cloud:
  - **Activadores de Cloud Firestore.** Se puede controlar eventos en Cloud Firestore sin necesidad de actualizar el código del cliente [9]. Las funciones del SDK de Cloud Functions para Firebase exportan un objeto **functions.firestore** que permite crear controladores vinculados con eventos específicos.
  - **Activadores de Realtime Database.** Cloud functions permite ejecutar operaciones de base de datos con privilegios administrativos completos y garantiza que cada cambio en la base de datos se procese de forma individual. Para crear funciones nuevas para los eventos de Realtime Database [15] se usa **functions.database**. Para controlar cuándo se debe activar la función, se especifica el controlador de eventos (onWrite(), onCreated(), onUpdate(), onDelete()) y la ruta de acceso de la base de datos en la que se detectarán los eventos.

- **Activadores de Remote Config.** Se puede activar una función como respuesta a un evento de Firebase Remote Config [16], incluida la publicación de una nueva versión de configuración o la reversión a una versión anterior. Para activar una función de Remote Config, se usa el controlador onUpdate que proporciona **functions.remoteConfig**.
- **Activadores de Firebase Authentication.** Se puede activar Cloud Functions a la creación o eliminación de cuentas de usuario de Firebase [12]. Por ejemplo, se puede enviar un correo electrónico de bienvenida a un usuario que acaba de crear una cuenta en la app. Para crear una función se usa **functions.auth.user()**, y seguidamente se usan los eventos onCreate() u onDelete() para indicar que se active cuando se cree un usuario o cuando se borre.
- **Activadores de Google Analytics para Firebase.** Google Analytics para Firebase [14] proporciona informes de eventos que ayudan a comprender la forma en que los usuarios interactúan con la app. Cloud Functions admite el evento AnalyticsEvent de Google Analytics para Firebase. Este evento se activa cada vez que la actividad del usuario genera un evento de conversión. Por ejemplo, se puede escribir una función que se active cuando se genere el evento in_app_purchase para indicar que se produjo una compra directa desde la app. Se debe especificar el evento de Analytics que se desee que active la función con el método **functions.analytics.event()** y controlar el evento dentro de onLog().
- **Activadores de Firebase Crashlytics.** Se puede activar una función en respuesta a los eventos de problemas de Crashlytics [13], lo que incluye problemas nuevos, recurrentes y alertas de velocidad. Para activar una función de Crashlytics, primero se debe generar un IssueBuilder con **functions.crashlytics .issue()** y, posteriormente, llamar a la función issue-generation respeciva del compilador (onNew(), onRegressed(), onVelocityAlert()).
- **Activadores de Cloud Storage.** Se puede activar una función en respuesta a la carga, actualización o eliminación de archivos y carpetas en Cloud Storage [11]. Para crear una función se usa **functions.storage**. Según si se desea definir el alcance de la función para un depósito específico de Cloud Storage (functions.storage.bucket('bucketName').object()) o usar el depósito predeterminado (functions.storage.object()).
- **Activadores de Cloud Pub/Sub.** Pub/Sub de Google Cloud [10] es un bus de mensajes de distribución global que se escala de forma automática en la medida que se necesita. Se puede usar el controlador de eventos **functions.pubsub** para crear una función que controle los eventos de Google Pub/Sub. El controlador de eventos que se usa es onPublish().
- **Activadores HTTP.** Se puede activar una función a través de una solicitud de HTTP mediante **functions.https**. Esto permite invocar una función síncrona a través de los métodos HTTP admitidos (GET, POST, PUT, DELETE y OPTIONS). Para crear una función que controle eventos HTTP se usa functions.https, y seguidamente se usa el evento onRequest() compatible con routers y apps que administra el marco de trabajo web Express.

- **Sin mantenimiento:** El código JavaScript o TypeScript se implementa en los servidores de Google con el comando **firebase deploy** desde la línea de comandos. Después de eso, Firebase aumenta los recursos de procesamiento automáticamente según los patrones de uso de los usuarios. El desarrollador no tendrá que preocuparse por las

credenciales, la configuración de servidores, el aprovisionamiento de servidores nuevos ni por sacar de servicio a los servidores antiguos.

- **Protege la privacidad y seguridad lógica:** En muchos casos, los desarrolladores prefieren controlar la lógica de la aplicación en el servidor para evitar alteraciones del lado del cliente. A veces, no es recomendable permitir que se aplique ingeniería inversa a ese código. Cloud functions está completamente aislada del cliente, de manera que se puede estar seguro de su privacidad y de que siempre hará exactamente lo que el desarrollador quiera.

A continuación se describe el ciclo de vida de una función en segundo plano, para ilustrar su funcionamiento:

1. El desarrollador escribe el código para una nueva función, selecciona el proveedor de eventos (como Realtime Database) y define las condiciones según las que se debe ejecutar la función (por ejemplo que escuche en el nodo o rama **/Notifications/invitacion** de la base de datos).

2. El desarrollador implementa la función y ejecuta el comando **firebase deploy** para subirla al servidor. Firebase la conecta al proveedor de eventos seleccionado.

3. Cuando el proveedor de eventos genera un evento (**onCreate(), onDelete(), onUpdate(), onWrite()**) que coincide con las condiciones de la función, se invoca el código.

4. Si la función está ocupada con muchos eventos, Google crea más instancias para controlar el trabajo con más rapidez. Si la función está inactiva, se borran las instancias.

5. Cuando el desarrollador actualiza la función mediante la implementación del código actualizado, se borran todas las instancias de la versión antigua y se reemplazan por instancias nuevas.

6. Cuando un desarrollador borra la función, se borran todas las instancias y se quita la conexión entre la función y el proveedor de eventos.

Además de detectar los eventos con una función en segundo plano, se pueden llamar a las funciones directamente con una solicitud HTTP o una llamada del cliente.

Como todos los servicios que se pueden encontrar, la mayoría tienen la opción gratuita y la opción de pago. En este caso se ha elegido la parte gratuita de **Firebase**, que proporciona, para cada uno de estos 3 servicios, las siguientes características:

Firebase Authentication	
Autenticación telefónica para EE.UU, Canadá e India	10.000 por mes
Autenticación telefónica para todos los demás países	10.000 por mes
Otros servicios de autenticación	permitido

Cuadro 2.1: Características de la versión gratuita de Firebase Authentication

Firebase Realtime Database	
Conexiones simultáneas	100
GB almacenados	1 GB
GB descargados	10 GB/mes
Varias bases de dato por proyecto	no permitido

Cuadro 2.2: Características de la versión gratuita de Firebase Realtime Database

Firebase Cloud Functions	
Invocaciones	125K/mes
GB-segundo	40K/mes
CPU-segundo	40K/mes
Redes de salida	Sólo servicios de Google

Cuadro 2.3: Características de la versión gratuita de Firebase Cloud Functions

2.2. API de Google

Las API de Google es un conjunto de APIs que permiten la comunicación e integración de los Servicios de Google con otros servicios.

En este caso, para el proyecto se ha usado el servicio Google Maps para poder visualizar el mapa en la aplicación, y así poder ubicar tanto al usuario como a las demás personas en él.

En el siguiente capítulo se incluirán más detalles sobre su integración en la aplicación.

Capítulo 3

JPC Locator

JPC Locator es una aplicación móvil que permite localizar a personas que se encuentren en los grupos a los que pertenece el usuario. Para ello, se empieza por la autenticación del usuario usando **Firebase Authentication** (mediante el segundo tipo de Autenticación de entre los que se mencionaron en el capítulo anterior) proporcionándole a la app el correo y la contraseña con el que se registró el usuario.

Los datos de cada usuario (nombre, email, ubicación, token del dispositivo móvil, grupos creados, grupos a los que pertenece y el número de teléfono), los usuarios que hay asociados a cada grupo, las conversasiones que se producen en el chat de cada grupo se guardarán en la base de datos **Realtime Database**, además, gracias a ella, se llevarán a cabo las notificaciones ya que esta aplicación es **SERVERLESS**, es decir, no se requiere un servidor que controle las notificaciones que se realizan. Posteriormente se entrará más en detalle en este tema.

3.1. Estructura de la aplicación

En esta sección se describe la arquitectura de la aplicación, figura B.1. Como son cada una de las funcionalidades que tiene y como es su funcionamiento en la aplicación. Cada funcionalidad cuenta con dos ficheros:

- Un fichero Java que se va a encargar de tener el código de lo que va a realizar cada funcionalidad.
- Un fichero XML que se encargará de la parte visual de la funcionalidad.

En la figura B.2 se puede observar el diagrama de clases que presenta esta estructura.

3.1.1. Registro

Esta funcionalidad, como su propio nombre indica, permite el registro de usuarios en la aplicación. Para ello, se solicita al usuario que introduzca un nombre, su teléfono, email con el que se quiera registrar y la contraseña que quiera poner, repitiéndola una segunda vez para verificarla. Internamente se comprueba que los datos introducidos son correctos y no contienen datos o valores inválidos (que la contraseña no sea menor a 6 valores o superior a 16, además de comprobar de que el formato del email es correcto). En la figura 3.1 se muestra el aspecto que tiene esta parte de la aplicación:

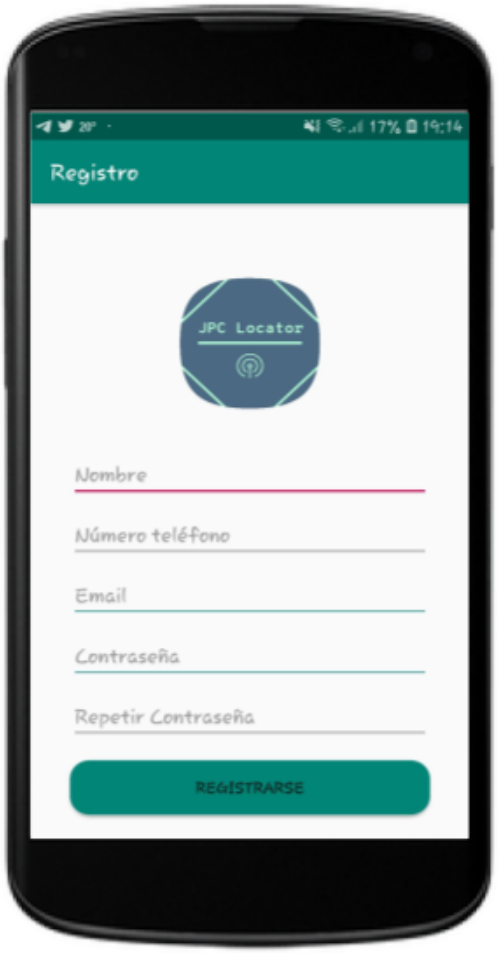

Figura 3.1: JPC Locator: Registro de usuarios

Cuando el usuario pulsa el botón para registrarse, esos datos se almacenan en la base de datos y **Firebase** se encarga de mandar un correo electrónico (que se puede personalizar en la consola de Firebase) al email con el que el usuario se registró para que pueda verificar la cuenta que creó (esto no se realiza por defecto. Es una de las posibilidades que Firebase proporciona a los desarrolladores).

3.1.2. Inicio de sesión

Esta funcionalidad se inicia con la aplicación. Si el usuario ya está registrado y no ha cerrado sesión no necesitará volver a iniciarla, ya que internamente se comprueba si el usuario ha cerrado sesión o no gracias a **Firebase Authentication**. Si se ha cerrado sesión, se le mostrará la vista del inicio de sesión. Una vez inicie la sesión, se actualizará el token del usuario, utilizado para remitirle las notificaciones.

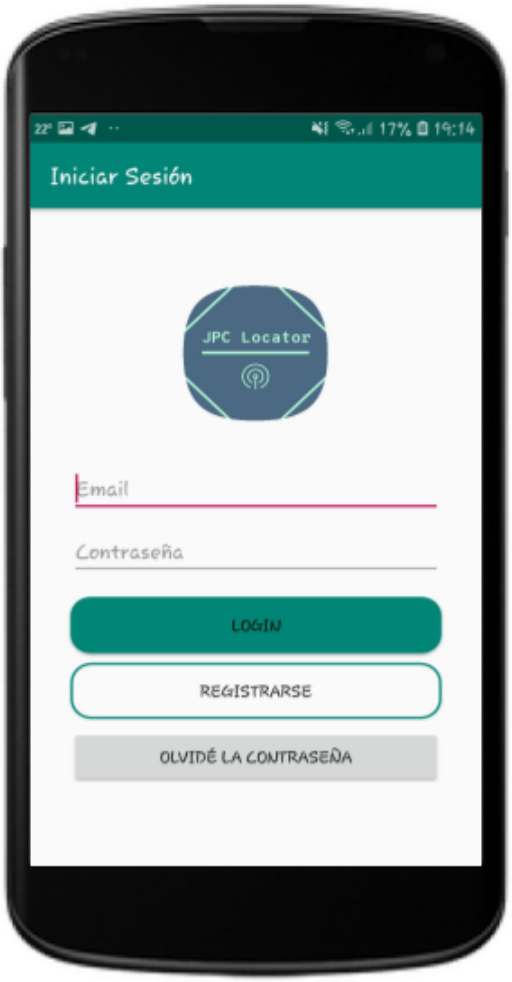

Figura 3.2: JPC Locator: Inicio de Sesión

Es muy sencilla, lleva a cabo el inicio de sesión del usuario en la aplicación, pero sólo si ha verificado previamente la cuenta cuando se ha registrado y si ha introducido correctamente los valores.

En esta funcionalidad es donde tenemos enlazados el **Registro** del usuario y la funcionalidad **Restablecer contraseña**, comentada a continuación.

3.1.3. Restablecer contraseña

Con esta funcionalidad, el usuario podrá restablecer la contraseña en caso de que se haya olvidado. El aspecto que tiene se muestra en la figura 3.3:

Figura 3.3: JPC Locator: Restablecer contraseña

Para ello, se introduce el email del usuario para enviarle un correo electrónico con un enlace que le lleva a la pagina que se indica en la figura 3.4:

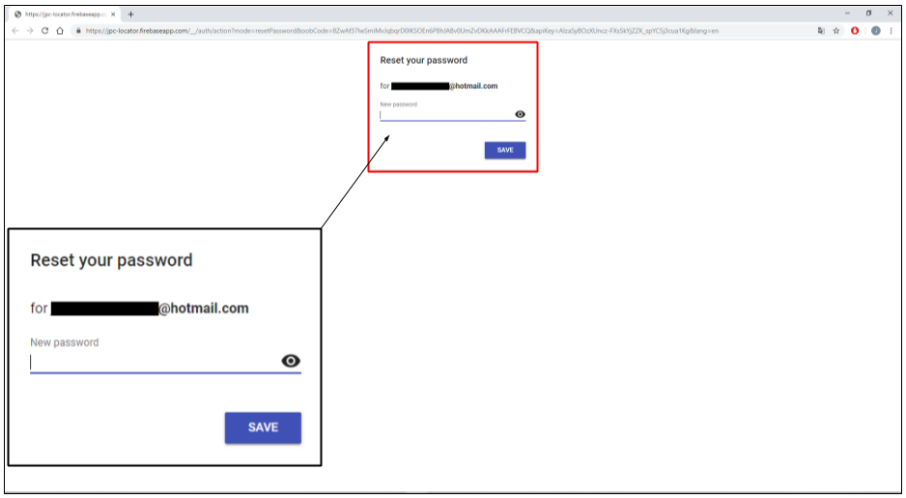

Figura 3.4: Página para restablecer contraseña de Firebase

3.1.4. Ventana de navegación

Esta funcionalidad es la que se abrirá una vez el usuario tenga la sesión iniciada. Esta parte cuenta con una ventana de navegación que contiene un menú con las diferentes funciones que la aplicación puede desempeñar. Figura 3.5:

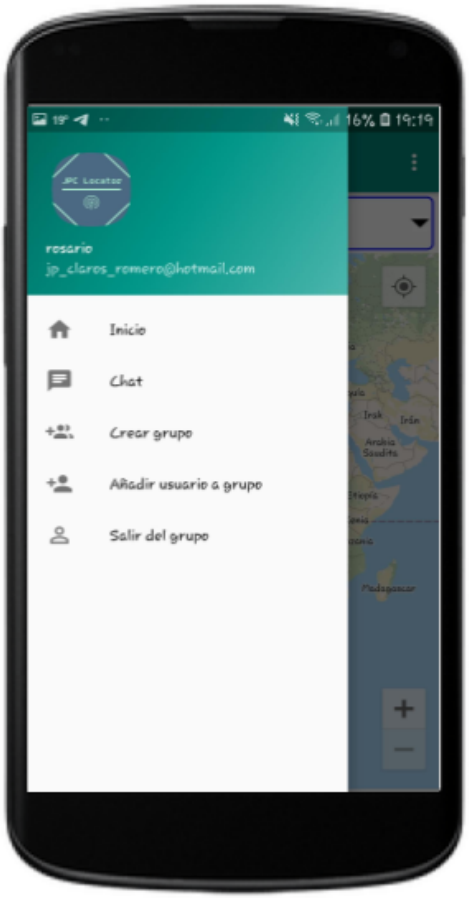

Figura 3.5: JPC Locator: Ventana de Navegación

Como se puede apreciar en la figura 3.5, se incluyen las funciones Inicio, Chat, Crear grupo, Añadir usuario y Salir del grupo, descritas en mayor profundidad en secciones posteriores de esta memoria.

Por último, esta funcionalidad cuenta con el botón Cerrar sesión, que permite llevar a cabo esta tarea. Figura 3.6:

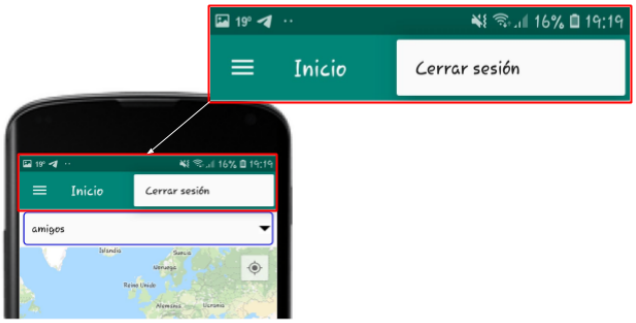

Figura 3.6: JPC Locator: Cierre de Sesión

Inicio

Esta pestaña **Inicio**, figura 3.7, se corresponde con lo que en Android se denomina **fragmento** (fragment[1]). Se va a encargar de contener un spinner[2] con los diferentes grupos a los que pertene el usuario y el mapa que la API de Google proporciona.

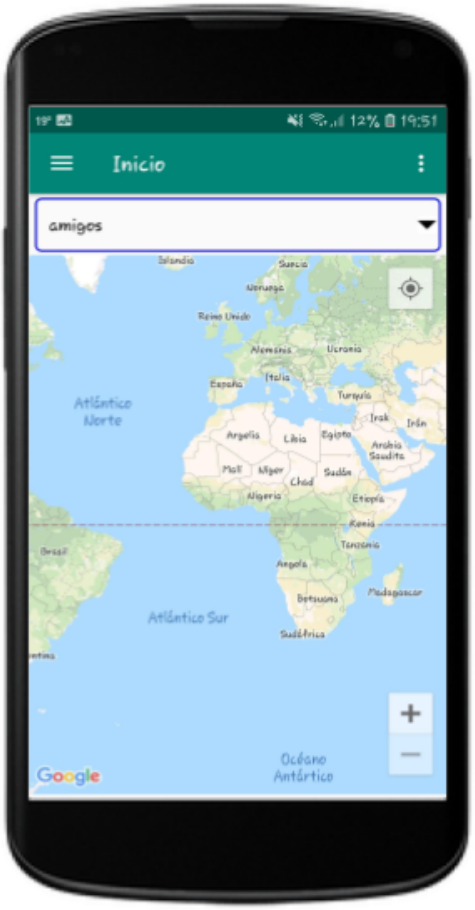

Figura 3.7: JPC Locator: Fragment principal. Mapa

Para integrar este mapa, los pasos [20] que hay que seguir son los siguientes:

1. Se añade los servicios de google en el Android Studio.
2. Se crea una nueva actividad (activity[3]). Esta activity será aquella que se llama **Google Maps Activity**. Al seleccionarla, se genera toda la estructura necesaria:
   - Se genera un fichero XML llamado **google_maps_api.xml** que es donde se pondrá la Key que la API de Google proporciona.
   - Se genera un fichero Java que contendrá el código de lo que se va a realizar en el mapa. Por ejemplo, subir la posición del usuario u obtener la latitud y longitud

[1] Un **fragment** es como una porción de la interfaz de usuario que puede añadirse o eliminarse de la interfaz de forma independiente al resto de elementos de la actividad, y que por supuesto puede reutilizarse en otras actividades.

[2] El widget **Spinner** de Android muestra una lista desplegable para seleccionar un único elemento y es equivalente al típico select de html o los ComboBox de otros entornos de desarrollo.

[3] Un **activity** es cada una de las pantallas o vistas que forman una aplicación.

de la base de datos (usando el fichero MapsPojo que contiene métodos **get**, para obtener estos datos de una manera mas cómoda), mostrar esta ubicación en el mapa, mostrar la posición de las demás personas mediante el uso de markers (son los punteros de color rojo que google maps usa cuando el usuario pulsa un encima de un establecimiento o lugar). Dichos markers tendrán un título (en este caso el nombre del usuario), texto (por ejemplo la distancia a la que está esa persona respecto del usuario), etc. Este fichero se genera con la estructura de un activity, pero para poder integrar la funcionalidad en la ventana de navegación, se adaptó dicha estructura a la de un fragment, que es un poco distinta a la de un activity ya que los métodos tienen nombres distintos y se usan de manera diferente.

- Se genera otro fichero XML que contendrá el fragment del mapa.

3. Se necesita una clave API (Key), que es una credencial centrada en el proyecto que sirve para dos propósitos:

   - **Identificación del proyecto:** Identificar la aplicación o proyecto que está realizando una llamada a la API.
   - **Autorización de proyecto:** Comprueba si a la aplicación que llama se le ha otorgado acceso para llamar a la API y si ha habilitado la API en el proyecto.

   Cuando se crea la clave API, se asocia con el proyecto. Al identificar el proyecto de llamada, la clave API permite que la información de uso se asocie con ese proyecto, y permite que las API de Google Maps Platform rechacen llamadas de otros proyectos.

   Para obtener la clave, se accede al enlace que aparece comentado en el fichero **google_maps_api.xml**. Cuando se accede a dicho enlace, se crea un nuevo proyecto y se accede a las credenciales. Una vez ahí, se le da a "añadir key" y se le da un nombre a la misma. Una vez que se está en la ventana de configuración, se incluye el nombre del paquete del proyecto Android en la sección correspondiente, además de la huella digital generado con la función hash SHA-1, que es diferente según el dispositivo donde se esté desarrollando. Una vez se da a Aceptar, se mostrará la clave que se genera con esos valores. Se incluye en el fichero XML asociándola a YOUR_KEY_HERE, ya que la aplicación accederá a esa parte del XML para obtener el valor y poder mostrar el mapa de Google.

Una vez se terminan estos pasos, el mapa de Google estará integrado en la aplicación.

Chat

Esta pestaña se encargará de mostrar dos cosas:

- Al acceder a ella, se mostrará una lista de los diferentes grupos, figura 3.8, a los que el usuario pertenece, que serán los diferentes chat a los que puede acceder. Para conseguir esto, se usa un ListView que contiene todos los grupos incluidos en la base de datos a los que el usuario pertenece. Una vez obtenidos de la base de datos, se almacenan en un contenedor y de ahí se pasan al ListView[4].

[4] Una vista **ListView** visualiza una lista deslizable verticalmente de varios elementos, donde cada elemento puede definirse como un Layout. Su utilización es algo compleja, pero muy potente. Diseñar un Layout individual que se repetirá en la lista.

- Si el usuario hace click en alguno de esos grupos, se abrirá el chat de ese grupo, figura 3.9, que es un activity que se encargará de obtener los mensajes que hay en la base de datos, y de escribir aquellos nuevos mensajes que se envíen.

Cuando se obtienen los mensajes, estos se muestran indicando quién lo remite, el propio contenido del mensaje y la fecha y hora de cuando se envió, diferenciando si el mensaje lo envió el usuario (se mostrará de color verde y a la derecha de la pantalla) o si lo envió otro usuario (se mostrará a la izquierda y con un color gris claro).

Para poder realizar todo esto, se usan los siguientes ficheros:

- Un fragment llamado GroupsFragment, que se encargará de mostrar esta lista de grupos en los que el usuario puede enviar mensajes.
- Un activity llamado GroupChatActivity que se encargará de traer los mensajes de la base de datos y de subir los nuevos mensajes que se escriban, con la siguiente información:
  - El nombre del usuario que lo envía.
  - El mensaje.
  - La fecha en la que envía el mensaje.
  - La hora.
  - El uid del usuario que lo remitió, para diferenciar al remitente. Si ese uid coincide con el usuario que está en la aplicación, se mostrarán los mensajes de una manera, y sino, de la otra.
  - El nombre del grupo en el que se escribió el mensaje. Este campo se pone porque será usado en la parte de notificaciones, se comentará en mayor detalle más adelante.
- Un fichero llamado Mensajes que contiene los valores: nombre, fecha, hora, mensaje y from (uid del usuario que lo mandó), junto con los métodos **get** y **set** para acceder a los valores o asignarles unos nuevos.
- Un fichero llamado MensajesAdapter que se encargará, mediante el uso de un RecyclerView[5], de darle formato a esos mensajes (asignarle un estilo u otro según el usuario y de enseñar la información del mensaje).

[5]Un **RecyclerView** es un contenedor de elementos en forma de lista al igual que la clase ListView. Aunque ambos tienen la misma función, este nuevo elemento permite "reciclar" los ítems que ya no son visibles por el usuario debido al scrolling.

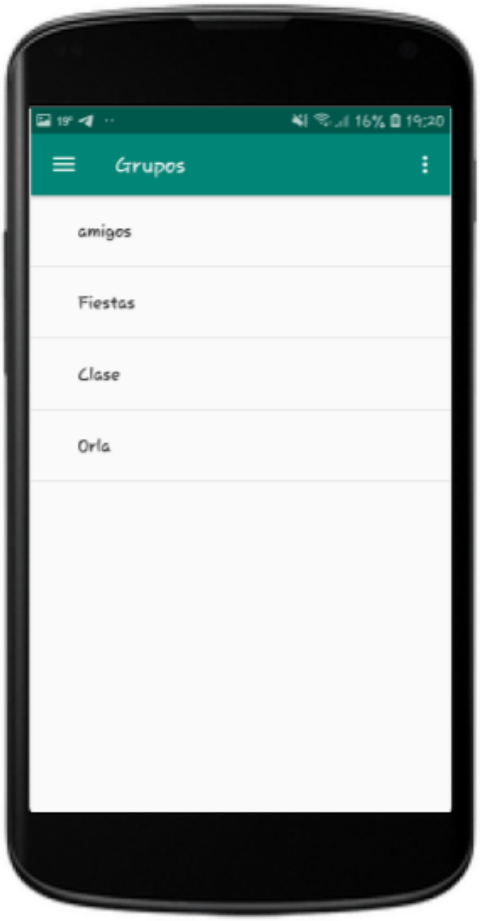

Figura 3.8: JPC Locator: Lista de grupos

Figura 3.9: JPC Locator: Chat

Crear grupo

Esta pestaña se encargará de crear los grupos que el usuario necesite. Para realizar esto, el fragment **AddGroupFragment** cuenta con una entrada de texto para escribir el nombre del grupo a crear, y un botón para que se ejecute la acción. Figura 3.10:

Cuando se rellena el campo **Nombre del grupo** y se pulsa el botón para crearlo, internamente se consulta la rama del usuario para comprobar si ya existe dicho grupo. Si la rama no existe, es que no se ha creado todavía ningún grupo, por lo que se procederá a crearlo. Si la rama existe, realizará una búsqueda de ese grupo. Si lo encuentra, se generará un Toast[6] informando de la existencia del grupo. Si no lo encuentra, se informará que se ha creado.

[6]Un **Toast** es un mensaje que se muestra en pantalla durante unos segundos al usuario para luego volver a desaparecer automáticamente sin requerir ningún tipo de actuación por su parte, y sin recibir el foco en ningún momento

Figura 3.10: JPC Locator: Crear Grupo

Añadir usuario

Esta pestaña, llevará a cabo la acción de añadir un usuario a un grupo, se abrirá sólo si el usuario tiene grupos creados a los que puede añadir usuarios. Si no tiene grupos creados, la aplicación le remitirá a la pestaña de crear grupos.

El fragment **AddUsuarioFragment** cuenta con un spinner para elegir el grupo entre los ya generados al que se quiere añadir el usuario, una entrada de texto para indicar el número de teléfono del usuario que se va a añadir, y un botón para llevar a cabo la acción. Figura 3.11:

Figura 3.11: JPC Locator: Añadir Usuario a Grupo

Cuando se selecciona el grupo, se escribe el número de teléfono del usuario a añadir y se pulsa el botón, internamente se busca al usuario que tenga ese número de teléfono, y cuando se encuentra, se guarda su uid. Una vez hecho esto, se comprueba en la rama **/Usuarios_por_grupo/id_grupo** de la base de datos si ya está ese usuario dentro del grupo. Si su uid ya está en el grupo, se le mostrará al usuario un Toast indicándoselo, sino, se escribirá en la rama **/Notifications/Grupo** la información necesaria para poder remitir a dicho usuario la notificación de invitación a grupo. El proceso de notificaciones se verá con mas detalle posteriormente.

Salir del grupo

Esta pestaña, que llevará a cabo la acción de poder salir de un grupo al que el usuario pertenezca, estará disponible sólo si el usuario pertenece a algún grupo. Si no es así, la aplicación le remitirá a la pestaña de crear grupos, indicándole mediante un Toast si quiere crear un grupo.

El fragment **LeaveGroupFragment** cuenta con un spinner para elegir el grupo, de entre aquellos a los que pertenece, del que el usuario quiera salir, y un botón para llevar a cabo la acción. Figura 3.12:

Figura 3.12: JPC Locator: Salir del Grupo

Cuando se selecciona el grupo y se pulsa el botón, internamente se eliminará el grupo del conjunto de grupos creados por el usuario y se propagará su eliminación a el reto de la arquitectura de la aplicación.

3.2. Notificaciones

Las notificaciones se utilizan para implementar las siguientes funcionalidades:

- Notificar a un usuario cuando una solicitud para unirse a un grupo
- Remitir la respuesta del usuario a dicha invitación.
- Notificar al usuario cuando alguna persona perteneciente a alguno de sus grupos se encuentra a menos de 500 metros.
- En el caso del chat, se enviará una notificación cuando una persona escriba un mensaje en uno de los grupos a los que pertenece el usuario.

Para controlar cuando y cómo se van a lanzar estas notificaciones, se usa el servicio **Firebase Cloud functions**. Como se dijo anteriormente, este servicio necesita un activador para ejecutar las funciones implementadas. En la implementación realizada en este trabajo final de grado se usó el activador **Realtime Database**, en el que se indica a las funciones implementadas que escuchen en la rama Notifications para llevar a cabo los tres primeros tipos de notificaciones. Para el caso del chat, se indica a las funciones que escuchen en la rama Chat

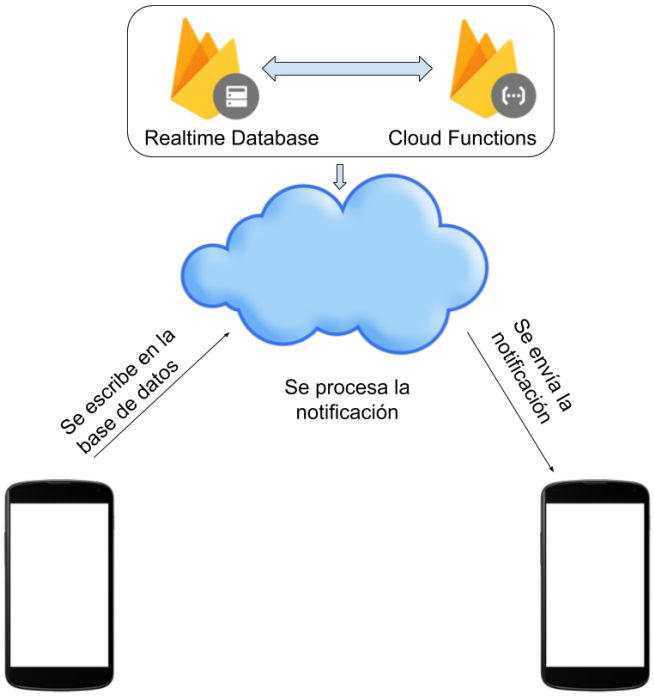

Figura 3.13: Estructura de una notificación

de la base de datos. La estructura que presenta el funcionamiento de las notificaciones es como se indica a continuación en la figura 3.13:

Seguidamente se tratará un poco más en profundidad el funcionamiento de cada una de estas notificaciones.

3.2.1. Invitación de usuario a grupo

Cloud function

Para esta notificación, la función escucha en el nodo **/Notifications/Grupo** y se escribe la siguiente información:

- El **uid** del emisor y del receptor de la notificación. Esta información se proporciona a través de la propia rama (/Notifications/Grupo/emisor_id/receptor_id).
- Dentro de la rama se puede observar la siguiente información:
  - **Nombre** del usuario emisor y receptor para incluirlo en el cuerpo de la notificación. En este caso, solo se usa el nombre del usuario emisor para pasarlo en el payload, indicando de esta manera en la aplicación qué usuario está enviando la notificación.
  - El **token** de cada uno de los usuarios para poder enviar la notificación mediante la función sendToDevice (payload[7], token). En este caso se usa el token del usuario receptor, el usuario a quien se está invitando al grupo.
  - **Nombre del grupo** al que se le está invitando. Este dato se usa en el cuerpo de la notificación como dato informativo, y para que se añada a la rama del usuario en caso de que acepte la invitación.

[7] El payload es el contenido que se le proporciona a la notificación (como el título de la misma, el mensaje, etc) en formato JSON

- **ID** del grupo al que se le está invitando. Esto se hace para que, cuando el usuario acepte la invitación, se le añada en su rama el grupo con es ID, que será el mismo para todos los usuarios que se encuentren en ese grupo.
- Dos variables **booleanas** que nos indique si el usuario acepta la invitación o si la rechaza. Se ha decidido usar dos variables ya que cuando se envía la invitación a grupo, se inicializan a **false**, y según se acepte o se rechace la invitación, se le da el valor **true** a una o a otra. Como el procesamiento de las cloud functions es muy rápido, se ejecutaba la función antes de que se le diera el valor **true** a las dos variables, por lo que siempre salía que se rechazaba la invitación. Esto fue lo que hizo que se tomara la decisión de usar dos variables booleanas en lugar de una. Posteriormente se entrará más en detalle en qué funcionamiento concreto realizan estas variables.

Una vez se obtienen esos datos de la base de datos, se crea un payload con el cuerpo que se indica en la figura 3.14:

```
const payload = {
   data: {
        id: "1",
        title: "Invitacion para unirte a grupo",
        body: `${nombreEmisor} quiere agregarte al grupo ${nombreGrupo}`,
        nombre: nombreEmisor,
        grupo: nombreGrupo,
        grupoId: grupoID,
        uidEmisor: usuarioEmisor,
        uidReceptor: usuarioReceptor
   }
};
```

Figura 3.14: Payload para invitación de usuario a grupo

Al payload se le puede proporcionar dos formatos de JSON: un data (que puede contener todo lo que queramos, asignando aquellos valores a variables con nombres que el desarrollador quiera darles) o un notification (que ya tiene definido los distintos campos que se pueden usar). En la implementación realizada se optó por un data que contiene un id (que se usará internamente en la aplicación para diferenciar los distintos tipos de notificaciones), un título y un cuerpo que es lo que se muestra a los usuarios cuando llega la notificación, y los valores que se han obtenido de la base de datos para cuando el usuario responda, tenga en cuenta qué rama de la base de datos seguir, además de los valores del grupo al que se quiera añadir en caso que responda afirmativamente.

JPC Locator

En la aplicación, cuando se quiere unir un usuario a un grupo, lo que se hace es escribir en la base de datos en la rama /Notifications/Grupo, junto con los uid del usuario emisor y receptor, los datos que se nombraron anteriormente.

Cuando se recibe este tipo de notificación, mediante el servicio **MessageService** que se encarga de esto con el método onMessageReceived, lo primero que se hace es que se comprueba el id del data que se recibe para poder identificar el tipo de notificación que es. En este caso es id = 1. Figura 3.15

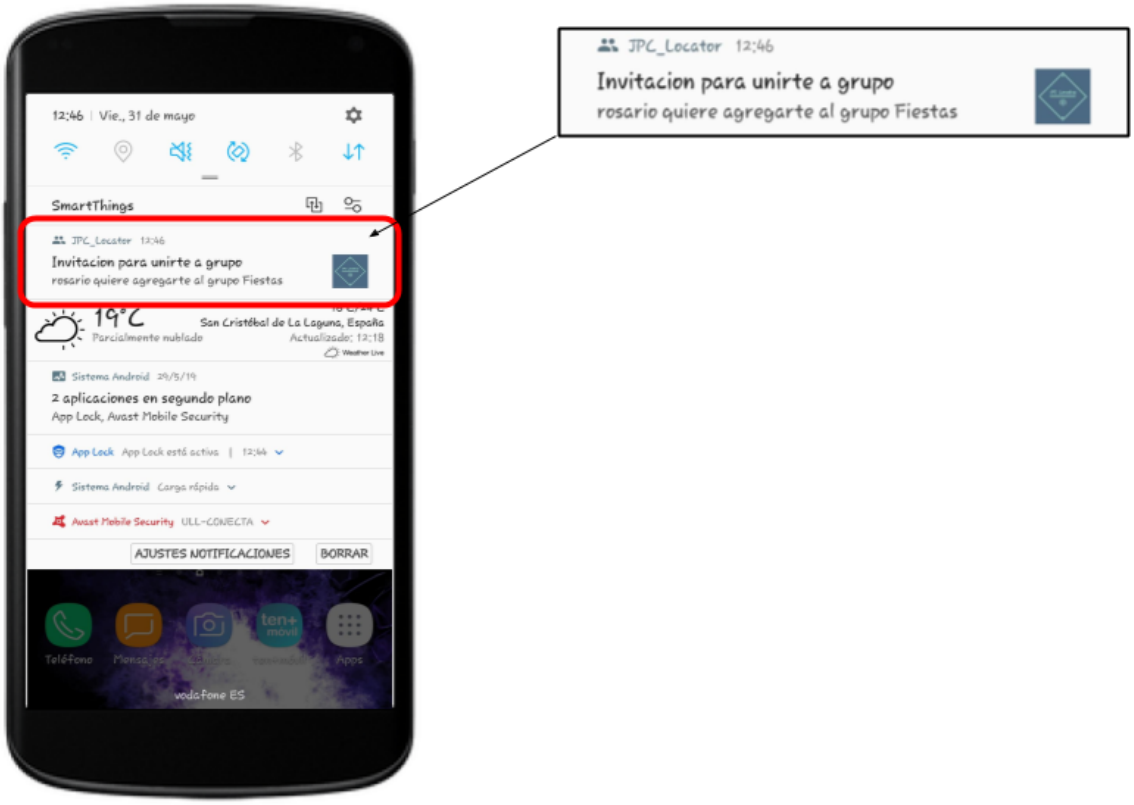

Figura 3.15: JPC Locator: Notificación para agregar usuario a un grupo

3.2.2. Respuesta para invitación de grupo

Cloud function

Para esta notificación, lo que se hace es que la función escuche en el nodo **/Notifications/Grupo** al igual que la anterior, junto con los uid del usuario emisor y receptor, pero en este caso solo se modificarán las variables booleanas que se nombraron anteriormente. Según el valor que se le de a estas variables, significará lo siguiente:

- Si la variable **recibido** es verdadera, significará que el usuario recibió la notificación y rechazó la invitación.
- Si la variable **unirse** es verdadera, significará que el usuario recibió la notificación y aceptó la invitación.

Esto se hace para elegir que tipo de mensaje enviar en la notificación. Una vez se tienen los datos que se quieren remitir, se crean los payload de las figuras 3.16 y 3.17:

```
const payload = {
    data: {
        id: "0",
        title: "Respuesta de peticion",
        body: `${nombreReceptor} ha rechazado la solicitud de unirse al grupo ${nombreGrupo}`
    }
};
```

Figura 3.16: Payload para cuando se rechaza la invitación

```
const payload = {
    data: {
        id: "0",
        title: "Respuesta de peticion",
        body: `${nombreReceptor} ha aceptado la solicitud de unirse al grupo ${nombreGrupo}`
    }
};
```

Figura 3.17: Payload para cuando se acepta la invitación

Al payload se le proporciona un data que contiene un id (que se usará internamente en la aplicación para diferenciar los distintos tipos de notificaciones), un título y un cuerpo que es lo que se muestra a los usuarios cuando llega la notificación.

JPC Locator

En la aplicación, cuando se recibe la notificación para unirse a un grupo y se hace click en ella, se abrirá un activity para este tipo de notificaciones. Este activity es el **RequestActivity**, que mostrará un texto indicando el nombre del usuario que envió la notificación y el grupo al que está invitando. Luego, este activity proporciona dos botones para que el usuario elija si se quiere unir o no al grupo. El activity se puede ver en la figura 3.18 y la notificación en la figura 3.19:

Figura 3.18: JPC Locator: RequestActivity

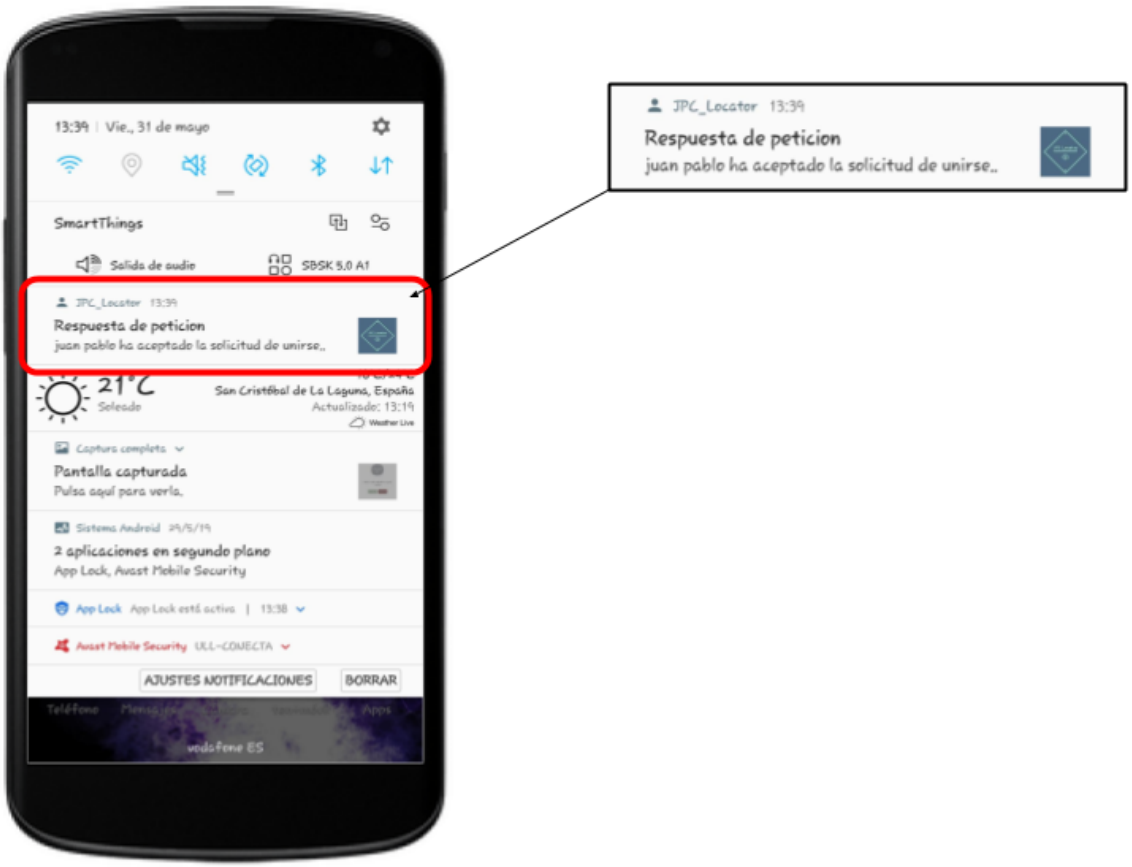

Figura 3.19: JPC Locator: Notificación respuesta para invitación a grupo

3.2.3. Notificación de cercanía

Cloud function

Para esta notificación, se crea una función que escuche en el nodo **/Notifications/Cerca**, que al igual que las anteriores notificaciones, la rama entera contendrá el uid del usuario emisor y el usuario receptor. Además de esta información, se proporciona adicionalmente los siguientes aspectos:

- La **distancia** a la que se encuentra el usuario de su amigo.
- El **nombre** del amigo que se encuentra a esa distancia.
- El **token** de la persona que está cerca, para notificarle que el usuario actual está cerca de ella.

Una vez se tienen esos datos, se genera el payload de la figura 3.20:

```
const payload = {
    data: {
        id: "3",
        title: "Usuarios cerca",
        body: `${nombre} esta a ${distancia} metros de ti.`
    }
};
```

Figura 3.20: Payload para notificar la cercanía de un usuario

El payload tendrá como id un 3, para indicarle a la aplicación que es la notificación de cercanía. Además, se le indicará el título de la misma junto a un cuerpo (body) que es lo que se mostrará a los usuarios.

JPC Locator

El aspecto de la notificación se puede ver en la figura 3.21:

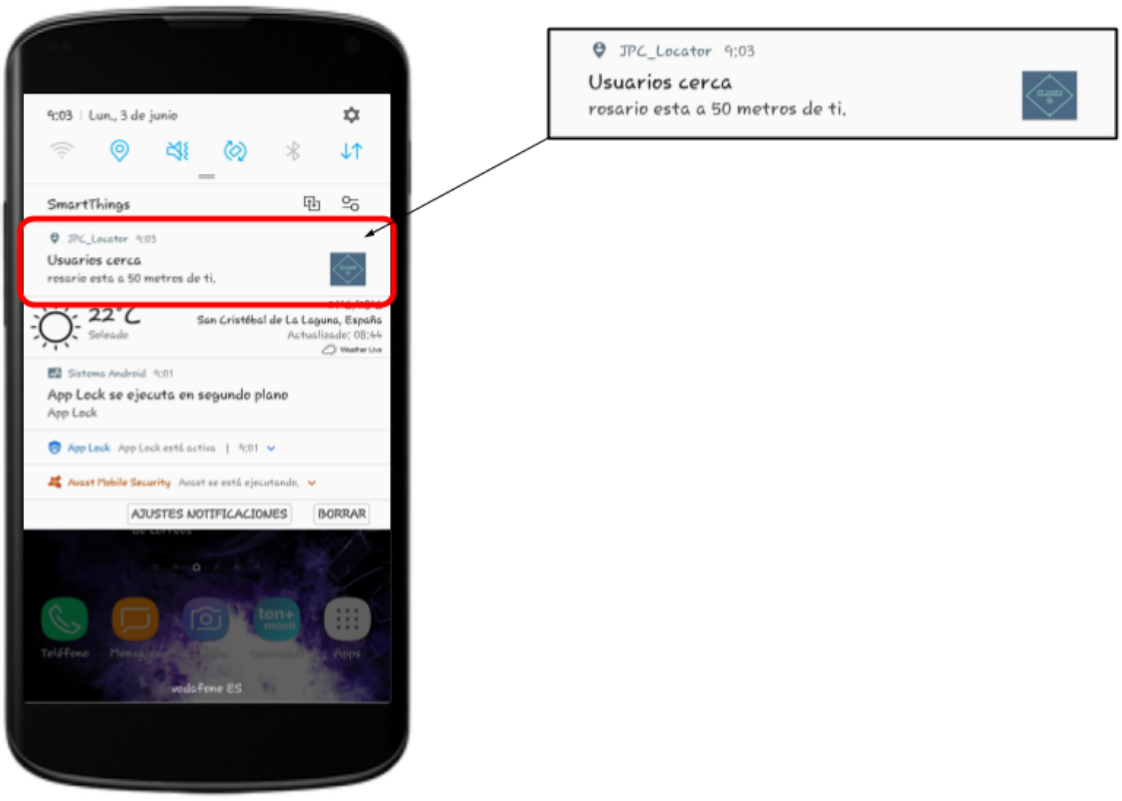

Figura 3.21: JPC Locator: Notificación de cercanía de un usuario

3.2.4. Notificación de mensaje de chat

Cloud function

Para esta notificación, se crea una función que escuche en el nodo **/Chat/grupo_id/mensaje_id**. Se creará una rama por cada mensaje que se envíe, en la que cada una tendrá la siguiente información:

- La **fecha** en la que se envió el mensaje.
- El **uid** del usuario que envió el mensaje.
- El **nombre** del grupo en el que se envió el mensaje.
- La **hora** en la que se envió el mensaje.
- El **mensaje**.
- El **nombre** del usuario que envió el mensaje.

Una vez se tienen esos datos, se genera el payload de la figura 3.22:

```
const payload = {
    data: {
        id: "2",
        title: `${nombre} ha mandado un mensaje al grupo ${grupo}`,
        body: mensaje,
        grupoId: grupoID,
        grupo: grupo
    }
};
```

Figura 3.22: Payload para notificar cuando se manda un mensaje

El payload tendrá como id un 2, para indicarle a la aplicación que es la notificación de chat. Además, se le indicará el título de la misma (informando quien envió el mensaje y en que grupo) junto a un cuerpo (body) en el que se le mostrará a los usuarios el mensaje que se ha remitido. El ID del grupo en el que se manda el mensaje y el nombre del mismo para que, internamente en la aplicación, se abra el chat de dicho grupo cuando el usuario pulse en la notificación. En esta notificación no se le indica el token del usuario receptor ya que es un conjunto de usuarios (todos aquellos que se encuentren en el grupo), por lo que se hace es obtener todos los uid de dichos usuarios e ir mandando el payload a cada uno a medida que se va seleccionando sus tokens.

JPC Locator

El aspecto de la notificación que se recibe es el que se muestra en la figura 3.23:

Figura 3.23: JPC Locator: Notificación de mensaje de chat

Capítulo 4

Seguridad: cifrado y reglas de Firebase

En toda aplicación que gestiona datos personales se tienen que incluir medidas de seguridad que protejan la confidencialidad de los datos y el acceso a los mismos.

En este caso, en el desarrollo de la aplicación **JPC Locator** se aplican las reglas de Firebase en la base de datos para controlar el acceso a los datos y el protocolo criptográfico **Diffie-Hellman**, su versión sobre curvas elípticas para reducir el consumo de recursos y mejorar la eficiencia del sistema, para generar la clave secreta con la que se va a cifrar los mensajes. El algoritmo de cifrado seleccionado es el **AES**.

4.1. Reglas de Firebase

Las reglas de Firebase que se aplicaron a la aplicación se van a encargar de cubrir los siguientes aspectos de seguridad relacionados con el control de acceso a datos:

- No se va a permitir leer ni escribir datos en la base de datos a menos que el usuario que lo vaya a realizar esté autenticado (esto se realiza gracias a una variable auth que nos proporciona esta información).
- Además de la medida de seguridad anterior, para el caso de los datos de cada usuario, no se va a permitir que un usuario pueda escribir en un nodo o rama de los Usuarios a no ser que esa rama donde vaya a escribir datos sea la suya. Es decir, se va a comprobar su **uid** para ver si coincide con el de la rama (ya que cada rama de la base de datos de los Usuarios tienen como id el uid del usuario que se registra).

A continuación, en la figura 4.1 se pueden observar como es la estructura de estas reglas:

Figura 4.1: Reglas de la base de datos de Firebase

4.2. Protección de la confidencialidad

4.2.1. Cifrado de clave secreta

Para llevar a cabo el cifrado de los mensajes que se envían en la aplicación, **JPC Locator** usará cifrado de clave privada (también conocido como cifrado simétrico o cifrado de clave secreta).

Los sistemas de cifrado de clave privada o secreta [22] utilizan una sola clave que comparten el remitente y el destinatario. Ambos deben poseer la clave; el remitente cifra el mensaje mediante la clave y el destinatario descifra el mensaje con la misma clave. Para poder establecer una comunicación privada, tanto el remitente como el destinatario deben mantener la clave en secreto.

El cifrado de clave privada tradicionalmente requiere una clave para cada par de personas que necesitan comunicarse de forma privada. Normalmente, el número necesario de claves aumenta considerablemente a medida que se incrementa el número de participantes. Concretamente, para un grupo de una cantidad **N** de personas que utilizan un criptosistema de clave secreta, es necesario distribuir una cantidad de claves equivalente a **N*(N-1)/2** [26]. Para el caso de la aplicación desarrollada, la idea que se sigue es utilizar una misma clave para todos los usuarios pertenecientes a un grupo. La primera vez que se añade un usuario a un grupo, se aplicará Diffie-Hellman elíptico (posteriormente se describirá en mayor detalle) para generar una clave entre el usuario nuevo y el usuario que lo invita. En relación a los nuevos usuarios que se añadan al grupo, cada vez que se vaya a incluir un nuevo usuario se volverá a aplicar el protocolo de Diffie-Hellmann entre el usuario que invita a los demás y el nuevo usuario. La clave generada en este momento, será remitida al resto de usuarios del grupo cifrándola con la clave anterior del grupo.

4.2.2. Diffie-Hellman sobre curvas elípticas

El protocolo criptográfico **Diffie-Hellman** [3], debido a Whitfield Diffie y Martin Hellman, es un protocolo de generación y establecimiento de claves entre partes que no han tenido contacto previo sin necesidad de contar un canal seguro.

El sistema de Diffie-Hellman se basa en la idea de que dos interlocutores pueden generar conjuntamente una clave compartida sin que un intruso que esté escuchando las comunicaciones pueda llegar a obtenerla. En la figura 4.2 se muestra su funcionamiento:

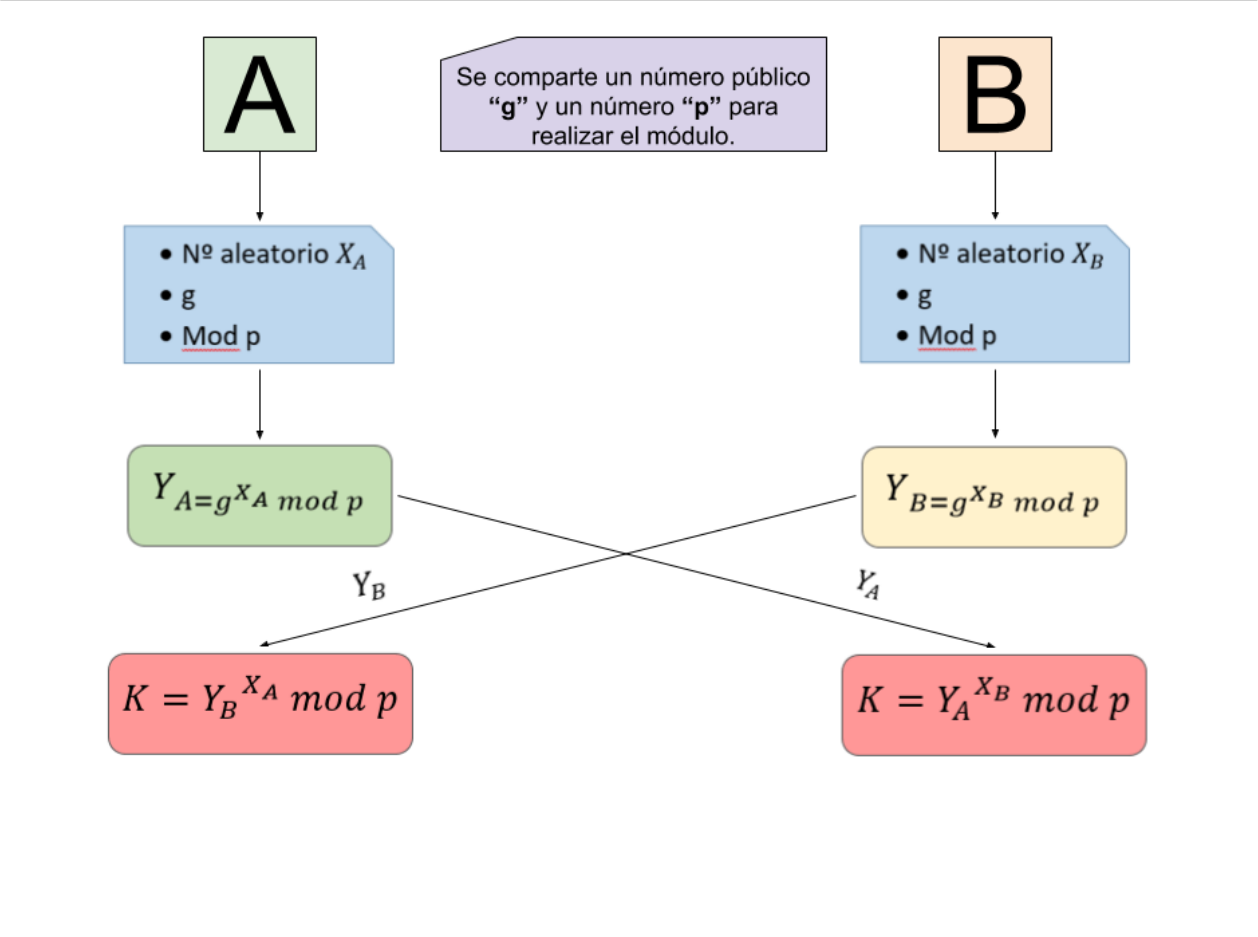

Figura 4.2: Funcionamiento del protocolo Diffie-Hellman

Los pasos que sigue son estos:

- En primer lugar, se eligen dos números que **A** y **B** comparten: un número primo **p** y una base **g** que es un generador del grupo $\mathbb{Z}_p$.
- A continuación, A y B eligen un número secreto aleatorio que sea menor a ese número primo p (X_a <p , X_b <p).
- A envía a B el valor Y_a, y B envía a A el valor Y_b.
- Por último, tanto A como B elevan ese valor que reciben a su número secreto (haciendo módulo p a ese resultado), y de esa manera ya tendría cada interlocutor la clave secreta.

Para la aplicación se usó el protocolo **Diffie-Hellman sobre curvas elípticas**. Los pasos que se siguen con esta versión, a diferencia de la original, son los siguientes:

- En lugar de compartir la base **g**, A y B comparten una curva elíptica *E* que está definida en $\mathbb{Z}_p$.
- Comparten un punto *P* de dicha curva.
- Como números secretos escogen X_a y X_b, que pertenecen a $\mathbb{Z}_p$ y están en la curva elíptica *E*.
- Para obtener las claves públicas (Y_a y Y_b), se multiplica la clave secreta por el punto *P* que se nombró anteriormente.
- Por último, A y B intercambian sus claves públicas para poder obtener la clave secreta compartida. Para ello, multiplican su clave secreta por la clave pública del otro, obteniendo así esta clave secreta que será la misma para los dos.

4.2.3. Algoritmo de cifrado Advanced Encryption Standard (AES)

Una vez se tiene la clave secreta compartida, se va a llevar a cabo el cifrado de los mensajes. Para ello, se hace uso de la librería llamada **Spongy Castle** [1].

Esta librería es de código abierto y proporciona algoritmos criptográficos para el cifrado de mensajes (como el algoritmo AES, que es el que se usa en este caso).

El algoritmo AES [27], fue aprobado por **Federal Information Processing Standards**, es un algoritmo criptográfico que puede usarse para proteger datos electrónicos. El algoritmo es un cifrado simétrico de bloques que puede encriptar (cifrar) o desencriptar (descifrar) información.
El cifrado convierte los datos a una forma ininteligible llamada **texto cifrado**. Cuando se descifra el texto cifrado, los datos vuelven nuevamente a su formato original, llamado texto sin formato o **texto plano**.

El algoritmo AES es iterativo y es capaz de usar claves criptográficas de **128**, **192** y **256 bits** para cifrar o descifrar bloques de datos de 128 bits.

Este estándar puede ser utilizado por los departamentos y agencias federales cuando una agencia determina que la información (no clasificada) requiere protección criptográfica. Además de este escenario, este estándar no sólo se usa en organizaciones gubernamentales, sino que también se aplica para fines comerciales y organizaciones privadas.

Para llevar a cabo el cifrado, se coge a la matriz de estado [2] del texto plano y se le aplican las cuatro funciones *AddRoundKey(), SubByte(), ShiftRow(), MixColumn()*.

[1] **Spongy Castle** es un reempaquetado de todas las funciones que están en la librería Bouncy Castle [7], incluida en el SDK de Android

[2] Es una matriz rectangular de bytes, que posee 4 filas y N columnas, siendo N = Tamaño del bloque elegido / 32.

AddRoundKey

Se realiza sobre la matriz de estado la operación **or exclusiva** con la subclave correspondiente de cada ronda. El bloque resultante será el bloque que se use para la siguiente ronda. El funcionamiento de esta operación la podemos ver en la figura 4.3:

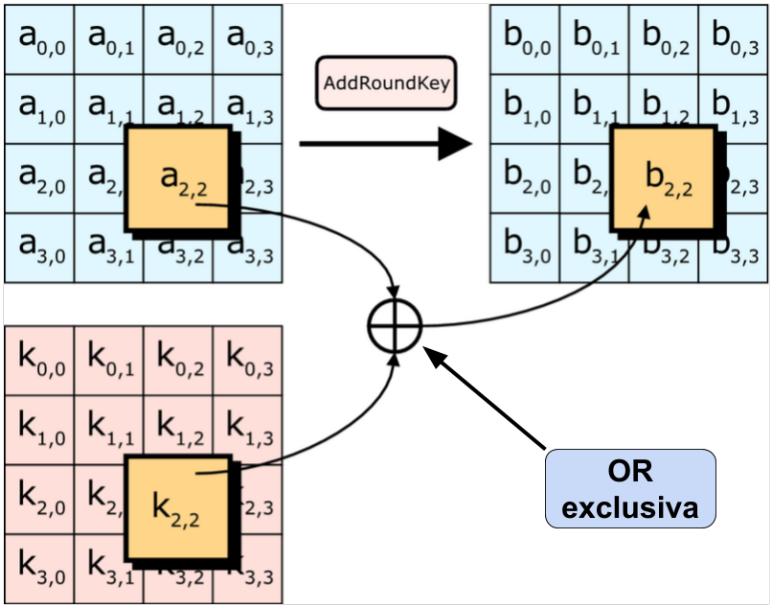

Figura 4.3: Operación AddRoundKey

Las n subclaves (10, 12 o 14, según el tamaño del bloque y de la clave) se generan internamente a partir de la clave original y realizando una serie de operaciones.

SubByte

Se realiza la sustitución de cada uno de los bytes **XY** de la matriz de estado por el byte correspondiente de la caja **S-Box**[3]. La primera parte del byte de la matriz de estado indica la fila y la segunda parte la columna de la caja S-Box, figura 4.5. El funcionamiento de esta operación la podemos ver en la figura 4.4.

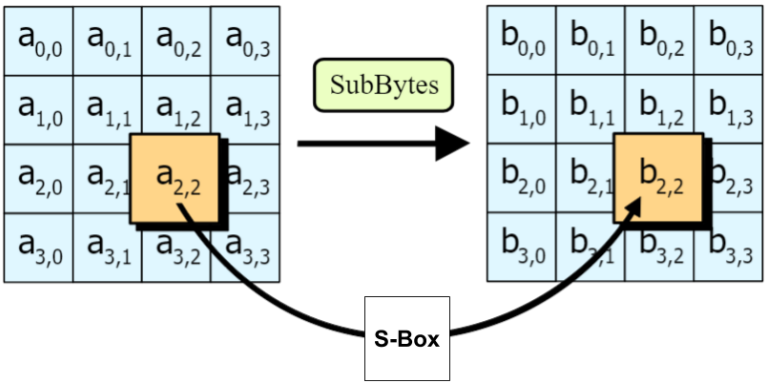

Figura 4.4: Operación SubByte

[3]Una caja S o S-Box es un componente básico de algoritmos de clave simétrica que intervienen en la deficnición de la operación de sustitución.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figura 4.5: Caja S-Box

ShiftRow

En esta función lo que se hace es rodar hacia la izquierda el número de posiciones según la fila. Para la primera fila no se hace nada, para la segunda se rueda una posición, para la tercera fila se ruedan dos posiciones y para la cuarta se ruedan tres posiciones. La figura 4.6 representa un esquema de esta función.

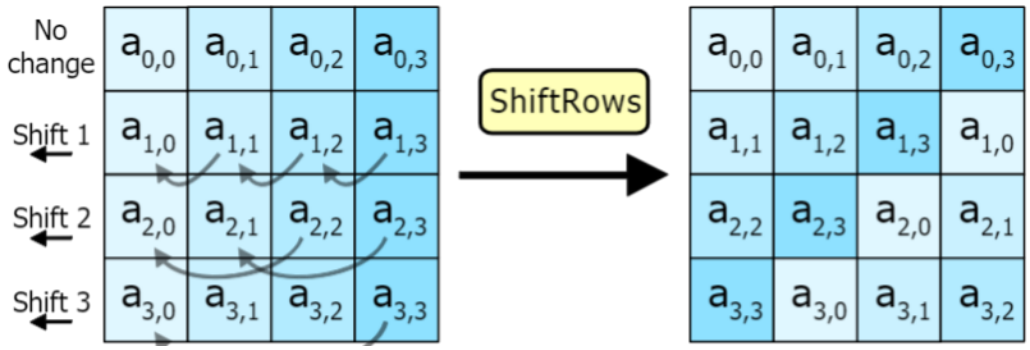

Figura 4.6: Operación ShiftRow

MixColumns

Esta función es un poco más compleja que las anteriores. Consiste en multiplicar cada una de las columnas de la matriz de estado por una constante que ya esta declarada, obteniéndose una nueva matriz de estado cuando se multiplican todas las columnas. En la figura 4.7 se puede observar como es esta operación:

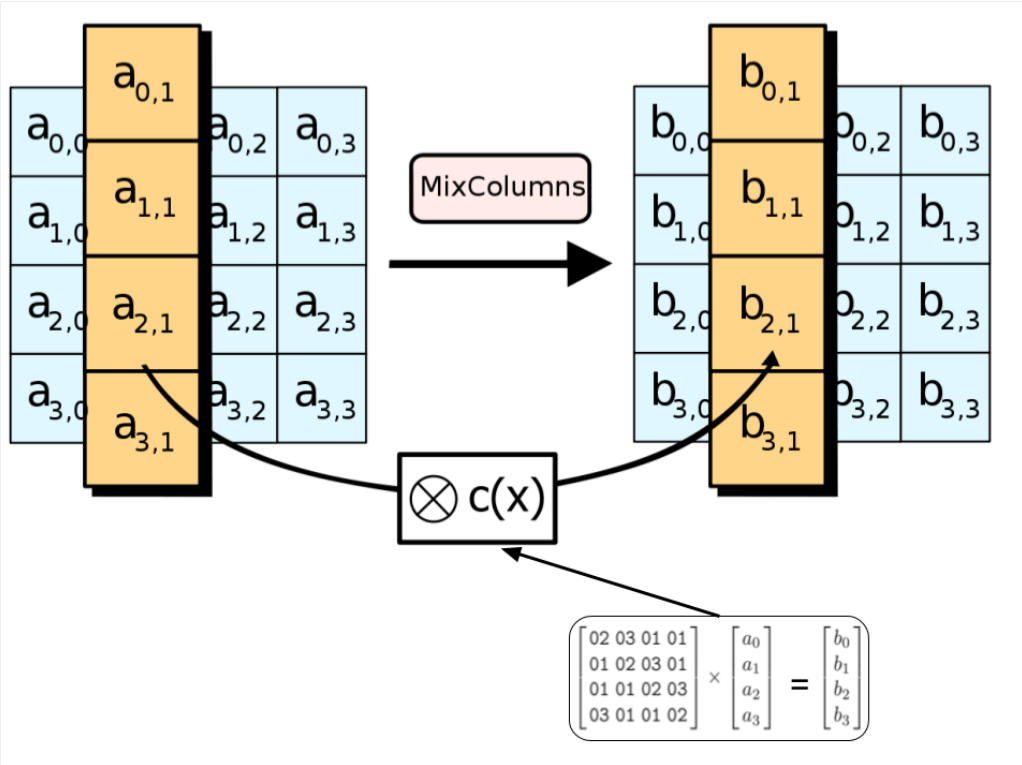

Figura 4.7: Operación MixColumns

El orden que se sigue para el cifrado del texto plano es el siguiente:

1. Se le aplica la función AddRoundKey() a la matriz de estado con la matriz de la clave original.

2. Una vez sale de la función, se realizarán n-1 vueltas ejecutando las funciones en el siguiente orden (siendo n = 10, 12 o 14 dependiendo del tamaño del bloque y clave. Tabla 4.1): *SubByte(), ShiftRow(), MixColumn() y AddRoundKey()* respectivamente. En esta parte, AddRoundKey() se aplicará con las distintas n-1 subclaves que se tienen generadas.

Clave/Bloque	Bloque = 128	Bloque = 192	Bloque = 256
Clave = 128	10	12	14
Clave = 192	12	12	14
Clave = 256	14	14	14

Cuadro 4.1: Tabla de iteraciones según el tamaño de bloque y clave en bits

3. Finalmente, una vez se terminan esas vueltas, se ejecutan *SubByte(), ShiftRow() y AddRoundKey()* respectivamente como último paso, obteniendo así el texto cifrado. En esta parte, AddRoundKey() se aplica con la última subclave.

Para descifrar el mensaje, se realizan las mismas operaciones pero a la inversa, es decir, se realizan en el texto cifrado. Un esquema del algoritmo AES lo podemos ver en la figura incluida en el apéndice C.

Capítulo 5

Conclusiones y líneas futuras

En la primera parte de este trabajo fin de grado se ha realizado un análisis de los antecedentes que ya existen en cuanto a solución desarrollada, además de las tecnologías de comunicación (BLE, LTE, WiFi Direct, GPS) y desarrollo utilizadas.
El desarrollo de este proyecto ha permitido el aprendizaje de tecnologías de localización y transferencia de información, además del aprendizaje de desarrollo de aplicaciones Android, ya era la primera vez que el desarrollador utilizaba muchos de los elementos necesarios para llevar a cabo todas las funcionalidades del sistema.
La aplicación que se ha creado, **JPC Locator**, no sólo es la solución al problema que se presenta en grandes eventos, el poder localizar a familiares o amigos que se pierdan en estos sin tener que recurrir a las llamadas telefónicas. El ámbito de aplicación puede extenderse a otros escenarios como es el caso de los cuerpos de emergencias que coordinan la seguridad de estos eventos (ya sea protección civil o cualquier otro equipo de emergencia similar). Pueden usar JPC Locator para localizarse entre ellos y gestionar mejor la seguridad, sin tener que estar todo el tiempo con los sistemas de comunicación que utilizan (walkie talkie) por el problema que se explicó anteriormente, que en estos sistemas puede pasar como las llamadas telefónicas, el excesivo ruido puede impedir que escuchen con claridad la mayoría de información que se proporcionan entre ellos.
Si bien esta aplicación esta bastante completa en cuanto a las funcionalidades que inicialmente se plantearon alcanzar, hay algunos aspectos que se pueden mejorar:

- Una de las mejoras que se pueden aplicar es el poder iniciar sesión a través de alguno de los proveedores nombrados anteriormente (Google, Facebook, etc).
- Además de los proveedores, poder registrarse usando el teléfono móvil únicamente.
- En el chat, además de enviar mensajes, se puedan remitir imágenes o vídeos, para que de esta manera la información que se comparta entre los usuarios sea mas precisa y entendible (indicando por estos medios si está en el sitio del mapa en un sitio elevado o subterráneo).
- Aportar más funcionalidades al mapa de Google que tiene la aplicación, como podría ser la posibilidad de ver imágenes o navegar por el mapa, de igual forma que Google Maps permite, para que de esta manera el usuario tenga más información de la zona y la localización sea mas eficaz.
- Añadir perfiles de los usuarios. Cada usuario tenga su propio perfil (con foto, descripción y, además, pueda modificar los datos que usó para el registro). Cada usuario pueda ver el perfil de los otros usuarios, pero no todos los datos, sino los que el usuario quiera mostrar.

Estas mejoras le aportarían a la aplicación un gran valor añadido.

Capítulo 6

Summary and Conclusions

In the first part of this project, an analysis of the antecedents that already exist in terms of the solution developed, the communication and development technologies has been carried out.
The development of this project has allowed to learn localization and information transfer technologies, as well how to develop applications on Android. It was the first time that the developer used many of the necessary elements to carry out all the system functionalities
The application that has been created, **JPC Locator**, is not only the solution to the problem that arises in large events, being able to locate family or friends who are lost without having to use phone calls. The scope of application can be extended to other scenarios such as the case of emergency bodies that coordinate security in these events (either civil protection or any other similar emergency team). They can use JPC Locator to locate each other and to better manage security, avoiding to use the communication systems they use (walkie talkie) because of the problem explained above, that in these systems can happen as telephone calls, excessive noise can prevent them from clearly hearing most of the information provided between them.
Although this application is quite complete in terms of the functionalities that were initially planned to achieve, there are some aspects that can be improved:

- One of the improvements that can be applied is to log in through one of the providers mentioned above (Google, Facebook, etc).
- In addition to the providers, being able to register using the mobile phone only.
- In the chat, in addition to sending messages, you can send images or videos, so that the information shared between users is more accurate and understandable (indicating by these means if it is on the site map in an elevated or underground).
- Provide more functionality to the Google map that has the application, such as the ability to view images or navigate the map, just as Google Maps allows, so that the user has more information of the area and the location is more effective.
- Add user profiles. Each user has his own profile (with photo, description and also can modify the data used for registration). Each user can see the profile of other users, but not all the data, but what the user wants to show.

These improvements would add great value to the application.

Capítulo 7

Presupuesto

En este capítulo se incluyen los costes estimados del proyecto, distinguiéndose los recursos que han sido necesarios a nivel de hardware, software y de personal.

7.1. Costes de Software

Software	Coste
IDE Android Studio	0 €
Licencia desarrollador Android	22 €
Firebase Realtime Database con Plan Spark	0 €
Firebase Authentication con Plan Spark	0 €
Cloud Functions de Firebase con Plan Spark	0 €
Total	22 €

Cuadro 7.1: Tabla costes software

7.2. Costes de Hardware

Software	Coste
Móvil Samsung Galaxy S7	305 €
Ordenador portátil	540 €
Total	845 €

Cuadro 7.2: Tabla costes software

7.3. Costes del personal

A continuación, se exponen los costes de los recursos humanos necesarios para el desarrollo de la aplicación. Se ha tomado como como referencia una jornada laboral de 4 horas durante 5 días cada semana, y el precio por hora de trabajo está en torno a unos 15 €:

Software	Jornadas	Coste
Análisis de las tecnologías a utilizar	10	600 €
Formación de desarrollo android	15	900 €
Estudio de sistemas de seguridad	8	480 €
Desarrollo e implementación de la aplicación	30	1800 €
Detección y corrección de errores	10	600 €
Total	73	4380 €

Cuadro 7.3: Tabla costes de recursos humanos

7.4. Costes totales

Software	Coste
Coste software	22 €
Coste hardware	845 €
Coste RRHH	4380 €
Total	5247 €

Cuadro 7.4: Tabla costes totales

Apéndice A

Diagrama de Caso de Uso

A.1. Diagrama de caso de uso de la aplicación

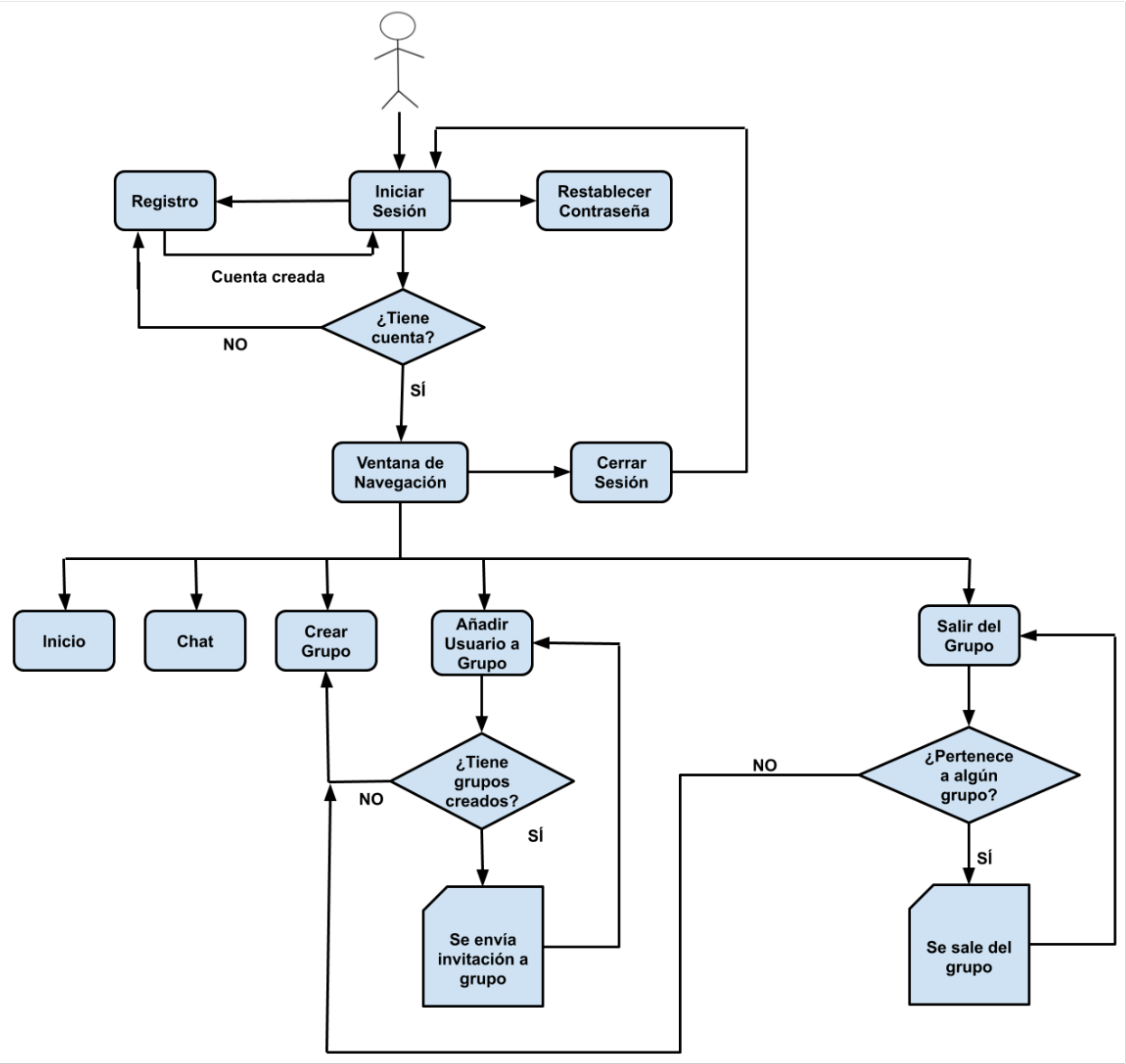

Figura A.1: Diagrama de caso de uso de la aplicación

Apéndice B

Estructura de ficheros y diagrama de clases

B.1. Estructura de ficheros

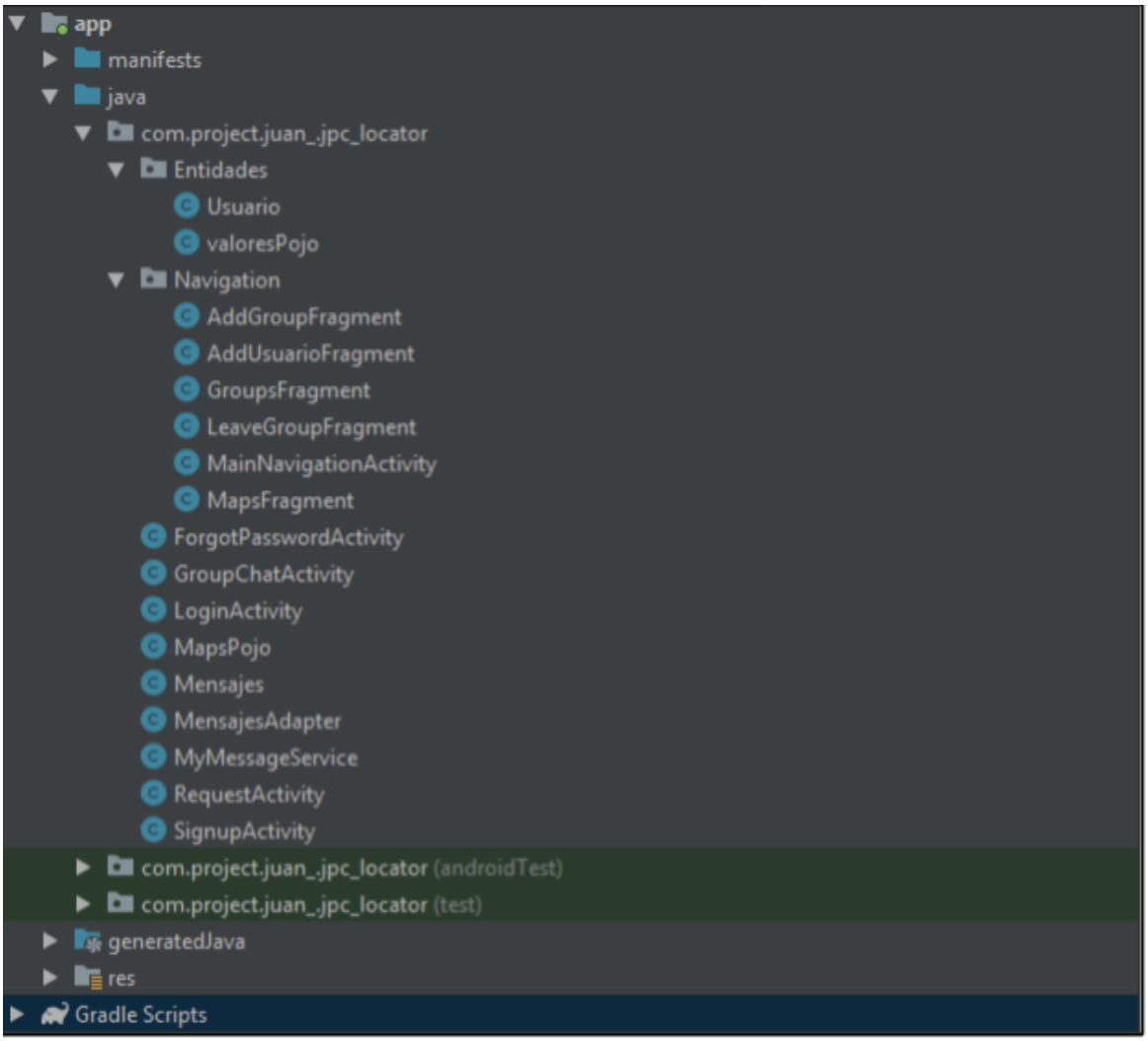

Figura B.1: Estructura de Ficheros

B.2. Diagrama de clases

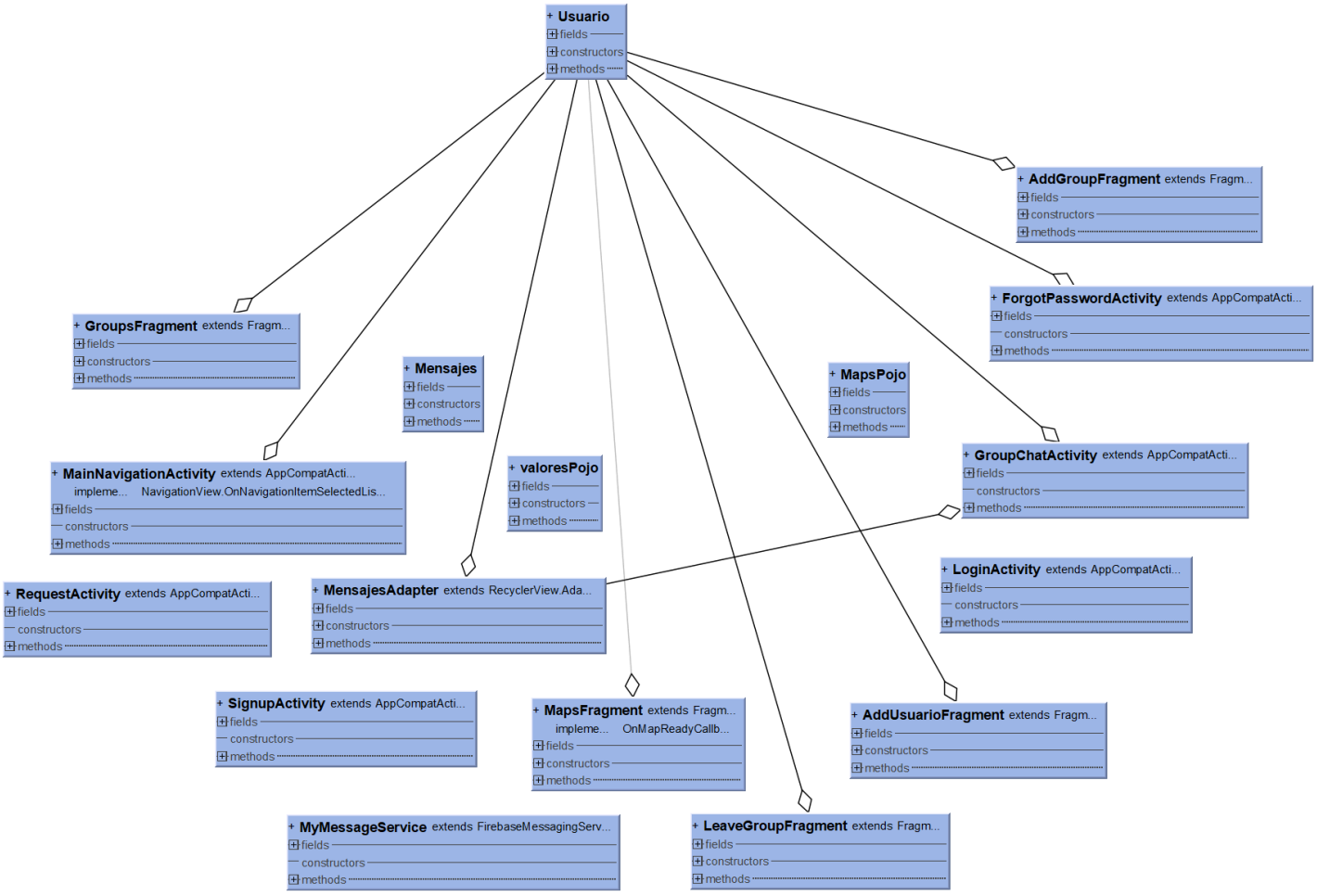

Figura B.2: Diagrama de Clases

Apéndice C

Algoritmo de cifrado AES

C.1. Esquema del algoritmo AES

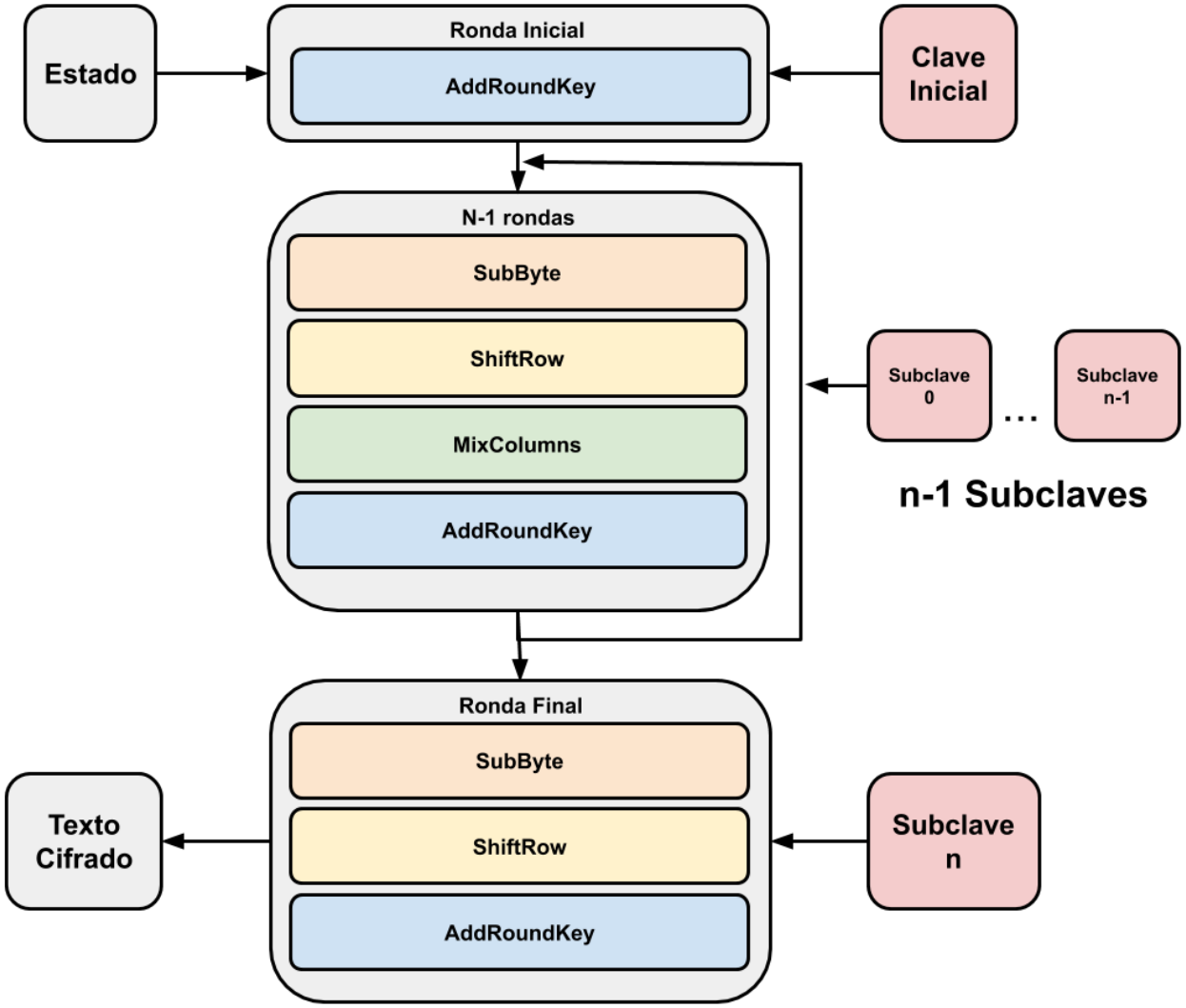

Figura C.1: Algoritmo AES

Bibliografía

[1] ¿qué es lte direct? https://www.adslzone.net/redes/que-es-el-lte-direct/. Accessed: 2019-03-14.

[2] ABC. ¿qué es el wifi direct y para que se usa? https://www.abc.es/tecnologia/consultorio/20150212/abci-wifi-direct-como-usar-201502111739.html. Accessed: 2019-03-14.

[3] Mejor Antivirus. Diffie-hellman (criptografía). http://www.mejor-antivirus.es/terminologia-informatica/que-es-diffie-hellman.html. Accessed: 2019-06-07.

[4] Colin Boyd. *Protocols for authentication and key establishment / Colin Boyd, Anish Mathuria*. Berlin [etc], Springer, 2003.

[5] Carlos Cabello. ¿wifi direct, así puedes sacar partido a este protocolo de comunicación en el hogar? https://www.nobbot.com/tecnologia/mi-conexion/wifi-direct. Accessed: 2019-03-14.

[6] Manuel Pérez Cardona. Firebase, qué es y para qué sirve la plataforma de google. https://www.iebschool.com/blog/firebase-que-es-para-que-sirve-la-plataforma-desarroladores-google-seo-sem/. Accessed: 2019-05-27.

[7] Bouncy Castle. The legion of the bouncy castle. https://www.bouncycastle.org/. Accessed: 2019-06-07.

[8] OpenID Connect. Welcome to openid connect. https://openid.net/connect/. Accessed: 2019-05-27.

[9] Firebase. Activadores de cloud firestore. https://firebase.google.com/docs/functions/firestore-events. Accessed: 2019-05-27.

[10] Firebase. Activadores de cloud pub/sub. https://firebase.google.com/docs/functions/pubsub-events. Accessed: 2019-05-27.

[11] Firebase. Activadores de cloud storage. https://firebase.google.com/docs/functions/gcp-storage-events. Accessed: 2019-05-27.

[12] Firebase. Activadores de firebase authentication. https://firebase.google.com/docs/functions/auth-events. Accessed: 2019-05-27.

[13] Firebase. Activadores de firebase crashlytics. https://firebase.google.com/docs/functions/crashlytics-events. Accessed: 2019-05-27.

[14] Firebase. Activadores de google analytics para firebase. https://firebase.google.com/docs/functions/analytics-events. Accessed: 2019-05-27.

[15] Firebase. Activadores de realtime database. https://firebase.google.com/docs/functions/database-events. Accessed: 2019-05-27.

[16] Firebase. Activadores de remote config. https://firebase.google.com/docs/functions/rc-events. Accessed: 2019-05-27.

[17] Firebase. Cloud functions para firebase. https://firebase.google.com/docs/functions. Accessed: 2019-05-27.

[18] Firebase. Firebase authentication. https://firebase.google.com/docs/auth. Accessed: 2019-05-27.

[19] Firebase. Firebase realtime database. https://firebase.google.com/docs/database/. Accessed: 2019-05-27.

[20] Google. Get started. https://developers.google.com/maps/documentation/android-sdk/start. Accessed: 2019-05-29.

[21] Manuel J. Gutiérrez. Qué es wifi direct y en qué se diferencia del bluetooth. https://elandroidelibre.elespanol.com/2016/10/wifi-direct-vs-bluetooth.html. Accessed: 2019-03-14.

[22] IBM. Cifrado de clave privada. https://www.ibm.com/support/knowledgecenter/es/SSMKHH_9.0.0/com.ibm.etools.mft.doc/ac55940_.htm. Accessed: 2019-06-07.

[23] Jezabel Molina-Gil Alexandra Rivero-García Iván Santos-González, Pino Caballero-Gil. Decentralized authentication for opportunistic communications in disaster situations. https://link.springer.com/chapter/10.1007/978-3-319-67585-5_55. Accessed: 2019-03-14.

[24] Life360. Localizador familiar y movil. https://play.google.com/store/apps/details?id=com.life360.android.safetymapd&hl=es. Accessed: 2019-03-14.

[25] OAuth2.0. Oauth 2.0. https://oauth.net/2/. Accessed: 2019-05-27.

[26] Jean-François Pilou. Criptografía de clave privada (o clave secreta). https://es.ccm.net/contents/126-criptografia-de-clave-privada-o-clave-secreta. Accessed: 2019-06-07.

[27] FIPS (Federal Information Processing Standards). Advanced encryption standard (aes). https://csrc.nist.gov/csrc/media/publications/fips/197/final/documents/fips-197.pdf. Accessed: 2019-06-08.