Aurelio Martínez Barceló

SORORIDAPP

TRABAJO DE FIN DE GRADO

dirigido por Pere Millán Marco

Grado en Ingeniería Informática

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2021

Resumen.

Proyecto de desarrollo de una aplicación personal Android de geolocalización, donde se pretende ayudar a mujeres en necesidad de socorro por acoso, solicitando ayuda a usuarios cercanos. Sororidapp pretende ser la ayuda necesaria para todas las situaciones de acoso que viven las mujeres en todos los países del mundo, sobre todo en la noche, siendo así una aplicación desarrollada en beneficio de la sociedad general. Esta aplicación está desarrollada en Android Studio tanto en lenguaje Java como en Kotlin, según la necesidad. También utiliza diferentes APIs de Google (p.e: Maps API, Directions API), así como Firebase como base de datos en tiempo real y autenticación. Asimismo, dicho desarrollo se ha concluido con un prototipo funcional de la aplicación descrita, y con la suma de conocimientos de uso de APIs de Google, profundización del desarrollo de interfaces de usuario para aplicaciones móviles, una ampliación del conocimiento en programación Java de cara a las funcionalidades de Android y Google, y una base sólida, desde cero, de programación en Kotlin para dispositivos móviles Android, lenguaje de programación oficial de Android. Aparte se practican los conceptos de usabilidad y funcionalidad a la hora de crear una interfaz de usuario útil para una aplicación de este ámbito.

Resum.

Projecte de desenvolupament d'una aplicació personal Android de geolocalització, on es pretén donar ajut a dones que necessitin auxili per un cas d'assetjament, sol·licitant ajuda a usuaris propers. Sororidapp pretén ser l'ajuda necessària per a totes les situacions d'assetjament que viuen les dones en tots els països del món, sobretot a la nit, sent així una aplicació desenvolupada en benefici de la societat general. Aquesta aplicació està desenvolupada en Android Studio tant amb llenguatge Java com Kotlin, segons la necessitat. També utilitza diferents APIs de Google (p.e: Maps API, Directions API), així com Firebase com a base de dades en temps real i autenticació. Així mateix, aquest desenvolupament s'ha conclòs amb un prototip funcional de l'aplicació descrita, i amb la suma de coneixements d'ús d'APIs de Google, aprofundiment del desenvolupament d'interfícies d'usuari per a aplicacions mòbils, una ampliació del coneixement en programació Java de cara a les funcionalitats d'Android i Google, i una base sòlida, des de zero, de programació en Kotlin per a dispositius mòbils Android, llenguatge de programació oficial d'Android. A part es practiquen els conceptes d'usabilitat i funcionalitat a l'hora de crear una interfície d'usuari útil per a una aplicació d'aquest àmbit.

Abstract.

Development project of an Android geolocation app for personal use where the aim is to help women who needs it in a harassment case, asking for help from nearby users. Sororidapp aims to be the necessary help for all situations of harassment experienced by women in all countries of the world, especially at night, thus being an application developed for the benefit of society in general. This application is developed in Android Studio using both Java and Kotlin languajes, as needed. It also uses different Google APIs (e.g., Maps API, Directions API), as well as Firebase as a real-time database and authentication. This development has also been concluded with a functional prototype of the application described and the sum of knowledge of using Google APIs, deepening the development of user interfaces for mobile applications, an expansion of knowledge in Java programming for the functionalities of Android and Google, and a solid foundation, from scratch, in Kotlin programming for Android mobile devices, the official programming language of Android. Apart from that, the concepts of usability and functionality will be practiced when creating a useful user interface for an application in this field.

Índice

Índice de Tablas

Índice de Figuras

1 Introducción

En el día a día, un grandísimo porcentaje de la población está acompañada en todo momento por su dispositivo móvil. Teniendo esto en mente, la forma más lógica de tratar de ayudar a las personas parece obvia: desarrollar una aplicación para estos dispositivos, ya que será donde más gente la tenga a su disposición.

Por otro lado, en nuestra sociedad, y más ahora volviendo a la nueva normalidad, ocurren cada día más ataques machistas o abusos de poder hacia las mujeres de nuestro país y en el mundo en general. Estas situaciones se dan normalmente cuando una mujer se encuentra aislada, sin poder pedir ayuda a nadie: son perseguidas, increpadas, acosadas o incluso violadas, aprovechando su momento de soledad, donde la agresión resulta más sencilla y asequible.

Pese a que estas acciones se dan en todo tipo de situaciones y franjas horarias, sí que hay una clara tendencia a ocurrir en ambientes de ocio nocturno, cuando además de solas, las mujeres atacadas pueden encontrarse ebrias o cansadas y somnolientas, lo cual alienta a los agresores conocedores de su estado y su debilidad.

Otra problemática, explicada y contrastada de forma extendida más adelante, es que, pese a que no todos los hombres realicen este tipo de acoso, una mujer no puede saber si un desconocido será un acosador o no, por lo que muchas veces no pueden confiar en ellos para pedir auxilio. De igual forma ocurre con los cuerpos de seguridad del estado en los que muchas veces se confía y su actuación deja que desear, o ellos mismos abusan de su poder para realizar acciones como las nombradas anteriormente.

El objetivo, pues, es desarrollar una aplicación que permita a las mujeres pedir auxilio cuando se encuentren en este tipo de situaciones y que puedan confiar en que la ayuda prestada por otras usuarias de la aplicación es de confianza. Una aplicación de mujeres ayudando a mujeres. De ahí le viene el nombre: Sororidapp.

El proyecto nace de la necesidad de mejora de nuestra sociedad, así como del amparo y la ayuda a las mujeres en estas situaciones. Parte también de la impotencia de salvaguardarlas. Puesto que no se puede estar en todas partes para ayudar, facilitar que puedan ayudarse entre ellas es la manera que encuentro de mejorar la situación. Porque donde haya una mujer en peligro, habrá otra dispuesta a ayudarla.

Esta memoria de TFG está estructurada de la siguiente forma. En el apartado 2 se indican los objetivos y funcionalidades principales de la aplicación. En el apartado 3 se muestran los resultados del estudio de mercado realizado. El apartado 4 se dedica a los requisitos funcionales y no funcionales, así como a los casos de uso. El apartado 5, se describen las diferentes tecnologías y librerías utilizadas. En el apartado 6 se realiza el análisis de costes y ganancias, así como un plan de negocio. En el apartado 7 se desarrolla la explicación de la implementación del proyecto. El apartado 8 simula el uso de la aplicación, dando a su vez explicaciones de las diferentes decisiones de diseño realizadas y el apartado 9 contiene una lista de las pruebas realizadas para comprobar el funcionamiento de la aplicación. Finalmente, el apartado 10 contiene las principales conclusiones del TFG y el apartado 11 cierra la memoria con el listado de referencias utilizadas.

2 Descripción General

2.1 Concepto y resumen

Este Trabajo de Fin de Grado es una idea de aplicación de carácter social. La idea de Sororidapp nace de la situación innecesaria de acoso y misoginia que viven mujeres de nuestro país y de todos los países del mundo día tras día, siendo acosadas en su vida diaria, pero sobre todo en la nocturna y cuando más solas y vulnerables se encuentran. Se pretende con esta aplicación proporcionar ayuda veloz y de confianza a las mujeres que lo necesiten, poniéndolas en contacto con otras mujeres de la zona, para que puedan ir en su ayuda y, si la situación lo requiriera, ponerse en contacto con el servicio de emergencias o los cuerpos de seguridad del estado. Asimismo, también cumple como mapa, y ayuda a las usuarias a encontrarse localizadas en todo momento, manteniéndolas informadas de su ubicación en tiempo real.

Con este proyecto se practican los conceptos de programación Android, tanto Java dominando aún más este lenguaje, como Kotlin aprendiéndolo desde cero; el diseño de interfaces de usuario para aplicaciones móviles, así como los conceptos esenciales de usabilidad, para desarrollarla de la mejor manera posible de cara al usuario final (tamaño de los botones, posición, colores distinguibles, iconos reconocibles, …).

Más concretamente, las funciones a realizar por la aplicación serán las siguientes:

- Poder pedir ayuda a las usuarias cercanas y, por otro lado, facilitar dar la ayuda desde dentro de la aplicación en un mapa-guía hasta la mujer en peligro.
- Poder realizar llamadas al servicio de emergencias desde la aplicación.
- Facilitar un mapa funcional para guiar en el camino a la usuaria que lo requiera.
- Mantener siempre a la usuaria en conocimiento de la dirección postal actual para que pueda indicar donde se encuentra.

Por otro lado, se ha realizado un estudio de mercado en forma de encuesta a más de 100 mujeres de distintos rangos de edad, en cuyas respuestas se han basado las decisiones tomadas y donde se ha plasmado en datos reales la necesidad incesante de un proyecto de este tipo.

Una usuaria de la aplicación deberá ser invitada por otra usuaria, a su vez anteriormente invitada, de forma que se pueda confiar en el correcto estado de la base de usuarias de la aplicación, puesto que, si tienen acceso a ella, es porque otra mujer le ha dado permiso. Asimismo, el permiso otorgado se puede revocar en cualquier momento, por lo que, si alguien hace un mal uso del sistema, se le puede echar de éste.

Al usar la aplicación, una mujer sabrá en todo momento la ubicación en la que se encuentra en tiempo real, indicándole la dirección postal en la que se encuentra, para poder ubicarse, indicárselo a algún conocido o a la policía.

También tendrá la posibilidad de llamar a emergencias con un solo clic, si así lo ve necesario, sin necesidad de marcar un número que, aunque sea sencillo, puede dificultar el acceso en según qué situaciones, como por ejemplo en un ataque de ansiedad o en estado de embriaguez.

Por otro lado, puede marcar si se encuentra disponible para ayudar a las personas cercanas, de forma que se mostrará en las aplicaciones de las demás usuarias. Dicho esto, se permitirá a las usuarias saber el número de mujeres que se encuentran en un radio cercano general y, de igual manera, las usuarias que se encuentren muy cerca de ella.

Si la usuaria se encuentra incomoda, pero no cree que requiera llamar a la policía o, por otro lado, no se fía de hacerlo por cualquier motivo y prefiere la ayuda y el apoyo ciudadano, podrá notificar a las usuarias cercanas que se encuentra en un potencial peligro, compartiendo su ubicación de forma que otras usuarias cercanas podrán ir en su ayuda inmediatamente.

Cuando alguna otra usuaria acceda a la notificación que recibe al haber alguien que pide ayuda en su zona cercana, entrará a un mapa donde verá en tiempo real su ubicación y la de la mujer que pide ayuda, así como la ruta más rápida a seguir para llegar hasta ella. Al mismo tiempo, a la mujer que pide auxilio se le mantendrá al tanto de la cantidad de mujeres que se encuentran viendo el mapa y que van en su ayuda.

Si la mujer en peligro considera que ya se encuentra a salvo o a resguardo, podrá cancelar la solicitud de ayuda, de forma que se notificará a las demás usuarias y se dejará de compartir su ubicación.

3 Estudio de mercado

Este apartado cuenta con dos secciones muy diferenciadas. En primer lugar, se muestran y analizan las encuestas realizadas a un grupo de mujeres para conseguir información para perfilar el proyecto. En segundo lugar, se analizan dos aplicaciones similares, que pueden ser competencia de Sororidapp y de las cuales se puede aprender sobre las necesidades de un sistema similar, así como de sus flaquezas y fortalezas.

3.1 Encuestas

Al inicio del proyecto, antes de empezar cualquier desarrollo, se realizó una encuesta [1] a mujeres de diferentes rangos de edad. La encuesta fue contestada por un total de 130 mujeres.

De todas ellas, pese a ser en gran medida jóvenes, en parte por el hecho de que muchos casos de acoso (como ya nombramos antes) ocurren de noche en ambientes de ocio, se han separado en las siguientes franjas de edad

Figura 1 Rangos de edades

De los cuales, el 44,6% se encuentran entre 16-18 y el 51,5% entre 19-25; cumpliendo así con la premisa de ser adolescentes/jóvenes adultas que presumiblemente tienen una vida nocturna más activa, así como, en el primer rango, imposibilidad de conducir, lo que las obliga a coger autobuses o a caminar hacia los sitios; situaciones en las que pueden ser asaltadas o acosadas.

A la pregunta de si se había sentido alguna vez acosada o perseguida, la inmensa mayoría contestaron que sí; dando a entender la necesidad de un cambio en esta sociedad, así como la gran posibilidad de utilidad de una aplicación como la planteada.

Figura 2. Porcentaje de Víctimas de acoso

Pese a que no todas han sido víctimas de una situación de este tipo, todas ellas coinciden en que sí se trata de una realidad.

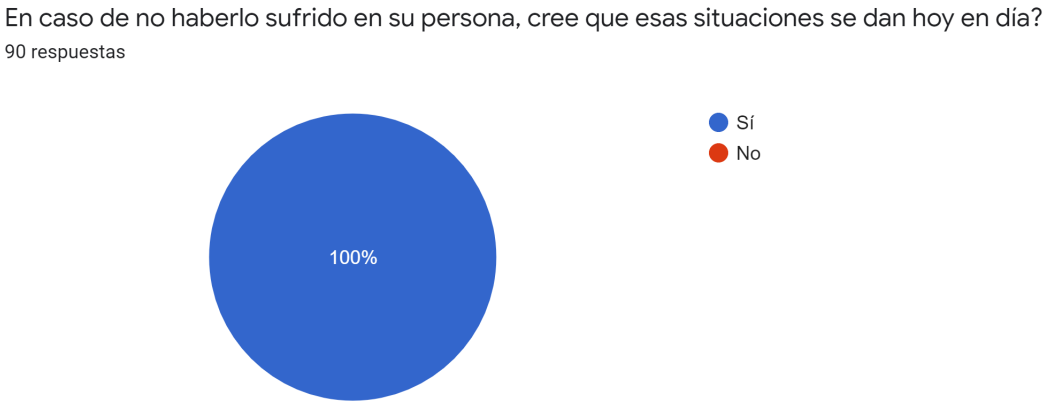

Figura 3. Porcentaje de Conocedoras de casos de acoso

En la pregunta de respuesta abierta sobre qué han hecho en esas situaciones, en términos generales se detona una impotencia y un miedo atroces; dado que la mujer promedio reacciona, de forma completamente entendible, huyendo o no haciendo nada, del pánico que siente en ese momento.

En la siguiente pregunta nos basamos para decidir parte del sistema de la aplicación, pues solo un 24,6% afirma que llamaría a la policía en una situación así y, lo que es más preocupante, un 18,5% de las encuestadas confirman que en esa situación no llamarían a la policía.

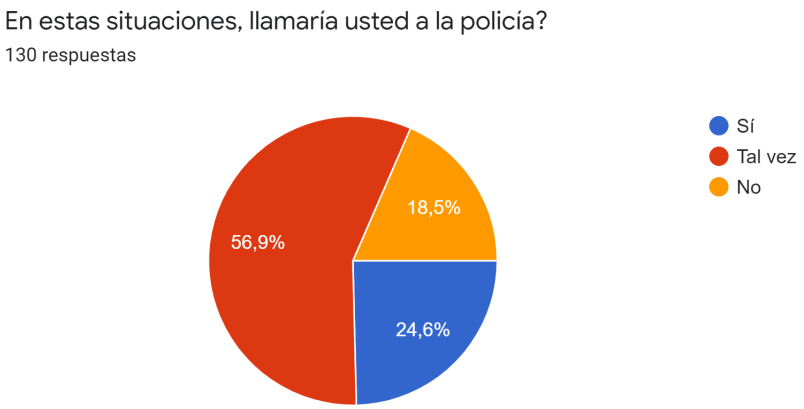

Figura 4 Porcentaje de Mujeres que llamarían a la policía

Estos datos nos indican que no podemos basar el funcionamiento general de la aplicación en la llamada a la policía; pues el número de usuarias a las que les resultaría útil es potencialmente del 24,6%.

Siguiendo con la encuesta se procede a dar una mínima descripción de Sororidapp:

"SororidApp es una aplicación de auxilio a mujeres que lo necesiten. La idea es que se pueda avisar a usuarios cercanos para que presten ayuda lo antes posible y, si es necesario, avisar a la policía."

Tras ella se realizan las siguientes preguntas respecto la idea, la utilidad y las aplicaciones competencia.

Para empezar, vemos en la siguiente figura que, en los términos generales, a las mujeres encuestadas no les importaría el hecho de que la mujer que venga en su auxilio sea una completa desconocida. Concretamente, al 93,1% de la población. De igual manera, a la siguiente pregunta de si iría en socorro de una completa desconocida, vemos que exactamente el mismo porcentaje estaría dispuesto a hacerlo.

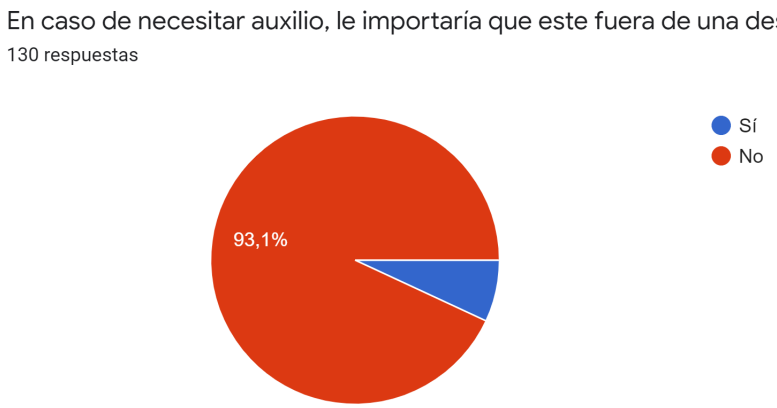

Figura 5. Porcentaje de mujeres dispuestas a ser ayudadas por desconocidas

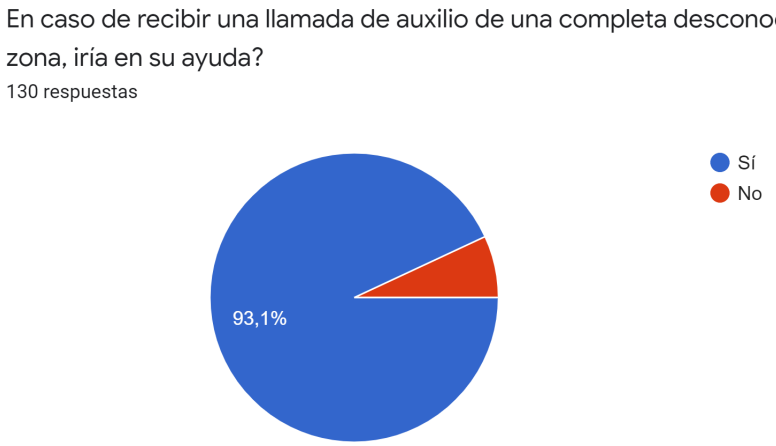

Figura 6. Porcentaje de mujeres que irían en auxilio de completas desconocidas

Esto da a entender que, en la aplicación, un sistema de "Quid pro quo" entre las usuarias tendría cabida y sentido, pues la misma cantidad de mujeres que aceptarían la ayuda de desconocidas, sería la de mujeres que están dispuestas a ir en la ayuda de otras. El hoy por ti mañana por mí planteado puede funcionar por mero sentido de la sororidad.

La siguiente pregunta y sus diferentes justificaciones plantea un punto de inflexión en la aplicación, pues la mitad de las mujeres encuestadas, el 46,2% de ellas, no se sentiría segura al usar la aplicación, si es consciente de que ésta está al alcance de hombres y la pueden usar de igual manera que las mujeres.

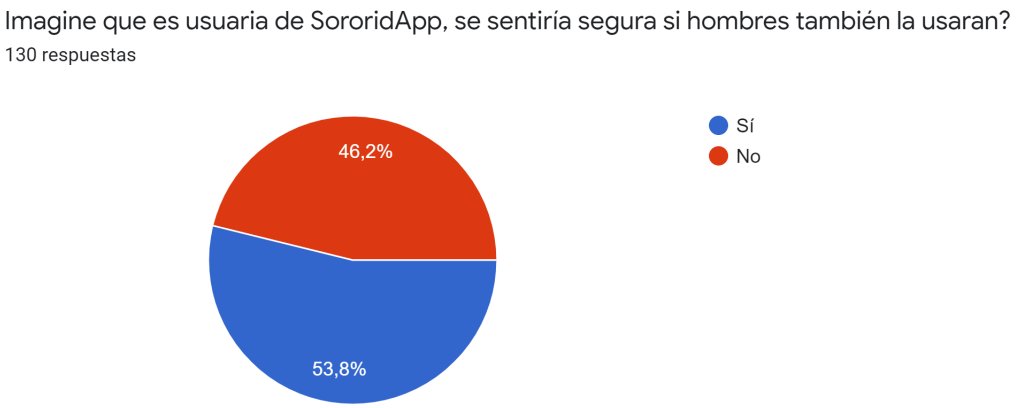

Figura 7 . Porcentaje de mujeres que se sienten seguras al usar la aplicación a la vez que hombres

Algunas de estas mujeres justificaron luego que esta negativa se debía a que no pueden saber si el hombre que la usa es un acosador también, que se pueda aprovechar de la información brindada por la aplicación para realizar sus abusos a usuarias desafortunadas.

Pero, por el contrario, en sentido general no les importaría que varones usaran Sororidapp para apoyar a las mujeres en auxilio, si saben que tienen un sello o una garantía de que éstos van con buena fe.

"Quiero aclarar que no me importaría que usasen la aplicación hombres si yo tuviese la total certeza de que se puede confiar en ellos y no tengo que tener miedo de que me puedan agredir"

"En la pregunta de si me sentiría segura si los hombres también la usarán entraríamos en términos dudosos, puesto que dependería del motivo por el cual se han descargado la aplicación. Puse que sí entendiendo que todos los de la aplicación van con buena fe a ayudar y no por intereses propios, sino sería distinta mi respuesta."

Por todo esto, he priorizado la importancia de controlar el acceso a la aplicación, para que en todo momento las usuarias tengan una seguridad de que las personas a las que le notifican o comparten la ubicación no van a suponer un peligro, ni va a ser peor el remedio que la enfermedad.

Por último, se les preguntó por la utilidad que le veían a una aplicación de este tipo, así como si conocían o utilizaban alguna similar, para conocer si realmente tendría cabida en el mercado una aplicación así o si debería ser desestimada por tener demasiada competencia o haber sobreestimado el público potencial.

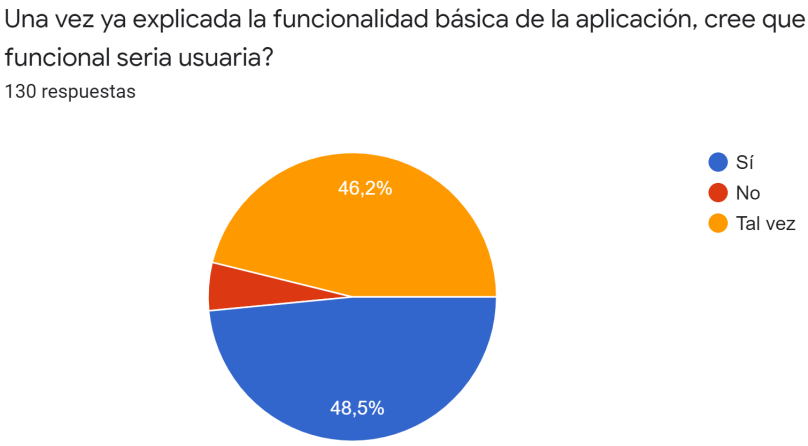

Figura 8. Encuesta sobre si usarían la aplicación en caso de salir al mercado

Figura 9. Pregunta sobre el conocimento de aplicaciones similares

Como podemos observar en la Figura 8, solo el 5,4% de las mujeres encuestadas se negaría a utilizar la aplicación, es decir, de las 130 mujeres encuestadas, solo 7 de ellas aseguran que no utilizarían la aplicación. Más allá de eso, observamos también que, de todas ellas, solo 13, el 10%, conocen aplicaciones similares, por lo que pudieran perder el interés de descargar y utilizar Sororidapp.

En la última, preguntamos de qué aplicación similar se trataba y de las 13 mujeres que afirmaron conocer alguna aplicación similar solo 2 recordaron el nombre de la aplicación. Las aplicaciones en cuestión fueron: Sister y AlertCops. Con esto procederé en el siguiente apartado, a analizar la competencia en estas dos aplicaciones y ver sus puntos débiles y fuertes respecto Sororidapp.

3.2 Estudio de mercado: Aplicaciones similares

3.2.1 AlertCops

La primera aplicación que se analizará será AlertCops [2] que se trata de una aplicación del Ministerio de Interior del Gobierno de España. Se trata de la aplicación de la Policía y de la Guardia Civil.

Figura 10. Imagen de alertcops, aplicacion de la policía

Utilidades:

- Chat con la Policía en caso de ser víctima o testigo de un delito. Respuesta inmediata.
- Avisos por zonas basados en los partes policiacos, para poder prevenir problemas.
- Capacidad de compartir tu ubicación con contactos seleccionados y los servicios de emergencia.
- Botón SOS que permite avisar de inmediato a las personas del apartado anterior; si se trata de una persona de un colectivo vulnerable, se provee de un nivel extra de seguridad de las fuerzas policiales cercanas.

Ventajas:

Línea directa con la policía a modo de SOS, chats, notificaciones y avisos de incidentes previos. Al ser una aplicación desarrollada por el Ministerio, se espera que en este ámbito sea superior a todas las alternativas.

Se encuentra adaptada para personas sordas e invidentes, y con traducción automática.

Desventajas:

No aprovecha para nada la base de usuarios para la ayuda según tu ubicación; solo puedes compartir tu ubicación o pedir ayuda a contactos previamente seleccionados, por lo que si te encuentras en una zona aislada, en un pueblo cercano en las fiestas por ejemplo, no será de mucha ayuda; aparte, depende directamente de la Policía que, a la vez de tener sus ventajas ya nombradas, es un problema en todas esa mujeres que confirman no confiar en estas instituciones de forma ciega, y que prefieren la ayuda de otras mujeres que han vivido sus mismas situaciones de acoso.

Por último, nombrar que no se trata de una aplicación centrada en situaciones de acoso misógino, sino en delitos de ámbito general y de todo tipo, por lo que potencialmente prestará menos atención a estas situaciones, o dejará funcionalidades sin implementar por ser una aplicación de uso más general.

Según su página de la Play Store [3] esta aplicación cuenta con más 500.000 descargas de entre las cuales los usuarios le han otorgado una valoración de 3.9 estrellas, por lo que se entiende que cumple debidamente con lo prometido. Pese a esto, no se trata de una competencia directa con Sororidapp, ya que solo una pequeña parte de esta trata lo mismo que mi aplicación.

3.2.2 Sister

Sister [4] se trata de una aplicación española de ayuda en situaciones de acoso a las mujeres, al igual que Sororidapp, desarrollada por Wave App S.L. Se autodenominan como una app de mujeres para mujeres.

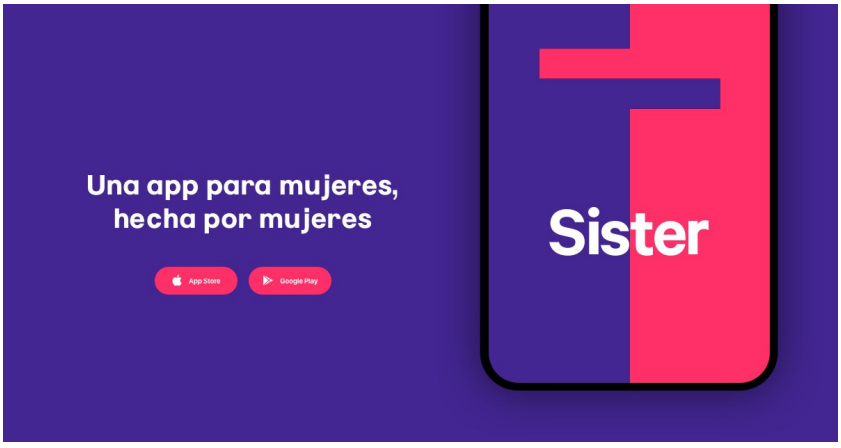

Figura 11 . Sister app

Utilidades:

- Elegir tus contactos de confianza y compartir tu ubicación en tiempo real.
- Seguir la ruta más segura, viendo las zonas más y menos seguras de tu ciudad.
- Activar una alarma disuasoria. Posibilidad de hacer sonar una alarma falsa para ahuyentar a los agresores.
- Pedir ayuda a tus contactos y, si quieres, a los servicios de emergencias.
- Registrar pruebas: tomamos audios e imágenes y los guardamos en la nube.
- Sister implementa un botón rápido de petición de ayuda con el botón de subida de volumen.

Ventajas:

Al ser una aplicación de mujeres, quizás entiendan mejor que yo la realidad pese a que haya hecho la encuesta a tantas voluntarias, pues ellas lo han vivido en sus propias carnes y eso les puede dejar mejor visión para desarrollar la aplicación. En materia más directa, esta aplicación cuenta con un mapa dinámico de ataques, para que las usuarias puedan evitar zonas donde se hayan realizado ataques previamente. Es una buena utilidad para prevenir situaciones peligrosas

Desventajas:

Nuevamente esta aplicación, desde mi punto de vista, peca de prevenir, pero no cuidar en el caso de algo espontáneo. Si bien puedes notificar guardianes, si recibieras un ataque directo de forma no esperada, no podrías pedir ayuda de forma rápida, más allá que a las autoridades que, pese a que a priori puede parecer suficiente, muchas mujeres no confían en ellos y preferirían no realizar la llamada de socorro. Por último, he de decir que las notificaciones se realizan por WhatsApp o SMS y no por la aplicación en si por lo que, si tuviera éstas silenciadas, cosa bastante habitual, no se darían cuenta de la llamada de socorro. Algunas usuarias, además, tienen quejas de que su uso es complicado; esto probablemente, debido a que se implementan muchas funcionalidades secundarias, que distraen de las más importantes.

Sister en Google Play [5] cuenta con más de 10.000 descargas, a priori, en el estado español, lo que es un número elevado, pero muy lejano a las 500.000 de AlertCops; y también tiene una cualificación general de 3,9 estrellas. Para finalizar, sí que habría que compartir mercado directo con Sister, pero su expansión por este no es extensa en demasía: no se trata de un superventas a batir para poder sacar al mercado Sororidapp.

4 Definición de requisitos

En este apartado se detallarán primero los requisitos funcionales del sistema, tras estos, se detallarán los requisitos no funcionales. Para finalizar el apartado se mostrará el diagrama de casos de uso.

4.1 Requisitos funcionales

4.1.1 RF-00 Iniciar sesión

RF-00	**Iniciar sesión**
Versión	1.0
Actores	Usuarios
Descripción	El usuario deberá iniciar sesión en la aplicación con su número de teléfono para poder usarla
Prioridad	Alta
Dependencias	No

Tabla 1 RF-00 Iniciar sesión

4.1.2 RF-01 Mostrar ubicación

RF-01	**Mostrar ubicación**
Versión	1.0
Actores	Sistema
Descripción	El sistema será capaz de mostrar la descripción de la ubicación del usuario mediante su dirección
Prioridad	Media
Dependencias	No

Tabla 2 RF-01 Mostrar Ubicación

4.1.3 RF-02 Mostrar Mapa

RF-02	**Mostrar mapa**
Versión	1.0
Actores	Sistema
Descripción	El sistema será capaz de mostrar la versión en blanco del mapa de Google mapas para el usuario
Prioridad	Media
Dependencias	No

Tabla 3 RF-02 Mostrar Mapa

4.1.4 RF-03 Mostrar número de personas Cercanas

RF-03	**Mostrar número de personas Cercanas**
Versión	1.0
Actores	Sistema
Descripción	El sistema será capaz de mostrar la cantidad de usuarios cercanos y muy cercanos de tu ubicación actual, que se encuentran disponibles.
Prioridad	Alta
Dependencias	RF-01 RF-04

Tabla 4 RF-03 Mostrar número de personas Cercanas

4.1.5 RF-04 Marcar disponibilidad

RF-04	**Marcar disponibilidad**
Versión	1.0
Actores	Usuario
Descripción	El usuario podrá marcar en todo momento si se encuentra disponible o no para ayudar al resto de usuarios
Prioridad	Alta
Dependencias	No

Tabla 5RF-04 Marcar disponibilidad

4.1.6 RF-05 Llamar al 112

RF-05	**Llamar al 112**
Versión	1.0
Actores	Usuario
Descripción	El usuario podrá llamar al 112 de forma rápida pulsando un botón.
Prioridad	Alta
Dependencias	No

Tabla 6 RF-05 Llamar al 112

4.1.7 RF-06 Pedir Ayuda a Usuarias

RF-06	**Pedir Ayuda a Usuarias**
Versión	1.0
Actores	Usuario
Descripción	El usuario podrá pedir auxilio a todos los demás usuarios de los alrededores que se encuentren disponibles
Prioridad	Máxima
Dependencias	RF-03 RF-04

Tabla 7 RF-06 Pedir Ayuda a Usuarias

4.1.8 RF-07 Dar Ayuda a Usuarias

RF-07	**Dar Ayuda a Usuarias**
Versión	1.0
Actores	Usuario
Descripción	El usuario podrá conceder la ayuda solicitada a una usuaria que la esté pidiendo en sus alrededores si éste se encuentra disponible
Prioridad	Máxima
Dependencias	RF-03 RF-04 RF-06

Tabla 8 RF-07 Dar Ayuda a Usuarias

4.1.9 RF-08 Buscar en el Mapa

RF-08	**Buscar en el mapa**
Versión	1.0
Actores	Usuario
Descripción	El usuario podrá utilizar un buscador para encontrar la dirección a la que se quiere dirigir.
Prioridad	Media
Dependencias	No

Tabla 9 RF-08 Buscar en el Mapa

4.1.10 RF-09 Marcar nueva ubicación en el mapa

RF-09	**Marcar nueva ubicación en el mapa**
Versión	1.0
Actores	Usuario
Descripción	El usuario podrá marcar en el mapa una nueva ubicación destino simplemente pulsando en ella.
Prioridad	Media
Dependencias	RF-02

Tabla 10 RF-09 Marcar nueva ubicación en el mapa

4.1.11 RF-10 Mostrar ubicación marcada

RF-10	**Mostrar ubicación marcada**
Versión	1.0
Actores	Sistema
Descripción	El sistema mostrará al usuario la dirección de la ubicación marcada previamente en el mapa como destino
Prioridad	Media
Dependencias	RF-02 RF-09

Tabla 11 RF-10 Mostrar ubicación marcada

4.1.12 RF-11 Centrar el Mapa en tu ubicación actual

RF-11	**Centrar el Mapa en tu ubicación actual**
Versión	1.0
Actores	Sistema, Usuario
Descripción	El mapa se abrirá centrado en la posición actual del usuario. Asimismo, el usuario podrá pulsar un botón para poder centrarla de nuevo en sí mismo.
Prioridad	Baja
Dependencias	RF-02

Tabla 12 RF-11 Centrar el Mapa en tu ubicación actual

4.1.13 RF-12 Invitar a Usuarias

RF-12	**Invitar a Usuarias**
Versión	1.0
Actores	Usuario
Descripción	El usuario podrá introducir números de teléfono de futuras usuarias para invitarlas a usar la aplicación.
Prioridad	Alta
Dependencias	RF-00

Tabla 13 RF-12 Invitar a Usuarias

4.1.14 RF-13 Cerrar sesión

RF-13	**Cerrar sesión**
Versión	1.0
Actores	Usuario
Descripción	El usuario podrá cerrar sesión en la aplicación siempre que ésta se encuentre abierta.
Prioridad	Media
Dependencias	RF-00

Tabla 14 RF-13 Cerrar sesión

4.1.15 RF-14 Borrar usuaria

RF-14	**Borrar usuaria**
Versión	1.0
Actores	Usuario
Descripción	El usuario podrá borrar su cuenta de la aplicación.
Prioridad	Baja
Dependencias	RF-00

Tabla 15 RF-14 Borrar usuaria

4.1.16 RF-15 Dejar de invitar a usuarias

RF-15	**Dejar de invitar a usuarias**
Versión	1.0
Actores	Usuario
Descripción	El usuario podrá dejar de dar permisos a una usuaria a la que ya había invitado con anterioridad
Prioridad	Alta
Dependencias	RF-12

Tabla 16 RF-15 Dejar de invitar a usuarias

4.1.17 RF-16 Control de Sesiones

RF-16	**Control de Sesiones**
Versión	1.0
Actores	Sistema
Descripción	El sistema solo dejara utilizar la aplicación a usuarios que inicien sesión con un número invitado por otro usuario
Prioridad	Máxima
Dependencias	RF-11 RF-00

Tabla 17 RF-16 Control de Sesiones

4.1.18 RF-17 Generar Caminos

RF-17	**Generar Caminos**
Versión	1.0
Actores	Sistema
Descripción	El sistema creara el camino optimo entre la ubicación actual y la ubicación destino, independientemente del origen de esta ubicación
Prioridad	Media
Dependencias	RF-02, RF-09/RF-08

Tabla 18 RF-17 Generar Caminos

4.2 Requisitos no funcionales

4.2.1 Requisito de Rendimiento

La aplicación deberá funcionar de forma fluida y eficiente, dado que un tiempo de demora elevado resultará en una experiencia insatisfactoria y negativa para el usuario.

4.2.2 Requisito de Fiabilidad

La aplicación dará siempre datos fiables en los cuales el usuario final puede confiar, datos como el número de usuarios o la ubicación aproximada no serán dados de forma errónea.

4.2.3 Requisito de Disponibilidad

La aplicación ha de estar disponible para el usuario final siempre que éste disponga de los medios necesarios para su funcionamiento correcto. Cuando el dispositivo móvil del usuario cuente con conexión a internet, así como de un sensor GPS, la aplicación ha de estar completamente disponible.

4.2.4 Requisito de Seguridad

La aplicación mantendrá los datos de los usuarios, como el número de teléfono o las ubicaciones de forma segura en las bases de datos, siempre que el usuario no indique lo contrario, cumpliendo así con la nueva Ley Orgánica de Protección de Datos y Garantía de Derechos Digitales (LOPDGDD, 2018).

4.2.5 Requisito de Exclusividad

La aplicación se podrá usar solo por usuarios verificados anteriormente por otros mediante las invitaciones, garantizando así la seguridad y la confianza de las usuarias respecto el resto de ellas.

4.3 Diagrama de Casos de Uso

Para el siguiente diagrama de casos de uso he utilizado la herramienta en línea Visual Paradigm Online [6] para representar los procesos de acciones que sigue el sistema y el usuario a lo largo de la utilización de la aplicación.

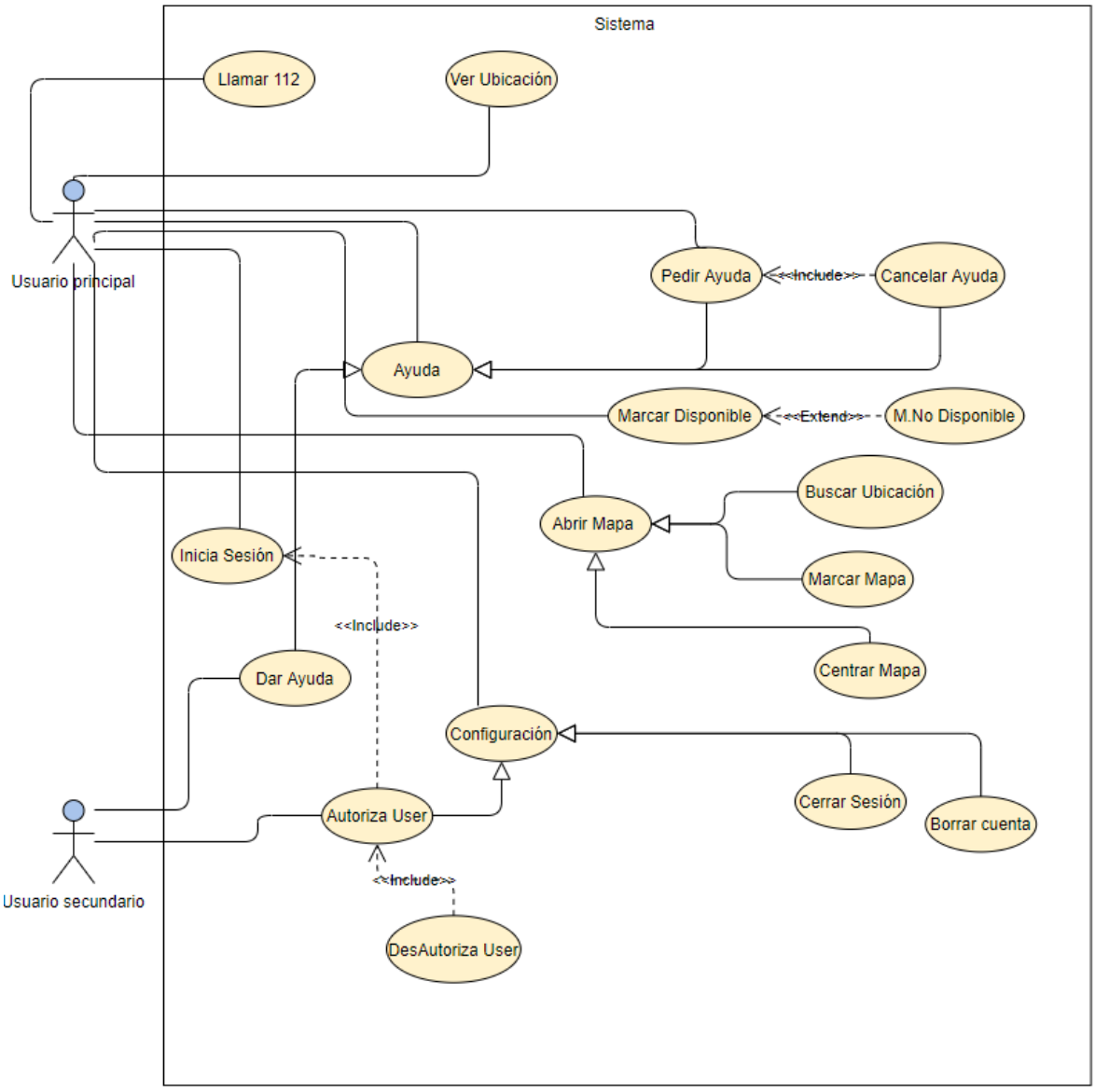

Figura 12 Diagrama de casos de uso

5 Tecnologías Utilizadas

A lo largo de este apartado se proporciona una explicación de la estructura utilizada en el proyecto, el patrón MVC y, a continuación, se ven los aspectos más importantes de las librerías, tanto de Firebase como de Google, y de las Kotlin Coroutines.

El proyecto de Sororidapp se ha desarrollado en Android Studio con el lenguaje de programación Java como lenguaje principal y contando con el lenguaje Kotlin cuando es necesario. Dentro del proyecto se ha implementado la arquitectura MVC estándar donde se separa el funcionamiento de éste en Modelo-Vista-Controlador, donde las diferentes layouts en los distintos XML hacen de vista, mientras que hacen de controlador las diferentes activities y fragments y, por último, el modelo lo tendremos en el RealTime Database de Firebase.

Este patrón presenta la interfaz que ve el usuario y con la que puede interactuar, de la forma que se recoge en el controlador, que recibe datos del modelo de datos de Firebase y los actualiza cuando el controlador lo indica, actualizando después la vista en consecuencia, tal y como se explica en la siguiente imagen [7].

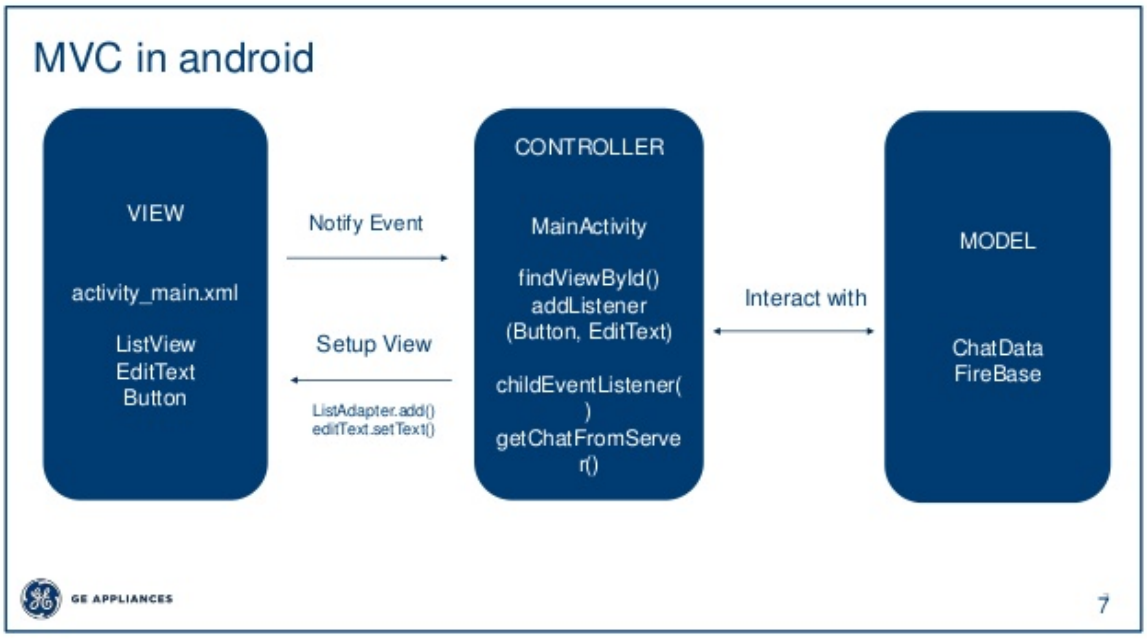

Figura 13 Patrón MVC

En el proyecto se han utilizado diferentes librerías de Google y Firebase para el funcionamiento global de la aplicación. Éstas son: Firebase-Analytics, Firebase-Database, Firebase-Auth, Firebase-SafetyNet, Google-Maps y Google-Locations. Por último, se utilizan también las Kotlin-Coroutines.

5.1 Firebase

5.1.1 Firebase Auth

Es la librería utilizada para gestionar los usuarios de la aplicación, Firebase proporciona funcionalidades de inicio de sesión y gestión de cuentas con variedad de credenciales diferentes de las cuales, en base a las diferentes necesidades del sistema, se ha seleccionado el log-in con número de teléfono y verificación por SMS. En la figura 14 podemos ver la lista completa de proveedores de sign-up de Firebase.

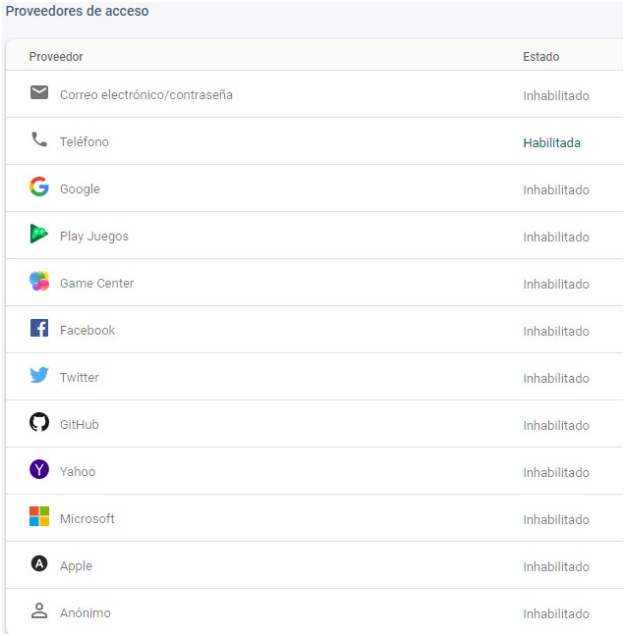

Figura 14 Lista de proveedores de sign-up de Firebase

La selección del servicio de proveedores de acceso mediante el teléfono ha sido elegida de entre todos los demás debido a que es una forma que dificulta en gran medida la realización de cuentas múltiples o cuentas falsas, así como que facilita el rastreo de cuentas que puedan poner en riesgo el sistema. Además, como en el uso a diario de la aplicación serán las usuarias las que inviten a otras potenciales usuarias de su círculo de confianza al uso de la aplicación, éste se puede gestionar de forma sencilla si el acceso se realiza mediante el número de teléfono.

Otra característica que tiene sentido de esta forma de acceso es que el propio proveedor de Firebase detecta el número de intentos de inicio de sesión y los limita por número de teléfono; si éste intenta entrar más de 5 veces en 4 horas, por lo que protege al sistema de un ataque que intente colapsarlo en base a infinidad de intentos de inicio de sesión desde números automatizados.

Este servicio generará un código de verificación que se enviará al número introducido en forma de SMS. Normalmente, si el dispositivo lo permite, se auto introducirá de forma que verificará al usuario sin necesidad de que este haga nada más que pulsar el botón final o, también, si es un usuario asiduo, es posible que se auto verifique sin necesidad de que se envíe el SMS en cuestión. Si ninguna de estas dos opciones automáticas ocurriera, el usuario recibirá el SMS con el código de verificación de 6 dígitos y deberá introducirlo en el panel que se indica en pantalla; si el código es correcto se validará el usuario.

Con esta tecnología se han completado los requisitos funcionales, RF-00 Iniciar Sesión, RF-13 Cerrar Sesión y RF-14 Borrar cuenta.

5.1.2 Firebase RealTime DataBase

Como servicio de base de datos utilizaremos el servicio de Firebase RealTime DataBase, que nos permite tener una base de datos en la nube de fácil acceso a todos los usuarios de la aplicación.

El servicio de Firebase cuenta con dos sistemas de bases de datos diferentes: la RealTime DataBase y el FireStore DataBase, siendo este último el más reciente en ser añadido. En la página principal de Firebase de comparativa de Bases de datos [8] podemos encontrar las similitudes y diferencias de los dos sistemas, así como una pequeña encuesta donde según tus respuestas puedes ver si se trata una mejor opción la base de datos en tiempo real o el FireStore.

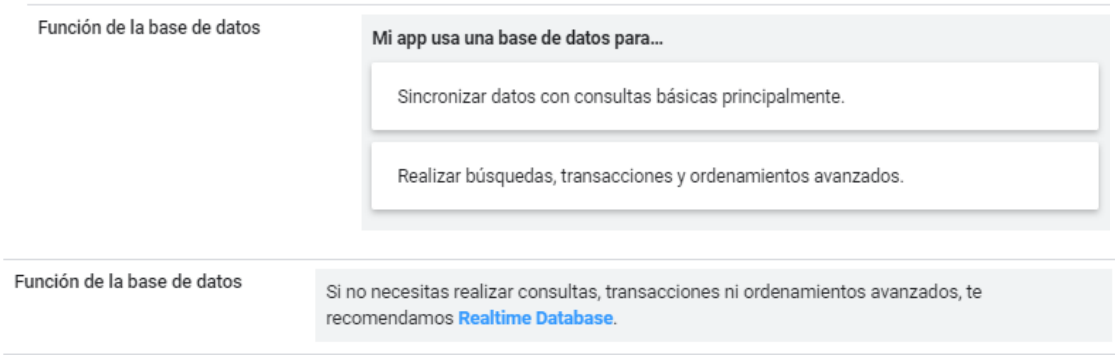

Figura 15 Ejemplo de pregunta y respuesta del test de Firebase DB

A continuación, detallo las características, ventajas o desventajas, que han llevado a la decisión del RealTime DataBase sobre el FireStore Storage para Sororidapp, todas ellas obtenidas de la página anteriormente nombrada.

RealTime DataBase:

- Almacena datos como un gran árbol JSON. De forma que estos datos son muy fáciles de almacenar si son datos simples, como es el caso.

- Soporte sin conexión para clientes iOS y Android, de forma que las actualizaciones hechas sin conexión se realizarán en la base de datos final de forma automática en cuanto el usuario cuente con la conexión necesaria.

- Las consultas pueden acceder a los datos en cualquier nivel de detalle, hasta valores de nodo de hoja individuales en el árbol JSON. De forma que se puede acceder directamente al valor deseado, si se lleva un buen control de índices.

- Latencia extremadamente baja, es la opción ideal para sincronizar estados con frecuencia. Esta característica es primordial para el uso y la importancia que tienen los accesos rápidos en la aplicación.

- Se cobra solo por ancho de banda y almacenamiento, pero con una tarifa mayor.

FireStore Storage:

- Almacena datos como colecciones de documentos, lo cual podría dificultar el uso de datos simples respecto al caso de la RealTime DataBase.

- Las consultas siempre deben mostrar documentos completos. Esto tiene su problemática cuando queremos acceder a un dato concreto con un índice concreto, como se realiza en Sororidapp.
  -
- Se cobra principalmente por operaciones ejecutadas en la base de datos (lecturas, escrituras y eliminaciones) y, con una tarifa menor, por ancho de banda y almacenamiento.

En conclusión, se puede afirmar que, pese a ser las dos opciones perfectamente válidas y habiéndose podido desarrollar de una manera similar con FireStore Storage, me he decantado por la opción original de Firebase, RealTime DataBase, debido a que el consumo de los accesos, muy numerosos pero de poca cantidad de datos, son más accesibles en cuanto a tiempo de respuesta así como en cuanto a precio, característica que se analizará de forma más exhaustiva más adelante, respecto a sus alternativas en la otra opción.

Figura 16 Respuestas sobre el proyecto a la encuesta de Firebase DB

Podemos observar que, en base a la encuesta ya mencionada, se trata de un acierto la elección de la RealTime DataBase.

Por último, dentro de la misma RealTime DataBase podemos definir unas normas de seguridad que regulan quién puede leer o escribir en la base de datos, y con una sencilla configuración podemos definir que solo los usuarios verificados por móvil (el servicio que utilizamos), puedan realizarlo.

```
"rules": {
      ".read": "(auth != null) && (auth.provider == 'phone')",
      ".write": "(auth != null) && (auth.provider == 'phone')"
      }
```

Código 1. Ejemplo de Reglas de Firebase RealTime DataBase

Pese a todo, si tratáramos de escalar Sororidapp más allá de una primera versión prototipo, que es el caso, a una versión definitiva donde se pudiera contar con más de 200.000 usuarias de forma simultánea, quizás se debería contemplar la idea de migrar la aplicación a FireStore, pues en estos números se debe dividir la base de datos de tiempo real en diversas de ellas para poder seguir funcionando, mientras que la escalabilidad en FireStore Storage es automática y cuenta con hasta 1 millón de conexiones simultáneas, número que se planea aumentar en el futuro, sin necesidad de desdoblar el servidor.

5.1.3 Firebase SafetyNet

Un servicio de App Check, como se dice en la página de Firebase App Check [9], es lo siguiente:

> *"App Check funciona junto con otros servicios de Firebase para ayudar a proteger sus recursos de backend contra abusos, como fraudes en la facturación o phishing. Con App Check, los dispositivos que ejecutan su aplicación utilizarán una aplicación o un proveedor de certificación de dispositivo que certifique uno o ambos de los siguientes:*
>
> *Las solicitudes se originan en su aplicación auténtica*
>
> *Las solicitudes se originan en un dispositivo auténtico y sin manipular"*

App check tiene diferentes proveedores, según sea la plataforma en la que se vaya a requerir:

- En Android el proveedor de certificación será SafetyNet, razón por la que es utilizado, pues ésta es la plataforma de Sororidapp.
- En iOS se cuenta con DeviceCheck o App Attest.
- En Aplicaciones Web, se utilizará Recaptcha v3.

En definitiva, es una capa más de seguridad para la aplicación y su ecosistema que, pese a no ser un "blindaje" definitivo, sí que puede parar muchos intentos de ataque, como el phishing ya mencionado y que es necesario tener en cuenta. Éste se basa en la certificación de los proveedores (SafetyNet), para confirmar la autenticidad de la aplicación que está realizando la llamada al servicio en cuestión, en este caso el RealTime DataBase.

En resumen, el AppCheck es a la veracidad de la aplicación, lo que la autenticación de usuarios a estos; con el esfuerzo redoblado de estas dos utilidades se consigue una gran seguridad tanto de la aplicación, como de sus datos o sus usuarios.

5.2 Google

5.2.1 Google Maps SDK

En Sororidapp se dispone de mapas en diversos escenarios, ya sea para uso personal en busca de guía o en el momento de ayudar a una persona en peligro. Estos mapas vienen proporcionados por el SAAS[1] de Google Maps SDK for Android [10].

[1] Software As A Service

Este software proporciona no solo todos los mapas estáticos con los que cuenta Google Maps, sino que también incluye la implementación y el control de los gestos, así como de las interfaces básicas del mismo.

Como bien se indica en la documentación del SDK, cada llamada a un mapa se deberá realizar en SupportMapFragment, y cualquier zoom, movimiento o cambio de capas no repercutirá en una nueva carga de mapa. En cuanto al gasto, para el uso que se le da en Sororidapp, la librería consta de un uso completamente gratuito independientemente del uso. Sin embargo, si en versiones posteriores de la aplicación se decidiera, por ejemplo, implementar el uso de StreetView, se debería tener en cuenta el uso de este en la factura generada por el SDK, pues no se encuentra incluido como Mapa Nativo, que son los que cuentan con esta gratuidad.

Sobre este tipo de mapas se podrán trazar rutas o caminos, así como poner diferentes pines de forma totalmente gratuita, sin necesidad de actualizar el mapa o acceder a otro tipo de éstos.

Para el uso de esta función se ha habilitado una clave API para poder limitar el uso de este SDK en el momento en que se requiera, debido a que se haya introducido alguna característica con coste y se deba controlar el uso para evitar que terceros se aprovechen de la cuenta de Sororidapp para su propio beneficio.

Esta clave API será limitada al uso en aplicaciones Android. Para restringir el uso se utilizará el nombre del "package" de la aplicación, junto con la huella digital SHA-1 del proyecto. Gracias a este servicio, se da funcionamiento al requisito funcional RF-02 Mostrar mapa.

5.2.2 Google Geocoding API

API de Geocoding de Google es la encargada de convertir las direcciones postales en coordenadas y viceversa (Reverse Geocoding); permitiendo así tener una versión humanamente legible de las coordenadas obtenidas del GPS similar a utilizar un buscador para encontrar una ubicación en un mapa.

El uso de esta API se realiza mediante llamadas HTTPS, y serán realizadas en la aplicación tanto para el uso de Geocoding como para el Reverse Geocoding. Esta llamada será siempre de la siguiente forma:

"https://maps.googleAPIs.com/maps/API/geocode/outputFormat?parameters"

Donde en "outputFormat" se declarará el formato de salida de la información recibida de la llamada realizada que puede ser tanto XML, como JSON, siendo este último el utilizado en el caso de Sororidapp, así como el recomendado en la documentación de la API. "parameters" por el contrario, será diferente según el uso que se le esté dando a la llamada Si se trata de Geocoding, se tratará de la dirección de la cual se quiere obtener información y si se tratara de una llamada de Reverse Geocoding, este parámetro constaría de la latitud y la longitud del punto a tratar, ambas se muestran en el ejemplo siguiente, obtenido de la documentación de Geocoding API [11]

Ejemplo de geocoding:
https://maps.googleAPIs.com/maps/API/geocode/json?address=1600+Amphitheatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY

Ejemplo de reverse geocoding:

https://maps.googleAPIs.com/maps/API/geocode/json?latlng=40.714224,-73.961452&key=YOUR_API_KEY

En SororidApp, como ya se ha comentado anteriormente, se utilizarán ambos servicios de forma independiente. En el caso del Geocoding, menos importante, se utilizará para dar la funcionalidad del requisito funcional RF-08 Buscar en el mapa, implementado con la respuesta de esta API el movimiento del mapa al punto buscado por el usuario con una dirección.

En el caso del Reverse Geocoding, se utilizará en la pantalla principal de la aplicación, donde sin necesidad de entrar en el mapa, se mostrará la descripción de la ubicación donde se encuentra el usuario. Esto, realizando una llamada a esta API con las coordenadas obtenidas del GPS del dispositivo móvil del usuario, completando así el requisito funcional RF-01 Mostrar Ubicación. Además, al marcar el usuario una ubicación en el mapa, la dirección de ésta será mostrada por pantalla, de forma que esta API se utiliza también para suplir las necesidades del RF-10 Mostrar Ubicación Marcada.

5.2.3 Google Directions API

Directions API [12] proporciona soporte a rutas tanto caminando, como es el caso en Sororidapp, en bicicleta o conduciendo, mediante una llamada HTTPS. La llamada soporta el uso de waypoints, que consisten en puntos indicados por los que el camino respuesta final ha de pasar. Pese a no ser utilizados en el proyecto, sí que pueden ser interesantes en el futuro si se quiere implementar alguna herramienta con estas características como pudiera ser llegar a casa por calles iluminadas, por lo que al introducir en la llamada los puntos de las calles iluminadas, el camino deberá pasar por ahí de forma obligatoria.

La llamada HTTPS es similar a las de la API, anterior siguiendo el formato siguiente:

https://maps.googleAPIs.com/maps/API/directions/outputFormat?parameters

"outputFormat", al igual que en el caso anterior podrá ser tanto XML como JSON, siendo este último el recomendado y el utilizado en todas las llamadas a esta API en Sororidapp. "parameters", por el contrario, en este caso se tratará del origen y el destino del camino solicitado, pudiendo ser introducido en forma de coordenada, como se realiza en Sororidapp, dirección o Id de ubicación. Además de otros parámetros de configuración de opciones de tráfico y similares se deberá indicar el modo en que el camino será solicitado: en este caso al ser caminando, indicaremos "walking".

Esta llamada devolverá un JSON de donde podremos obtener una lista de puntos a seguir para generar el camino deseado hasta el punto destino marcado. El camino se realizará de la forma más eficiente respecto al tiempo de viaje, es decir, la forma más rápida de realizar el camino será la que se devuelva en la llamada. Sin embargo, también se tendrán en cuenta otros factores, como la distancia total recorrida en el camino o el número de turnos, para dar con la respuesta optima a la solicitud.

En la ruta proporcionada se podrá encontrar organizada de la siguiente manera:

Para cada ruta tendremos diferentes piernas[2] que consistirán en la ruta desde la ubicación hasta un objetivo. Las rutas sin waypoints consistirán en una única pierna mientras que las que si tenga waypoints tendrán una pierna por cada waypoint al que tengan que asistir. Por cada pierna habrá a su vez diferentes pies[3], que serán los pasos a seguir para realizar el camino de forma correcta, corresponden con lo que el GPS por voz te diría, ordenes como "gire a la derecha" o "continúe recto hasta X". Por último, por cada pie, se contará con diferentes puntos, los cuales se utilizarán para crear la Polyline en el mapa, con el camino que puede seguir el usuario.

Con esta librería se da respuesta a los requisitos funcionales de generación de caminos, es decir, el RF-17 Generar Caminos.

5.3 Kotlin Coroutines

5.3.1 Kotlin

Kotlin es el lenguaje de programación oficial establecido por Google para el desarrollo de aplicaciones Android, ganando popularidad y funcionalidades rápidamente según pasa el tiempo. Pese a ser ideado para el desarrollo Android, también es utilizado para el desarrollo de iOS, aplicaciones web y multi-plataforma

Es un lenguaje de programación estático que funciona sobre una JVM[4]. Gracias a esto, una de las principales características de Kotlin es la compatibilidad directa que tiene con el lenguaje de programación Java, pudiendo cada uno ejecutar código del otro lenguaje. Esto nos permite realizar librerías o funciones Kotlin indistintamente en el proyecto. Tal es la compatibilidad de estos lenguajes, que el propio Android Studio cuenta con su propio traductor de Java a Kotlin.

[2] Termino empleado en la documentación oficial, diferentes pasos dentro de una ruta

[3] Termino empleado en la documentación oficial, diferentes pasos dentro de una pierna

[4] Java Virtual Machine, Máquina Virtual de Java

En general, el uso de Kotlin reduce el peligro de generar *NullPointerExceptions,* pues Kotlin tiene una muy buena gestión de las referencias nulas. Tanto es así en el momento de la declaración, donde se puede declarar la variable con la posibilidad de ser nula añadiendo el carácter "?" en su tipo de variable, como se muestra en el siguiente ejemplo, donde la variable activity tanto puede ser de tipo Activity como nula.

```
private var activity: Activity? = null
```

Código 2. Ejemplo de inicialización de variable con posibilidad de ser nula

Por otro lado, se tiene en Kotlin un modo de acceso seguro a variables, que garantiza que no se generaran excepciones de las ya nombradas, pues consiste en que, al acceder a ésta, solo ejecuta la acción si no se trata de un nulo, es decir, facilita la posibilidad de que las variables sean nulas. De igual manera que en la inicialización para utilizar este acceso seguro, se empleará el carácter "?" como en el ejemplo a continuación, donde la variable polyline solo se configurar al color en cuestión si no es nula. Si lo es, no se configurará, pero tampoco lanzará una *NullPointerException.*

```
lineOptions?.color(Color.MAGENTA)
```

Código 3. Ejemplo de acceso a variable con posibilidad de ser nula

5.3.2 Coroutines

Las *coroutines* de Kotlin son un aspecto clave en éste, y el motivo por el cual se utiliza el lenguaje en el proyecto: se trata de funciones asíncronas que administran tareas largas sin bloquear el *thread* principal. El uso de estas funciones nace de que la alternativa tradicional, el **AsynkTask,** ha pasado a ser *deprecated* u obsoleta, con lo que su uso no es recomendable.

En resumen, una corrutina es un patrón de diseño que proporciona Kotlin para las acciones asíncronas pesadas. Éstas proporcionan la posibilidad de elegir el hilo en el que se ejecuta la tarea, evitando así bloquear hilo principal de la aplicación; permiten ejecutar infinidad de tareas en el subproceso seleccionado y facilitar la escritura de código pues permiten escribir código asíncrono de forma secuencial, como si se tratara de código, síncrono.

Como bien se indica en la página de las Corrutinas [13], en estas se utilizan despachadores para determinar qué hilos las ejecutan. Estos despachadores son los tres siguientes:

- Dispatchers.Main, se trata del despachador que ejecuta la corrutina en el hilo principal de la aplicación. Solo se debe usar este despachador para interactuar con la IU o realizar trabajos muy rápidos y de poca carga pues entro caso se puede bloquear la aplicación entera.

- Dispatchers.IO, este despachador está optimizado para realizar E/S de disco o red fuera del subproceso principal. Las llamadas HTTPS se deberán realizar con este despachador para no bloquear el hilo principal.

- Dispatchers.Default, este despachador está optimizado para realizar trabajo que usa la CPU de manera intensiva fuera del subproceso principal. Algunos casos prácticos de ejemplo son clasificar una lista y analizar JSON.

En Sororidapp se utilizará principalmente el despachador de entrada y salida, es decir, el Dispatchers.IO.

Se podrán ver ejemplos de las corrutinas implementadas más adelante, en el apartado 7 de implementación del proyecto.

6 Análisis de costes y ganancias

En este apartado se analizarán los costes económicos de las librerías utilizadas y, a continuación, se plantearán las opciones de monetización de Sororidapp (ingresos económicos) de las que se disponen.

6.1 Análisis de coste

De las librerías explicadas con anterioridad, las que cuentan con un precio mayor de 0€ son las siguientes: Maps SDK, Geocoding API, Directions API, Firebase Realtime DataBase, FirebaseAuth como se muestra en la siguiente imagen de la tabla de costes de la plataforma de Google Cloud.

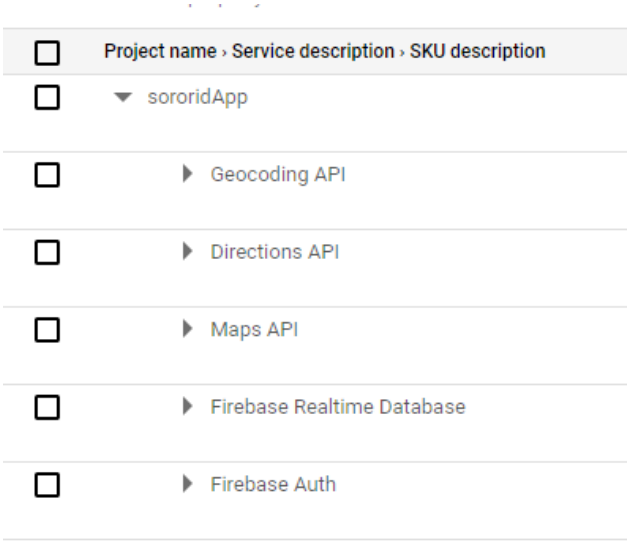

Tabla 19 Tablas de costes de Google Cloud platform

De entre éstos se deben diferenciar entre los servicios de Firebase, que cuentan con su propio plan de gastos, que se explicará más adelante, y las API de Google Cloud, que se analizarán a continuación.

Todas estas APIs tienen un sistema de pago por uso [14], según se hagan más llamadas a la librería se cobrarán, no cuentan con ningún tipo de nivel de uso gratuito, pero sí que contienen funciones que no son de pago. Este último es, por ejemplo, el caso de Maps SDK, que, pese a tener opciones de pago como el *Streetview* ninguna de ellas es utilizada dentro de Sororidapp en la actualidad, por lo que el uso de esta librería resulta en un coste nulo. Por otro lado, las llamadas a las otras dos API de Google, Geocoding y Directions, sí serán de pago y acarrearán un gasto.

- GeoCoding API: cuenta con un solo tipo de librería, cuyo coste está dividido en dos niveles diferentes según el uso de esta, si se realizan muchas llamadas a la librería (más de 100.000), el precio por unidad de éstas bajará.

- Directions API: en este caso se cuenta con dos tipos de librería Directions y Directions Advanced, siendo esta última, lógicamente, más cara. Como en Sororidapp solo se usarán las básicas no es necesario preocuparse por este segundo nivel. De igual manera que en la librería anterior, el precio se divide en dos niveles según el uso.

A continuación, se muestran las tablas de precios de las librerías anteriores.

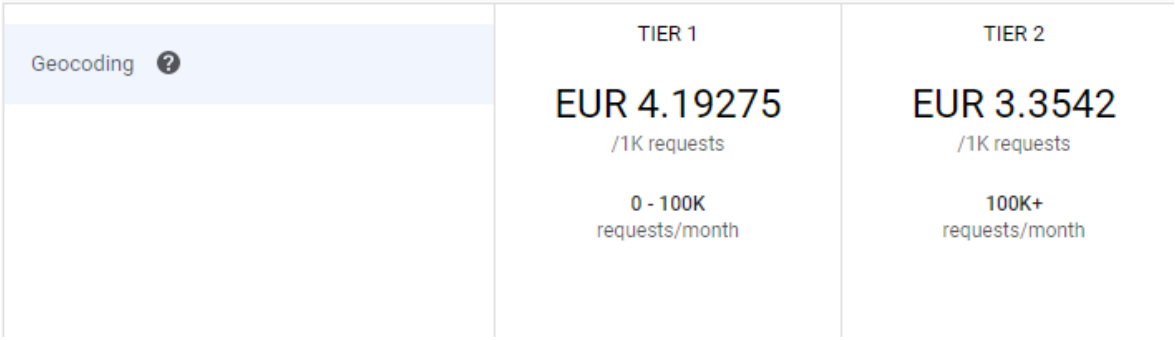

Figura 17 Tabla de precios de la API Geocoding

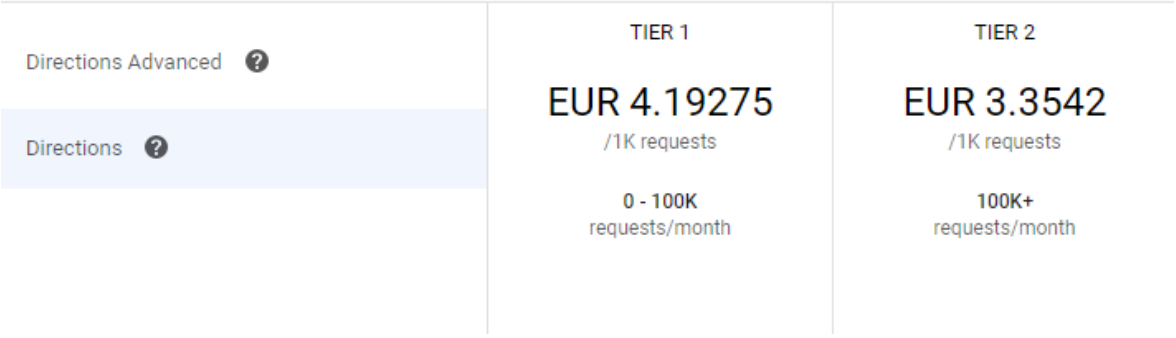

Figura 18 Tabla de precios de la API Directions

Con esta información parece inviable realizar una aplicación "independiente", pues los gastos podrían ser infinitamente mayores que los beneficios. Por suerte, se cuenta con 200$ de créditos cada mes mientras tengas en tu proyecto una API de mapas. Desde Google indican que la mayoría de las veces este crédito sobra para paliar los gastos ocasionados por las librerías. Aparte de este crédito, al iniciar el proyecto se pudieron solicitar 300$ de créditos extra a caducar en 3 meses (estos créditos caducaron durante el desarrollo de la aplicación).

Por último, se tiene la posibilidad de pedir un permiso de aplicación sin ánimo de lucro o de bien general [15], con lo que Google otorgaría un 3º crédito extra para reducir los gastos al mínimo para este tipo de propósitos. Este crédito sería de 250$, a sumar a los 200$ anteriores, se contaría con un total de 550$ al mes para el sufragio de los gastos generados por estas librerías.

Durante el desarrollo de la aplicación, estos 200 dólares de crédito han sido más que suficientes para no significar ningún gasto.

En las funciones de Firebase se cuenta con dos planes de uso [16]: el plan Spark, sin cargo y con limitación de uso, para iniciarse en el uso de las funciones y en proyectos "de prueba", y el plan Blaze, que partiendo del nivel gratuito del plan Spark, al acabar éste, se empezará a cobrar por uso, con un precio variable según la función. De los dos planes, el único compatible con el uso de Google Cloud Platform es el plan Blaze, por lo que en este proyecto el plan es, de forma obligatoria, el plan Blaze.

En cómputo global, de las diferentes funciones contamos con los siguientes niveles gratuitos y con un precio a partir de estos niveles:

- Firebase Auth: De esta característica se cuenta con 20.000 verificaciones mensuales gratuitas, la mitad en Estados Unidos, Canadá y la India y el resto para el resto de los países. A partir de éstas, se cobrarán a un valor de 1 céntimo de dólar las primeras y 6 céntimos de dólar las segundas. Hay que tener en cuenta que solo son computadas las verificaciones correctas y finalizadas.

- RealTime DataBase: Para la base de datos contamos con 200.000 conexiones simultáneas de forma gratuita. Por lo único que se cobrará en este campo es por el volumen de carga y descarga, siendo el precio de éstos de 5 dólares por GB de datos almacenados a partir del segundo, pues el primero es el nivel gratuito del plan Spark; para la descarga, se contará con 10 GB de descarga gratuitos por mes, después de éste el precio será de 1 dólar por GB excedido.

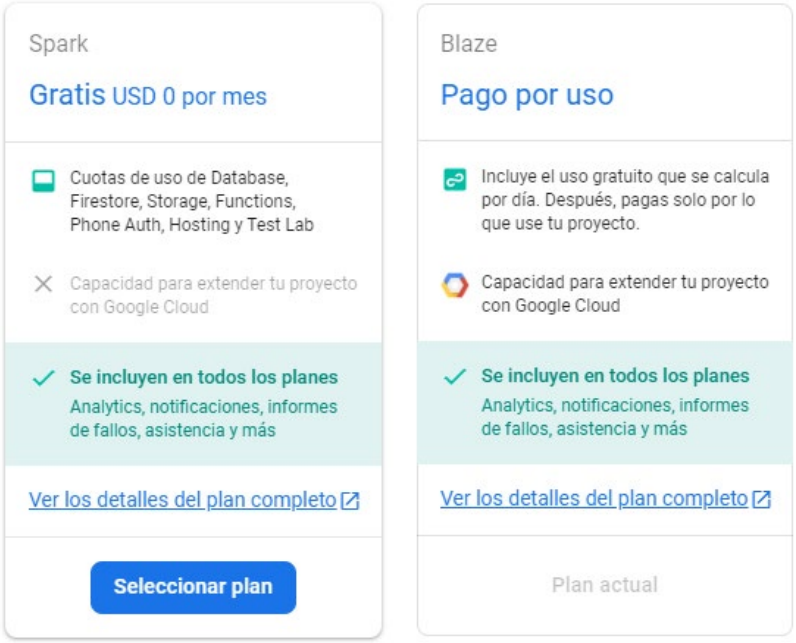

Figura 19 Imagen resumen de las diferencias entre los planes de pago de Firebase

6.2 Opciones de monetización

Las opciones de monetización de una aplicación de este tipo pueden resultar un dolor de cabeza mayor del que pudiera parecer, pues al ser una aplicación de uso en caso de necesidad, implementar publicidad, que normalmente se trata de una forma bastante intrusiva, puede condenar a la aplicación a ser completamente inútil.

Por otro lado, la alternativa de convertirla en una aplicación de pago puede llegar a ser contraproducente, pues puede reducir el número de usuarias interesadas, lo cual en este caso es doblemente negativo, no solo de la manera estándar, pues todas las aplicaciones quieren muchas descargas, sino que la base del funcionamiento de Sororidapp parte de tener una base de usuarias grande para poder encontrar auxilio independientemente de donde te encuentres. Reducir el número de usuarias puede crear un efecto de bola de nieve y una fuga de usuarias, que dejen de usarla debido a que es inservible porque varias veces la han requerido y no ha sido de utilidad. Sin obviar que la idea de la aplicación tiene el objetivo de ser descargada por todas y requerida por ninguna.

Por lo que, eliminadas estas posibilidades y tras un proceso de reflexión importante, se ha llegado a la conclusión de que la monetización de Sororidapp tiene dos opciones: Ser subvencionada externamente, ya sea por el gobierno o por una empresa privada, que quiera hacerse cargo, o directamente compre la aplicación; o encontrar una forma de publicidad que no sea intrusiva y que no obstaculice de ninguna forma el correcto y normal uso del sistema.

Profundizando más en esta última opción, se ha pensado el posible desarrollo de un botón de publicidad que, de manera totalmente voluntaria y pedida por las usuarias, proceda a introducir un anuncio temporal de forma que, sin obstaculizar el funcionamiento y siempre que éstas quieran, se pueden ver anuncios para dar fondos al sistema. Este concepto parte del botón de anuncio por vida extra típico de los videojuegos, pero sin recompensa.

Pese a ser, quizás, un tanto idealista, creo que merece por lo menos un intento para crear un sistema o una relación simbiótica en cuanto que viendo anuncios voluntariamente las usuarias mantienen viva la aplicación y, a cambio, quizás la aplicación marque la diferencia de encontrarse la usuaria en una situación peligro

7 Implementación

En este apartado se recogen los aspectos más destacados de la implementación de Sororidapp. En primer lugar, se detallan las activities y fragments que forman las pantallas más importantes de la aplicación. A continuación, se indican las principales clases en lenguaje Kotlin para implementar tareas en segundo plano. El apartado concluye con algunos detalles de clases de apoyo a la aplicación.

7.1 Activities y fragments

7.1.1 PhoneLogin

Esta actividad es la que realiza todo el proceso de inicio de sesión, explicada en la guía de la autenticación por teléfono [17] y con la actividad pública [18] disponible en el github de Firebase, basándose en esta última para los fragmentos esenciales y más básicos, pero siendo adaptada a las necesidades del sistema.

Aparte de la gestión de interfaz de usuario necesaria para recoger la información otorgada por la usuaria, la actividad cuenta con los siguientes fragmentos de código que gestionan la autenticación:

La siguiente función recoge el número de teléfono introducido por la usuaria y crea un proveedor de autenticación en móvil con él.

```
private void startPhoneNumberVerification(String phoneNumber) {
    // [START start_phone_auth]
    progres.setVisibility(View.VISIBLE);
    PhoneAuthOptions options =
        PhoneAuthOptions.newBuilder(mAuth)
            .setPhoneNumber(phoneNumber)        // Phone number to verify
            .setTimeout(120L, TimeUnit.SECONDS) // Timeout and unit
            .setActivity(PhoneLogIn.this)       // Activity (for callback binding)
            .setCallbacks(mCallbacks)           // OnVerificationStateChangedCallbacks
            .build();

    PhoneAuthProvider.verifyPhoneNumber(options);
    // [END start_phone_auth]
}
```

Código 4. startPhoneNumberVerification.

Esta llamada cuenta con 3 métodos *callback* diferentes definidos:

- onVerificationCompleted: Método que salta si la operación ha tenido éxito y además se ha auto detectado el código enviado. En consecuencia, el sistema introducirá el código en el campo correspondiente en lugar de hacerlo la usuaria.

- onVerificationFailed: La verifciacion ha fallado, se notifica a la usuaria a qué se debe.

- onCodeSent: El código se ha enviado de manera correcta, se actualiza la interfaz de usuario para permitir la introducción de éste.

Cuando se envíe el código de verificación, se ejecutará el siguiente fragmento:

```
private void verifyPhoneNumberWithCode( String code) {
        PhoneAuthCredential credential =
PhoneAuthProvider.getCredential(mVerificationId, code);
        signInWithPhoneAuthCredential(Objects.requireNonNull(credential));
 }
```

Código 5. VerifyPhoneNumberWithCode

```
    private void signInWithPhoneAuthCredential(PhoneAuthCredential credential) {
        progres.setVisibility(View.VISIBLE);
        mAuth.signInWithCredential(credential)
                .addOnCompleteListener(this, task -> {
                    if (task.isSuccessful()) {
                        FirebaseUser user = task.getResult().getUser();
                        updateUI(user);
                    } else {
                        Toast.makeText(this,
getString(R.string.InicioError),Toast.LENGTH_SHORT).show();
                        Log.w(TAG, "signInWithCredential:failure", task.getException());
                    }
                });
    }
```

Código 6. signInWithPhoneAuthCredential

Con estas dos funciones se intenta verificar el código de acceso. Si es satisfactorio, se procede a actualizar la interfaz de usuario, yendo a la pantalla de MainActivity (con las configuraciones de sesión necesarias). Si no, se notifica a la usuaria.

Ésta puede decidir que se reenvíe el código de verificación, que se haría con el siguiente fragmento:

```
private void resendVerificationCode(String phoneNumber,
PhoneAuthProvider.ForceResendingToken token) {
        PhoneAuthOptions options =
            PhoneAuthOptions.newBuilder(mAuth)
                .setPhoneNumber(phoneNumber)
                .setTimeout(120L, TimeUnit.SECONDS)
                .setActivity(PhoneLogIn.this)
                .setCallbacks(mCallbacks)
                .setForceResendingToken(token)
                .build();
        PhoneAuthProvider.verifyPhoneNumber(options);
    }
```

Código 7. Reenviar código de Verificación

Donde con el código de reenvío obtenido del primer envío se puede solicitar el reenvío del código de verificación.

Se cumple así con el requisito funcional *RF-00 Inicio de sesión.*

7.1.2 MainActivity

MainActivity es la actividad principal de la aplicación. Muestra la cantidad de gente cercana (y muy cercana) en un Textview; da la opción a estar disponible o no con un SwitchCompat, y contiene 4 botones; el botón de llamada a la policía, el de abrir los mapas que llevará a la MapActivity, el de pedir ayuda o notificar que estás bien y el de configuración, que lleva a MainLoginActivity. Se muestra también la dirección postal de la ubicación del usuario en tiempo real en otro campo de texto. Todo esto es explicado y plasmado en imágenes más adelante en la memoria.

Con el correcto funcionamiento de las caracterizas nombradas para esta actividad se cumple con los siguientes requisitos funcionales: RF-01, RF-03, RF-04, RF-05, RF-06 y RF-07.

Al iniciar la actividad se inicializan los servicios de SafetyNet, así como un listener de autenticación encargado de acabar la aplicación si se cierra la sesión, puesto que esto solo ocurrirá si se trata de un usuario no autorizado. Este *listener* se añadirá en el método onStart() y será eliminado en onStop().

```
FirebaseApp.initializeApp(this);
FirebaseAppCheck FirebaseAppCheck = FirebaseAppCheck.getInstance();
FirebaseAppCheck.installAppCheckProviderFactory(
      SafetyNetAppCheckProviderFactory.getInstance());

mAuth = FirebaseAuth.getInstance();
user = mAuth.getCurrentUser();

authListener = FirebaseAuth -> {
   FirebaseUser user1 = mAuth.getCurrentUser();
         Toast.makeText(MainActivity.this,"Usuario No Invitado, Saliendo de la aplicacion",
Toast.LENGTH_LONG).show();
   if (user1 == null) {
      finish();
   }
};
```

Código 8. Inicialización de SafetyNet, FirebaseAuth, FirebaseUser y authListener

Esta actividad crea 3 referencias diferentes a Firebase, para actualizar tu información personal, la información de gente disponible y la información de gente que pide ayuda, como se muestra a continuación:

```
public static final String END_POINT_BASE_URL = "https://sororidapp-default-rtdb.europe-
west1.firebasedatabase.app";
…
helpDBreference=FirebaseDatabase.getInstance(END_POINT_BASE_URL).getReference("
emergencies");
locDBreference=FirebaseDatabase.getInstance(END_POINT_BASE_URL).getReference("pl
aces");
mDatabaseReference=FirebaseDatabase.getInstance(END_POINT_BASE_URL).getRefere
nce("users/"+user.getPhoneNumber());
```

Código 8. Inicialización de referencias de bases de datos.

Lo primero que se realiza en esta actividad, tras estas inicializaciones, es comprobar los permisos, tanto de ubicación requeridos por Android, como de utilización de la aplicación, es decir, comprueba si la usuaria está invitada a ella. El primero de ellos se hace de la forma estándar de comprobación de permisos, como se indica en la guía de developers de Android [19]. Para comprobar si debe tener acceso, se accederá mDatabaseReference donde, si tiene acceso, aparecerá la lista de usuarios que le autorizan. Si ésta no se encuentra, se cerrará sesión, lo que dispara el AuthStateListener ya explicado y cerrará la aplicación. Con esta acción se cumple con el RF-16 Control de Sesiones.

Al crear la actividad se llamará a startLocationUpdates() que creará un looper para actualizar la información de ubicación. El tiempo estándar de loop serán 60 segundos, con un máximo de velocidad de 30.

Esta función se realiza mediante el uso de un LocationRequest con el que se creará un LocationSettingRequest. Cuando el primero tenga las configuraciones del tiempo del Loop, una vez construido, este último se usará para comprobar si se cumplen las condiciones necesarias. Por último, se configurará el getFusedLocationProviderClient con el callback que se quiere ejecutar a cada actualización de ubicación. En nuestro caso, se ejecutará el siguiente código donde location es locationResult.getLastLocation(), es decir, la última ubicación conocida por el proveedor de ubicación:

```
public void onLocationChanged(Location location) {
            if (location !=currentLocation){
               currentLocation = location
               new DirectionGetter( MainActivity.this,1)
      .execute(getUrlDirection(location));
        }
     }
```

Código 9. Callback OnLocationChanged

Todo este método recién explicado se ha desarrollado en base y con la ayuda del tutorial de la página CODEPATH RETRIEVING LOCATION WITH LOCATIONSERVICES API [20]

Se llama también a createNotificationChannel(), función que crea el canal de notificaciones, para poder crear notificación *Push* cuando alguien requiera ayuda.

Al marcarnos como disponibles, compartiremos la ubicación en la segunda referencia locDBreference y mostrándonos disponibles para los demás usuarios cercanos en su panel informativo.

Al estar disponible, se empezará a hacer un listener de la referencia de ayuda en tu zona, de forma que, si se detecta alguna solicitud de ayuda, se generará una notificación única llevándote al mapa personalizado donde ir a ayudar a la persona que pide ayuda, a través de un pending intent a HelpMapActivity.

Al pulsar en el botón de ayuda, empezaremos a compartir la ubicación en la referencia a la base de datos “helpDBreference”, de forma que los usuarios cercanos y disponibles en nuestros alrededores recibirían la notificación recién explicada, al tener un listener de esa misma base de datos.

Como ya se ha explicado, en el cliente de ubicación tendremos un método callback que se llamará cada vez que cambie la ubicación en el bucle creado; dentro de esta cogeremos la location pasada y llamaremos a new DirectionGetter(MainActivity.this, 1).execute(getUrlDirection(location)), de forma que como se ha indicado anteriormente en el apartado de explicación de las API, se conseguirá la dirección postal de la ubicación pasada por parámetro, para así poder mostrarla.

A la vuelta se ejecutará el callback de la interfaz *TaskLoadedCallback* implementado en el MainActivity.this. Así pues, tendremos en esta actividad la implementación de

onDirectionTaken, donde se realizará la actualización de todo lo orquestrado hasta ahora: se actualizarán las coordenadas de la usuaria en *locDBreference* y se actualizarán o inicializarán también los listeners a las bases de datos. Esto ocurrirá cuando sea el primer acceso, teniendo que inicializar éstos o cuando la usuaria se haya desplazado y tenga que modificarlos.

Así pues, tanto para los usuarios cercanos como para los notificados al pedir ayuda, se contemplarán a los usuarios que estén dentro de tu misma zona, sabiéndolo con la referencia a locDBreference, y que se encuentren a una distancia absoluta de menos de 2 kilómetros.

En el método onDestroy() se borrarán todos los listeners creados para la base de datos, así como el cliente de ubicación, acabando debidamente así con el ciclo de vida de la actividad.

7.1.3 MapActivity

Esta actividad es la actividad principal de uso del mapa, implementando aquí todas las librerías explicadas y cumpliendo sus requisitos funcionales.

Se trata de la actividad donde ver el mapa centrado en tu ubicación, con buscador y posibilidad de clicar para marcar caminos. El looper del cliente de ubicación se crea de igual manera en starLocatioUpdates() pero además, aparte del looper, aquí dentro se llamará a getLastlocation que es una task. Tras esto, se añadirá un onSuccesListener donde se guardará la ubicación en la variable y se iniciará el mapa con ella, consiguiendo así que se inicialice el mapa centrado en la ubicación de la usuaria. Este proceso se muestra en el fragmento de código siguiente.

```
getFusedLocationProviderClient(this).getLastLocation().addOnSuccessListener(location ->
{
        if(location!=null){
            currentLocation=location;
            mapFragment.getMapAsync(MapActivity.this);
          }
       });
```

Código 10. Obtención de primera ubicación

```
getFusedLocationProviderClient(this).requestLocationUpdates(mLocationRequest, locCB
=new LocationCallback() {
                @Override
                public void onLocationResult(@NotNull LocationResult locationResult) {
                    onLocationChanged(locationResult.getLastLocation());
                }
```

```
}, Looper.myLooper());
```

Código 11. Inicializacion del looper de ubicación

Asimismo, la función que se llama en el Código 10, se encarga de inicializar el mapa. Se llama de este modo para asegurar que disponemos de la ubicación necesaria para inicialarla. La función en cuestión, al finalizar, llamará a la funcion onMapReady, que es donde se encuentran realmente las inicializaciones:

```
@Override
public void onMapReady(@NotNull GoogleMap map) {
    mMap = map;
    if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(MapActivity.this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 44);
    } else {
        mMap.setMyLocationEnabled(true);
        mMap.setTrafficEnabled(false);
        mMap.setOnMyLocationButtonClickListener(this);
        mMap.setOnMapClickListener(this);
        mMap.moveCamera(CameraUpdateFactory.zoomTo(16));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(new LatLng(
currentLocation.getLatitude(), currentLocation.getLongitude() )));
    }
}
```

Código 12. Inicialización del mapa

Como podemos observar, se concretan que debe tener la ubicación de la usuaria activada, el tráfico desactivado (pues no es necesario para el uso de la aplicación), se inicializan los listeners que se han implementado y que regulan el botón para centrar el mapa en la ubicación de la usuaria, así como el registro de las pulsaciones en el mapa. Y por último se configuran el *zoom* y la posición iniciales.

Al clicar en el mapa, la función básica del SDK devolverá las coordenadas de ese punto. Con ellas se realizará la llamada a Directions que tiene siempre como origen la ubicación actual de la usuaria y como destino tendrá esas coordenadas obtenidas. También, con ellas, se utilizará la API de Geocoding para poder mostrar la dirección de la ubicación destino y, al ir avanzando por la calle, el looper irá actualizando las coordenadas actuales de la usuaria, con las que se volverá a usar la librería de Geocoding para mostrar su dirección actual y evitar que ésta se pierda.

7.1.4 MainLoginAcitivity

Esta actividad no tiene mucho misterio, incluye 3 botones diferentes con los cuales se puede cerrar sesión, de la misma manera que se cierra en la MainActivity, pero esta vez de manera voluntaria por parte de la usuaria.

El botón de borrar cuenta, aparte de pedir confirmación, se diferencia del anterior en que en este caso se borra al usuario de la lista de autorizados, aunque sigue quedando constancia de su uso en la base de datos, ya que es peligroso que la usuaria pueda desaparecer completamente del registro, sobre todo si se hace un mal uso de la aplicación.

Por último, queda el botón de acceso a la actividad AddUserActivity.

7.1.5 MapHelpActivity

Se trata de la actividad donde ayudar a las demás usuarias, y es una versión simplificada de MapActivity. En este MapFragment se mostrarán en tiempo real tu ubicación y la ubicación de la usuaria en peligro, así como el camino más rápido para llegar a ésta.

Si la mujer en peligro decide que ya está a salvo, se te mostrará un aviso y se borrarán tanto el camino como la ubicación de la susodicha, pues ya no es necesario conocerlas y se ha vuelto al estado normal de la aplicación, donde todo el mundo se encuentra a salvo.

Esta actividad es muy similar a la anteriormente explicada, con la salvedad de que no tiene onMapClick, ni buscador de ubicación, ya que el objetivo es ir sí o sí a donde la mujer en auxilio se encuentra y no ir a otros sitios. No tiene nada que no sea estrictamente necesario para poder ir en ayuda de la usuaria en apuros.

En esta actividad se usan los servicios de la API de directions para recrear en tiempo real el camino más corto entre tu ubicación y la de la usuaria en peligro. En onCreate() se crea una referencia a la base de datos donde se encuentra la ubicación de la usuaria en peligro, la cual se destruirá junto con el looper de ubicación en el método OnDestroy(). Asimismo, también se notifica a la usuaria en peligro que se está yendo en su ayuda, o se deja de decir si se sale de mapa porque haya dejado de ir por cualquier motivo.

7.1.6 AddUserActivity

Actividad que, junto a UserAdapter y ViewHolderFicha, muestra una lista dinámica de los números a los que has invitado a usar la aplicación. Muestra también un botón con icono para añadir usuarias que, al pulsarlo, mostrará un pop-up para introducir el número.

Por seguridad y coherencia, con la aplicación no se permite a un usuario introducir su propio número en la lista.

Por último, para eliminarlos de la lista, los números se pueden desplazar hacia los dos lados, con lo que saltará un pop-up de seguridad para confirmar que en efecto quieres eliminar el número.

7.1.7 LoadActivity

Se trata de la actividad más sencilla de la aplicación, pero no por ello es menos importante. Su función consiste en comprobar si hay un usuario con la sesión iniciada. Si es así, abre la MainActivity de forma normal. Si no es así, guía a la usuaria a la actividad de inicio de sesión.

```
public class LoadActivity extends AppCompatActivity {
    private  FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mAuth = FirebaseAuth.getInstance();

        setContentView(R.layout.activity_load);
        Intent intent;
        FirebaseUser user = mAuth.getCurrentUser();
        if( user ==null){
            intent = new Intent(this, PhoneLogIn.class);
        }
        else{
            intent = new Intent(this, MainActivity.class);
        }
        startActivity(intent);
    }
}
```

Código 13. LoadActivity

7.2 Clases Kotlin

Las clases Kotlin implementadas se deben todas a la necesidad de una corutina y todas ellas están apoyadas en la interfaz TaskLoadedCallback, para procesar el resultado de las llamadas.

Las clases DataParser, FetchURL, PointParser, TaskLoadedCallback, así como los métodos utilizados para generar las URL para las llamadas HTTPS, han sido inspiradas en el siguiente proyecto de GitHub de Vysh [22], que se encontró fortuitamente al principio del desarrollo, y ayudó a sentar las bases de lo que acabaría siendo Sororidapp.

En cada una de las clases Kotlin, contamos con el método startBackground(), que será el encargado de iniciar la courutine en esa clase. Como ejemplo, a continuación se introduce la versión de la clase PointParser:

```
open fun startBackground(jsonData: JSONObject,mContext: Context, a: Activity) {
    this.taskCallback = mContext as TaskLoadedCallback
    this.activity = a
    // inicializamos la coroutine
    CoroutineScope(Dispatchers.IO).launch {
        try {
            val result = doInBackground(jsonData)
            withContext(Dispatchers.Main) {
                try {
                    onPost(result)
                } catch (e: Exception) {
                    e.printStackTrace()
                }
            }
        } catch (e: Exception) {
            e.printStackTrace()
        }
    }
}
```

Código 14. Ejemplo de Coroutine

Como se puede observar con los comentarios del código, el primer paso es iniciar la coroutine con el dispatcher que se quiera utilizar. A continuación, la función doInBackGround viene a ser el apartado del mismo nombre del obsoleto AsynkTask. Por último, donde se llama al metodo onPost se ejecutará todo en lo que la versión obsoleta se ejecutaba en el postExecute.

7.2.1 CoordGetter

Clase Kotlin que, dada una URL que cumple las características de la API de Google Geocode, coge la ubicación en la dicha URL y devuelve la coordenada de dicha ubicación.

De igual manera que las anteriores, ejecuta el acceso HTTP en una coroutine de Kotlin en background. Luego devuelve en el postExecute el callback onCoordTaken pasado a la clase donde se llamó, devolviendo la latitud y longitud deseadas de esta llamada.

7.2.2 DirectionGetter

Clase Kotlin que, dada una URL que cumple las características de la API de Google Geocode, descarga el contenido del HTTP en background y comparte el resultado con la activity que la invocó, en el taskCallback onDirectionTaken. Devolviendo la comunidad, el país, la dirección entera y un entero para diferenciar, en el caso de que sea una llamada de la MapActivity, si se trata de una llamada para la ubicación actual o para el destino.

7.2.3 FetchURL

Clase Kotlin para la corrutina que, dada una URL que cumpla la API de Google Directions, descarga todo el contenido de la llamada API, prepara el JSON object para pasarlo a la función PointParser. Esta clase se encarga de hacer toda la conexión en un hilo secundario para no colapsar el hilo principal de la aplicación. La llamada de Locations se encarga de devolver el camino generado en Google Maps desde el punto inicial al punto final, paso a paso como camina una persona.

7.2.4 PointParser

Clase Kotlin para la corrutina que dado, un objeto JSON generado por FetchURL tras la llamada de la API de Directions, lo trata con la clase DataParser(), con la que genera la PolylineOptions. Dado este resultado, lo configura con el color y grosor deseados, y lo devuelve a la clase desde donde se llamó con el método Callback onTaskDone, ya sea MapActivity o HelpMapActivity.

7.3 Clases secundarias o de apoyo

7.3.1 TaskLoadedCallback

Interfaz java de los callbacks a implementar para el debido uso de las coroutines de Kotlin. Con esta interfaz se puede variar la acción a realizar según donde se llame, es decir, los métodos de las clases Kotlin darán lugar a una respuesta de esta interfaz, que será implementada en la clase que la utilice. Y así se puede diferenciar el tratamiento del resultado en MainActivity respecto el mismo resultado en MapActivity.

Los métodos incluidos en la interfaz son: onTaskDone, que será la respuesta a PointParser, onCoordTaken como respuesta a CoordGetter y onDirectionTaken como respuesta a DirectionGetter.

7.3.2 DataParser

Clase Java con dos funciones que se encargan de generar el camino de origen a destino.

Parser: Función que se encarga de ver de forma exhaustiva el JSON devuelto y pasado por parámetro. Cuando llega a un nodo de información de polyline llama a decodePoly; y repite hasta que ha acabado el camino. Entonces devuelve el polylineOptions completo de la ruta.

DecodePoly: Función que, pasado un nodo de Polyline codificado por Google, se encarga de descodificarlo para poder trabajar con él. Este método fue obtenido de la página de JEFRREYSAMBELS [22] aunque actualmente se encuentra descontinuada.

7.3.3 AdapterNumber

Clase que extiende a RecyclerView.Adapter<ViewHolderFicha> y que proporciona soporte de los números de teléfono de la lista al Recycler View.

Contiene 3 Funciones sencillas:

- onCreateViewHolder: función que devuelve una vista del elemento personalizado ViewHolder.
- onBindViewHolder: función que dado un holder personalizado y una posicion de la lista de contactos, llama al onBind del holder para incluir la información, es decir, el numero correspondiente.
- deleteItem: Función que sirve para borrar una entrada de la lista de números.

7.3.4 ViewHolderFicha

ViewHolder personalizado para incluir los números de teléfono en el TextView correspondiente.

8 Sororidapp, guía de uso y decisiones de diseño

En este apartado se comentará un proceso general de uso de la aplicación de una usuaria normal. Mientras se avance en el proceso, se comentarán las decisiones de diseño que han sido tomadas para crear una interfaz de usuario que, pese a no ser perfecta, cumple con unos mínimos de usabilidad, siendo así una base sólida de la que partir para que, junto al *feed-back* de las diferentes usuarias pueda ser mejorado hasta dar con la versión definitiva.

Lo primero a comentar es que, pese a no ser los tonos púrpuras y morados los más usados en aplicaciones, se ha decidido basar la paleta de colores en éstos, pues son los colores adoptados como símbolo del movimiento feminista, dando a entender así que Sororidapp es una alidada más y fortaleciendo la imagen de marca que se pretende dar a las usuarias.

Al entrar en la aplicación por primera vez, la usuaria verá un formulario de inicio de sesión donde deberá introducir su número de teléfono junto a su prefijo, este último predefinido según la configuración del sistema, pero modificable por si no se tratara de éste.

Figura 20 Pantalla principal inicio de sesion Sororidapp

Al introducir el número de teléfono y pulsar en entrar, el sistema de autenticación por móvil de Firebase enviará a la usuaria un SMS que contendrá un código de verificación de usuaria. Mientras se espera la respuesta del servidor, se mostrará en pantalla un progress bar

a modo de respuesta visual, para que la usuaria sepa que, en efecto, se está procesando su solicitud.

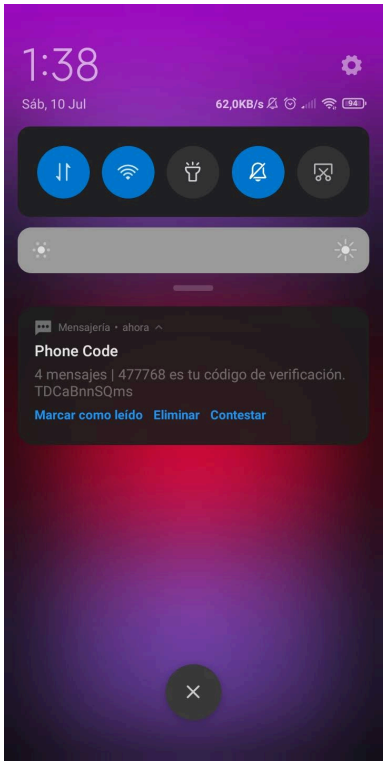

Figura 21 Recepción de mensaje de verificación al hacer Log-in

Una vez recibido el mensaje de texto con el código, éste deberá ser introducido por el usuaria y enviado para completar la verificación de usuaria. Este contará con la opción de solicitar que se reenvíe el mensaje de verificación, si no lo recibiera o hubiera algún problema.

El código del mensaje, si fuera auto detectado por la aplicación, se introducirá en el campo correspondiente de forma inmediata, ahorrando el trabajo a la usuaria. Nuevamente, mientras se procesa la solicitud de verificación, un progress bar será mostrado para transmitir a la usuaria que ha realizado bien el proceso y se está gestionando su verificación. Esto mismo se hará también si la usuaria solicita que se reenvíe el código de verificación

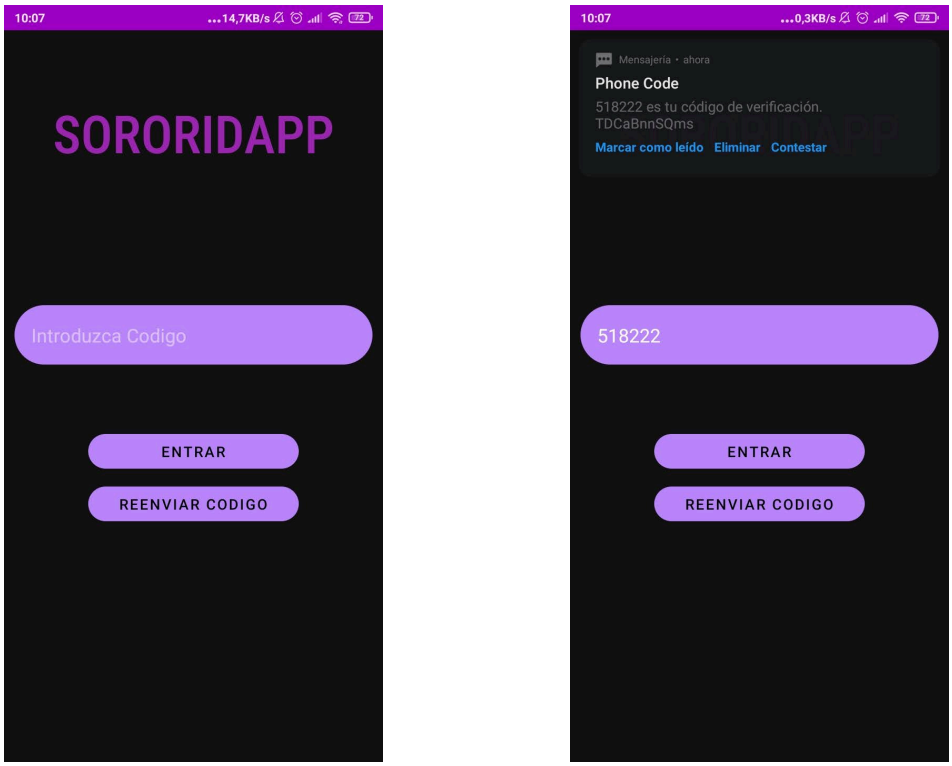

Figura 22 Introduccion del codigo de verificación (imagen placeholder)

Cuando la usuaria introduzca, de forma satisfactoria, el código de verificación pasará la pantalla principal de la aplicación donde dispondrá de diferentes posibilidades dentro de la misma. En primera instancia contara con un botón de Toggle arriba del todo para marcar si se encuentra disponible o no para el resto de las usuarias, al lado de este un botón de acceso a opciones extra explicadas más adelante.

Bajando a lo largo de la pantalla verá, en este orden, la ubicación en la que se encuentra en tiempo real, a la derecha de la cual dispone de un botón icónico del mapa para poder abrir éste; debajo, la indicación de usuarias cercanas (y muy cercanas) a ella, las cuales serán notificadas si la usuaria decidiera pulsar el siguiente botón.

En el centro de la pantalla el botón de ayuda, con el acceso más asequible, y perfectamente diferenciable tanto respecto a su forma, más estridente, como a su color, que no encaja en la gama cromática para que se pueda diferenciar, por ejemplo, en estado de embriaguez o en un vistazo instantáneo mientras huye del agresor.

Por último, la usuaria contará con un boto tan accesible y diferenciable como el anterior, que le permitirá llamar al número de emergencias 112.

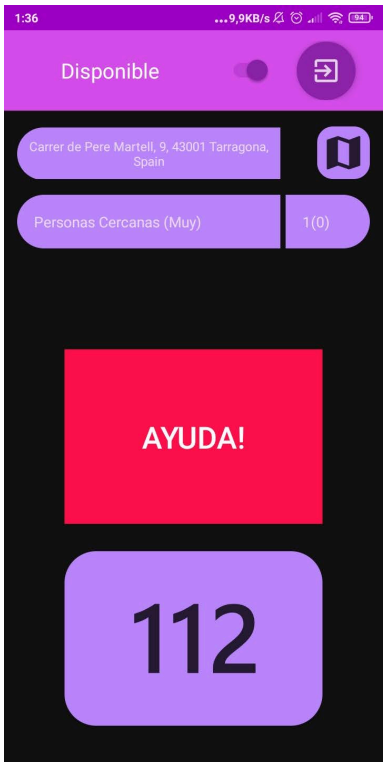

Figura 23 Pantalla principal de Sororidapp a salvo, MainAcitvity

Al pulsar el botón de ayuda, la usuaria notará un cambio drástico en la pantalla principal, pues el botón de ayuda, que ahora será el botón de encontrarse a salvo, ha pasado a ser verde, más llamativo aún, si cabe, que el color rojo original del botón. Esto está hecho, aparte de por los mismos motivos que en su anterior estado, por si la usuaria activara la opción de forma involuntaria por un descuido, fuera fácil verlo. En este caso se ha optado por la fácil cancelación de errores, antes que en asegurar que la acción es apropósito, pues cuantas más facilidades tenga una usuaria en apuros, de mayor utilidad le será la aplicación.

Aparte de este cambio, se añadirá también una nueva línea de información donde la usuaria puede ver cuántas usuarias externas han accedido a ir en su ayuda, para calmarse si vienen en su ayuda o plantearse llamar a la policía, si se diera el caso de que ninguna de éstas viene en su ayuda. Esta información será el número de usuarias, pero nunca su ubicación será compartida con la usuaria en peligro o con las demás, para salvaguardar su seguridad y privacidad.

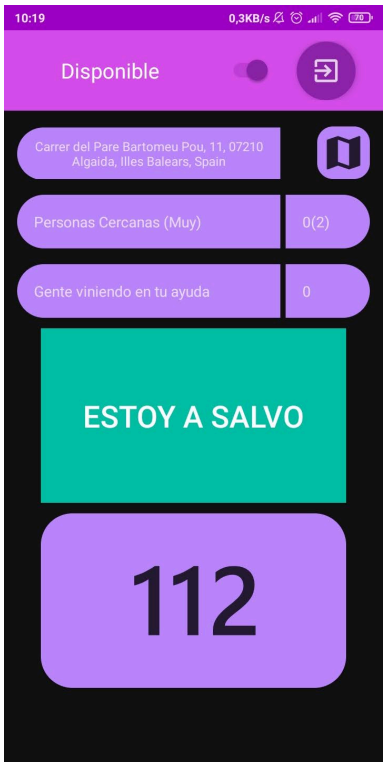

Figura 24 Pantalla principal de Sororidapp pidiendo auxilio, MainAcitvity

Cuando la usuaria principal entre en este modo, las usuarias cercanas serán notificadas con sonido y una notificación *Push* de la siguiente manera:

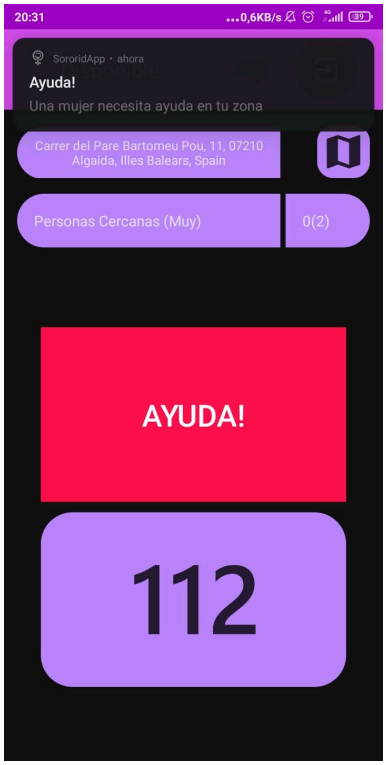

Figura 25 Usuaria cercana recibiendo solicitud de ayuda

Al pulsar en la notificación, la usuaria será llevada a la pantalla de mapa siguiente, donde podrá ver en tiempo real, la ubicación de la persona en peligro y el camino a recorrer para llegar hasta ella de forma óptima. Además, mientras esta segunda usuaria está yendo en su ayuda, la primera será notificada en el campo correspondiente, sabiendo el número de personas que viene en su ayuda.

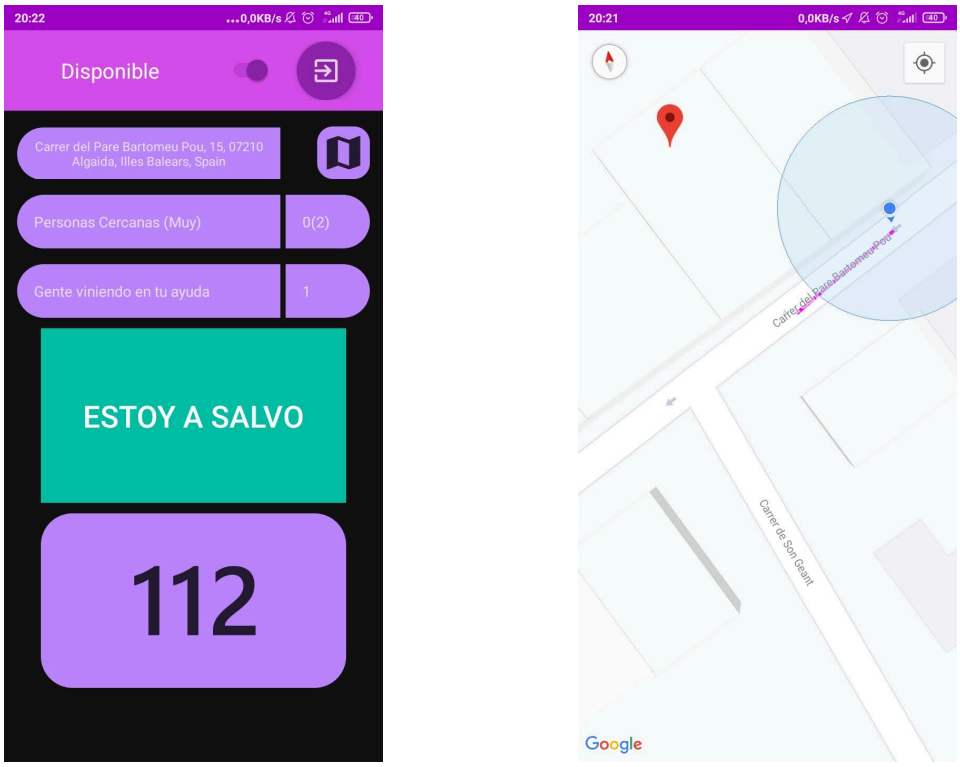

Figura 26 Pantalla de las usuarias al pulsar la notificación

Si la usuaria considera que ya ha pasado el peligro o la notificación ha sido por error, al pulsar de nuevo el botón central se volverá al estado normal de la aplicación; notificando a todas las usuarias que venían en su ayuda, que la persona en peligro ya se encuentra a salvo y dejando de guiarlas hasta ella.

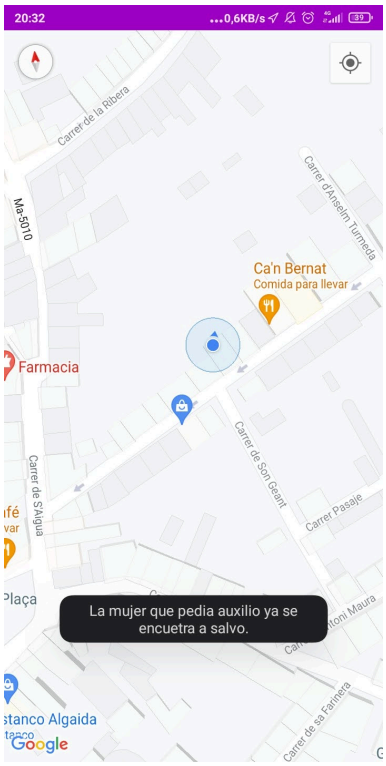

Figura 27 Notificación de que la mujer ya se encuentra a salvo

Si en lugar del botón de ayuda, la usuaria decidiera usar el de 112, se procedería a llamar a los servicios de emergencia, como el botón propiamente dice: al 112.

Cuando la usuaria entre al mapa, éste empezará centrado en la misma, mostrando nuevamente la ubicación en la que se encuentra en un mensaje emergente. En la pantalla se verá principalmente el mapa, pero la usuaria contará también con un buscador para realizar búsquedas de la ubicación a la que desea dirigirse, así como un botón estándar de mapa que centra la vista nuevamente en la usuaria.

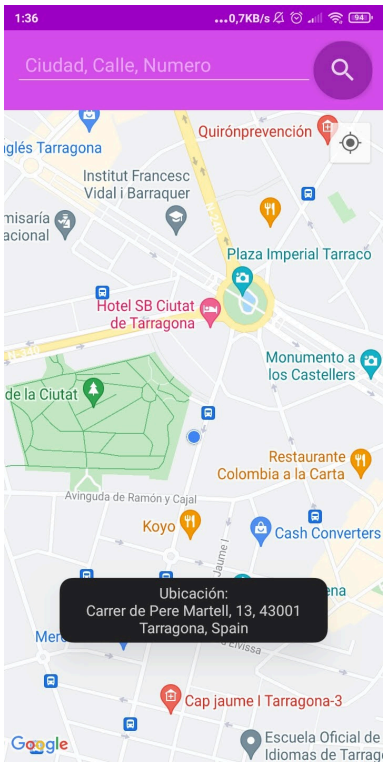

Figura 28 Pantalla del mapa en su estado inicial

Si la usuaria toca un punto en el mapa, automáticamente se generará el camino directo desde su ubicación actual hasta el punto seleccionado, originando una imagen como la de la figura 29, de igual manera se generará si, en su lugar, el destino es seleccionado mediante el uso del buscador.

Al mismo tiempo, se mostrará nuevamente con un mensaje Toast la dirección del destino al que se está dirigiendo y durante el trayecto se mantendrá informada de la misma manera a la usuaria de su ubicación, por si necesitara de llamar a la policía, por ejemplo, pudiera informarles de forma directa en la ubicación en la que se encuentra y hacia dónde se dirige y así, recibir una mejor ayuda de su parte.

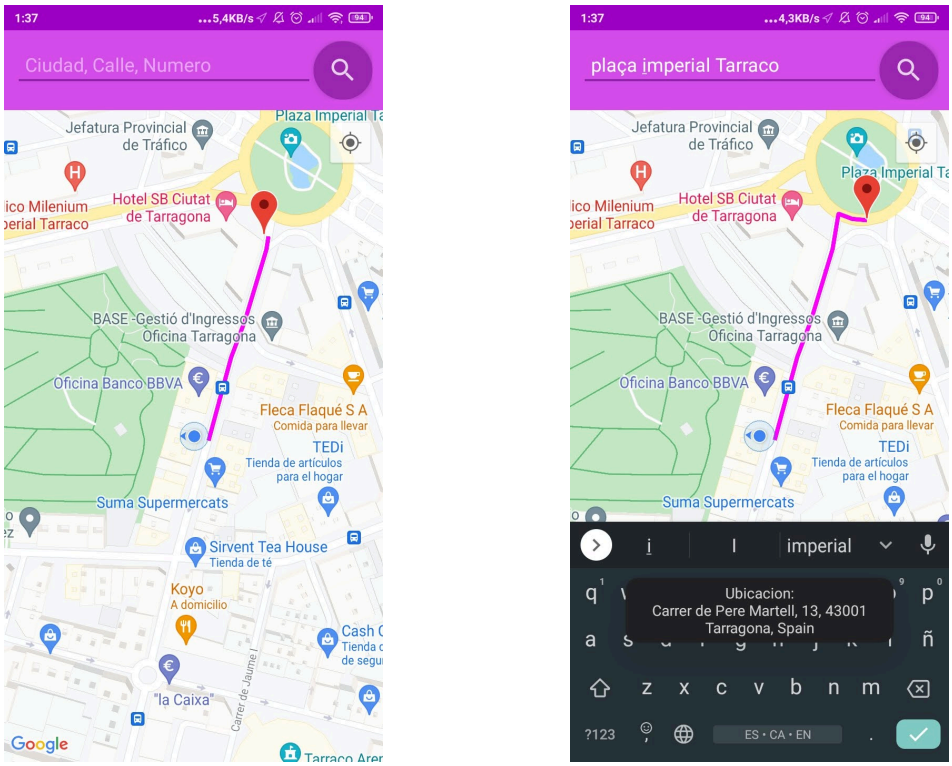

Figura 29 Generación de caminos en mapa mediante pulsación y búsqueda (MapActivity)

Volviendo a la pantalla principal, pulsando en las opciones extra, podremos observar la siguiente pantalla, contando con botones de cerrar sesión, borrar cuenta e invitar usuario.

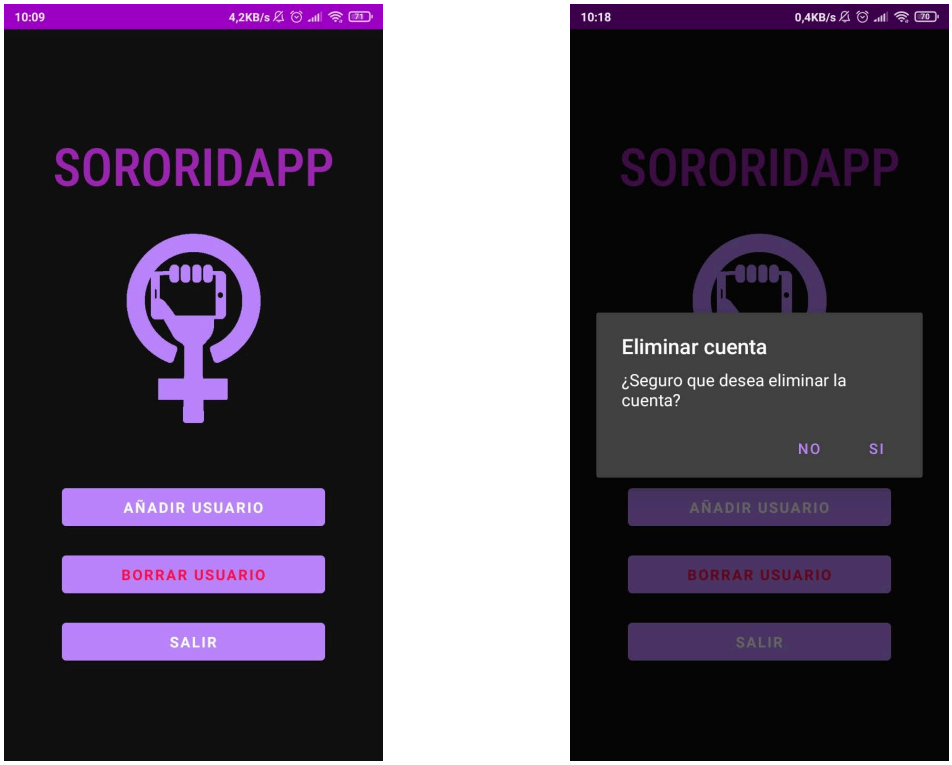

Figura 30 Pantalla de gestión de usuarios (MainLoginAcitivity)

En este caso, el botón de cerrar sesión y el de borrar cuenta hacen lo que su nombre indica, cerrar la sesión actual del usuario en la aplicación con consecuencia final la necesidad de hacer nuevamente el log-in ya explicado. Asimismo, al borrar se genera el mismo estado que al cerrar sesión para el usuario, la diferencia está en la base de datos y la gestión de usuarios de donde se eliminará el número, de la usuaria en cuestión. En este último caso, se pide confirmación a la usuaria dado que es una acción irrevocable.

Por último, pero no por ello menos importante, se cuenta con el acceso a la pantalla de invitación de usuarias, donde cada usuaria podrá invitar a las usuarias que crea conveniente de forma que ellas puedan usar también la aplicación. La idea con esto es hacer un cribaje de las personas que descarguen la aplicación, de las cuales solo las invitadas anteriormente por otras usuarias podrán utilizar la aplicación.

Esta invitación consistirá en introducir su número de teléfono y es revocable en cualquier momento, simplemente eliminando el número de la lista esta usuaria quedará des invitada por tu parte. Esto se debe al planteamiento de que no haya hombres usuarios como norma general o como objetivo, pero si pudiera haberlos como apoyo puntual, o si una usuaria hiciera un mal uso de la aplicación, podría revocársele el uso.

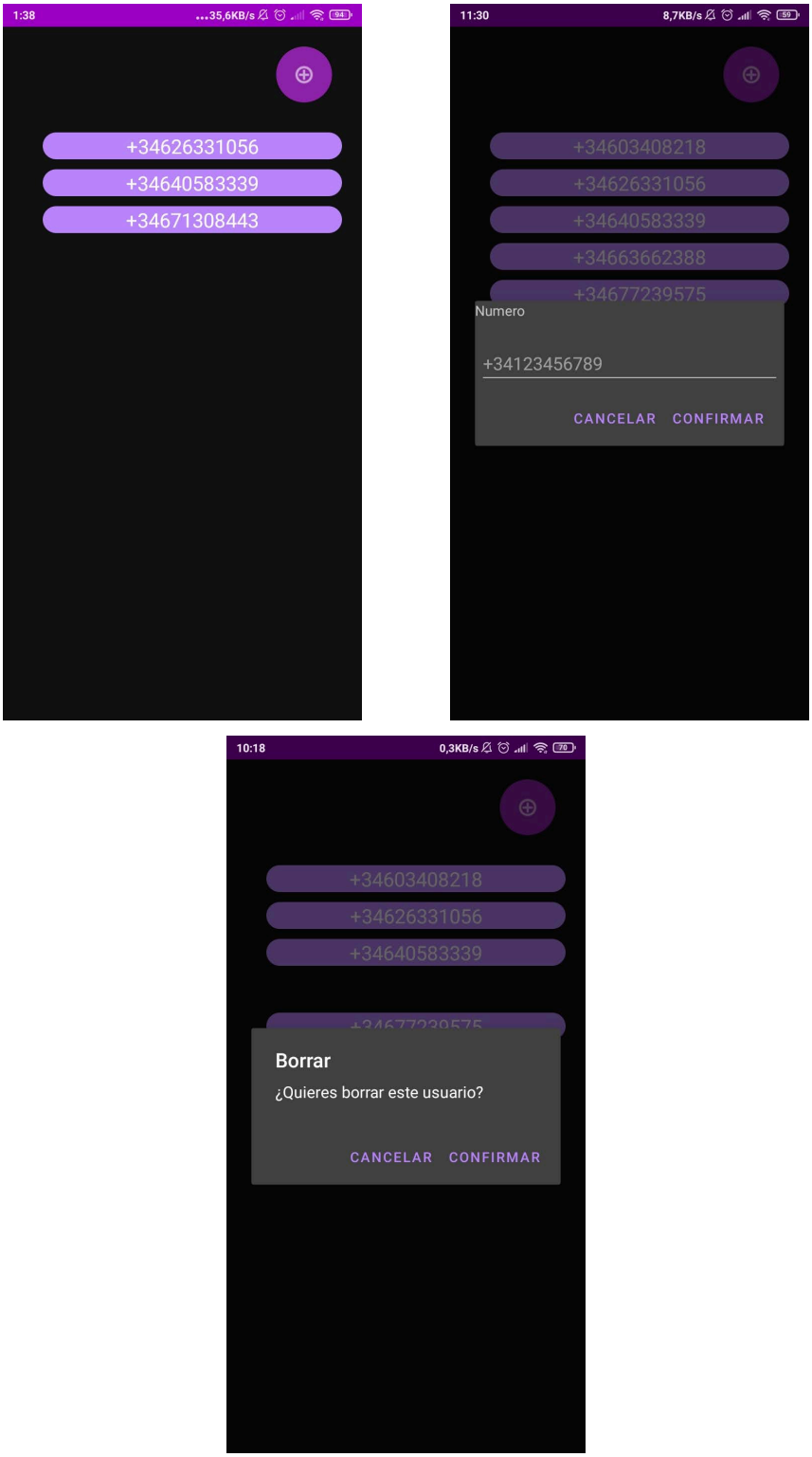

Figura 31 Pantalla para gestionar invitaciones a otras usuarias a usar la aplicación

9 Juego de pruebas

En este apartado se indica el conjunto de pruebas realizadas para comprobar el correcto funcionamiento de Sororidapp.

Tan importante es en un proyecto el trabajo de desarrollo en si, como la comprobación y el juego de pruebas de todo lo desarrollado, dado que, si no se puede confiar en que funciona todo correctamente, entonces no es de ninguna utilidad.

A continuación, se definen todas las pruebas realizadas sobre las funcionalidades desarrolladas para Sororidapp:

1- La usuaria puede iniciar sesión vía telefónica correctamente.
2- La ubicación indicada en la app es correcta.
3- La traducción de ubicación a dirección postal se realiza correctamente.
4- Se detectan correctamente las usuarias en los alrededores.
5- Se envían correctamente solicitudes de ayuda.
6- Llegan correctamente notificaciones de solicitudes de ayuda.
7- Se generan correctamente caminos en el mapa.
8- Se abre correctamente la ubicación centrada en la usuaria.
9- Se realizan correctamente las búsquedas en el buscador del mapa.
10- Se cierra sesión de forma correcta.
11- Se borra la cuenta de forma correcta.
12- Se invita a usuarias correctamente.
13- Se des invita a usuarias correctamente.
14- La aplicación no funciona para personas no invitadas.
15- Se deja de mostrar como disponible a una usuaria al ella marcarlo correctamente.
16- Se notifica correctamente que la mujer en peligro ya está a salvo.
17- Se notifica cuántas personas están yendo en su ayuda correctamente.
18- Se llama al número de emergencias correctamente.

Con todas estas pruebas satisfactorias, se puede concluir que la aplicación funciona correctamente en términos generales, ya que cumple con lo establecido en todos sus apartados.

10 Conclusiones

A continuación, en este último apartado, se dan las conclusiones extraídas del análisis y la realización tanto del proyecto Android, como de su memoria, donde se contrastan los resultados.

En general, y como primera reflexión, quería concluir este Trabajo de Fin de Grado como un éxito, ya no solo porque se ha conseguido llevar a cabo un prototipo bastante solido de la aplicación deseada, sino por todos los conocimientos que el desarrollo de éste me ha otorgado.

Gracias al desarrollo de Sororidapp, he pasado de apenas conocer la existencia del lenguaje Kotlin, a tener una base sólida de éste y ser consciente de su potencial y su auge en el mercado. Asimismo, he visto tremendamente incrementadas mis capacidades generales en cuanto a desarrollo Android, ya sea en la lógica con el conocimiento ampliado de Java, el lenguaje principal del proyecto, o con el mayor conocimiento del que ahora dispongo sobre el funcionamiento del sistema operativo Android, sus patrones, como el MVC, o su ciclo de vida.

Se ha adquirido una mayor constancia de la importancia de la interacción de la persona con el ordenador, o en este caso el móvil, y de lo vital que es para un buen funcionamiento de todo sistema, que la interfaz de usuario sea usable, cómoda e intuitiva.

Se ha vivido de primera mano gran parte del proceso de desarrollo de una aplicación desde el concepto, pasando por el análisis de mercado y necesidades, hasta llegar a las primeras versiones prototípicas de lo que puede llegar a ser un sistema real y competente.

Se ha conocido la triste pero indudable realidad de que las buenas ideas no se venden solas y que no todo sistema puede dar beneficios; que quizás tengas que comprometer parte del ideal de la vista inicial para que el proyecto siga adelante y de lo realmente costosos que pueden llegar a ser los servicios prestados por empresas de SAAS.

Se ha mejorado la transmisión de ideas y conceptos, ya sea de manera formal o coloquial, para dar a entender o poder explicar el objetivo o el problema que se está afrontando. Teniendo en cuenta la vital importancia del trabajo en equipo y la comunicación en esta industria, considero que, pese a no ser palpable, es de increíble valor.

Para finalizar, como se ha comentado anteriormente, este TFG se trata de un prototipo por lo que hay multitud de opciones o utilidades que no han sido implementadas en esta primera versión, pero sí podrían realizarse en el futuro. Por ejemplo, se pueden ampliar las funcionalidades de los mapas, incluyendo Streetview o ampliando la capacidad de caminos para poder realizarlos con parámetros como “pasando por zona iluminada” y otras variantes. Además, se podría empezar un registro similar al de AlertCops para marcar las zonas donde ha habido mayor incidencia de casos y que así las usuarias pudieran evitarlas.

11 Referencias

[1] Encuesta realizada: https://docs.google.com/forms/d/e/1FAIpQLSfsfuYnI62hBBX1lxtYc27aVmexgNogaJxB3FvS4oHxb196wQ/viewform?usp=sf_link

[2] AlertCops: https://alertcops.ses.mir.es/mialertcops/

[3] Google Play Store AlertCops: https://play.google.com/store/apps/details?id=com.alertcops4.app&hl=es

[4] Sister: https://www.joinsister.com/es/

[5] Google Play Store Sister:

https://play.google.com/store/apps/details?id=com.wavelt.sister&hl=es&gl=US

[6] Visual Paradigm Online:

https://online.visual-paradigm.com/drive/#diagramlist:proj=0&new=UseCaseDiagram

[7]Imagen MVC:

https://www.codementor.io/@dragneelfps/implementing-mvc-pattern-in-android-with-kotlin-i9hi2r06c

[8] Firebase DataBases:

https://Firebase.google.com/docs/database/rtdb-vs-firestore

[9] Firebase App Check:

https://Firebase.google.com/docs/app-check

[10] Maps SDK for Android documentation:

https://developers.google.com/maps/documentation/android-sdk/overview?hl=en_US

[11] Geocoding API documentation:

https://developers.google.com/maps/documentation/geocoding/overview?hl=en_US#before-you-begin

[12] Directions API: https://developers.google.com/maps/documentation/directions/get-directions?hl=en_US

[13] Developers Coroutines: https://developer.android.com/kotlin/coroutines-adv?hl=es-419

[14] Pricing Maps API: https://cloud.google.com/maps-platform/pricing

[15] Public Programs, Non-Profit apps: https://developers.google.com/maps/billing/public-programs

[16] Planes de uso de Firebase: https://Firebase.google.com/pricing?authuser=0

[17] Phone Auth: https://Firebase.google.com/docs/auth/android/phone-auth

[18] PhoneAuthActivity.java : https://github.com/Firebase/snippets-android/blob/375a84f96080a69a9d89aa0139d3c2e32ef475a1/auth/app/src/main/java/com/google/Firebase/quickstart/auth/PhoneAuthActivity.java#L104-L111

[19] Check Permision: https://developer.android.com/training/permissions/requesting

[20] Codepath Retrieving Location with LocationServices API: https://guides.codepath.com/android/Retrieving-Location-with-LocationServices-API

[21] Proyecto de github Vysh01/android-maps-directions: https://github.com/Vysh01/android-maps-directions/tree/master/app/src/main/java/com/thecodecity/mapsdirection/directionhelpers

[22] Data Parser(down): https://jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-API-with-java