UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

TESIS

Desarrollo de un sistema IoT para la mejora de la seguridad ciudadana en una Smart City en el Perú

PARA OBTENER EL TÍTULO PROFESIONAL DE:

LICENCIADO EN CIENCIA DE LA COMPUTACIÓN

Elaborado por:

JAVIER ENRIQUE VILLEGAS HERRERA

Asesor:

Dr. JOSÉ MANUEL CASTILLO CARA

LIMA - PERÚ

2019

Resumen

Hoy en día existen muchos problemas sin solución en nuestra sociedad, siendo uno de estos problemas la violencia de género. Específicamente, en los últimos años se ha intensificado la violencia contra la mujer, que actualmente se ha vuelto un problema sin solución en América Latina.

En promedio, actualmente existe un 38 % de mujeres que han declarado ser víctimas de violencia en América Latina siendo uno de los porcentajes más alto en el Mundo. Hoy en día, nuestro país ocupa el sétimo lugar entre los países de la región con los índices más alarmantes en estos temas.

Una respuesta rápida a estas preocupantes cifras es la implementación de un sistema distribuido de monitorización de personas en tiempo real, recolectando información de sus ubicaciones a través de sensores en una Smart City. Todo esto será posible mediante el uso de tecnologías y métodos de geolocalización.

La característica principal de este sistema es la conectividad entre la víctima y el agresor con el policía más cercano a través de alertas generadas cuando se produzca la violación del perímetro de seguridad de la víctima, permitiendo la rápida acción del policía y evitar una posible agresión.

A lo largo de este trabajo se desarrollarán las aplicaciones para la víctima, el agresor y policía. Estas aplicaciones serán desarrolladas en dispositivos móviles Android, aprovechando las herramientas que proveen estos dispositivos de bajo costo. Así mismo, se implementarán alertas dentro de una arquitectura fog computing utilizando el paradigma de procesamiento de eventos complejos.

Índice general

Índice de figuras

Índice de cuadros

Índice de Acrónimos

API Application Programming Interface

CEPAL Comisión Económica para América Latina y el Caribe

FCM Firebase Cloud Messaging

GPS Global Positioning System

GPRS General Packet Radio Service

HTTP HyperText Transfer Protocol

HTTPS HyperText Transfer Protocol Secure

IMEI International Mobile Equipment Identity

IMSI International Mobile Subscriber Identity

IoT Internet of things

JSON JavaScript Object Notation

MAC Media Access Control

MIT Massachusetts Institute of Technology

MIMP Ministerio de la Mujer y Poblaciones Vulnerables

mAh Miliamperio-hora

OECD Organización para el Desarrollo y Coperación Económica

RUP Rational Unified Process

SSL Secure Sockets Layer

SQL Structured Query Language

TLS Transport Layer Security

URL Uniform Resource Locator

XML Extensible Markup Language

Capítulo 1

Introducción

En este capítulo introductorio se comenta sobre las motivaciones que han llevado al desarrollo de este trabajo, además de los objetivos que se desean alcanzar; mostrando finalmente la estructura de la memoria.

1.1. Motivación

Hoy en día existen muchos problemas sin solución en nuestra sociedad, siendo uno de estos problemas la violencia de género. La triste realidad es que América Latina y el Caribe tienen la segunda tasa más alta de feminicidios de todas las regiones del mundo, la cual sólo es superada por África.

A nivel mundial en el 2014, según la Organización para el Desarrollo y Coperación Económica (OECD), se han registrado un 38 % de mujeres que han declarado ser víctimas de violencia en América Latina y un 36 % en nuestro país [1].

En el 2016, la Comisión Económica para América Latina y el Caribe (CEPAL) presentó estadísticas sobre violencia de género, en la cual el Perú ocupa el sétimo lugar entre los países de la región con los índices más alarmantes [2].

Viendo las estadísticas de los últimos años dadas por el Ministerio de la Mujer y Poblaciones Vulnerables (MIMP), el registro de feminicidios en nuestro país asciende a 1073 mujeres que han perdido la vida entre enero del 2009 y junio del 2018, ocurriendo 124 y 121 casos, en el 2016 y 2017 respectivamente [3]. Esto es un alarmante crecimiento del 20 % en comparación del 2015.

En países como en India ya se están abordando temas de violencia de género para tratar de frenar de alguna manera esta situación crítica. Para afrontar este problema se han propuesto conferencias y plataformas de programas de información, todas estas con un elemento en común: Seguridad en una Smart City [4]. Es decir, integrar el concepto de seguridad ciudadana con aplicaciones de monitorización de una ciudad en tiempo real [5] recolectando información de sensores a través del Internet de las Cosas(IoT).

De esta forma, se busca a través de este trabajo disminuir drásticamente estas lamentables estadísticas en el Perú, mediante el diseño y desarrollo un sistema que permita monitorizar en todo momento a personas [6] integrando las relaciones víctima, agresor y policía.

A grandes rasgos, este sistema permitirá que las personas que hayan sufrido violencia ya sea hombre o mujer, puedan tener la seguridad de mantenerse alejado a una distancia prudente de su agresor y en caso se rompa este perímetro, alertar al policía más cercano permitiendo su rápida acción contribuyendo a disminuir la cantidad de agresiones.

Bajo este contexto, se comenzó a definir los objetivos a alcanzar en este trabajo para poder brindar un ambiente de seguridad y tranquilidad a la víctima.

Un gran aliado para este trabajo será el desarrollo sobre dispositivos móviles Android ya que estos dispositivos aparte de ser de muy bajo costo, proveen herramientas como el sensor *GPS* que permiten localizar a una persona en tiempo real.

1.2. Objetivos

El objetivo principal de este trabajo consiste en el diseño y desarrollo de un sistema de localización para víctima, agresor y policía. Este trabajo se puede puede dividir en objetivos menores para poder ofrecer un mayor detalle.

- **Desarrollar una arquitectura distribuida tipo fog computing.** Un objetivo importante será desarrollar un sistema distribuido fog

computing en contraposición de una arquitectura centralizada como cloud computing.

- **Utilizar una metodología para el proceso de creación del sistema.**

  Se tiene como objetivo la utilización de una metodología para el desarrollo del sistema teniendo como principal resultado obtener un plan formal de trabajo que permita la trazabilidad del sistema.

- **Desarrollar la Ingeniería del Software.**

  Este punto tiene como fin describir el proceso de Ingeniería de software del sistema, haciendo énfasis los procesos fundamentales como el entorno y descripción del sistema, análisis de requisitos y diseño e implementación del sistema.

- **Utilizar herramientas para sistemas en tiempo real.**

  Se busca implementar el sistema bajo tecnologías que permitan controlar actividades críticas del sistema en pequeños intervalos de tiempo.

- **Implementar la transferencia segura de datos en el sistema.**

  Se realiza este trabajo con el fin de desarrollar un sistema consistente y robusto sin dejar de lado la protección de datos de los usuarios.

- **Desarrollar aplicaciones móvil en Android.**

  Se busca aprovechar los recursos brindados por dispositivos móviles de bajo costo como el uso del sensor integrado *GPS*.

- **Optimizar recursos utilizados por el sistema.**

  Este punto tiene como fin reducir drásticamente el consumo energético del dispositivo móvil y consumo de memoria.

1.3. Estructura de la Memoria

A fin de proporcionar al lector una idea global del contenido de esta memoria, a continuación se redacta una breve descripción sobre cada uno de los capítulos:

Capítulo 1: Introducción
En este capítulo introductorio se comentan sobre las motivaciones que han llevado al desarrollo de este trabajo, además de los objetivos que se desean alcanzar; mostrando finalmente la estructura de la memoria.

Capítulo 2: Estado del Arte
En este apartado se analizarán algunos trabajos similares que hayan marcado un precedente y que han alimentado la motivación en la realización de este trabajo. Se tendrá una conclusión acerca de lo que se pretende hacer en este desarrollo en contraste a los trabajos analizados en este capítulo.

Capítulo 3: Metodología y Herramientas
En este capítulo se describirá la metodología usada a lo largo del desarrollo de este trabajo y las herramientas que han sido usadas durante la implementación del sistema.

Capítulo 4: Ingeniería del Software
En este punto se expondrán los puntos más resaltantes de la Ingeniería del software utilizados para sentar las bases de este trabajo. Teniendo como fases: el entorno y descripción del sistema, análisis de requisitos, modelado de datos, diseño e implementación.

Capítulo 5: Estructura de SAVIAS
En este apartado se analizarán los componentes de SAVIAs y su interacción entre ellos, así mismo, se detallará cómo SAVIAs realiza los procesos de análisis de ubicaciones en tiempo real y envío de alertas a policías más cercanos.

Capítulo 6: Casos de estudio

En este capítulo se mostrarán las funcionalidades implementadas en las aplicaciones para los roles de víctima, agresor y policía que han sido desarrolladas a lo largo de este trabajo.

Capítulo 7: Conclusiones y trabajo a futuro

En este punto se expone una observación general de este trabajo y todas las conclusiones que se han podido rescatar de nuestra implementación. Además, desarrollar diferentes propuestas que pueden ser interesantes a desarrollar en futuro.

Capítulo 2

Estado del Arte

Una forma de tener una idea más clara sobre lo que se plantea realizar en este trabajo, es investigar sobre trabajos similares o previos que posean características importantes que puedan ser útiles y dar una mejor idea en cuanto a funcionalidad.

Es así que gracias a esta investigación y al tener información acerca de estos trabajos, se puede tener una idea mucho más definida acerca de características y funcionalidades que permitirán añadir y/o descartar funcionalidades previamente pensadas para el desarrollo de este trabajo.

El siguiente paso será discutir desarrollos sobre arquitecturas existentes, cloud y fog computing, que tienen relación directa con la idea principal de nuestro trabajo y que nos hayan ayudado a afrontar la implementación.

Adicionalmente, se mostrará algunas aplicaciones móviles más resaltantes orientadas a la prevención de violencia de género.

2.1. Sistemas de localización Víctima - Agresor

El sistema de localización Víctima-Agresor en ambientes indoor y outdoor [7] está orientado hacia personas que gozan de detención domiciliaria o movilidad restringida; este hecho está permitiendo una rápida localización del agresor.

A su vez, este sistema realiza una monitorización en tiempo real y garantiza a la víctima que su agresor se mantenga a una distancia prudente, ofreciéndole tranquilidad.

Para este sistema se ha diseñado una arquitectura cloud que consta de dos plataformas, una de estas es la plataforma de monitorización y comunicación basado en sensores *GPS*, permitiendo interactuar en entornos abiertos y cerrados. Por otro lado, posee una plataforma web que permite informar a las víctimas si su agresor ha roto los protocolos de distancia máxima.

Uno de los principales conceptos que rige el diseño de sistemas de monitorización es evitar la fuga del agresor, es por eso, la existencia de sensores *GPS* como teléfonos, pulseras electrónicas, etc. que a pesar de sus limitaciones en ambientes indoor, cumplen con el propósito fundamental que es localizar al agresor en cualquier momento.

Tal como se muestra en la Figura 2.1, el sistema de localización cuenta con tres tipos de dispositivos móviles, el de la víctima, el del agresor y el faro, y dos servidores, un servidor web y un servidor encargado de la recepción de la información de los faros y de los dispositivos móviles.

El funcionamiento interno del sistema es el siguiente: al agresor se le asigna un dispositivo no removible y a la víctima un dispositivo móvil con el software del sistema.

Como ya se ha dicho este sistema permite obtener la posición en todo momento de los dispositivos involucrados, en este caso los del agresor y la víctima; creando previamente sus roles y privilegios en el sistema, como su relación Víctima-Agresor y su distancia máxima legal.

Estos datos de localización de la víctima y el agresor llegan al servidor encargado de la recepción y se encarga de actualizar su posición en la base de datos.

Una vez ingresado los datos, se garantiza poder calcular de manera permanente la distancia entre ellos y permite que al ser esta distancia menor a su distancia máxima legal, esta ruptura de distancia se manifiesta mediante un mensaje de texto alertando a la víctima indicándole la presencia de su agresor y en caso de ser reiterativo se enviará un mensaje al administrador del caso, quién se encargará de tomar las acciones que considere adecuadas.

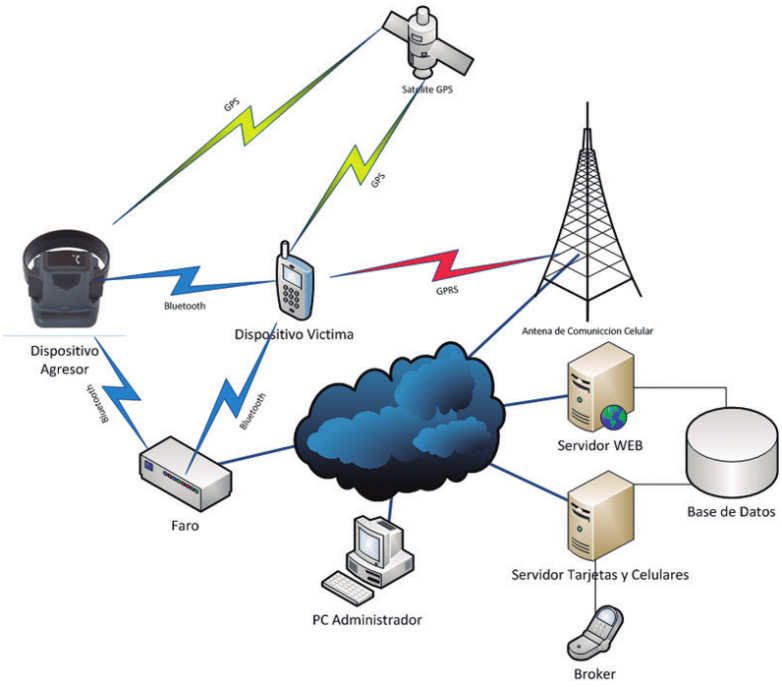

FIGURA 2.1: Arquitectura cloud del Sistema Víctima-Agresor. Fuente: Revista S&T.

2.2. Detección de Violencia de Género mediante pulsera GPS en España

Se ha implementado desde el año 2009 en varias comunidades de España. Las primeras en ponerlo en funcionamiento han sido Cataluña, Baleares y Madrid, aunque con el paso de los meses varias otras se han unido al uso de estas pulseras para intentar combatir la violencia de género.

Esta pulsera electrónica (ver Figura 2.2) busca la efectividad de la prohibición de aproximación, la tele-asistencia, que ofrece una atención más completa a las mujeres con orden de protección.

El sistema de detección de proximidad [8] está compuesto por dos equipos, el de la víctima y el del agresor. La víctima porta un aparato, con la apariencia y el tamaño de un teléfono móvil que emite una aguda señal de aviso en el caso de que el agresor rompa la barrera de los 500 metros de distancia con respecto a la víctima. En el instante en que se emite esa señal, se está produciendo una llamada de aviso a la central de emergencias.

FIGURA 2.2: Pulsera electrónica para la lucha de violencia doméstica en España. Fuente: *SINC*

2.3. App-Elles, la aplicación para mujeres víctimas de violencia

Este sistema desarrollado por Association Resonantes [10], tiene como principales características alertar a los contactos más cercanos a la víctima en caso de una emergencia y obtener ayuda profesional e información acerca de temas de violencia.

Su principal característica se da frente a una emergencia, la víctima genera una alerta mediante el uso de una aplicación móvil y ésta llega a sus contactos más cercanos, activando inmediatamente el GPS y ubicándola en un mapa, y de igual manera el micrófono es activado enviando un mensaje de voz en vivo.

Esta aplicación tiene la peculiaridad que puede ser sincronizada con un brazalete y cumplir con todas las funcionalidades sin la necesidad de utilizar el smartphone.

La arquitectura Cloud de App-Elle está representada en la figura 2.3 en la que puede verse la interacción entre los componentes que la forman. Por ejemplo se muestra la integración entre el brazalete y la aplicación móvil mediante el sistema Need, y del mismo modo se ve reflejada la comunicación de la aplicación móvil con el sistema de alerta WaryMe y el web services Odiwi.

FIGURA 2.3: Arquitectura Cloud de App-Elle. Fuente: *Association Resonantes*.

2.4. Aplicaciones móviles relacionadas a la prevención de violencia de género

Las aplicaciones que se listarán a continuación tienen como principal propósito servir de soporte e información para casos similares de violencia como agresión sexual, acoso y abuso.

2.4.1. Hands Away

Esta aplicación móvil [11] fue diseñada específicamente para combatir la violencia de género contra las mujeres en el espacio público debido a la gran cantidad de acoso y abuso sexual en Francia.

Sus funcionalidades más resaltantes son:

- Crear aletas precisas y geolocalizadas en caso de agresión sexual.
- Testificar sobre alguna agresión sexual que se ha experimentado o presenciado.

- Ejercer el rol de "street angel"que permita traer apoyo o ser testigo de un ataque de agresión sexual, creando un entorno de diálogo con la víctima.

2.4.2. AlertCops, canal interactivo para alertar sobre delitos

La aplicación móvil AlertCops [9] impulsada por el Ministerio del interior de España, tiene como objetivo principal que el ciudadano tenga acceso a los servicios públicos de seguridad. Es decir, permitirá la participación directa de las personas en seguridad ciudadana mediante alertas hacia las Fuerzas y Cuerpos de Seguridad de Estado(Policía y guardia civil), estas alertas pueden ser sobre un acto delictivo, violencia de género o incidencia de seguridad del que está siendo víctima o testigo.

La interfaz que muestra las distintas opciones de AlertCops se puede ver en la figura 2.4.

AlertCops tiene como principales beneficios:

- Facilitar un nuevo canal de comunicación entre los ciudadanos y las Fuerzas y Cuerpos de Seguridad de Estado.
- Mejorar los tiempos de respuesta al ciudadano y del proceso de información.
- Agilizar el protocolo de demanda de información y respuesta, obteniendo desde el inicio información del demandante de ayuda, tales como: posicionamiento, tipo de incidente que está sufriendo u otros datos relevantes

FIGURA 2.4: Aplicación móvil AlertCops para seguridad ciudadana. Fuente: *Ministerio del Interior de España.*

2.4.3. I'm Here For You

Esta aplicación [12] desarrollada por la Universidad de Loyola de Chicago para dispositivos iOS tiene como funcionalidades resaltantes:

- Alimentar a una base de datos acerca de situaciones de violencia, stalking y ataques sexuales.
- Estar conectado directamente a los servicios y recursos acerca de violencia.
- Obtener información como ayudar a alguien que ha experimentado violencia.

2.4.4. Aplicaciones fuentes de información: iMatter, GVA, Ni más ni menos, Domestic Violence Prevention, R.I.S.E Social Media App

Estas aplicaciones se caracterizan por ser fuentes de información para la prevención acerca de la violencia de género.

R.I.S.E Social Media App [13] desarrollada por la organización OCTEVAW, actualmente disponible sólo para iOS. Agrupa a un grupo de organizaciones dedicadas a erradicar la violencia hacia la mujer, a través del liderazgo, educación, vocación y acción política orientado hacia la persona que ha experimentado abuso. Esta aplicación es un medio social que ayuda a la prevención de violencia.

Funcionalidades a resaltar:

- **Intervención:** una persona es testigo de un incidente de violencia y no sabe que hacer, mediante esta aplicación se puede cargar una base de datos de posibles situaciones y consejos para las intervenciones y apoyo.
- **Soporte:** se tiene acceso a una extensa base de datos y ubicaciones de los servicios de apoyo cerca de donde se encuentra.
- **Comunidad:** permite compartir opiniones y preguntas sobre diversos temas.
- **Educación:** compartir mensajes en redes para educar al mundo sobre la violencia de género.
- **Información:** permite acceder a recursos que permite ayudar a personas que luchan contra la violencia de género.

Particularmente, iMatter [14] provee de actualizaciones regulares de series de mensajes positivos, videos y preguntas que dan a conocer signos de alerta sobre cualquier indicio de violencia de género. También provee servicios de apoyo a personas que han experimentado situaciones de violencia.

Así mismo, GVA [15] realiza una labor similar a iMatter siendo de carácter informativa pero profundiza en temas de actualidad, sugiriendo temas de interés para personas identificadas con este tema.

Por último, Domestic Violence Prevention [16] realiza tareas de capacitación en temas como identificar factores asociados a convertirse en agresor, identificar formas de violencia que afecta a mujeres y niños e identificar tácticas y señales utilizadas por los agresores en casos de violencia domestica. Contiene recursos y fuentes donde se puede extraer información acerca de estos temas. Adicionalmente, posee una sección de contactos de emergencia a disposición de los usuarios las 24 horas del día los 7 días a la semana, como por ejemplo: la Línea Nacional de Violencia Doméstica.

2.5. Arquitectura Fog Computing

Fog computing es una tecnología emergente dentro del Internet de las cosas [45] y puede ser considerado como una extensión natural del cloud computing. La principal característica de una arquitectura Fog es que cada componente toma decisiones independientes del Cloud, bajo este enfoque se busca optimizar y reducir el consumo de energía [46] . Este tipo de paradigma está dividido en dos niveles: core y edge.

Básicamente, el core level está formato por los principales componentes de un entorno cloud, es decir, una base de datos, aplicación web, brokers, etc. [47] y el edge level está formado por los dispositivos periféricos que forman la arquitectura, por ejemplo sensores inalámbricos y dispositivos móviles [48, 49].

El enfoque Fog Computing ha sido implementado en aplicaciones de análisis de datos para realizar prediciones y toma de decisiones en tiempo real.

2.6. Conclusiones

Después de hacer una breve descripción de aplicaciones y sistemas centralizados (cloud computing) que pueden contribuir a este trabajo se

pueden adoptar las siguientes características:

- Localización mediante dispositivos móviles.
- Sistema víctima - agresor en tiempo real con perímetros de seguridad.
- Emisión de señal a una central de emergencias del sistema de violencia de género en España.
- Comunicación directa de un ciudadano con las fuerzas policiales mediante alertas desde un dispositivo móvil.

A diferencia de los trabajos anteriormente comentados, en este trabajo se busca:

- Desarrollar un sistema que involucre la interacción de tres roles: víctima, agresor y policía.
- Implementar una arquitectura fog computing que realice monitorización en tiempo real de estos roles y que permita generación y recepción de alertas.
- Interacción víctima - policía más cercano mediante alertas permitiendo un rápida acción.
- Como se ha podido observar las aplicaciones móviles que hoy existen son de carácter preventivo e informativo. En este trabajo se pretende realizar aplicaciones que trabajen en la fase de ejecución de una orden de alejamiento legal entre víctima y agresor.
- Desarrollar aplicaciones móviles con distintas funcionalidades para los roles víctima, agresor y policía.

Capítulo 3

Metodología y Herramientas

En este capítulo se definirá la metodología de trabajo que se adoptará durante el desarrollo, las herramientas y tecnologías utilizadas para la construcción del sistema.

3.1. Metodología

Como se ha descrito anteriormente en las secciones de Objetivos y Conclusiones del capítulo de Estado del Arte, secciones 1.2 y 2.6 respectivamente, este sistema consistirá fundamentalmente de dos plataformas: una plataforma web y otra móvil. Bajo este concepto, se definirá una metodología de desarrollo de software adecuada que permita seguir el desarrollo de este sistema de manera estructurada.

Las metodologías de desarrollo de software nacen debido a la necesidad de estructurar, planificar y controlar el proceso del desarrollo del software de manera formal. La elección de la metodología de desarrollo a usar depende de cuán claro se tiene el producto final.

En este trabajo se utilizará una metodología híbrida [17] propiamente diseñada entre la metodología *Rational Unified Process (RUP)* y una metodología Ágile como lo es *SCRUM*.

En este caso se tiene claro la idea de un primer prototipo con funcionalidades básicas para lo cuál necesitaremos de la metodología *RUP* y para luego ir agregando nuevas funcionalidades se utulizará la metodología *SCRUM* generando entregas continuas de valor para el cliente.

Una de las razones que inclinaron a escoger la metodología *SCRUM* es que el desarrollo de software para dispositivos móviles es muy diferente al software tradicional en muchos aspectos, ya que deberá satisfacer de manera especial los requisitos y restricciones convirtiéndolos en retos para el equipo de diseño y desarrollo.

Los retos que un diseño de software para dispositivos móviles [18] debe alcanzar son los siguientes:

- Problemas de establecimiento de comunicación.
- Cuestiones de portabilidad.
- Cuestiones de experiencia de usuario.
- Trabajar sobre estándares, protocolos y tecnologías de redes.
- Privacidad especial y necesidades personalizadas.

La utilización de esta metodología tiene por finalidad solucionar los puntos expuestos anteriormente. *RUP* se caracteriza por ser una metodología formal, que tiene como objetivo producir documentación y obtener un mayor beneficio al manejar y guiar el proceso de diseño e implementación del sistema. En contraste a *RUP*, una de las características importantes que tiene *SCRUM* es la realización de reuniones continuas sobre la documentación, que en realidad es poca; haciendo partícipe al cliente de manera activa, aportando nuevas funcionalidades y permitiendo una detección temprana de errores en los requisitos o implementación.

Es por esto, que para la etapa de análisis de requisitos y diseño del sistema se ha utilizado la metodología *RUP*, apoyándonos en la capacidad de documentación que lo caracteriza. Es decir, se generará artefactos como especificaciones de caso de uso, diagramas de clases y de componentes para el primer prototipo que será nuestro producto mínimo viable.

Una vez realizada esta etapa, se adopta la postura de la metodología *SCRUM*, trabajando con el cliente de manera progresiva y conjunta. Se realizan prototipos sobre el producto mínimo viable que serán probados por el cliente

en reuniones cada quince días, esto permitirá que el producto final se ajuste de manera rápida a la visión y expectativas del cliente.

En el apéndice C se da un mayor detalle acerca del backlog y sprints utilizados en esta metodología.

3.2. Herramientas y tecnologías

A continuación, se muestran las tecnologías y herramientas que se van a utilizar a lo largo de este trabajo. Se hará una breve reseña por cada una de ellas resaltando sus principales características.

3.2.1. Android

Android [19] es un sistema operativo basado en el núcleo de Linux. Es un sistema operativo personalizable y fácil de utilizar que incluye una gran cantidad dispositivos en todo el mundo. Siendo una de las plataformas que posee la mayor cantidad de desarrolladores.

Android es muy popular gracias a que pone a completa disposición todas las herramientas para la creación de aplicaciones, manipulación de hardware de cada dispositivo. También permite la rápida adaptación de las aplicaciones a diversos dispositivos de distintas características.

Adicionalmente, contribuye a desarrollar eficientemente aplicaciones a través de Android Developoer Tools, ya que ofrece diversas características trabajando con lenguaje de programación JAVA.

Este sistema operativo móvil ha sido el elegido para el desarrollo de las aplicaciones móviles que se desarrollarán en este trabajo.

3.2.2. Android Studio

Android Studio [20] es el editor oficial de Android. Tiene como propósito construir aplicaciones de manera rápida y con gran calidad para dispositivos Android.

Dicho de otra manera, el objetivo de Android Studio es la automatización del proceso de construcción del software. Puede configurar un proyecto para incluir las bibliotecas locales y externas. También, tiene como ventaja instalar prototipos de aplicaciones en un emulador mucho más rápido que en un dispositivo real.

Las aplicaciones móviles implementadas en este trabajo serán desarrolladas a través de este editor.

3.2.3. Localización mediante satélites y alternativas

Global Positioning System (GPS) es el sistema de navegación portable de algunos dispositivos que utiliza una red de satélites alrededor del planeta permitiendo su geolocalización.

Los dispositivos Android que se utilizarán en este desarrollo poseen un sensor que recepciona señales desde los satélites. Esta señal permite al dispositivo calcular cuán lejos se encuentra del satélite basado en el tiempo que demora en llegar la señal. Combinando las señales de almenos cuatro satélites, el dispositivo calcula su posición.

Otro método de obtener la ubicación de un dispositivo, es usando triangulación de antenas de telefonía móvil [21], el cual trabaja de manera similar al *GPS* pero utiliza las señales de estas antenas. Aunque la ubicación obtenida no es la más precisa, es la más rápida y consume menos batería. Una variante de este método es utilizando redes Wi-Fi.

Android tiene la capacidad de obtener ubicaciones mediante estos métodos, agrupándolos en dos grandes categorías *GPS_PROVIDER* para ubicación a través de *GPS* y *NETWORK_PROVIDER* para posicionamiento a través de triangulación. Siendo estas dos formas de ubicación en tiempo real las que serán utilizadas por las aplicaciones desarrolladas.

3.2.4. Node.js

Básicamente, Node.js [22] es una librería de entorno de ejecución de E/S dirigida por eventos. Tiene como principal característica que se ejecuta sobre el intérprete de JavaScript creado por Google V8.

La utilización de Node.js hoy en día se ha incrementado debido a la capacidad que posee de realizar tareas asíncronas a un gran velocidad y rendimiento. Resalta por razones como:

- Desarrollo más rápido.
- Aplicaciones más rápidas mejorando la experiencia de usuario.
- Menor coste de infraestructura.
- Flexible.
- Gran cantidad de módulos por ejemplo: *HyperText Transfer Protocol Secure (HTTPS)*, *Secure Sockets Layer (SSL)*.

Se ha elegido Node.js como servidor web debido a que en este desarrollo se trabajará con un gran número de clientes, que realizarán diversas tareas de registro de información y consultas en simultáneo.

A diferencia de servidores web como Java y PHP que para una nueva conexión genera un hilo de aproximadamente 2MB de memoria, teniendo como limitante al tamaño de memoria RAM del servidor; Node.js resuelve el problema de la falta de memoria cambiando la forma en que se realiza una conexión. En lugar de generar un nuevo hilo para cada conexión, se ejecuta un nuevo evento soportando de decenas de miles de conexiones concurrentes.

Módulos utilizados en este trabajo junto a Node.js

1. **Express** [23]: es una infraestructura de aplicaciones web Node.js mínima y flexible que proporciona un conjunto sólido de características para las aplicaciones web y móviles.

Esta infraestructura posee métodos de enrutamiento de *Uniform Resource Locator (URL)* y métodos como *GET, POST* que serán utilizados con frecuencia en este trabajo.

2. **Assert** [24]: este módulo ofrece un conjunto sencillo de pruebas que se puede utilizar para probar inconsistencias al iniciar el servicio. Generalmente es para uso interno de Node.js.

3. **HTTPS** [25]: es el protocolo *HyperText Transfer Protocol (HTTP)* sobre *Transport Layer Security (TLS)/SSL*. Para Node.js se implementa como un módulo adicional.

4. **Body-parser** [26]: es un middleware que se utiliza para traducir la información que llega al servidor en un objeto *JSON*.

5. **Request** [27]: está diseñado para realizar llamadas *HTTP* de la manera más simple para Node.js.

6. **Mongodb** [28]: es el driver oficial de MongoDB para Node.js, ofrece una interfaz asíncrona entre el servidor y la base de datos MongoDB.

7. **Mysql** [29]: es el driver de Node.js para mysql. Posee licencia del *Massachusetts Institute of Technology (MIT)*. Tiene la capacidad que cada método que se invoca es ejecutado de manera secuencial.

8. **Geolib** [30]: este módulo ofrece realizar operaciones geoespaciales básicas como calcular la distancia entre dos puntos, convertir coordenadas decimales a sexagesimales y viceversa.

9. **Fs** [31]: permite la lectura de ficheros de manera asíncrona y síncrona.

3.2.5. JavaScript Object Notation

JavaScript Object Notation (JSON) [32] nació como una alternativa a *Extensible Markup Language (XML)* y una de las mayores ventajas que tiene su uso es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser

usado para el intercambio de información entre distintas tecnologías. Además debido a su naturaleza y al ser más compacto suele ser mucho más rápido trabajar con *JSON* antes que con *XML*.

El principal motivo por el que se ha decidido el uso *JSON* para esta implementación ya que es un formato diseñado para el intercambios de datos. *JSON* describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos.

3.2.6. HttpClient de Apache

El envío de datos en este trabajo es una parte muy importante ya que nuestras aplicaciones necesitan interactuar de manera continua con el servidor.

Una manera de enviar datos al servidor es mediante el cliente *HTTP* de Apache [33] para Android cuya librería es ***org.apache.http.****, esta librería provee la clase ***HttpClient*** que ayudará a hacer el envío de datos de una manera segura y eficiente.

Este servicio se ha implementado de manera que se pueda enviar datos a través de un Objeto *JSON* mediante una petición *HTTP* del tipo *POST*.

3.2.7. Google Maps API v2

Esta *Application Programming Interface (API)* [34] creado por Google permite interactuar con la herramienta de mapas en el dispositivo Android tal y como se muestra en la Figura 3.1.

Como principal características de este *API* podemos destacar:

- Integración con los servicios de Google Play (Google Play Services) y la Consola de *APIs*.
- Utilización a través de un nuevo tipo específico de fragment (MapFragment).

- Utilización de mapas vectoriales, lo que repercute en una mayor velocidad de carga y una mayor eficiencia en cuanto a uso de ancho de banda.
- Mejoras en el sistema de caché, lo que reducirá en gran medida las famosas áreas en blanco que tardan en cargar.
- Los mapas en 3D, es decir, podremos mover nuestro punto de vista de forma que lo veamos en perspectiva.

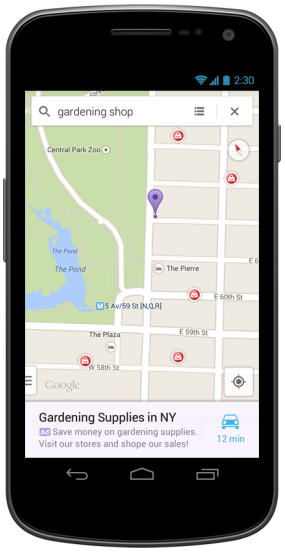

FIGURA 3.1: Google Maps en Android.
Fuente: Google Developers.

3.2.8. Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) una infraestructura [35] segura y escalable desarrollada por Google, que permite enviar información desde un servidor a un dispositivo y recibir mensajes desde otros dispositivos.

Haciendo una analogía, esta infraestructura trabaja de manera similar a un servicio de mensajería, el cual lleva la información hacia el dispositivo de destino.

Posee características importantes como:

- Envío de mensaje a un sólo dispositivo.

- Envío de mensaje a un grupo de dispositivos.
- Envío de mensajes a dispositivos pertenecientes a una categoría.

En este trabajo se ha utilizado este servicio debido a las características expuestas anteriormente, teniendo como ventaja alta compatibilidad con las herramientas descritas en este capítulo como por ejemplo *JSON*.

En la figura 3.2 se muestra un esquema general del funcionamiento de *FCM*.

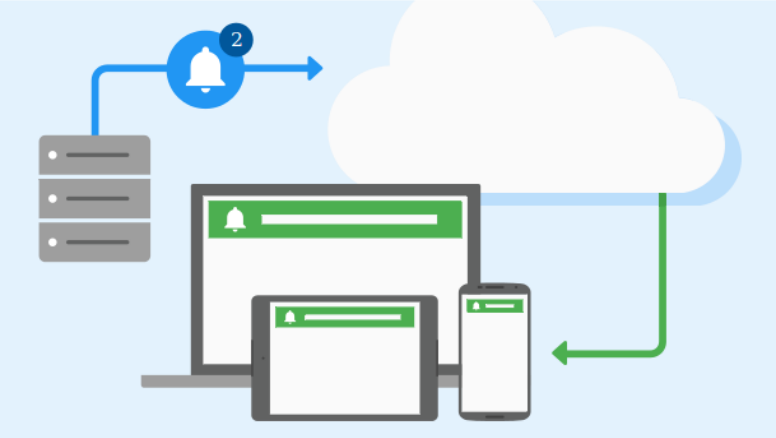

FIGURA 3.2: Esquema general de *FCM*.
Fuente: Google Developers.

3.2.9. SQLite

SQLite [36] es una herramienta que permite el almacenamiento de información de manera sencilla, eficaz y rápida en equipos con pocas capacidades de hardware, es decir, tiene una alta compatibilidad con dispositivos Android.

SQLite permite desarrollar consultas tipo *Structured Query Language (SQL)* desde las más básicas hasta las más complejas.

Aunque la velocidad de herramienta es gracias a la lectura y escritura directamente en ficheros, tiene como principal problema el almacenamiento en texto plano de la información.

3.2.10. MySQL

MySQL [37] es un sistema de administración de bases de datos relacional desarrollado por Oracle. Es una de la base de datos más populares del mundo, ya que es muy utilizado en aplicaciones web, como Joomla, Wordpress, Drupal y por grandes empresas como Wikipedia, Facebook, Google.

En este sistema es usado ya que existe una baja concurrencia en la modificación de datos y en la lectura de datos. Otra de las razones por la que es utilizada en este sistema es que permite gestionar de usuarios rápidamente.

3.2.11. MongoDB

MongoDB [38] es una base de datos no relacional, ágil, escalable que posee un gran rendimiento tanto para lectura como para escritura de una gran cantidad de datos.

Esta base de datos tiene la particularidad que los datos no se almacenan en tablas como en las bases de datos relacionales, en su lugar MongoDB guarda estructuras de datos en documentos similares a *JSON* con un esquema dinámico.

La principal característica por la que se ha decidido usar MongoDB es que en este trabajo se requiere un entorno de alto rendimiento o de alta concurrencia de acceso, es decir, se requiere un acceso rápido y con alta carga de consultas.

3.2.12. Apache Flink - CEP

Apache Flink [39] es un framework open-source orientado a eventos, siendo una de las herramientas de big data para aplicaciones distribuidas. Tiene como prinipal característica el procesamiento de flujos de datos, es decir, procesa millones de eventos a una baja latencia, manteniéndo los datos consistentes a una alta disponibilidad.

En este trabajo se ha trabajado con Flink CEP, que es la biblioteca de procesamiento de eventos complejos de Flink. Le permite detectar fácilmente ciertos patrones entre eventos complejos de un flujo de datos interminables,

dichos eventos pueden construirse a partir de secuencias coincidentes. Se ha utilizado esta librería para la implementación de alertas en tiempo real.

Capítulo 4

Ingeniería del Software

En este capítulo se entrará en detalle en la Ingeniería del Software que se ha desarrollado y seguido en este trabajo, en la cual se centrarán las bases para realizar una correcta implementación del software. Comprende secciones como entorno y descripción del sistema, análisis de requisitos y diseño e implementación del sistema.

4.1. Entorno y descripción del sistema

En la sección 1.1 se habló de la problemática existente en América Latina, en el Perú y de las cifras alarmantes de violencia de género contra la mujer. Aunque ya se ha mencionado a grandes rasgos bajo que circunstancias estará este sistema, en esta sección se ampliará la descripción del contexto.

Antes de describir los posibles escenarios, primero se debe listar los componentes que forman parte de este sistema. De manera básica, se trata de dispositivos móviles que obtienen información de su ubicación mediante *GPS* por ejemplo, y a su vez esta información es enviada a un servidor y éste le responde con un mensaje de confirmación de envío. Esto se puede ver en la figura 4.1 que muestra la interacción entre los componentes del sistema.

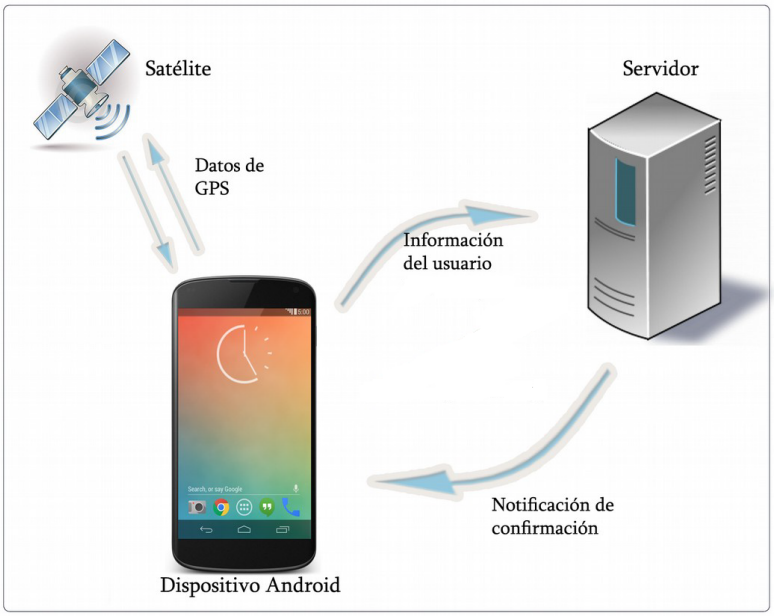

FIGURA 4.1: Interacción entre satélite, móvil y servidor. Fuente: Elaboración propia.

4.1.1. Descripción general del sistema

En este trabajo se presenta un sistema en tiempo real que se encargará de realizar los objetivos trazados anteriormente. Este sistema llamado *Surveillance and Alarm for gender VIolence Application*, *SAVIA* por sus siglas en inglés, está compuesto fundamentalmente de dos componentes: un servidor web llamado *SAVIAs* y un grupo de aplicaciones móviles que cumplen distintas funciones de acuerdo a los distintos roles que se tienen.

A continuación, se lista las aplicaciones móviles que formarán parte del sistema *SAVIA*. En primer lugar, se tienen las aplicaciones llamadas *SAVIApp* y *SAVIApol* para la víctima y el policía respectivamente, estas aplicaciones permitirán el envío de la ubicación cada minuto hacia *SAVIAs*. Caso especial es del agresor, ya que por ahora tendrá una aplicación llamada *SAVIAtt* en un dispositivo móvil, más adelante se reemplazará este dispositivo por un grillete electrónico que cumpla las mismas funciones de la aplicación.

La figura 4.2 muestra los aplicaciones por roles y su interacción con *SAVIAs*, las tres aplicaciones envían su ubicación pero sólo podrán recibir notificaciones de alertas *SAVIApp* y *SAVIApol*.

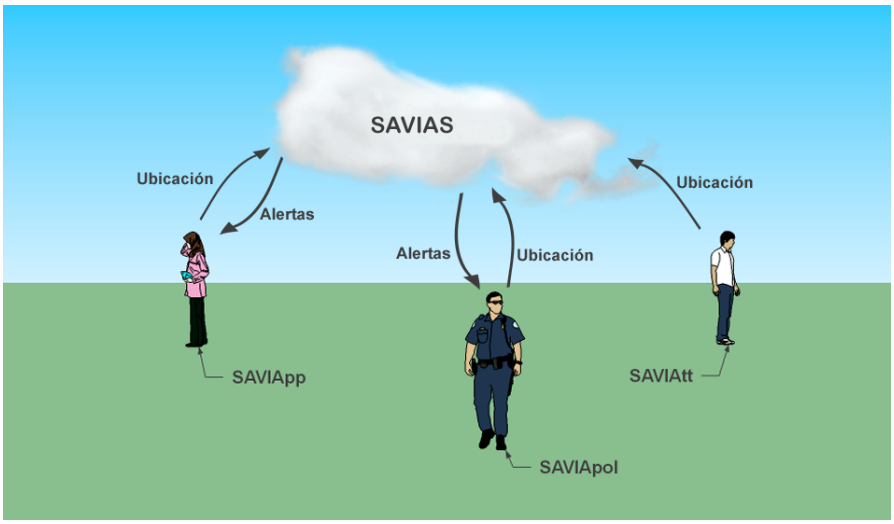

FIGURA 4.2: Descripción general del sistema SAVIA. Fuente: Elaboración propia.

El envío de ubicación está acompañado del envío de información del dispositivo que servirán para su identificación en *SAVIAs*, siendo estos datos los siguientes:

- Dirección *MAC* del dispositivo.
- Número *IMEI* del dispositivo.
- Número *IMSI* del dispositivo.

La extracción de la ubicación se realiza principalmente a través de dos métodos: *GPS* y *Triangulación*.

4.1.2. Escenario de acción de SAVIA

Como escenario principal(ver figura 4.3) en el que trabajará *SAVIA*, supone la existencia de una orden de alejamiento que prohíbe al agresor acercarse a menos de una cierta distancia a la víctima y que los usuarios se encuentran debidamente registrados en la base de datos de *SAVIAs*.

Para la explicación del escenario de acción de este sistema, se ha tomado como ejemplo un perímetro de seguridad de un kilómetro y adicionalmente, se ha definido un perímetro de prevención de dos kilómetros de radio alrededor de la víctima.

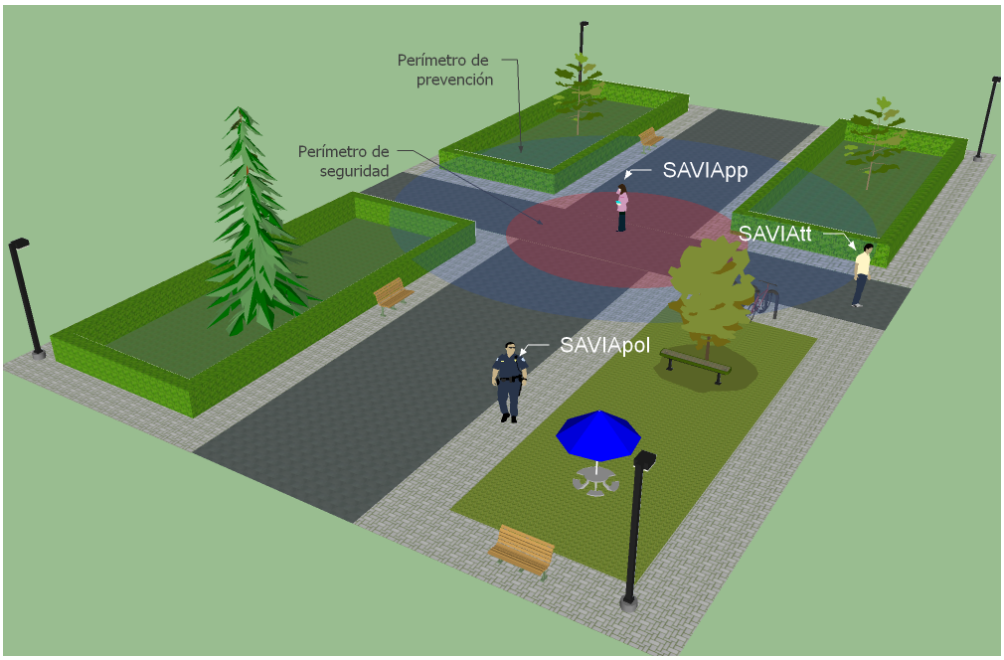

FIGURA 4.3: Escenario de acción de SAVIA.

Si *SAVIAs* detecta que el agresor se encuentra dentro del perímetro de prevención(región azul de la figura 4.3) activará automáticamente el sensor *GPS* tanto de la víctima como el del agresor, con el fin de ganar precisión para la monitorización de estas personas. Caso contrario, víctima y agresor se posicionarán bajo *Triangulación*, es decir, aproximar la posición del dispositivo mediante métodos de triangulación de antenas de telefonía móvil.

Estos datos de ubicación e información del dispositivos son enviados a *SAVIAs* en un objeto *JSON* mediante el protocolo *HTTPS*.

Simultáneamente al envío de datos, las aplicaciones móviles guardarán los datos enviados en una base de datos en el propio dispositivo utilizando el motor SQLite para Android. Esta base de datos será accedida sólo por un usuario administrador del sistema en caso sea necesario mediante una resolución legal.

Este sistema tiene la capacidad de enviar alertas automáticas en caso el agresor rompa el perímetro de seguridad(ingreso a la región roja de la figura 4.3). Estas alertas serán registradas en *SAVIAs* y a su vez, son enviadas al policía más cercano a la víctima, permitiendo su rápida acción.

SAVIApp también posee la posibilidad de enviar alertas al policía más cercano a través de un botón implementado en la aplicación móvil. Adicionalmente *SAVIApp* posee funcionalidades como:

- Ver registro de ubicación del usuario en una franja de tiempo.
- Ver posición de sus agresores en tiempo real a través de un mapa.
- Ver registro de ubicación de sus agresores en una franja de tiempo.
- Búsqueda de policías por kilómetros.

Por otra parte, *SAVIApol* permite visualizar un registro histórico de las alertas y monitorizar alertas en tiempo real. La monitorización de alertas tiene como condiciones que éstas deberán seguir activas y que hayan sido asignadas al policía que quiere visualizar alguna de ellas. Las alertas en tiempo real serán mostradas a través de un mapa posicionando a la víctima, agresor y policía en un misma sección de mapa, mostrando distancias actualizadas entre los implicados.

La aplicación para policía posee funciones adicionales como:

- Ver posición de todas las víctimas y agresores en tiempo real a través de un mapa.
- Ver registro de ubicaciones de víctimas y agresores en una franja de tiempo.
- Búsqueda de víctimas y agresores por kilómetros.
- Monitorización en tiempo real de personas previamente seleccionadas sin importar el rol que tengan a través de un mapa.

4.2. Análisis de requisitos

Un requisito [40] no es más que una condición o capacidad que debe tener el sistema para lograr los objetivos trazados.

Los requisitos se pueden clasificar en funcionales y no funcionales, pero para ofrecer un mejor entendimiento de estos se usará clasificación de requisitos según Pohl [41]. Esta clasificación divide a los requisitos no funcionales en requisitos de calidad y restricciones.

A continuación se describe de manera detallada los requisitos del sistema.

4.2.1. Requisitos funcionales

- **Obtención de la ubicación desde las aplicaciones *SAVIApp*, *SAVIAtt* y *SAVIApol*.**
  Obtener la posición del dispositivo en un intervalo de un minuto mediante *Triangulación* o *GPS*.

- **Envío de ubicación desde las aplicaciones *SAVIApp*, *SAVIAtt* y *SAVIApol* a *SAVIAs*.**
  Cada vez que se detecte una nueva ubicación, enviar esta ubicación a *SAVIAs*.

- **Envío de la información de los dispositivos agresor, víctima y policía a *SAVIAs*.**
  Al realizar el envío de ubicación, enviar la información del dispositivo como dirección *MAC*, número *IMEI* e *IMSI* a *SAVIAs*.

- **Almacenamiento de los datos enviados en los dispositivos agresor, víctima y policía.**
  Almacenar los datos enviados a *SAVIAs* en el dispositivo móvil de manera permanente y restringida.

- **Cambio automático de *Triangulación* a *GPS* y viceversa.**
  Permitir que las aplicaciones *SAVIAtt* y *SAVIApp* cambien automáticamente el modo de obtención de coordenadas a través de *Triangulación* y/o *GPS* según decida *SAVIAs*.

- **Recolección de la información que envían las aplicaciones a *SAVIAs*.**
  SAVIAs permitirá recolectar la información que se envía a través de las aplicaciones de manera automática.

- **Análisis de la información del dispositivo que envían las aplicaciones por parte de *SAVIAs*.**
  SAVIAs permitirá analizar de manera automática los datos de información del dispositivo para realizar la verificación de seguridad y evitar posible manipulación de las credenciales del dispositivo.

- **Análisis de cada ubicación enviada a *SAVIAs*.**
  SAVIAs analizará cada ubicación recibida a través de las aplicaciones. Este análisis permitirá el cálculo de la distancia entre víctimas y agresores, verificando los perímetros de seguridad y de prevención.

- **Envío de alertas a policías más cercano.**
  Al romperse algún perímetro de seguridad por parte del agresor, *SAVIAs* enviará automáticamente una alerta al policía más cercano a la víctima permitiendo su rápida acción.

- **Recepción de alertas por parte de *SAVIApol***
  SAVIAs permitirá el envío de alertas al policía más cercano, para esto *SAVIApol* debe recibir estas alertas generadas por el sistema.

- **Visibilidad del agresor por parte de la víctima.**
  SAVIApp tendrá la capacidad de ver la posición del agresor en tiempo real y en franjas de tiempo.

- **Visibilidad de policías por parte de la víctima.**
  SAVIApp tendrá la capacidad de ver la posición de policías en tiempo real.

- **Visibilidad de agresores y víctimas por parte del policía.**
  SAVIApol podrá monitorizar la posición de agresores y víctimas en tiempo real y en franjas de tiempo.

- **Controlar el zoom automático en la visualización de la información en mapas de las aplicaciones.**
  Manejar el zoom automático en las aplicaciones *SAVIApp* y *SAVIApol* que utilizan mapas permitiendo una visualización personalizada y cómoda.

- **Visualización de alertas a través de *SAVIApol*.**
  Los policías podrán visualizar las alertas generadas por *SAVIAs* en tiempo real a través de la aplicación *SAVIApol*.

4.2.2. Requisitos de calidad

- **Recolección sin interrupciones de ubicación y envío de datos a *SAVIAs*.**
  El servicio de envío y recolección de ubicación debe trabajar sin interrupciones aunque se cierren las aplicaciones. El envío de los datos se hará las 24 horas del día, 7 días a la semana, siendo un minuto el intervalo de envío.

- **Mantener actualizado el identificador único de dispositivo.**
  Se deberá tener actualizado los identificadores únicos de los dispositivos para recepción de alertas.

- **Servicio de recepción de alertas siempre activo.**
  Este sistema deberá garantizar que servicio de recepción de alerta siempre esté disponible en *SAVIApol*. Este servicio estará disponible las 24 horas del día, 7 días a la semana.

- **Interfaz de navegación para las aplicaciones *SAVIApp* y *SAVIApol*.**
  Implementación de una interfaz de navegación amigable y funcional para las aplicaciones *SAVIApp* y *SAVIApol*.

- **Controlar saturación de *SAVIAs* causadas por alertas.**
  Evitar la saturación de *SAVIAs* fijando un tiempo mínimo de tres minutos para cada envío de alerta por usuario, garantizando así que no se generen multiples alertas para una casuística.

- **Transferencia segura de datos entre *SAVIAs* y las aplicaciones.**
  Este sistema deberá garantizar la seguridad de la transferencia de información entre *SAVIAs* y las aplicaciones.

4.2.3. Restricciones

- **Aplicaciones móviles en dispositivos Android.**
  Para este sistema sólo se utilizarán dispositivos que tengan como sistema a **Android**, debido a su bajo costo y a la fácil utilización de recursos que brinda.

- **Requisitos mínimos de dispositivos Android.**
  Las aplicaciones desarrolladas deben tener como mínimo versión de **Android 4.2**. Adicionalmente, deberán de contar con sensor *GPS*.

- **Servidor web para *SAVIAs*.**
  SAVIAs tendrá como motor al lenguaje basado en Javascript, **Node.js** y tendrá como interfaz para el servicio web bajo el protocolo *HTTPS* al framework **Express**.

- **Gestor de base de datos en *SAVIAs*.**
  El servidor tendrá como gestores de base de datos a *MYSQL* y a *MongoDB*.

- **Requisitos mínimos de hardware de *SAVIAs*.**
  SAVIAs deberá trabajar sobre un sistema que permita realizar las tareas asignadas, para esto debe tener como requisitos mínimos de hardware:

  - Procesador core 2 dúo de 2.3 GHz.
  - Memoria RAM de 1GB.
  - Disco duro de 500GB.

- **Tiempo máximo de respuesta de *SAVIAs*.**
  El tiempo máximo de respuesta de *SAVIAs* deberá ser 10 segundos.

- **Formato para envío de datos a *SAVIAs*.**
  El formato que se utilizará para el envío de datos a *SAVIAs* será el formato *JSON* ya que describe los datos con una sintaxis dedicada que se usa para identificar y gestionar los datos. Tiene como ventaja su alta compatibilidad con **Android** y **NodeJS**.

- **Protección de datos del sistema.**
  Se tiene como restricción legal que la información en el almacenamiento en el dispositivo móvil y servidor estén protegidas mediante métodos que avalen la seguridad de estos datos.

4.3. Modelado de datos

En la sección 4.1 se detalló información acerca de las personas, objetos y aplicaciones que interactúan entre sí y tomando como referencia estos datos se mostrará la relación que existe entre ellos (ver figura 4.4). A continuación, se listan estos datos detallando lo que denotan para el sistema.

- Preferences:

  SERVICE_RUN: valor que indica el estado del servicio al iniciar la aplicación por primera vez.

  FCM_TOKEN: valor único que identifica al dispositivo para la recepción de alertas para el policía. En caso del agresor y víctima, éste tendrá el valor de **null**.

  ID_USER: identificador único del usuario en el sistema.

- Data:

  _id: identificador único de cada elemento de Data.

  latitud: latitud del dispositivo.

  longitud: longitud del dispositivo.

  MAC: dirección MAC del dispositivo.

IMEI: número IMEI del dispositivo.

IMSI: número IMSI del dispositivo.

TimeDate: hora en que se obtuvo la ubicación.

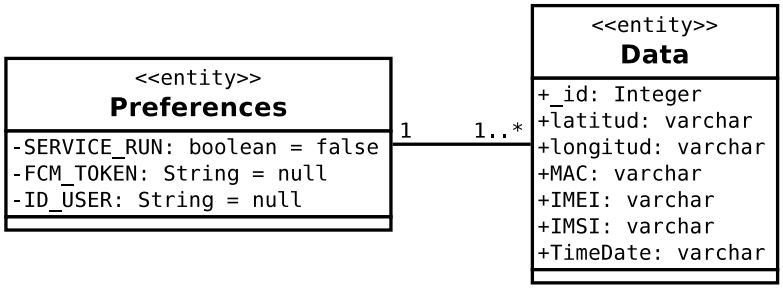

FIGURA 4.4: Relación entre preferences y data.

4.3.1. Actores

Los actores son las personas o entidades que participan en un caso de uso. Los actores de este sistema son:

- **Agresor**

  Este actor representa a la persona que tiene una orden de restricción legal hacia otra.

- **Víctima**

  Representa a la persona que ha sido víctima del agresor. Tiene privilegios de ver ubicación en tiempo real y en franjas de tiempo de su agresor.

- **Policía**

  Es aquel encargado de velar por la seguridad de la víctima. Tiene privilegios de ver ubicación en tiempo real y en franjas de tiempo de víctimas y agresores; como también puede monitorizar alertas en tiempo real.

4.3.2. Diagramas de casos de uso

A continuación se puede observar en las figuras 4.5, 4.6 y 4.7 los diagramas de caso de uso para agresor, víctima y policía respectivamente. En el Apéndice

A se puede encontrar las especificaciones de los casos de uso que detallan uno a uno los casos de uso para los distintos actores.

FIGURA 4.5: Diagrama de caso de uso para el agresor.

FIGURA 4.6: Diagrama de caso de uso para la víctima.

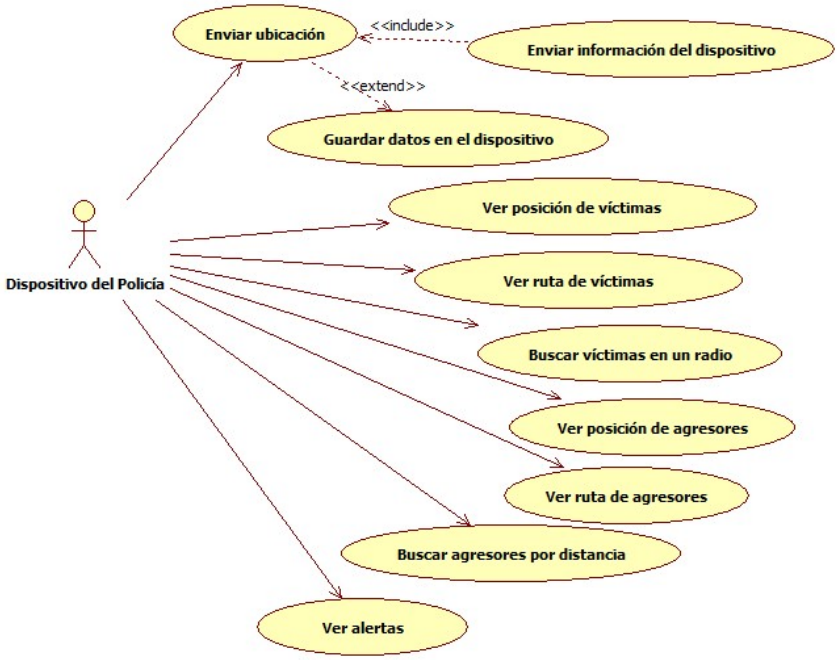

FIGURA 4.7: Diagrama de caso de uso para el policía.

4.4. Diseño e implementación

Una vez conocidos el entorno del sistema, la definición de los requisitos y las funcionalidades necesarios del sistema, en este apartado se busca ser más profundo en detalle acerca de la arquitectura, los componentes y entidades que formarán parte de este desarrollo.

4.4.1. Arquitectura de SAVIA

En este punto tendrá como finalidad detallar la estructura operacional de SAVIA, la cual se puede observar en la figura 4.8. La arquitectura de SAVIA está conformada por aplicaciones móviles para los roles de agresor, victima y policía, un servidor web cuyo entorno de desarrollo fue *Node.js*, una base de datos no relacional basada en documentos como *MongoDB* y una base de datos relacional como *MySQL*.

Adicionalmente se muestra un servicio de notificaciones *Firebase*, este servicio permitira a *SAVIAs* enviar notificaciones push los dispositivos móviles.

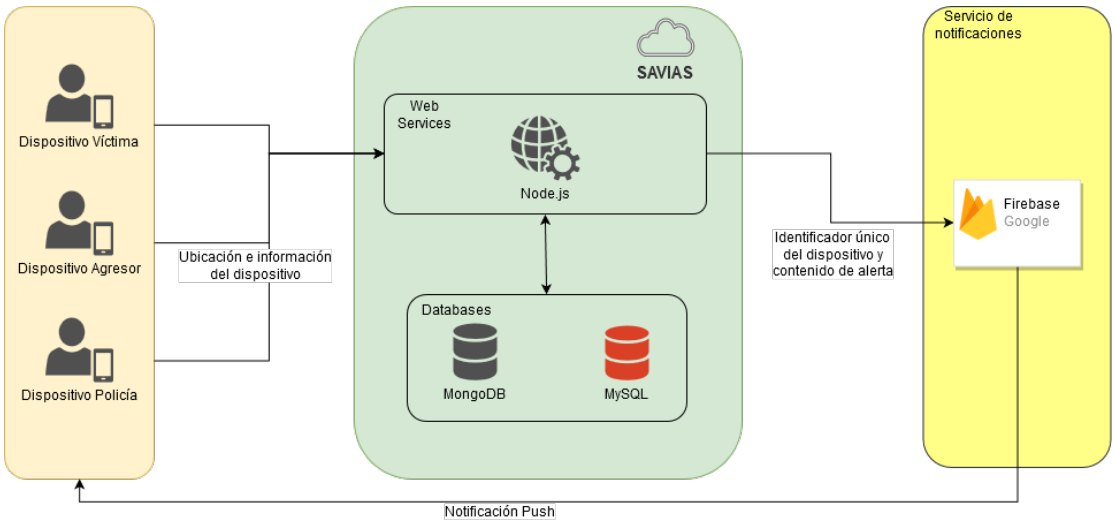

FIGURA 4.8: Arquitectura de SAVIA. Fuente: Elaboración propia.

4.4.2. Diagramas de diseño

En este punto se utilizarán diagramas que den niveles distinto de detalles como son los diagramas de componentes, diagramas de clases y diagrama de

secuencia dando detalles a alto y bajo nivel respectivamente.

Un diagrama de componentes representa cómo un sistema es dividido en componentes y muestra las dependencias entre estos componentes. Para visualizar la estructura de este sistema se hará uso de un diagrama de componentes.

Así mismo, los diagramas de clases muestran las clases del sistema, junto con sus atributos y operaciones, las interrelaciones que existen entre ellas. Son útiles para conocer el dominio del sistema y de ese modo, se pueda tener una visión clara y unificada de la estructura que tendrá nuestro sistema como se pueden observar en las figuras 4.10, 4.11 y 4.12 para las aplicaciones *SAVIAtt*, *SAVIApp* y *SAVIApol* respectivamente.

De esta misma manera, un diagrama de secuencia muestra las interacciones entre un conjunto de objetos de una aplicación a través del tiempo. Para este caso, se muestra el diagrama de secuencia 4.13 para el caso de uso "Ver ruta de agresores" del dispositivo del Policía, descrito en la sección A.3.

A continuación, se muestra la figura 4.9 que describe el diagrama de componentes que interactúan en la plataforma SAVIA. En este diagrama se puede ver que las aplicaciones *SAVIAs*, *SAVIAtt*, *SAVIApp* y *SAVIApol* tienen como única interfaz de comunicación el protocolo HTTPS, es decir, las aplicaciones móviles se comunican con *SAVIAs* exponiendo APIs bajo un servicio web implementado con el framework Express en Node.js.

Las aplicaciones móviles cuentan en su desarrollo con módulos en común los cuales son los siguientes: módulo de almacenamiento de base de datos en el propio dispositivo, módulo de obtención de información del teléfono (direcciones IMEI, IMSI y MAC), módulo obtención de coordenadas (latitud y longitud) y envío de objeto JSON mediante el protocolo HTTPS.

Para el caso de la aplicación *SAVIApp*, ésta cuenta con módulos de realizar búsquedas sobre determinados agresor y envío de alertas. Y la aplicación *SAVIApol* cuenta adicionalmente con los módulos de recepción de alertas, vizualización de alertas y permitir realizar búsquedas de personas en tiempo real y rutas en un determinado periodo de tiempo.

También resalta el componente Firebase Cloud Messaging que permite el envío de alertas a los dispositivos móviles y como se puede ver éste módulo interactúa directamente con *SAVIAs* y *SAVIApol*. En el siguiente capítulo detallaremos cómo interanctúan estos tres módulos para la generación y envío de alertas a dispositivos móviles.

Diagramas de componentes

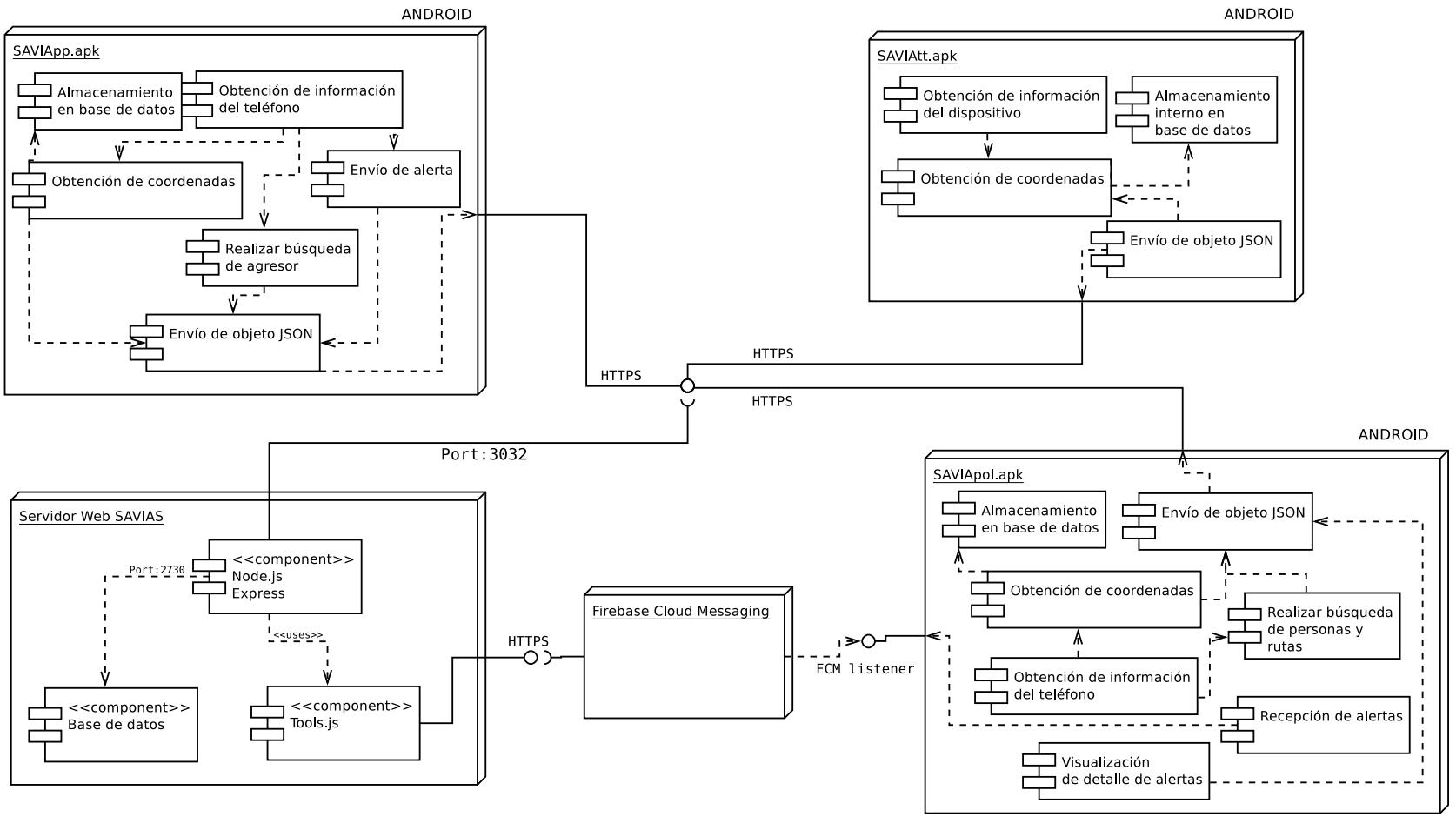

FIGURA 4.9: Diagrama de componentes que interactúan en el sistema *SAVIA*.

Diagramas de clases

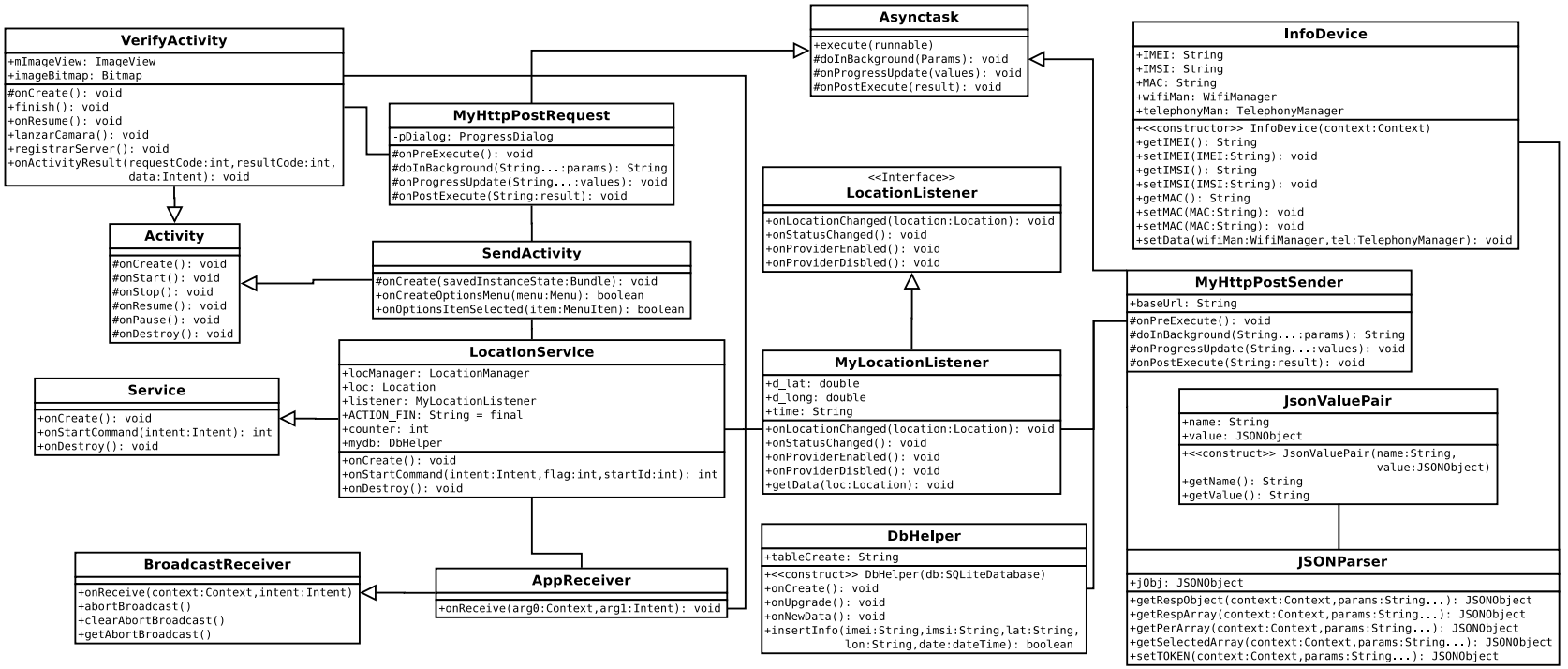

FIGURA 4.10: Diagrama de clases para *SAVIAtt*.

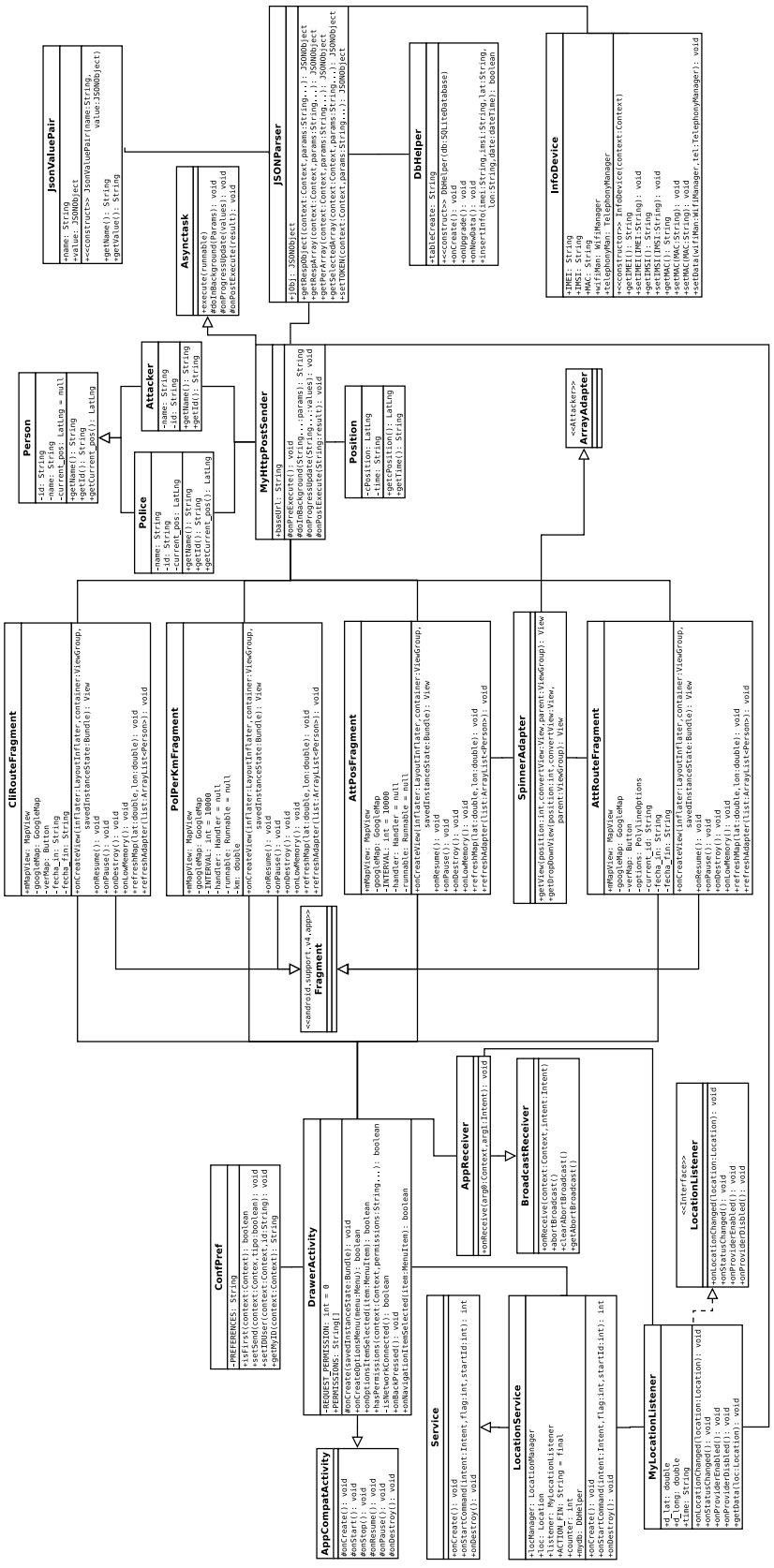

FIGURA 4.11: Diagrama de clases para *SAVIApp*.

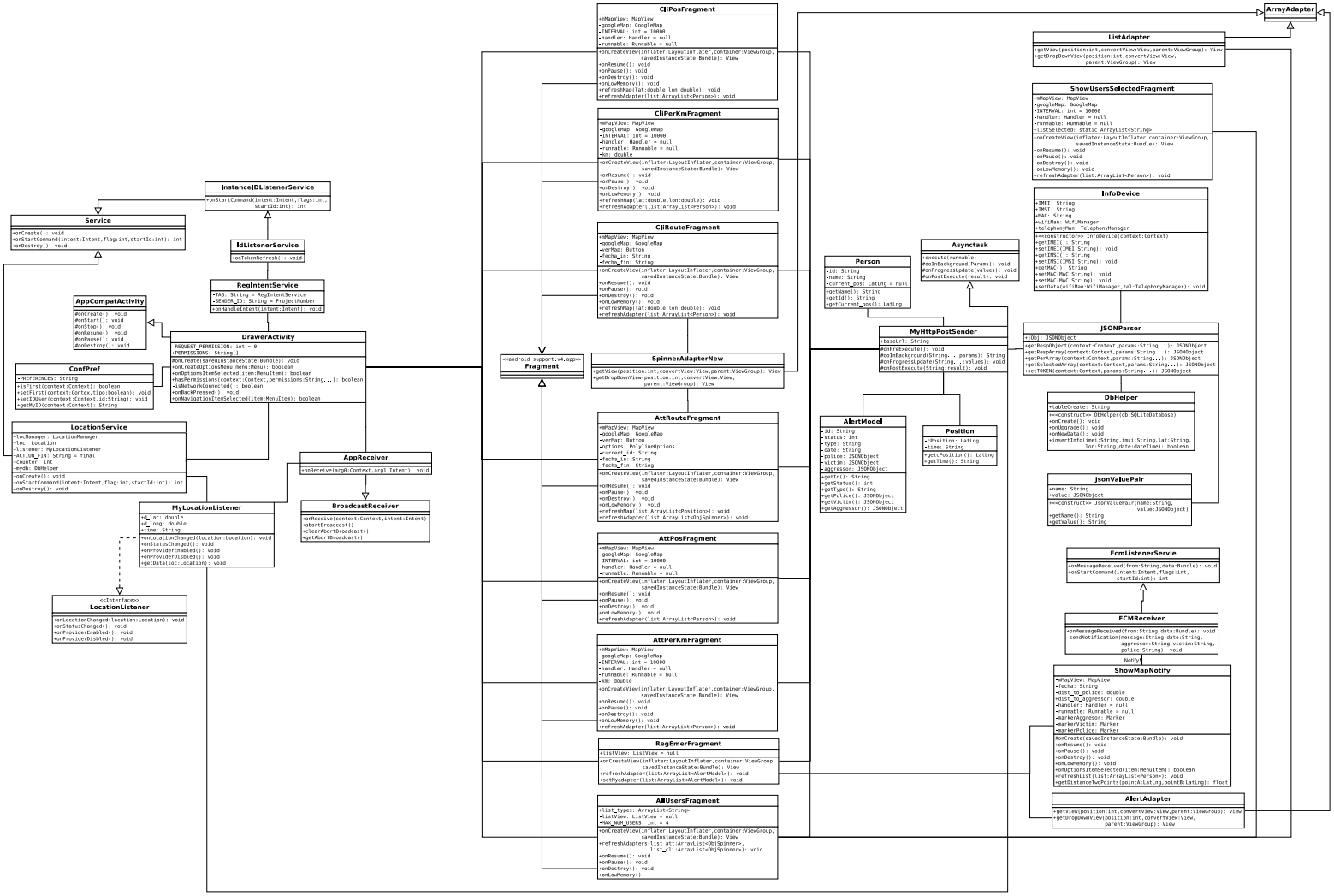

FIGURA 4.12: Diagrama de clases para *SAVIApol*.

Diagramas de secuencia

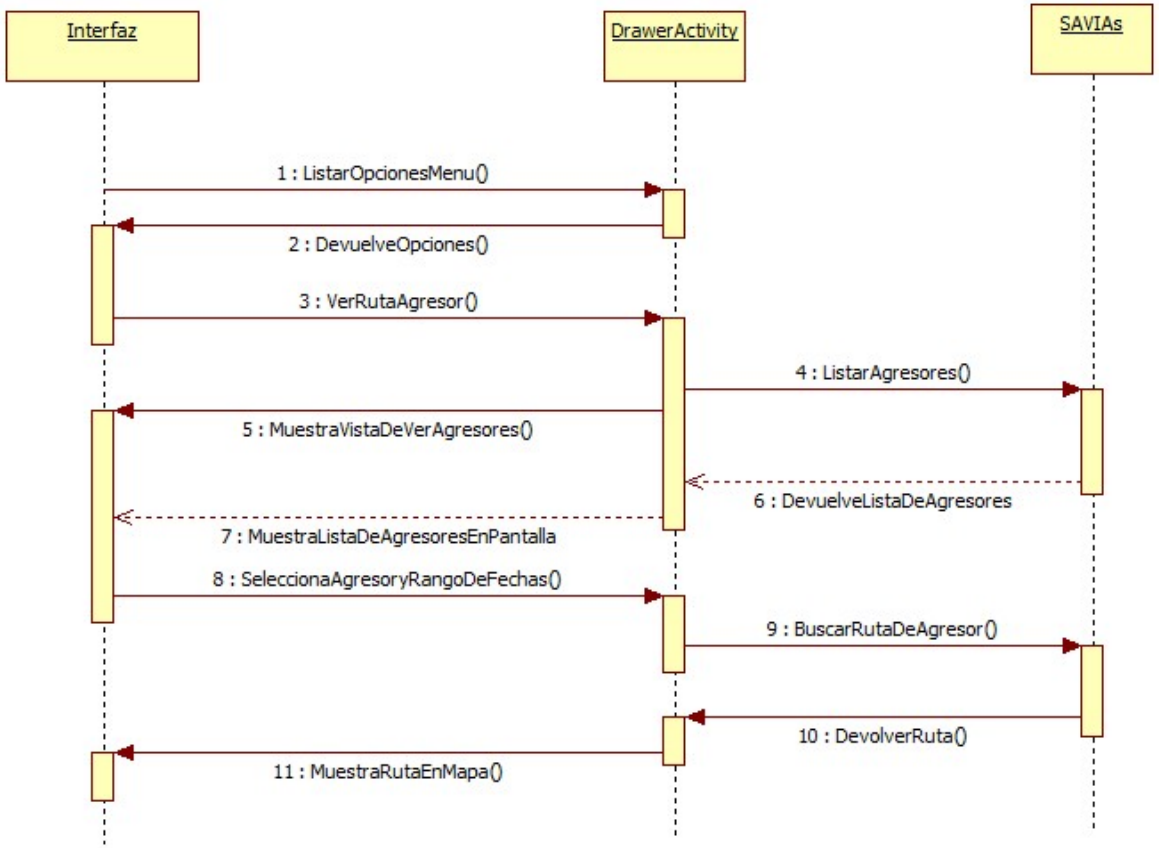

FIGURA 4.13: Diagrama de secuencia para el caso de uso "Ver ruta de agresores" de *SAVIApol*.

Capítulo 5

Estructura de *SAVIAs*

En este apartado se analizarán los componentes de SAVIAs y su interacción entre ellos, así mismo, se va a profundizar en los procesos de análisis de ubicaciones en tiempo real y envío de alertas a policías más cercanos.

5.1. Componentes

Para tener una idea más detallada que el diagrama de componentes descrito en la sección anterior 4.4 se muestra en la figura 5.2, un diagrama que muestra los componentes que interactúan entre sí para llevar a cabo las tareas de *SAVIAs*.

SAVIAs tiene como principal componente una aplicación basada en Node.js, esta aplicación expone una interfaz trabaja bajo el protocolo HTTPS, esta interfaz ejerce la función de gateway permitiendo comunicar *SAVIAs* con las aplicaciones móviles desarrolladas como *SAVIAtt*, *SAVIApol* y *SAVIApp*.

Este componente está compuesto básicamente de tres módulos que se encargan de tareas específicas como:

- Direccionamiento de eventos: Se encarga de interpretar las peticiones HTTPS y direccionar hacia los módulos de experiencia de usuario o análisis y registro de ubicación.
- API de experiencia de usuario: Contiene básicamente todas funcionalidades implementadas en los dispositivos móviles de consulta de usuario de la aplicación.

- Análisis y registro de ubicación: Se comunica con los componentes de registro de base de datos y generación de alerta.

Adicionalemte, se cuenta con dos componentes que se encargan del registro en la base de datos (MongoDB y MSQL) y generación de alertas en tiempo real.

Este último componente realiza la tarea comunicación con el servicio de Firebase Cloud Messaging mediante una interfaz bajo HTTPS, en caso de querer enviar una alerta hacia un dispositivo móvil en específico.

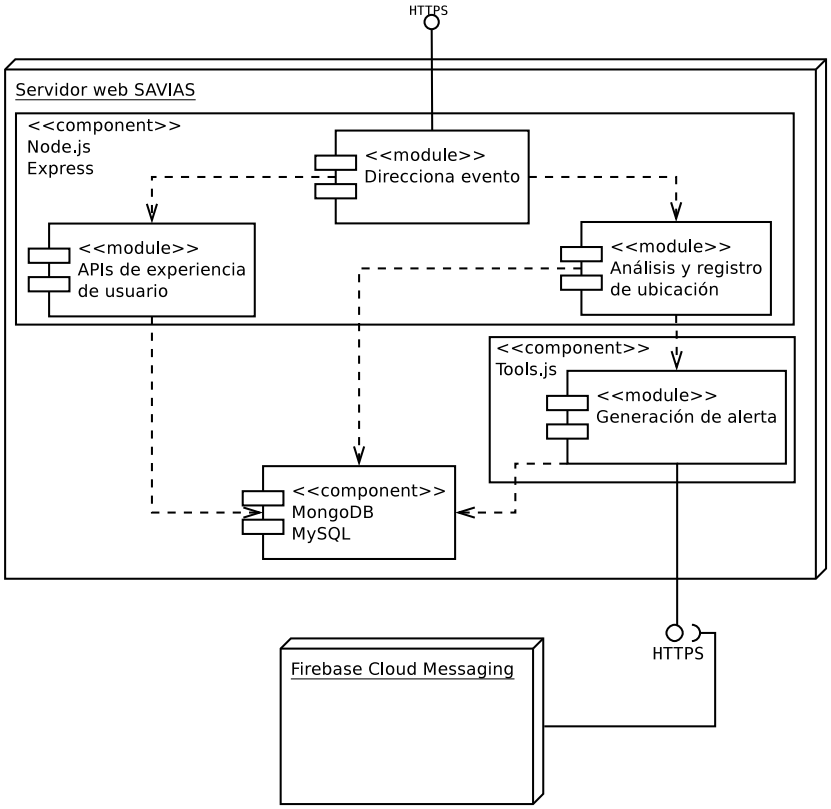

FIGURA 5.1: Diagrama de componentes que interactúan en el servidor. Fuente: Elaboración propia.

5.2. Características

Con el fin de profundizar en las características de *SAVIAs*, se analizarán procesos como el análisis de ubicaciones y el envío de alertas a policías más cercanos.

En el presente trabajo se utiliza como servidor web a **Node.js** conjuntamente con **Express**, debido a que se realizarán una gran cantidad de tareas a la vez. Estas tareas tendrán que ser realizadas en tiempo de respuesta muy pequeños.

5.2.1. Análisis en tiempo real de ubicaciones

Una de las tareas más importantes que realiza *SAVIAs* es la detección del quebrantamiento del perímetro de prevención y del perímetro de seguridad de la víctima por parte del agresor.

Para detallar este proceso, suponemos que existe una víctima, agresor y policía debidamente registrados en *SAVIAs*, caso contrario, los datos que envíen serán ignorados por el sistema. Dado que la víctima tiene asociado a un agresor que tiene una orden de restricción de 1km y una perímetro de prevención de 2km, estos tres usuarios se encontrarán continuamente enviando sus ubicaciones al servidor.

En este punto se hablará de la implementación de alertas que notificarán a las aplicaciones *SAVIApol* en una arquitectura Fog Computing; sin embargo, en el apéndice B se encuentra una primera implementación de detección de alertas para una arquitectura cloud bajo este mismo contexto.

Optimización del Core Level en una arquitectura Fog Computing

Como se comentó anteriormente (subsección 3.2.12), se ha usado Flink-CEP para la detección de alertas en tiempo real manejando los eventos generados, en este caso localizaciones, por las aplicaciones *SAVIAtt* y *SAVIApp* que nos permitirán optimizar el core level de una arquitectura Fog Computing.

En comparación con la implementación de alertas en una arquitectura cloud, Flink-CEP tiene las siguientes mejoras:

- Analizar en tiempo real las posiciones de los usuarios enviadas por las aplicaciones móviles.
- Eliminar el número de consultas en la base de datos para el cálculo de distancia entre *SAVIAtt* y *SAVIApp*.

- Agregar un nivel más de seguridad sobre *SAVIApp* permitiendo predecir eventos futuros y enviar alertas de prevención basado en los patrones de localización entre *SAVIAtt* y *SAVIApp*, permitiendo crear un entorno con un grado mayor de control por *SAVIAs*.

El proceso por el cual Flink-CEP detecta, analiza y genera una alerta es el siguiente:

- Reconocimiento de patrones: CEP considera el orden de los eventos y las relaciones que hay entre ellos para generar un patron.
- Análisis de eventos: Analiza las relaciones que hay entre los eventos evaluando los eventos recibidos.
- Acciones: Toma deciones de acuerdo al análisis previo.

Bajo el escenario de acción de *SAVIAs*, se ha logrado implementar tres tipos de alerta:

In situ: Este caso se da cuando *SAVIAtt* rompe el perímetro de seguridad de *SAVIApp*. En la siguiente imagen 5.2 se muestra cómo cada evento es recibido y es evaluado de acuerdo a la orden de restrición dada. Si la distancia es mayor que la establecida CEP descarta los eventos, caso contrario, envía una alerta a *SAVIApol*.

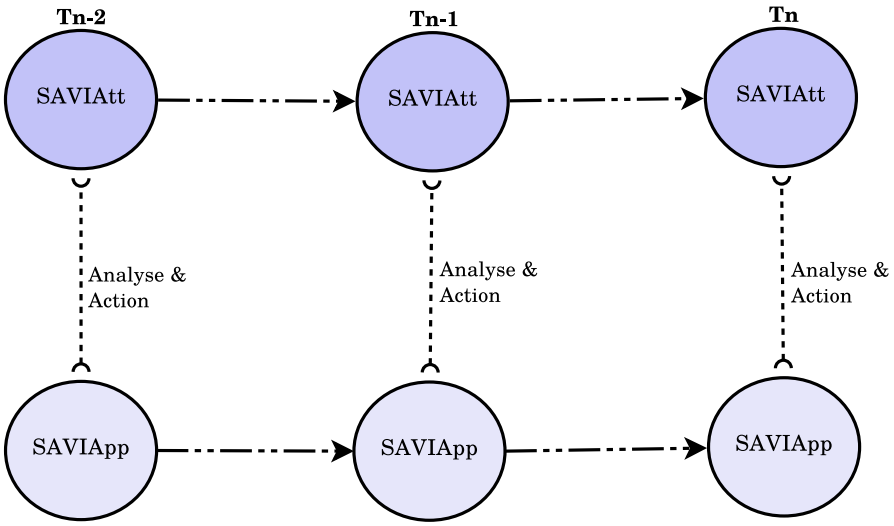

FIGURA 5.2: Tipo de alerta: In situ. Fuente: Elaboración propia.

Eventos predictivos: Este caso se da cuando se identifica si un evento, $Pos(T)$, en el futuro violará la orden de restricción. Para realizar esto, se establece un vector de distancia entre las dos últimas posiciones enviadas, en ese caso $Pos(T_n)$ y $Pos(T_{n-1})$ tal como se muestra en la figura 5.3.

Una vez que las posiciones de los roles de *SAVIAtt* y *SAVIApp* han sido predecidas, $Pos(T_{n+1})$, se evalúan sobre la orden de restricción y si ésta ha sido violada se genera una alerta al igual que el tipo In situ.

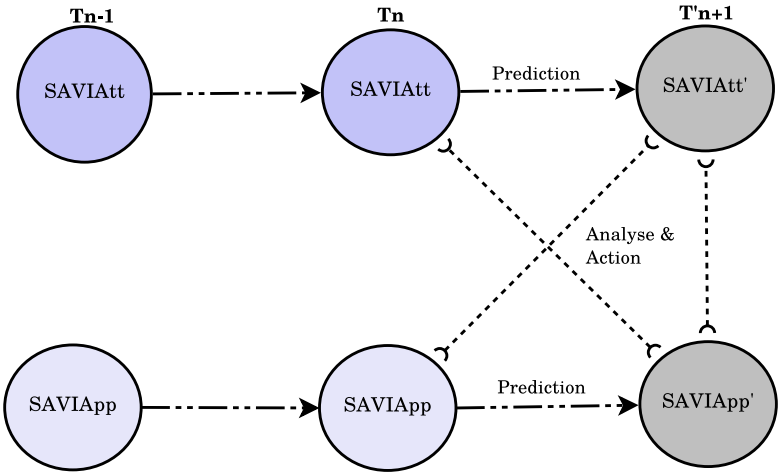

FIGURA 5.3: Tipo de alerta: Eventos predictivos. Fuente: Elaboración propia.

Eventos de contexto cercano: Este tipo de alerta busca prevenir proximidad entre *SAVIAtt* y *SAVIApp* rompiendo el perímetro de seguridad incluso en espacios de tiempos diferentes.

La figura 5.4 muestra cómo este tipo de alerta es generada. En este caso, *SAVIAs* analiza la última posición de *SAVIAtt*, $Pos(T_n)$, y verifica con las posiciones anteriores de *SAVIApp*, $Pos(T_{n-1}), Pos(T_{n-2})$. Si *SAVIAs* detecta que la orden de restricción ha sido violada se genera una alerta, y en caso contrario, estos eventos serán descartados.

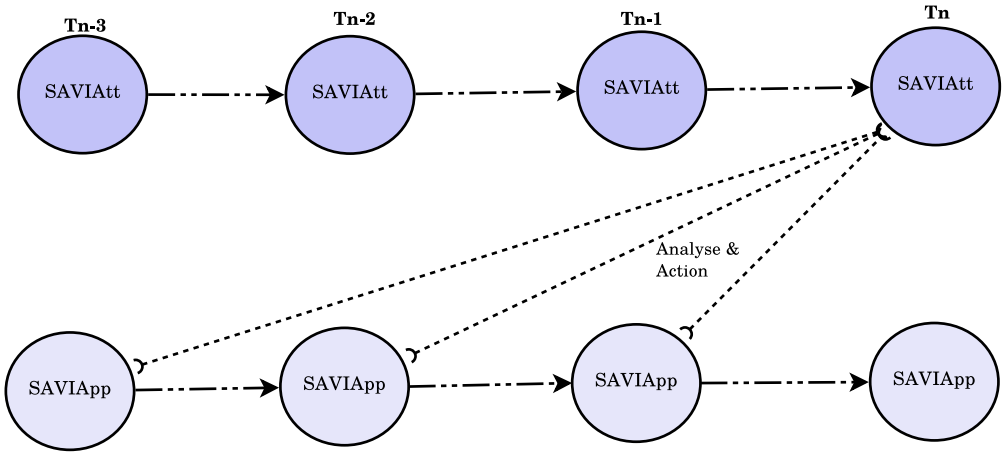

FIGURA 5.4: Tipo de alerta: Eventos de contexto cercano. Fuente: Elaboración propia.

5.2.2. Envío de alertas a través de Firebase Cloud Messaging

Existen dos tipos de alertas que serán gestionadas por *SAVIAs*: alertas automáticas y alertas manuales. Una alerta automática es generada cada vez que se ha detectado el quebrantamiento del perímetro de seguridad de la víctima por parte del agresor. Así mismo, una alerta manual es generada cada vez que la víctima pulsa el botón de alerta en su aplicación. Ambas alertas son derivadas al policía más cercano.

A continuación detallamos el procedimiento del envío de alerta desde *SAVIAs* a *SAVIApol*.

Como paso previo para esta implementación, se debe obtener una clave de autorización(API Key) para el uso de FCM API, el cual se obtiene al registrar el servidor en Google Developers Console.

Una vez obtenido el API key, se procede a obtener un código único del dispositivo al que se quiere enviar la alerta. Esto se realiza utilizando la clase **com.google.android.gms.iid.InstanceID** en Android. Este código llamado **token** es obtenido tal y como se muestra en el código 5.1. Luego **token** es enviado a *SAVIAs* para su registro.

```
InstanceID instanceID = InstanceID.getInstance(this);
String token = instanceID.getToken(SENDER_ID,
        GoogleCloudMessaging.INSTANCE_ID_SCOPE, null);
```

CÓDIGO 5.1: Uso de InstanceID.

Después de haber registrado el token del dispositivo en los datos del policía en la base de datos, éste queda disponible para realizar cualquier envío de alertas desde el servidor.

Este envío se realiza mediante la infraestructura FCM de Google, para esto se utiliza el módulo **request** para **Node.js**. Este módulo permite realizar una petición del tipo POST a través de HTTP, comunicando de esta manera con **Google connection server** que interactúa con un dispositivo móvil destino, en nuestro caso, el policía más cercano.

El código 5.2 muestra la construcción del request JSON que se utiliza para comunicar *SAVIAs* con **Google connection server**. Si no ha ocurrido ningún inconveniente, muestra un mensaje de envío exitoso.

```
request(
    //METODO Y CABECERAS DEL REQUEST
    { method: 'POST',
    uri: 'https://gcm-http.googleapis.com/gcm/send',
    headers: {
        'Content-Type': 'application/json',
        'Authorization':'key= API_KEY'
    },
    //CONSTRUCCION DEL MENSAJE
    body: JSON.stringify({
        "registration_ids" : [token],
        "data" : {
            "msj" : mensaje,
        },
        "time_to_live": 108
```

```
            })
        }
        //MANEJO DE ERRORES
        , function (error, response, body) {
            //MOSTRAR ERROR
         }
    );
```

CÓDIGO 5.2: Estructura del request a GCM.

Capítulo 6

Casos de estudio

En este capítulo se mostrarán todas las funcionalidades implementadas en las aplicaciones para los roles de víctima, agresor y policía que han sido desarrolladas a lo largo de este trabajo.

6.1. Aplicación agresor: *SAVIAtt*

El desarrollo de esta aplicación tiene como única finalidad el envío de la posición del agresor cada minuto hacia *SAVIAs*. Cuenta con sólo una interfaz que permite realizar acciones de verificación de datos y permisos, Una vez realizada la verificación, *SAVIAtt* activará el servicio de envío de ubicación en tiempo real.

6.1.1. Interfaz

La figura 6.1 muestra la interfaz de esta aplicación que tiene como principales funcionalidades comprobar si existe conexión a internet (ver figura 6.2), verificar los permisos necesarios(en caso que el dispositivo tenga Android versión 6) antes de empezar con el envío de datos.

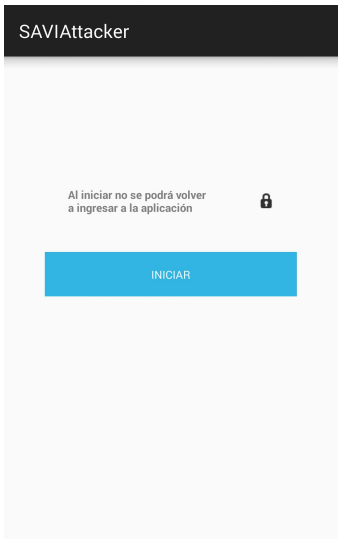

FIGURA 6.1: Interfaz de *SAVIAtt*.

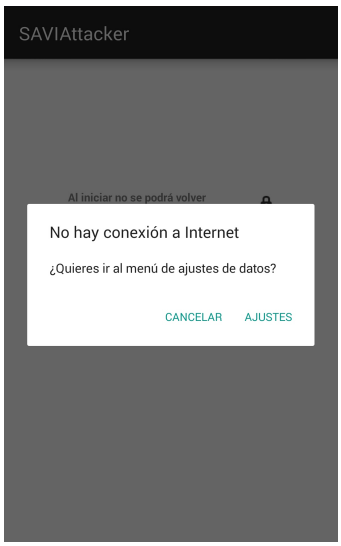

FIGURA 6.2: Verificación de internet de *SAVIAtt*.

6.2. Aplicación víctima: *SAVIApp*

La aplicación *SAVIApp* además de adoptar el servicio de envió de ubicación en tiempo real de *SAVIAtt* posee las siguientes funcionalidades:

- Ver mi ruta.
- Ver agresor.
- Ver ruta de agresor.

- Buscar policías por kilómetro.
- Enviar alerta.

6.2.1. Interfaz

A través de la interfaz principal de *SAVIApp*, mostrada en la imagen 6.3, el usuario podrá interactuar y disponer de la funcionalidades que posee la aplicación.

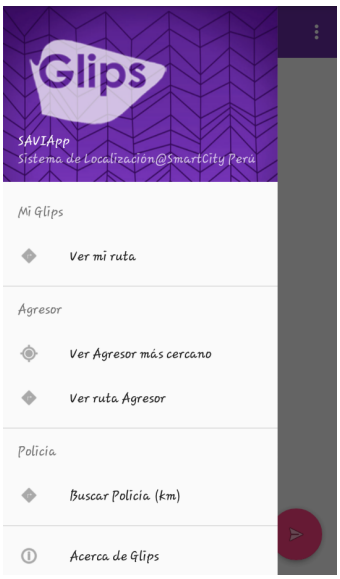

FIGURA 6.3: Interfaz de *SAVIApp*.

6.2.2. Ver mi ruta

Esta funcionalidad permite que el usuario vea su ruta histórica en un lapso de tiempo determinado. En la figura 6.4 muestra un grupo de puntos en el mapa unidos por una línea, donde se ha fijado como punto inicial (verde) a las 2015-12-21 02:39 y como punto final (amarillo) 2015-12-25 02:39. También se puede ver más detalle de un punto en particular pulsando sobre el marcador del mapa.

FIGURA 6.4: Ver mi ruta.

6.2.3. Ver agresor

El usuario podrá ver ubicación de sus agresores en tiempo real. La imagen 6.5 representa la posición en tiempo real del agresor Juan García. Para poder ver la posición en tiempo real de otro agresor a la víctima, basta con pulsar sobre el nombre del agresor y seleccionar una opción de la lista agresores.

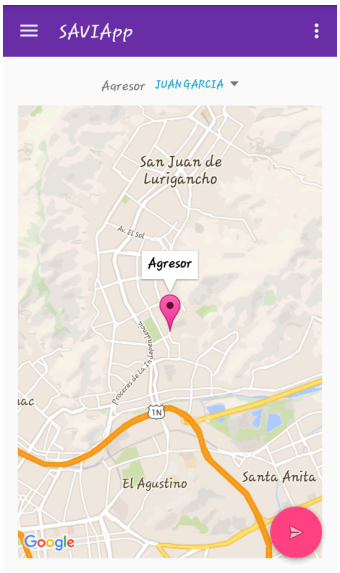

FIGURA 6.5: Ver agresor.

6.2.4. Ver ruta agresor

La imagen 6.6 describre la funcionalidad en la cual el usuario podrá ver la ruta histórica de sus agresores en un lapso de tiempo determinado. Aunque aquí se muestra información similar que en la funcionalidad "Ver mi ruta", también se podrá seleccionar a otro agresor.

FIGURA 6.6: Ver ruta agresor.

6.2.5. Buscar policías por kilómetro

La aplicación *SAVIApp* también permite que el usuario pueda buscar a los policías que se encuentran a su alrededor, también puede modificar el radio de búsqueda. En el ejemplo que muestra la figura 6.7, se puede ver que existe un policía en un radio de 10 kilómetros fijado por el usuario. Cabe resaltar que el valor por defecto del radio de búsqueda es de 1 kilómetro.

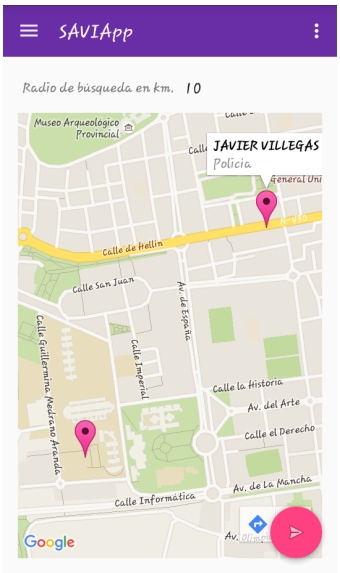

FIGURA 6.7: Buscar policía más cercano.

6.2.6. Enviar alerta

Una de las principales funcionalidades de *SAVIApp* es la posibilidad de generar alertas de manera manual al policia más cercano, utilizando un botón de pánico. Este botón de forma circular se encuentra disponible en la parte inferior derecha, tal como se muestra en la figura 6.8.

FIGURA 6.8: Enviar alerta al policía más cercano.

6.3. Aplicación policía: *SAVIApol*

La aplicación *SAVIApol* además de adoptar el servicio de envió de ubicación en tiempo real de las aplicaciones *SAVIAtt* y *SAVIApp* posee las siguientes funcionalidades:

- Ver agresores/víctimas.
- Ver ruta de agresor/víctima.
- Buscar agresores/víctimas por kilómetro.
- Ver más personas.
- Ver alertas.
- Recepción de alertas

6.3.1. Interfaz

A través de esta interfaz, figura 6.9, el policía podrá disponer de la funcionalidades de la aplicación.

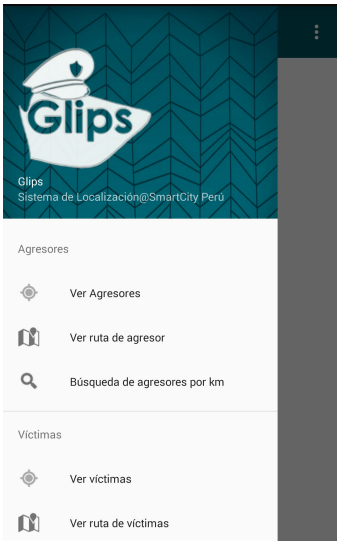

FIGURA 6.9: Interfaz de *SAVIApol*.

6.3.2. Ver agresores/víctimas

Esta funcionalidad el policía podrá ver ubicación de todos los agresores que han sido registrados en *SAVIAs* en tiempo real. De igual manera podrá hacerlo para las víctimas. En la figura 6.10 se muestra un único agresor en el mapa. Esta interfaz posee una casilla de marcado llamado "Auto ajuste" que permite activar o desactivar el ajuste el zoom de manera automática sobre el mapa.

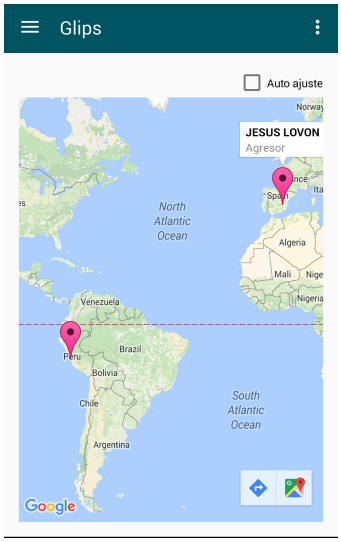

FIGURA 6.10: Ver todos los agresores.

6.3.3. Ver ruta de agresor/víctima

El policía podrá ver la ruta histórica de todos los agresores en un lapso de tiempo determinado, también se podrá seleccionar a otro agresor. De igual manera podrá hacerlo para las víctimas.

En la figura 6.11 muestra un grupo de puntos en el mapa unidos por una línea, donde se ha fijado como punto inicial (verde) a las 2015-12-30 03 y como punto final (amarillo) 2015-12-28 02:39. También se puede ver más detalle de un punto en particular pulsando sobre el marcador del mapa.

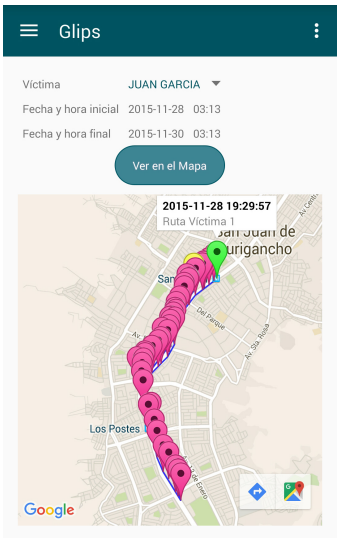

FIGURA 6.11: Ver ruta de víctimas.

6.3.4. Buscar agresores/víctimas por kilómetro

La aplicación *SAVIApol* también permite que el usuario pueda buscar a los policías que se encuentran a su alrededor, también puede modificar el radio de búsqueda. En el ejemplo que muestra la figura 6.12, se puede ver que existe un agresor en un radio de 10 kilómetros fijado por el usuario. Cabe resaltar que el valor por defecto del radio de búsqueda es de 1 kilómetro.

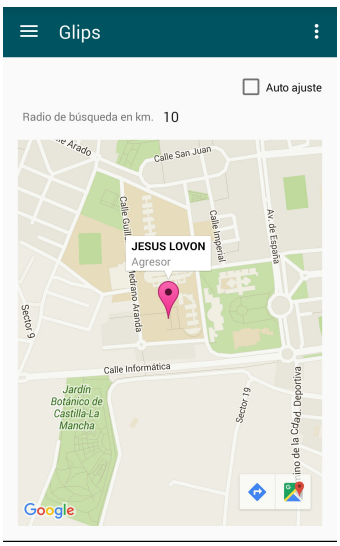

FIGURA 6.12: Buscar agresor por km.

6.3.5. Ver más personas

Esta funcionalidad permite que el policía podrá buscar hasta cuatro personas asociando el nombre de la persona a monitorizar, es decir, se podrá ver la posición en tiempo real de los usuarios seleccionados.

Seleccionar usuarios

El usuario selecciona a las personas a través del tipo y nombre tal como se muestra en la figura 6.13.

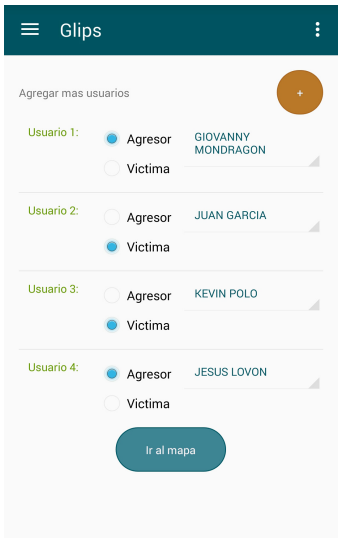

FIGURA 6.13: Seleccionar usuarios.

Ver usuarios seleccionados en mapa

La figura 6.14 muestra los usuarios seleccionados con su ubicación en tiempo real.

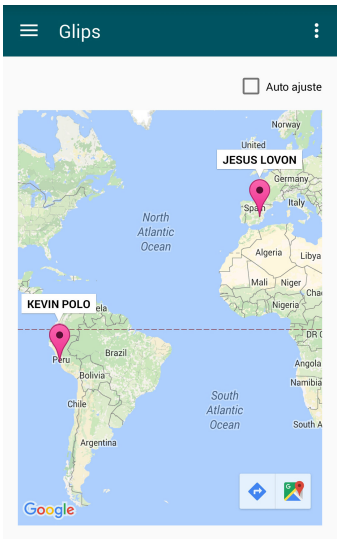

FIGURA 6.14: Ver usuarios seleccionados en mapa.

6.3.6. Ver alertas

Con esta funcionalidad mostrada en la figura 6.15, *SAVIApol* permite al policía ver el registro históricos de alertas generadas en *SAVIAs*.

FIGURA 6.15: Ver registro de alertas.

Mis alertas

Así mismo, seleccionando el marcardor "Mis alertas", el policía podrá ver las alertas que le han sido asignadas. La figura 6.16 muestra información básica

de una alerta generada (fecha, estado, nombre de policía y nombre de víctima) que ha sido asignada a un policía.

FIGURA 6.16: Ver mis alertas.

Detalle de alerta

Al presionar en una de las alertas, se muestra en la figura 6.17 un cuadro de diálogo que contiene información detallada de la alerta.

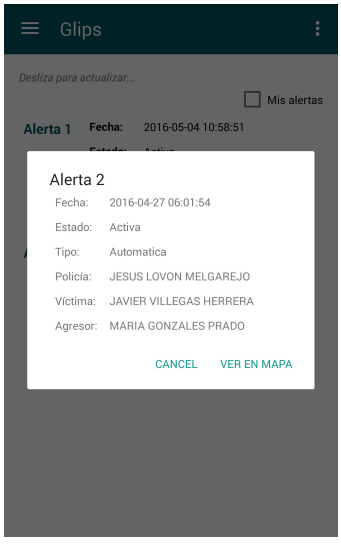

FIGURA 6.17: Ver más detalles de alerta.

Ver en mapa: Alerta automática

Esta opción (figura 6.18) muestra el detalle de una alerta automática, la víctima, agresor y policía implicados y la distancia en tiempo real entre ellos. Podrá ser monitorizada en tiempo real si la alerta se encuentra aún en estado .Activaz le ha sido asignada.

FIGURA 6.18: Ver alerta automática en mapa.

Ver en mapa: Alerta manual

De igual manera, en la figura 6.19 se muestra una alerta manual el instante donde fue enviada la alerta. En este caso, no se encuentra información del agresor ya que esta alerta ha sido generada de manera manual por *SAVIApp* utilizando la función .Enviar alerta".

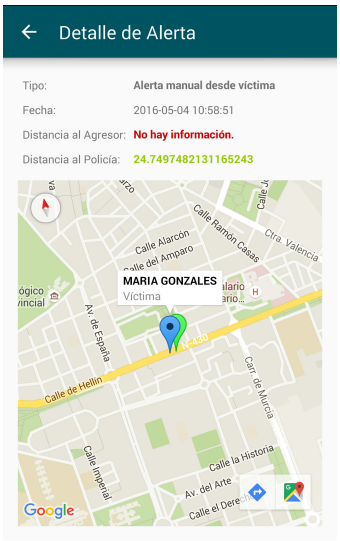

FIGURA 6.19: Ver alerta manual en mapa.

6.3.7. Recepción de alertas

SAVIApol tiene la capacidad de recibir alertas en todo momento del día, a continuación la figura 6.20 muestra las notificaciones que se generan en el dispositivo móvil a partir de la llegada de las alertas.

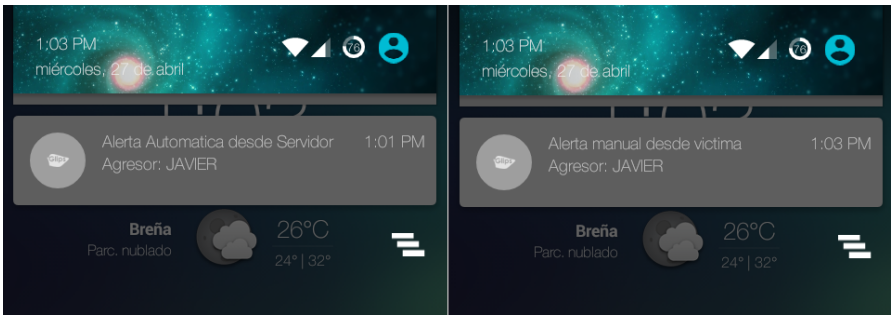

FIGURA 6.20: Recepción de alerta automática y manual de *SAVIApol*.

Capítulo 7

Conclusiones y trabajo a futuro

Una vez terminada las implementaciones de las aplicaciones de agresor, víctima y policía, se puede exponer las conclusiones que se han obtenido de este trabajo. Adicionalmente, mencionar trabajos a futuro para mejorar este proyecto.

7.1. Conclusiones

Una conclusión que ha dejado esta implementación de aplicaciones para el sistema es que se ha desarrollado de manera exitosa todos los objetivos trazados al iniciar este trabajo. Como en todo primer trabajo hay puntos por mejorar, se resaltan los siguientes puntos.

- **Cobertura y retardo del sensor *GPS*:** un punto a resaltar es la cobertura del sensor *GPS*. Aunque la precisión del *GPS* aumenta a medida que el usuario esté en exteriores, dificulta ciertas pruebas dentro de edificios.

  También se tomó muy en cuenta el retardo del sensor *GPS* ya que uno de los requisitos fue recolectar información del satélite cada minuto. Los resultados arrojaron que el sensor GPS demora unos 7 a 13 segundos en conseguir una respuesta del satélite.

  Por otro lado, al recolectar información del satélite cada 10 segundos, este retardo desapareció debido a que en este pequeño lapso de tiempo el servicio de *GPS* nunca se logró poner en modo suspensión.

- **Localización mediante *GPS* y *Triangulación*:** otro de los temas que se tuvo en discusión fue la precisión y el retardo del sensor *GPS*. Existen otras tecnologías que permiten conseguir la posición en la que se encuentra un usuario, una de ellas es mediante *Triangulación* que utiliza técnicas de triangulación de antenas de telefonía móvil.

  Aunque, el tiempo de respuesta de la *Triangulación* es mucho más corto en comparación con el *GPS*, la precisión de esta técnica tiene un error de aproximadamente 60 metros a la redonda. La diferencia es muy marcada ya que el *GPS*, a pesar de su retardo, tiene un error de 2 a 4 metros.

- **Búsqueda rápida de coordenadas al iniciar las aplicaciones:** como se ha visto en los puntos anteriores que la obtención de la ubicación depende en demasía del lugar donde se encuentra el dispositivo y del tiempo que se logre establecer una comunicación con el satélite. Por esta razón, las aplicaciones tendrán como prioridad obtener la ubicación mediante *Triangulación*. Esto permitirá tener una posición aproximada de manera rápida para ser enviada al servidor y permitir su análisis.

- **Velocidad de envío:** al implementar el envío como un servicio se ha ganado muchísimo en velocidad, ya que al utilizar un servicio el procesador mantiene este módulo en background y deja al usuario interactuar fluidamente con la interfaz de la aplicación.

  Evitando demoras innecesarias debido a que realiza el envío cada cierto tiempo mediante el uso de una tarea asíncrona avalado por la simplicidad de la estructura JSON.

- **Utilización de Express y NodeJS como servidor web:** otra ventaja es el uso de tecnologías que permitan realizar tareas asíncronas en lapso de tiempos críticos con tiempo de respuesta muy pequeños. Bajo el uso de estas herramientas, *SAVIAs* realiza las tareas asignadas de forma rápida y eficiente.

- **Análisis de ubicaciones tiempo real:** como ya se ha visto, *SAVIAs* realiza un análisis ubicación por ubicación de tal manera que detecta violaciones de perímetros de seguridad y permite alertar al policía más cercano.

  También verifica que los usuarios estén enviando ubicaciones en todo momento, y de no ser así, enviar una alerta al administrador del sistema para que tome las acciones adecuadas.

- **Seguridad en la comunicación:** otro factor resaltante en este trabajo es la implementación de un servidor web bajo el protocolo HTTPS que utiliza un certificado de seguridad para garantizar la comunicación segura entre servidor y las aplicaciones.

- **Experiencia de usuario:** al ser parcialmente un desarrollo para dispositivos móviles, se tiene como factor importante la experiencia del usuario con las aplicaciones. Teniendo como punto resaltante la interfaz sencilla y fácil de usar, mostrando notificaciones de alertas o errores.

- **Bajo consumo de batería de los dispositivos:** Aunque el rendimiento de estos dispositivos móviles se ha elevado drásticamente en los últimos años, se sigue teniendo a la batería como punto débil. Enfatizamos esta característica ya que este factor es importantísimo para este trabajo.

  Debido a que el consumo por parte del sensor *GPS* es altísimo, se opta por utilizar un sistema de recolección de ubicaciones híbrido. Es decir, estas aplicaciones utilizará el sensor *GPS* sólo cuando *SAVIAs* lo considere necesario, esto implica el uso de *Triangulación* cuando exista una distancia considerable entre víctima y agresor, dando como resultado un ahorro significativo en el consumo de batería.

  La reducción del consumo se ve claramente ya que al realizar pruebas con el sensor *GPS* se obtuvo casi 10mAh de consumo energético en un lapso de 10 minutos, en contraste a las pruebas con *Triangulación* donde se obtuvo medidas menores de 1mAh en el mismo lapso de tiempo.

7.2. Trabajo a futuro

En este punto se menciona qué mejoras podemos realizar a las aplicaciones agresor, víctima, policía y al sistema.

7.2.1. Implementar un sistema de localización en interiores

A lo largo de este trabajo se ha visto la necesidad de implementar un sistema de localización rápido, preciso y sin mucho consumo de batería. El procedimiento utilizado para la ubicación es muy útil en temas de rapidez y consumo de batería pero aún tiene problemas en precisión y más aún cuando se trata de interiores.

Un paso fundamental sería implementar un sistema que permita obtener la ubicación en interiores de manera precisa y de bajo consumo, como por ejemplo bajo la tecnología **Bluetooh 5.0** [43, 44].

Al implementar el sistema de posicionamiento en interiores garantizaría a la monitorización de reclusos en cárceles, a través de grilletes electrónicos que serán visualizados en las aplicaciones con ayuda de un mapa 3D del interior del centro penitenciario.

7.2.2. Realizar un estudio de performance entre alertas cloud y alertas fog

Debido a que en este trabajo se resalta la optimización del core level de una arquitectura fog, un punto atractivo de abordar sería realizar un estudio de rendimiento entre la generación de alertas cloud (arquitectura centralizada) y la generación de alertas con Flink-CEP (arquitectura fog).

7.2.3. Implementar alertas del tipo de no recepción de ubicaciones

Adicionalmente, se podría implementar un nuevo tipo de alertas que se generen a partir de la no recepción de ubicaciones después de un cierto tiempo

o una posible manipulación de los dispositivos. Esto permitirá asegurar al sistema que el usuario estará siempre disponible para su monitorización.

7.2.4. Mejora de la expericia de usuario de las aplicaciones *SAVIApp* y *SAVIApol*

También mejorar la experiencia de usuario en las aplicaciones de víctima y policía, adicionando información adicional en la representación de los usuarios mediante una fotografía.

7.2.5. Implementar la búsqueda de la ruta más corta hacia el agresor en la aplicación *SAVIApol*

Otra funcionalidad que podría ser de gran ayuda para el policía, sería el implementar la búsqueda de la ruta más corta para llegar hacia el agresor o la víctima. Dando un valor agregado a la monitorización de la alerta en tiempo real ya que esta funcionalidad otorgará una ruta más rápida y eficaz para llegar hacia los implicados y evitar una posible agresión.

7.2.6. Adaptación de SAVIAS a nuevas casuísticas

Aunque este trabajo se trata de una plataforma de seguridad, podría extenderse a realizar monitorizaciones de ancianos o personas con necesidades especiales siempre y cuando sea permitido en el marco legal Peruano, ya que hoy en día existe leyes que respaldan monitorización a través de la geolocalización sólo para actividades delictivas como la extorsión y el secuestro [42].

Otro tema que sería interesante abordar es distribuir de manera eficiente a los policías en ciudades donde se tenga un mayor índice de violencia, esto permitiría evitar de manera rápida posibles agresiones.

Bibliografía

[1] Organisation for Economic Co-operation and Development (2016), violence against women (indicator). https://data.oecd.org/inequality/violence-against-women.htm. Último acceso: 2018-07-31.

[2] Observatorio de Igualdad de Género de América Latina y el Caribe. https://oig.cepal.org/es/indicadores/feminicidio. Último acceso: 2018-07-31.

[3] Ministerio de la Mujer y Poblaciones Vulnerables - Estadísticas sobre Feminicidio. https://www.mimp.gob.pe/contigo/contenidos/pncontigo-articulos.php?codigo=39. Último acceso: 2018-07-31.

[4] Table conference on 'safe city is a smart city', hanns seidel stiftung. http://www.hss.de/fileadmin/india/downloads/Safe_City_is_a_Smart_City.pdf. Último acceso: 2018-06-02.

[5] Jiong Jin, Jayavardhana Gubbi, Slaven Marusic, and Marimuthu Palaniswami. An information framework for creating a smart city through internet of things. *Internet of Things Journal, IEEE*, 1(2):112–121, 2014.

[6] SMART CITIES. Trace analysis and mining for smart cities: issues, methods, and applications. *IEEE Communications Magazine*, 121, 2013.

[7] Madelayne Morales Rodriguez, Juan Camilo Acosta Escobar, Juan David Díaz Ramírez, and Darío Fernando Cortés Tobar. Sistema de localización agresor-victima en ambientes indoor y outdoor. *Sistemas & Telemática*, 10(23):51–63, 2012.

[8] El gobierno español autoriza la pulsera electrónica para maltratadores por violencia de género. http://www.agenciasinc.es/Noticias/El-gobierno-espanol-autoriza-la-pulsera-electronica-para-maltratadores-por-violencia-de-genero. Último acceso: 2018-05-20.

[9] Alertcops, ministerio del interior de españa. https://alertcops.ses.mir.es/mialertcops/info/info.xhtml. Último acceso: 2018-06-16.

[10] App-Elles - Alerter, En Parler, Agir. https://www.app-elles.fr/index_en.html. Último acceso: 2018-08-06.

[11] Hands Away, itunes - play store. https://www.handsaway.fr. Último acceso: 2018-08-06.

[12] I'm here for you, itunes. https://itunes.apple.com/us/app/here-for-you/id739824451?mt=8. Último acceso: 2018-05-20.

[13] R.i.s.e social media app, itunes. http://www.octevaw-cocvff.ca/projects/rise-social-media-app. Último acceso: 2018-05-20.

[14] imatter, itunes. https://itunes.apple.com/us/app/imatter/id953936692?mt=8. Último acceso: 2018-05-20.

[15] Gva, ni más ni menos, play store. https://play.google.com/store/apps/details?id=gva.artefinal.ni_mas_ni_menos. Último acceso: 2018-05-20.

[16] Domestic violence prevention, play store. https://play.google.com/store/apps/details?id=com.tracen.domviol. Último acceso: 2018-05-20.

[17] M Salman Bashir and M Rizwan Jameel Qureshi. Hybrid software development approach for small to medium scale projects: Rup, xp & scrum. *Cell*, 966:536474921, 2012.

[18] Vahid Rahimian and Raman Ramsin. Designing an agile methodology for mobile software development: A hybrid method engineering approach. In *Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on*, pages 337–342. IEEE, 2008.

[19] Android, the world's most popular mobile platform. `https://developer.android.com/about/index.html`. Último acceso: 2018-05-27.

[20] Android studio features. `https://developer.android.com/studio/features.html`. Último acceso: 2018-05-27.

[21] The difference between gps and gprs, opposing views. `http://science.opposingviews.com/difference-betwen-gps-gprs-17996.html`. Último acceso: 2018-05-27.

[22] About node.js. `https://nodejs.org/en/about/`. Último acceso: 2018-05-28.

[23] Infraestructura de aplicaciones web node.js, express. `http://expressjs.com/es/`. Último acceso: 2018-05-28.

[24] Assert node.js v6.2.0 manual & documentation. `https://nodejs.org/api/assert.html`. Último acceso: 2018-05-28.

[25] Https node.js v6.2.0 manual & documentation. `https://nodejs.org/api/https.html`. Último acceso: 2018-05-28.

[26] Body-parser: Node.js body parsing middleware. `https://github.com/expressjs/body-parser`. Último acceso: 2018-05-28.

[27] Simplified http request client. `https://github.com/request/request`. Último acceso: 2018-05-28.

[28] Node.js mongodb driver. `https://docs.mongodb.com/ecosystem/drivers/node-js/`. Último acceso: 2018-05-28.

[29] A pure node.js javascript client implementing the mysql protocol. https://github.com/felixge/node-mysql. Último acceso: 2018-05-28.

[30] Growing library to provide some basic geo functions. https://github.com/manuelbieh/Geolib. Último acceso: 2018-05-28.

[31] File system node.js v6.2.0 manual & documentation. https://nodejs.org/api/fs.html. Último acceso: 2018-05-28.

[32] Jsonobject, android developers. https://developer.android.com/reference/org/json/JSONObject.html?hl=es. Último acceso: 2018-05-27.

[33] Httpcomponents httpclient overview, apache. https://hc.apache.org/httpcomponents-client-4.5.x/index.html. Último acceso: 2018-05-27.

[34] Getting started, google maps android api. https://developers.google.com/maps/documentation/android-api/start. Último acceso: 2018-05-27.

[35] Cloud messaging, google developers. https://developers.google.com/cloud-messaging/. Último acceso: 2018-09-20.

[36] About sqlite. https://www.sqlite.org/about.html. Último acceso: 2018-09-20.

[37] Why mysql. http://www.mysql.com/why-mysql/. Último acceso: 2018-09-20.

[38] Reinventando la gestión de datos, mongodb. https://www.mongodb.com/es. Último acceso: 2018-09-20.

[39] Apache Flink - Stateful Computations over Data Streams. https://flink.apache.org/. Último acceso: 2018-05-28.

[40] Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, pages 1–84, Dec 1990.

[41] Klaus Pohl. *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.

[42] Ley stalker: Pnp y empresas firman protocolo de geolocalización, el comercio. `http://elcomercio.pe/lima/seguridad/geolocalizacion-celulares-pnp-y-operadoras-firman-protocolo-noticia-1848628`. Último acceso: 2018-09-20.

[43] Manuel Castillo-Cara, Jesús Lovón-Melgarejo, Gusseppe Bravo-Rocca, Luis Orozco-Barbosa, and Ismael García-Varea, "An Analysis of Multiple Criteria and Setups for Bluetooth Smartphone-Based Indoor Localization Mechanism," *Journal of Sensors*, vol. 2017, Article ID 1928578, 22 pages, 2017.

[44] Castillo-Cara, Manuel and Lovón-Melgarejo, Jesús and Bravo-Rocca, Gusseppe and Orozco-Barbosa, Luis and García-Varea, Ismael, .ªn Empirical Study of the Transmission Power Setting for Bluetooth-Based Indoor Localization Mechanisms", *Sensors*, vol. 17, Article ID 1318, 2017.

[45] Bonomi, Flavio and Milito, Rodolfo and Zhu, Jiang and Addepalli, Sateesh, "Fog Computing and Its Role in the Internet of Things", *ACM*, Article ID 2342513, 2012

[46] Alsaffar, Aymen Abdullah and Pham, Hung Phuoc and Hong, Choong-Seon and Huh, Eui-Nam and Aazam, Mohammad, .ªn architecture of IoT service delegation and resource allocation based on collaboration between fog and cloud computing", *Mobile Information Systems*, vol. 2016, 2016.

[47] A. Al-Fuqaha and M. Guizani and M. Mohammadi and M. Aledhari and M. Ayyash, Ïnternet of Things: A Survey on Enabling Technologies, Protocols, and Applications", *IEEE Communications Surveys Tutorials*, vol. 17, Article ID 1553-877X, 2347-2376 pages, 2015.

[48] M. B. A. P. Madumal and D. A. S. Atukorale and T. M. H. A. Usoof, .ᴬdaptive event tree-based hybrid CEP computational model for Fog computing architecture", *2016 Sixteenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, 2016.

[49] Alfonso Garcia-de-Prado and Guadalupe Ortiz and Juan Boubeta-Puig, ÇOLLECT: {COLLaborativE} ConText-aware service oriented architecture for intelligent decision-making in the Internet of Things", *Expert Systems with Applications*, vol. 85, 231-248 pages, 2017.

Apéndice A

Especificaciones de casos de uso

A.1. Agresor, víctima y policía

CU1: Enviar ubicación

Precondiciones

El usuario se encuentra debidamente registrado en la base de datos de *SAVIAs* y debe tener la aplicación instalada en el dispositivo.

Flujo principal

1. INGRESO
   El usuario ingresará a la aplicación. El sistema mostrará la interfaz al usuario.
2. CONEXIÓN A INTERNET
   El sistema comprobará que exista conexión a internet.
3. SERVICIO DE ENVÍO
   El sistema inicia automáticamente el servicio de envío de datos a *SAVIAs*.

Flujos alternativos

1. PERMISOS
   Del flujo principal INGRESO. En caso el dispositivo tenga versión de Android mayor a 6.0, el sistema mostrará un cuadro de diálogo señalando los permisos que el usuario deberá

permitir. Este mensaje sólo será mostrado la primera vez que el usuario entra a la aplicación.

2. MOSTAR CONFIGURACIÓN DE WIFI Y DATOS
   Del flujo principal CONEXIÓN A INTERNET. Si el sistema no detecta conexión a internet, mostrará un cuadro de diálogo que llevará al usuario a las configuraciones del sistema para activar las conexiones WIFI o de datos.
3. MENSAJE DE ERROR
   Del flujo principal SERVICIO DE ENVÍO. Si se muestra un mensaje de error indica que los datos del dispositivos no concuerdan con la base de datos de *SAVIAs*.

Inclusiones

1. ENVIAR INFORMACIÓN DEL DISPOSITIVO
   Al realizar el envío de ubicación cada minuto, también se envíará en el mismo paquete de datos la información del dispositivo como la dirección MAC, números IMEI e IMSI.

Extensiones

1. GUARDAR DATOS EN EL DISPOSITIVO
   Simultáneamente al envío de ubicación y de datos del dispositivo, esta información se guardará en una base de datos en el móvil. Esta base de datos sólo será accedida por un representante legal o administrador del sistema.

A.2. Víctima

CU1: Ver mi ruta

Precondiciones

La víctima se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN
   El usuario va a la interfaz de navegación y pulsa en "Ver mi ruta".
2. FECHAS DE BÚSQUEDA
   El usuario ingresa las fechas de búsqueda.
3. VER MAPA
   El usuario pulsa en el botón "Ver en el Mapa".
4. PROCESO DE CONSULTA
   El sistema procesa y devuelve el recorrido en el mapa.
5. DETALLE DE UBICACIÓN
   El usuario pulsa en cualquiera de los puntos que representan ubicaciones y el sistema mostrará el número y la hora de dicha ubicación.

Flujos alternativos

1. NO HAY RESULTADOS
   Del flujo principal PROCESO DE CONSULTA. Si no se encuentra resultados entre las fechas ingresadas, el sistema emitirá un mensaje en la pantalla informando que no se encontraron resultados.

CU2: Ver posición de agresor

Precondiciones

La víctima se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN
   El usuario va a la interfaz de navegación y pulsa en "Ver agresor".
2. CARGA DE AGRESORES
   El sistema carga en una lista desplegable todos los agresores del usuario.

3. SELECCIÓN DE AGRESOR
   El usuario selecciona el agresor que desea monitorizar.
4. UBICACIÓN DEL AGRESOR
   El sistema devuelve la posición en tiempo real del agresor seleccionado por la víctima.

Flujos alternativos

1. NO HAY AGRESORES
   Del flujo principal CARGA DE AGRESORES. El sistema muestra un mensaje alertando que no hay agresores asignados para el usuario. La lista desplegable queda vacía.
2. NO HAY INFORMACIÓN
   Del flujo principal UBICACIÓN DEL AGRESOR. El sistema muestra la última posición conocida en un mapa y a su vez un mensaje indicando que no hay información de la posición actual del agresor.

CU3: Ver ruta de agresor

Precondiciones

La víctima se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN
   El usuario va a la interfaz de navegación y pulsa en "Ver ruta de agresor".
2. CARGA DE AGRESORES
   El sistema carga en una lista desplegable todos los agresores del usuario.
3. SELECCIÓN DE AGRESOR
   El usuario selecciona el agresor que desea monitorizar.

4. FECHAS DE BÚSQUEDA

   El usuario ingresa las fechas de búsqueda.

5. VER MAPA

   El usuario pulsa en el botón "Ver en el Mapa".

6. PROCESO DE CONSULTA

   El sistema procesa y devuelve el recorrido en el mapa.

7. DETALLE DE UBICACIÓN

   El usuario pulsa en cualquiera de los puntos que representan ubicaciones y el sistema mostrará el número y la hora de dicha ubicación.

Flujos alternativos

1. NO HAY AGRESORES

   Del flujo principal CARGA DE AGRESORES. El sistema muestra un mensaje alertando que no hay agresores asignados para el usuario. La lista desplegable queda vacía.

2. NO HAY RESULTADOS

   Del flujo principal PROCESO DE CONSULTA. Si no se encuentra resultados entre las fechas ingresadas, el sistema emitirá un mensaje en la pantalla informando que no se encontraron resultados.

CU4: Buscar policía más cercano

Precondiciones

La víctima se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN

   El usuario va a la interfaz de navegación y pulsa en "Buscar policía por kilómetro".

2. DISTANCIA A BUSCAR

   El usuario ingresa la distancia a buscar en kilómetros.

3. MUESTRA DE POLICÍAS

   El sistema muestra los policías que se encuentran en el rango de búsqueda ingresado por el usuario. Muestra su ubicación en tiempo real.

4. DETALLE DE POLICÍA

   El usuario pulsa en el punto que representa una ubicación y el sistema mostrará el nombre del policía.

Flujos alternativos

1. DISTANCIA POR DEFECTO

   Del flujo principal DISTANCIA A BUSCAR. Si el usuario no ha ingresado ningún valor para la distancia, el sistema considera la distancia por defecto de un kilómetro.

2. NO HAY POLICÍAS

   Del flujo principal MUESTRA DE POLICÍAS. El sistema emitirá un mensaje informando que no hay policías en el radio de búsqueda ingresado. El mapa se muestra vacío.

CU5: Enviar alerta

Precondiciones

La víctima se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN

   El usuario va a la interfaz de navegación y pulsa en la barra de menú para cerrar la lista de opciones de la aplicación.

2. BOTÓN DE ALERTA

   El usuario pulsa el botón de alerta.

3. MENSAJE DE ENVÍO

   El sistema muestra un mensaje de envío exitoso.

Flujos alternativos

1. SATURACIÓN DE ALERTAS

   Del flujo principal MENSAJE DE ENVÍO. Si el usuario vuelve a emitir una alerta en un lapso de tiempo menor a tres minutos, el sistema mostrará que ya se ha enviado una alerta y que la alerta ya ha sido procesada con anterioridad.

A.3. Policía

CU1: Ver posición de víctimas

Precondiciones

El policía se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN

   El usuario va a la interfaz de navegación y pulsa en "Ver víctimas".

2. MUESTRA DE VÍCTIMAS

   El sistema devuelve la posición en tiempo real de las víctimas registradas en el sistema. Se muestran en un mapa.

3. DETALLE DE VÍCTIMA

   El usuario pulsa en el punto que representa una ubicación y el sistema mostrará el nombre de la víctima.

Flujos alternativos

1. NO HAY VÍCTIMAS

   Del flujo principal MUESTRA DE VÍCTIMAS. El sistema muestra un mensaje informando que no hay víctimas registradas en el sistema. El mapa se muestra vacío.

CU2: Ver ruta de víctimas

Precondiciones

El policía se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN
   El usuario va a la interfaz de navegación y pulsa en "Ver ruta de víctimas".
2. CARGA DE VÍCTIMAS
   El sistema carga en una lista desplegable todas las víctimas del sistema.
3. SELECCIÓN DE VÍCTIMA
   El usuario selecciona la víctima que desea monitorizar.
4. FECHAS DE BÚSQUEDA
   El usuario ingresa las fechas de búsqueda.
5. VER MAPA
   El usuario pulsa en el botón "Ver en el Mapa".
6. PROCESO DE CONSULTA
   El sistema procesa y devuelve el recorrido en el mapa.
7. DETALLE DE UBICACIÓN
   El usuario pulsa en cualquiera de los puntos que representan ubicaciones y el sistema mostrará el número y la hora de la ubicación.

Flujos alternativos

1. NO HAY VÍCTIMAS
   Del flujo principal CARGA DE VÍCTIMAS. El sistema muestra un mensaje informando que no hay víctimas en el sistema. La lista desplegable queda vacía.
2. NO HAY RESULTADOS
   Del flujo principal PROCESO DE CONSULTA. Si no se encuentra

resultados entre las fechas ingresadas, el sistema emitirá un mensaje en la pantalla informando que no se encontraron resultados.

CU3: Buscar víctimas por distancia

Precondiciones

El policía se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN
   El usuario va a la interfaz de navegación y pulsa en "Buscar víctima por kilómetro".
2. DISTANCIA A BUSCAR
   El usuario ingresa la distancia a buscar en kilómetros.
3. MUESTRA DE VÍCTIMAS
   El sistema muestra las víctimas que se encuentran en el rango de búsqueda ingresado por el usuario. Muestra su ubicación en tiempo real.
4. DETALLE DE VÍCTIMA
   El usuario pulsa en el punto que representa una ubicación y el sistema mostrará el nombre de la víctima.

Flujos alternativos

1. DISTANCIA POR DEFECTO
   Del flujo principal DISTANCIA A BUSCAR. Si el usuario no ha ingresado ningún valor para la distancia, el sistema considera la distancia por defecto de un kilómetro.
2. NO HAY VÍCTIMAS
   Del flujo principal MUESTRA DE VÍCTIMAS. El sistema muestra un mensaje informando que no hay víctimas en el radio de búsqueda ingresado. El mapa se muestra vacío.

CU4: Ver posición de agresores

Precondiciones

El policía se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN
   El usuario va a la interfaz de navegación y pulsa en "Ver agresores".
2. MUESTRA DE AGRESORES
   El sistema devuelve la posición actual de los agresores registrados en el sistema en tiempo real. Se muestran en un mapa.
3. DETALLE DE AGRESOR
   El usuario pulsa el punto que representa una ubicación y el sistema mostrará el nombre del agresor.

Flujos alternativos

1. NO HAY AGRESORES
   Del flujo principal MUESTRA DE AGRESORES. El sistema muestra un mensaje informando que no hay agresores registrados en el sistema. El mapa se muestra vacío.

CU5: Ver ruta de agresores

Precondiciones

El policía se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN
   El usuario va a la interfaz de navegación y pulsa en "Ver ruta de agresores".

2. CARGA DE AGRESORES

   El sistema carga en una lista desplegable todos los agresores del sistema.

3. SELECCIÓN DE AGRESOR

   El usuario selecciona el agresor que desea monitorizar.

4. FECHAS DE BÚSQUEDA

   El usuario ingresa las fechas de búsqueda.

5. VER MAPA

   El usuario pulsa en el botón "Ver en el Mapa".

6. PROCESO DE CONSULTA

   El sistema procesa y devuelve el recorrido en el mapa.

7. DETALLE DE UBICACIÓN

   El usuario pulsa en cualquiera de los puntos que representan ubicaciones y el sistema mostrará el número y la hora de la ubicación.

Flujos alternativos

1. NO HAY VÍCTIMAS

   Del flujo principal CARGA DE AGRESORES. El sistema muestra un mensaje informando que no hay agresores en el sistema. La lista desplegable queda vacía.

2. NO HAY RESULTADOS

   Del flujo principal PROCESO DE CONSULTA. Si no se encuentra resultados entre las fechas ingresadas, el sistema emitirá un mensaje en la pantalla informando que no se encontraron resultados.

CU6: Buscar agresores por distancia

Precondiciones

El policía se encuentra registrado en el servidor. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN

   El usuario va a la interfaz de navegación y pulsa en "Buscar agresor por kilómetro".

2. DISTANCIA A BUSCAR

   El usuario ingresa la distancia a buscar en kilómetros.

3. MUESTRA DE AGRESORES

   El sistema muestra los agresores que se encuentran en el rango de búsqueda ingresado por el usuario. Muestra su ubicación en tiempo real.

4. DETALLE DE AGRESOR

   El usuario pulsa el punto que representa una ubicación y el sistema mostrará el nombre del agresor.

Flujos alternativos

1. DISTANCIA POR DEFECTO

   Del flujo principal DISTANCIA A BUSCAR. Si el usuario no ha ingresado ningún valor para la distancia, el sistema considera la distancia por defecto de un kilómetro.

2. NO HAY AGRESORES

   Del flujo principal MUESTRA DE AGRESORES. El sistema muestra un mensaje informando que no hay agresores en el radio de búsqueda ingresado. El mapa se muestra vacío.

CU7: Buscar personas por tipo

Precondiciones

El policía se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN

   El usuario va a la interfaz de navegación y pulsa en "Ver más personas en el mapa".

2. CARGA DE USUARIOS

   El sistema cargará todos los usuarios en dos listas, una para los agresores y otra para las víctimas.

3. AGREGAR MÁS USUARIOS

   El usuario pulsa en el botón " + " para poder seleccionar a un usuario.

4. SELECCIÓN DE USUARIO

   El usuario selecciona de una lista desplegable según que tipo de usuario a seleccionar.

5. VER EN MAPA

   El usuario pulsa en el botón "Ver en el Mapa".

6. MUESTRA DE USUARIOS

   El sistema devuelve la posición en tiempo real de los usuarios seleccionados. Los usuarios se muestran en un mapa.

7. DETALLE DE USUARIO

   El usuario pulsa el punto que representa una ubicación y el sistema mostrará el nombre del usuario.

Flujos alternativos

1. NO HAY USUARIOS

   Del flujo principal CARGA DE USUARIOS. El sistema muestra un mensaje informando que no hay usuarios en el sistema. La lista desplegable queda vacía.

2. LÍMITE DE USUARIOS

   Del flujo principal AGREGAR MÁS USUARIOS. El sistema permite seleccionar como máximo cuatro usuarios. En caso de tratar de sobrepasar este límite se mostrará un usuario indicando que ya se encuentra en la cantidad máxima permitida.

3. NO HAY RESULTADOS

   Del flujo principal PROCESO DE CONSULTA. El sistema emitirá un mensaje en la pantalla informando que no se encontraron resultados.

CU8: Ver alertas

Precondiciones

El policía se encuentra registrado en *SAVIAs*. Y se encuentra en la interfaz de navegación.

Flujo principal

1. MENÚ DE NAVEGACIÓN

   El usuario va a la interfaz de navegación y pulsa en Registro de alertas".

2. CARGA DE ALERTAS

   El sistema cargará todas alertas mostrándolas en una lista. Se mostrará un mensaje indicando que para ver más detalles de la alerta se debe pulsar en una de éstas.

3. MIS ALERTAS

   El usuario pulsa en "Mis alertas". El sistema mostrará las alertas que han sido asignadas al usuario.

4. DETALLE DE ALERTAS

   El sistema mostrará información adicional de la alerta seleccionada. Como el estado y el tipo de la alerta.

5. VER EN MAPA

   El usuario pulsa en "VER EN MAPA".

6. VISUALIZACIÓN DE ALERTA

   El sistema muestra una interfaz que contiene detalles de la alerta, la distancia en tiempo real entre la víctima y el agresor y también la distancia entre la víctima y el policía. Adicionalmente posee las ubicaciones de los usuarios involucrados con la ayuda de un mapa.

7. DETALLES DE USUARIO

   El usuario pulsa el punto que representa una ubicación y el sistema mostrará el nombre del usuario y el rol que desempeña.

Flujos alternativos

1. NO HAY ALERTAS

   Del flujo principal CARGA DE ALERTAS. El sistema muestra un mensaje alertando que no hay alertas en el sistema. La lista queda vacía.

2. NO HAY RESULTADOS

   Del flujo principal MIS ALERTAS. El sistema emitirá un mensaje en la pantalla alertando que no se encontraron resultados.

3. ERROR EN LA UBICACIÓN

   Del flujo principal VISUALIZACIÓN DE ALERTA. El sistema notificará con un mensaje que hay problemas en el servicio de localización y por ende no puede mostrar información en tiempo real.

4. ALERTA FINALIZADA

   Del flujo principal VISUALIZACIÓN DE ALERTA. El sistema emitirá un mensaje que la alerta ya no está activa y no puede ser mostrada en tiempo real. Se muestra información del momento en que la alerta ha sido registrada en *SAVIAs*.

5. MONITORIZACIÓN DE ALERTA NO PERMITIDA

   Del flujo principal VISUALIZACIÓN DE ALERTA. El sistema emitirá un mensaje que la alerta no ha sido asignada al usuario y por lo tanto no puede ser mostrada en tiempo real. Se muestra información del momento en que la alerta ha sido registrada.

CU9: ABRIR NOTIFICACIÓN DE ALERTA

Precondiciones

El policía se encuentra registrado en *SAVIAs*. Y llega una notificación de alerta al dispositivo.

Flujo principal

1. INGRESO A LA ALERTA

   El usuario pulsa en la notificación.

2. VISUALIZACIÓN DE ALERTA

   El sistema muestra la una interfaz que contiene detalles de la alerta, la distancia en tiempo real entre la víctima y el agresor y la distancia en tiempo real entre la víctima y el policía. Adicionalmente posee las ubicaciones de los usuarios involucrados con la ayuda de un mapa.

3. DETALLES DE USUARIO

   El usuario pulsa el punto que representa una ubicación y el sistema mostrará el nombre del usuario y el rol que desempeña.

Flujos alternativos

1. ERROR EN LA UBICACIÓN

   Del flujo principal VISUALIZACIÓN DE ALERTA. El sistema notificará con un mensaje que hay problemas en el servicio de localización y por ende no puede mostrar información en tiempo real.

Apéndice B

Alertas en tiempo real en una arquitectura Cloud Computing

En este apartado se detallará una primera implementación de detección de alertas en tiempo real en un sistema distribuido tipo cloud.

Como primer paso, se fija automáticamente un perímetro de prevención de 2km alrededor de la víctima. El sistema recibe la posición de la víctima y busca en la base de datos su agresor asociado, devolviendo la posición actual de éste último. Paralelamente a esta recepción de ubicación, se guarda esta posición de la víctima en la base de datos.

Una vez que se tiene al agresor asociado a la víctima, se calcula la distancia que existe entre ellos, utilizando el módulo **Geolib**, si la distancia es mayor que el perímetro de prevención se envía la respuesta indicando que la operación se hizo de manera exitosa mediante la señal 'OK' e indicando que el servicio de localización sea mediante *Triangulación*.

Otro caso es cuando la distancia obtenida sea menor que 2km pero mayor que 1km. Lo que decide *SAVIAs* en esta ocasión es que la localización sea mediante GPS. En el código B.1 se muestra un ejemplo de la orden de cambio de sistema de localización por partes de *SAVIAs*.

```
var response=JSON.stringify({
    "status":"OK",
    "provider":"gps"
})
res.end(response);
```

CÓDIGO B.1: Orden de cambio a GPS.

En el caso que la distancia sea menor que 1km, el sistema generará automáticamente una alerta al policía más cercano.

Dicho de otra manera, el sistema selecciona al policía más cercano, calculando la posición de todos los policías del sistema con respecto a la víctima, el que tenga menor distancia será el seleccionado. Una vez seleccionado el policía, se le envía un mensaje con información de la alerta a través FCM, este procedimiento de envío de alerta será explicado más adelante.

La información que contiene una alerta es la siguiente:

- Identificador único de la alerta generada.
- Posición actual del agresor, víctima.
- Usuarios involucrados, es decir, nombres y apellidos de la víctima, agresor y policía.
- Distancia entre víctima y agresor y distancia entre víctima y policía.
- Fecha y hora de la generación de la alerta.
- Estado de la alerta, que en este caso es 'Activa'.
- Tipo de alerta, que en este caso es 'Automática'.

Adicionalmente al envío de la alerta, ésta es almacenada en una base de datos que nos permitirá tener un registro histórico de alertas.

Para el caso del análisis de ubicación del agresor, sólo se hará hasta la verificación de perímetro de prevención, es decir, si la distancia a la víctima asociada es menos de 2km se utilizará el posicionamiento mediante GPS.

El motivo por el que se realiza hasta este punto es que sólo se busca ganar precisión en la ubicación del agresor, permitiendo tener una alta fiabilidad en el cálculo de la distancia al analizar la posición de la víctima, dando como resultado la generación de alertas de manera efectiva.

Por ahora, para el policía no se ha registrado condiciones que permita la orden de cambio de sistema de localización a través de *SAVIAs*. En otras palabras, este procedimiento se limita a aceptar la ubicación del policía y almacenarlo en la base de datos.

A continuación, se muestra en la figura B.1 un diagrama de flujo que resume el procedimiento de análisis de coordenadas de *SAVIAs*.

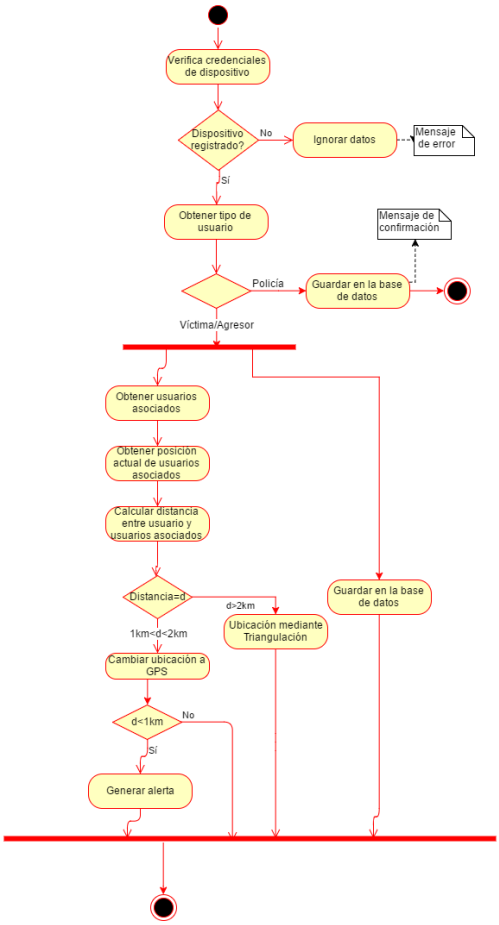

FIGURA B.1: Diagrama de secuencia de Análisis en tiempo real.

Apéndice C

Backlog y Sprints

En este apartado se hablará del backlog, también conocido como la pila de tareas o historias de usuario a realizar en la implementación para cumplir las funcionalidades necesarias del sistema.

Como se ha dicho antes, este trabajo se ha dividido en bloques de trabajo de cada quince días aproximadamente, a estos bloques le llamaremos sprints. En este caso, cada sprint estará marcado por una reunión entre el equipo de trabajo y el cliente. El resultado de un sprint completo, es un producto potencialmente entregable al cliente.

En estas reuniones se mostrará el producto al cliente para ser evaluado y posteriormente aceptado finalizando el trabajo, incorporando nuevas funcionalidades o en su defecto realizar algunas modificaciones.

A continuación se muestran los sprints con sus respectivas historias de usuarios priorizadas de mayor a menor valor realizadas a lo largo de este proyecto ordenados cronológicamente.

C.1. Sprint 1

Sprint 1 - 15/03/16
Como víctima quiero enviar alertas al policía algún policía.
Como víctima quiero realizar la búsqueda de policías por kilómetro.

Sprint 1 - 15/03/16
Como un usuario de la aplicación quiero una transferencia segura de datos.

CUADRO C.1: Historias de usuario y tareas del Sprint 1

C.2. Sprint 2

Sprint 2 - 05/04/16
Como administrador quiero que las alertas sean enviadas a la plataforma web.
Como policía quiero buscar víctimas y agresores por diámetro.
Como un usuario de la aplicación quiero una transferencia segura de datos.

CUADRO C.2: Historias de usuario y tareas del Sprint 2

C.3. Sprint 3

Sprint 3 - 12/04/16
Migración de base de datos de MySQL a MongoDB.
Cambiar servidor web Apache y PHP por Node.js y Express.
Como un usuario de la aplicación quiero una transferencia segura de datos.

CUADRO C.3: Historias de usuario y tareas del Sprint 3

C.4. Sprint 4

Sprint 4 - 02/05/16
Cambiar servidor web Apache y PHP por Node.js y Express.
Como un usuario de la aplicación quiero una transferencia segura de datos.

CUADRO C.4: Historias de usuario y tareas del Sprint 4

C.5. Sprint 5

Sprint 5 - 17/05/16
Como víctima quiero que la alerta generada sea enviada al policía más cercano.
Como un usuario de la aplicación quiero una transferencia segura de datos.

CUADRO C.5: Historias de usuario y tareas del Sprint 5

C.6. Sprint 6

Sprint 6 - 02/06/16
Realizar el análisis de ubicaciones llegadas al servidor en tiempo real.
Generación de alertas automáticas por violación del perímetro de seguridad.

CUADRO C.6: Historias de usuario y tareas del Sprint 6

C.7. Sprint 7

Sprint 7 - 15/06/16
Implementar perímetro de prevención para cambio entre GPS y Triangulación para ahorro de batería
Como usuario de la aplicación quiero que el consumo de batería sea mínimo dado que el GPS tiene un alto consumo.

CUADRO C.7: Historias de usuario y tareas del Sprint 7

C.8. Sprint 8

Sprint 8 - 05/07/16
Realización de pruebas en España con 4 usuarios en simultáneo.
Como policía quiero monitorizar la alerta en tiempo real para poder llegar más rapido a la víctima o agresor.

CUADRO C.8: Historias de usuario y tareas del Sprint 8

C.9. Sprint 9

Sprint 9 - 19/07/16
Como administrador quiero que se alerte cuando un usuario no envía datos en X minutos.
Como policía quiero encontrar la ruta más próxima al agresor o a la víctima.

Sprint 9 - 19/07/16
Optimización del sistema de envío de alertas a un sólo policía.
Como policía necesito un formulario indicando cómo se ha solucionado una alerta. Realizar pruebas con más usuarios para detección de errores.

CUADRO C.9: Historias de usuario y tareas del Sprint 9

C.10. Sprint 10

Sprint 10 - 02/08/16
Como policía quiero encontrar la ruta más próxima al agresor o a la víctima.
Quitar interfaz de verificación de las aplicaciones.
Corrección de errores en la interfaz de las aplicaciones.
Implementar un nuevo servicio de envío de ubicación en background que siga activo aunque aplicación esté cerrada.

CUADRO C.10: Historias de usuario y tareas del Sprint 10

C.11. Sprint 11

Sprint 11 - 17/08/16
Implementar un nuevo servicio de envío de ubicación en background que siga activo aunque aplicación esté cerrada.
Optimización del sistema de envío de alertas a un sólo policía.

CUADRO C.11: Historias de usuario y tareas del Sprint 11

C.12. Sprint 12

Sprint 12 - 31/08/16
Como policía necesito un formulario indicando cómo se ha solucionado una alerta. Realizar pruebas con más usuarios para detección de errores.
Como usuario quiero tener más detalles en mi perfil como por ejemplo una fotografía.

CUADRO C.12: Historias de usuario y tareas del Sprint 12

C.13. Sprint 13

Sprint 13 - 16/09/16
Migrar de Google Cloud Message a Firebase.

CUADRO C.13: Historias de usuario y tareas del Sprint 13

C.14. Sprint 14

Sprint 14 - 01/10/16
Actualizar paquetes del sdk de Android.
Crear y actualizar las nuevas claves de API para los servicios de google.

CUADRO C.14: Historias de usuario y tareas del Sprint 14

C.15. Sprint 15

Sprint 15 - 16/10/16
Corrección de errores en la aplicaciones al iniciar los servicios.
Iniciar con la redacción de tesis.

CUADRO C.15: Historias de usuario y tareas del Sprint 15

C.16. Sprint 16

Sprint 16 - 02/02/17
Realizar proceso de arranque automático del entorno SAVIA y sus servicios al iniciar el servidor.
Actualizar paquetes y librerías del servicio web SAVIA.

CUADRO C.16: Historias de usuario y tareas del Sprint 16

C.17. Sprint 17

Sprint 17 - 20/02/17
Migrar SAVIA a servidor on premise en CTIC - UNI para realización de pruebas masivas.
Actualización de servicios de Google y mantenimiento de la aplicación.

CUADRO C.17: Historias de usuario y tareas del Sprint 17

C.18. Sprint 18

Sprint 18 - 05/03/17
Creación de simulador para pruebas masivas de más de 600 usuarios.
Entrega de redacción final de tesis.

CUADRO C.18: Historias de usuario y tareas del Sprint 18