UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

Aplicación Móvil para reportar objetos perdidos

PATRICIO ANTONIO PINO SALAZAR

Profesores Guía: DANIEL MORENO
FEDERICO MEZA

Memoria para optar al título de
Ingeniero Civil en Computación

Curicó – Chile
26 de marzo de 2018

CONSTANCIA

La Dirección del Sistema de Bibliotecas a través de su encargado Biblioteca Campus Curicó certifica que el autor del siguiente trabajo de titulación ha firmado su autorización para la reproducción en forma total o parcial e ilimitada del mismo.

Curicó, 2019

Dedicado a mis padres.

AGRADECIMIENTOS

En primer lugar agradecer a mi padre Manuel Pino, mi madre Nancy Salazar y mi hermano Felipe, por el esfuerzo económico y constante preocupación para hacer de esta etapa universitaria un sueño hecho realidad.

Agradecer a mi amigos Patricio Valderrama y Yanira Guevara, por el constante apoyo en los momentos difíciles.

También agradecer a los profesores Daniel Moreno y Federico Meza, por su gran disposición y paciencia durante el desarrollo de este proyecto.

Por último, agradecer a todos mis amigos, compañeros y profesores, por hacer de mi paso por esta universidad una gran experiencia.

RESUMEN

Gran parte de las personas, alguna vez ha perdido un objeto que tiene un valor económico o sentimental añadido. En muchos casos este suceso, genera un sentimiento de angustia que desaparece solo si se recupera el objeto o uno idénticamente al original sin un costo para el afectado. Recuperar el objeto se vuelve complicado ya que existe una vaga idea de donde podría estar y recorrer el camino o una área no asegura encontrar el objeto. Si bien existen aplicaciones, plataformas y otro tipo de soluciones tecnológicas para abordar el problema de encontrar objetos perdidos, estas herramientas no ofrecen prestaciones particulares para hacer mas efectivo el reporte y la búsqueda de objetos.

El objetivo de este proyecto de título, es implementar una aplicación móvil que sirva de ayuda para aumentar las posibilidades de encontrar un objeto perdido, mediante la colaboración de otros usuarios, la georreferenciación, creación y visualización de reportes en la aplicación móvil.

Mediante un análisis de los resultados obtenidos por distintas pruebas, se puede concluir que la aplicación desarrollada durante este proyecto, cumple con las funcionalidades necesarias para aumentar las posibilidades de recuperar un objeto.

ABSTRACT

A lot of people have ever lost an object that has a value economic or sentimental added. In many cases this event generates a feeling of anguish that disappears only if the object is recovered or one identically original without a cost for the affected. Recovering the object becomes complicated already that there is a vague idea of where it could be and walk the road or an area not it ensures finding the object. While there are applications, platforms and other types of technological solutions to address the problem of finding lost objects, these tools do not offer particular benefits to make more effective the report and the search for objects.

The objective of this title project is to implement a mobile application that help to increase the chances of finding a lost object, through the collaboration of other users, georeferencing, creation and visualization reporting in the mobile application.

Through an analysis of the results obtained by different tests, is possible conclude that the application developed during this project complies with the necessary to increase the chances of recovering an object.

TABLA DE CONTENIDOS

página

Dedicatoria i

Agradecimientos ii

Tabla de Contenidos iii

Índice de Figuras vii

Índice de Tablas ix

Resumen xi

1. Introducción **12**

1.1. Contexto del proyecto 12
1.2. Definición del problema 13
1.3. Trabajo Relacionado 13
1.4. Propuesta de solución 13
1.5. Objetivos 14
1.6. Alcances 14
1.7. Resumen 15

2. Marco teórico **16**

2.1. Aplicación móvil para reportar objetos perdidos o encontrados 16
2.2. Componentes de la aplicación móvil 17
2.3. Elementos de dibujo en un mapa de Google 17
2.4. Sistema de coordenadas geográficas 18
2.5. Arquitectura Cliente-Servidor 19
2.6. API 20
2.6.1. APIs de servicios web 20
2.6.2. APIs basadas en bibliotecas 21
2.6.3. APIs basadas en clases 21
2.7. Resumen 21

3. Análisis del problema **22**
3.1. Definición del problema . . . 22
3.2. Estado del arte . . . 23
3.3. Solución . . . 26
3.3.1. Funcionalidades de la aplicación móvil . . . 26
3.3.2. Servicio web . . . 27
3.4. Solución alternativa . . . 28
3.5. Resumen . . . 28

4. Metodología de desarrollo **29**
4.1. Metodología . . . 29
4.1.1. Scrum . . . 29
4.1.2. Scrum modificado . . . 31
4.1.3. Planificación . . . 32
4.2. Requisitos . . . 32
4.3. Diseño . . . 35
4.3.1. Diagrama de Clases . . . 35
4.3.2. Casos de Uso . . . 37
4.3.3. Diseño de la Base de Datos . . . 41
4.3.4. Arquitectura física . . . 44
4.3.5. Diseño de Interfaz . . . 45
4.4. Resumen . . . 47

5. Desarrollo **49**
5.1. Ambiente operacional . . . 49
5.2. Instalación y configuración . . . 50
5.2.1. Android Studio . . . 50
5.2.2. Ruby on Rails . . . 51
5.2.3. Ngrok . . . 52
5.3. Implementación de la aplicación móvil . . . 53
5.3.1. Integración con la API de Google Map . . . 53
5.3.2. Agregar un marcador al Mapa . . . 56
5.3.3. Agregar una polilínea al Mapa . . . 58
5.3.4. Agregar un círculo al Mapa . . . 59

5.3.5. Servicio y solicitud de ubicación 60
5.3.6. Distancia a un reporte 63
5.3.7. Envío y Recepción de datos 65
5.3.8. Transformación de una imagen 68
5.4. Implementación del Servicio Web 69
5.4.1. Configuración de un nuevo servidor web 69
5.4.2. Modelos de negocio 69
5.4.3. Acciones del Controlador 70
5.5. Resumen 72

6. Pruebas y Resultados **73**
6.1. Pruebas 73
6.1.1. Cuestionario de preguntas de usuario 74
6.1.2. Pruebas de Caja Negra 77
6.2. Resultados 82
6.2.1. Resultados de Experiencias de Usuario 82
6.2.2. Resultados de Casos de Prueba 87
6.3. Análisis de los resultados 101
6.3.1. Resultado de los gráficos 102
6.3.2. Resultado de los casos de prueba 104
6.4. Resumen 104

7. Conclusiones **105**
7.1. Trabajo futuro 107

Glosario **108**

Bibliografía **109**

Anexos

A: Tablas de requisitos y Matriz de trazado **112**
A.1. Requisitos de Sistema 112
A.2. Matriz de trazado 116

B: Bosquejos de interfaz de usuario **117**

C: Código fuente **125**

C.1. Crear notificaciones en Android 125

C.2. Controlador de Reportes del Web Service 126

ÍNDICE DE FIGURAS

página

2.1. Sistema de coordenadas geográficas. 19

4.1. El proceso de Scrum. 30
4.2. Diagrama de Clases. 36
4.3. Caso de Uso: CU01, Inicio de Sesión. 37
4.4. Caso de Uso: CU02, Crear reporte. 38
4.5. Caso de Uso: CU03, Eliminar reporte. 39
4.6. Caso de Uso: CU04, Crear reporte perdido. 40
4.7. Caso de Uso: CU05, Crear reporte encontrado. 41
4.8. Modelo relacional. 42
4.9. Arquitectura física Cliente-Servidor. 44
4.10. Bosquejo de Visualización y creación de Reportes. 46
4.11. Bosquejo de crear reporte. 47

5.1. Túnel con Ngrok. 53
5.2. Dependencias de compilación para el proyecto en Android Studio. . . 54
5.3. Archivo xml con clave de Google Maps API. 55
5.4. Visualización del mapa de Google en un proyecto Android. 56
5.5. Marcador creado mediante una coordenada de latitud y longitud. . . 58
5.6. Polilínea creada mediante coordenadas de latitud y longitud. 59
5.7. Círculo creado mediante una coordenada de latitud, longitud y radio de 75 metros. 60

6.1. Gráfico con porcentajes de respuestas de la pregunta 2. 82
6.2. Gráfico con porcentajes de respuestas de la pregunta 3. 83
6.3. Gráfico con porcentajes de respuestas de la pregunta 4. 83
6.4. Gráfico con porcentajes de respuestas de la pregunta 5. 84
6.5. Gráfico con porcentajes de respuestas de la pregunta 6. 84
6.6. Gráfico con porcentajes de respuestas de la pregunta 7. 85
6.7. Gráfico con porcentajes de respuestas de la pregunta 8. 85
6.8. Gráfico con porcentajes de respuestas de la pregunta 9. 86
6.9. Gráfico con porcentajes de respuestas de la pregunta 10. 86

6.10. Gráfico con porcentajes de respuestas de la pregunta 11. 87
6.11. Mensaje de fallo de auntenticación. 88
6.12. Transición desde el inicio de sesión hacia la pantalla principal. 89
6.13. Mensaje de fallo al crear un reporte perdido o encontrado cuando faltan valores requeridos. 90
6.14. Mensaje de éxito cuando guarda un reporte en el Servicio Web. . . . 91
6.15. Información de un reporte al tocar el marcador e información detallada cuando se desliza el panel hacia arriba. 92
6.16. Imagen agregada a la creación de un reporte. 93
6.17. Mensaje de registro de usuario exitoso y redirección a la pantalla de Inicio se Sesión. 94
6.18. Mensaje de fallo cuando se intenta registrar un usuario con un email existente en la base de datos. 95
6.19. Mensaje de fallo cuando se intenta registrar un usuario con campos requeridos faltantes. 96
6.20. Eliminar un reporte desde el menú Mis Reportes y eliminación del reporte en el mapa. 97
6.21. Eliminar un punto del mapa cuando se crea la ubicación para el reporte. 98
6.22. Seleccionar número telefónico y realización de llamada al número. . . 99
6.23. Seleccionar número telefónico y realización de llamada al número. . . 100
6.24. Notificación creada cuando un usuario se acerca a un reporte. 101

B.1. Inicio de Sesión. 118
B.2. Registro de Usuario 119
B.3. Menú lateral. 120
B.4. Perfil de Usuario. 121
B.5. Mis Reportes. 122
B.6. Agregar ubicación a un reporte encontrado. 123
B.7. Agregar ubicación a un reporte perdido. 124

ÍNDICE DE TABLAS

página

3.1. Tabla comparativa del estado del arte 25

4.1. Tabla Atributo-Descripción para requisitos. 33
4.2. Requisito de Usuario RU01 33
4.3. Requisito de Usuario RU02 34
4.4. Requisito de Usuario RU03 34
4.5. Requisito de Usuario RU04 34
4.6. Requisito de Usuario RU05 34
4.7. Requisito de Usuario RU06 35

5.1. Hardware utilizado 49
5.2. Software utilizado 50

6.1. Caso de prueba 1 77
6.2. Caso de prueba 2 77
6.3. Caso de prueba 3 77
6.4. Caso de prueba 4 78
6.5. Caso de prueba 5 78
6.6. Caso de prueba 6 78
6.7. Caso de prueba 7 78
6.8. Caso de prueba 8 79
6.9. Caso de prueba 9 79
6.10. Caso de prueba 10 79
6.11. Caso de prueba 11 80
6.12. Caso de prueba 12 80
6.13. Caso de prueba 13 80
6.14. Caso de prueba 14 81
6.15. Caso de prueba 15 81
6.16. Caso de prueba 16 81

A.1. Requisito de Sistema RS01 112
A.2. Requisito de Sistema RS02 112

A.3. Requisito de Sistema RS03 113
A.4. Requisito de Sistema RS04 113
A.5. Requisito de Sistema RS05 113
A.6. Requisito de Sistema RS06 113
A.7. Requisito de Sistema RS07 114
A.8. Requisito de Sistema RS08 114
A.9. Requisito de Sistema RS09 114
A.10.Requisito de Sistema RS10 114
A.11.Requisito de Sistema RS11 115
A.12.Requisito de Sistema RS12 115
A.13.Requisito de Sistema RS13 115
A.14.Requisito de Sistema RS14 115
A.15.Requisito de Sistema RS15 115
A.16.Matriz de trazado: requisitos de usuario vs requisitos de sistema . . . 116

1. Introducción

1.1. Contexto del proyecto

Independiente de la época en la que vivimos, la humanidad tiene la necesidad de adquirir bienes, los cuales tienen un valor impuesto, el cual puede ser económico, o simplemente aprecio personal. Debido al valor que aplicamos sobre estos bienes, saber que extraviamos alguna de nuestras pertenencias nos genera un sentimiento de angustia, el cual solo desaparece si volvemos a encontrar nuestra pertenencia, o en algunos casos, si adquirimos una copia idéntica por un costo muy bajo o nulo.

Cuando sucede lo anteriormente descrito, nuestra primera reacción es poner en marcha un plan de búsqueda individual, que toma como punto de inicio el último lugar donde vimos el objeto y finaliza en el punto donde nos percatamos que ya no poseíamos dicho objeto. Si no tenemos éxito, una opción es realizar una búsqueda colaborativa, solicitando ayuda a nuestro grupo de amigos, familiares o simplemente con las personas que te encuentras.

Pero, ¿Qué sucede cuando este elemento es encontrado por una persona desconocida? Actualmente tenemos a disposición distintos dispositivos y servicios que nos permiten comunicarnos sin la necesidad de estar presencialmente en el mismo lugar, como lo son los celulares, computadores o redes sociales. Sin embargo, es un requisito de estos servicios el estar interconectado con nuestros conocidos, ya sea a través de su dirección, ya sea, dirección de email, número de teléfono o nombre de la persona para el caso de la redes sociales. El proyecto que proponemos busca expandir el rango de posibilidades de personas que puedan ayudar con la búsqueda de los objetos perdidos.

1.2. Definición del problema

El problema principal que busca abordar este proyecto, es proveer una aplicación móvil que permita principalmente, a usuarios que no necesariamente se conocen entre si, colaborar en la búsqueda de objetos perdidos en una área definida por un usuario. Ademas, se busca proveer una aplicación móvil de comunicación directa con las personas que potencialmente podrían encontrar el objeto perdido.

La creación de una aplicación móvil colaborativa, permitirá aumentar las posibilidades de éxito en la búsqueda de objetos, los cuales tienen un valor importante y diferente para cada persona.

1.3. Trabajo Relacionado

Actualmente existen aplicaciones móviles para las plataformas Android y iOS, que implementan el concepto de realizar reportes de objetos perdidos y encontrados. Algunas de esas aplicaciones son *Lost & Found* y *Find it- Lost and Found*. Sin embargo, muchas de ellas, no trabajan con ubicaciones georreferenciadas o carecen de detalle al momento de realizar el reporte. Otras aplicaciones existentes como y *Tile* hacen uso de dispositivos que se deben adherir a un objeto para que la aplicación informe sobre la ubicación.

Existen también páginas web donde un usuario puede buscar objetos que hayan sido encontrados por otros usuarios, solución muy distinta a como se quiere trabajar en este proyecto.

1.4. Propuesta de solución

Para el problema explicado en la sección 1.2, y considerando que Chile es el país líder en Latinoamérica en el uso de teléfonos inteligentes, con un 78 %[2] y que el acceso a internet ya se concentra desde estos dispositivos con un 77 %[13], se desea resolver este problema, mediante el desarrollo de un prototipo funcional de una aplicación móvil para el sistema operativo Android, la cual permita a cualquier usuario previamente registrado, realizar un reporte de pérdida geo localizado marcando una ruta posible por donde este objeto podría estar, así como también poder reportar un hallazgo. Además, se pretende que cualquier usuario que transite por una ruta

que contenga un objeto reportado como perdido, pueda recibir una notificación informándole de la posible existencia y la posibilidad de devolverlo a otro usuario.

Como característica principal de la solución, se pretende resolver con el uso de la Api de Google para Google Maps y con el principal hardware de GPS de los teléfonos inteligentes.

1.5. Objetivos

A continuación se exponen el objetivo general y los objetivos específicos que se espera lograr con el desarrollo del proyecto.

Objetivo general

- Crear una aplicación móvil que permita aumentar las posibilidades de encontrar objetos perdidos, mediante la colaboración y georreferenciación.

Objetivos específicos

- Especificar los requisitos de la aplicación móvil.
- Diseñar la arquitectura de la aplicación.
- Diseñar la interfaz gráfica de la aplicación.
- Determinar las tecnologías para el desarrollo de la aplicación basada en el uso de mapas en Android.
- Construir el sistema con las tecnologías investigadas previamente.
- Evaluar las funcionalidades de la aplicación.

1.6. Alcances

La siguiente lista de puntos, corresponde a los límites que abarca el proyecto.

- En este trabajo se espera implementar un prototipo funcional de la idea a desarrollar, por lo tanto se propone desarrollar una interfaz que permita probar la funcionalidad de la aplicación.

- En este proyecto no se considerará el desarrollo en los sistemas operativos IOS, Windows Phone, BlackBerry OS y otros.
- El proyecto no considerará pruebas de carga de trabajo considerando grandes cantidades de usuarios. La cantidad de usuarios de prueba se definirá posteriormente.
- En este proyecto no se considerará la traducción de la interfaz de usuario, a ningún idioma además del español.

1.7. Resumen

En este capítulo se expuso la introducción del problema definido que se espera solucionar en este proyecto a través de hipótesis. Adicionalmente se muestran los objetivos generales y específicos para el desarrollo de este proyecto. Brevemente se enfatiza en los alcances, para finalizar con un plan de trabajo que se utilizó para completar los objetivos anteriormente vistos.
En el siguiente capítulo se verá el Marco teórico del sistema, donde se profundizará acerca de los conceptos que tienen relación con el desarrollo del prototipo de la aplicación móvil y el servicio web.

2. Marco teórico

En este capítulo se profundizarán los conceptos básicos y principales que tienen relación al desarrollo de un prototipo funcional de una aplicación móvil para reportar objetos perdidos. En primera instancia se explicarán los elementos que la componen, tipos de datos utilizados y el estado del arte con una tabla comparativa.

2.1. Aplicación móvil para reportar objetos perdidos o encontrados

Una aplicación móvil para realizar reportes, consiste en un sistema que permite a un usuario realizar un reporte de un objeto perdido o encontrado en una zona en el mapa. En términos computacionales, la aplicación móvil debe tener las siguientes características:

- Captura de datos
- Manejo y almacenamiento de datos.
- Manipulación y análisis de datos.
- Presentación de datos

La aplicación a desarrollar debe proveer una interfaz que permita dar soporte para la creación de reportes que incluyen texto, fecha, hora, imagen e información referenciada geográficamente. Posteriormente, esta información se manipula para ser enviada para su almacenamiento. Se finaliza el proceso mostrando esta información al usuario.

2.2. Componentes de la aplicación móvil

Una aplicación móvil para reportar objetos esta compuesta por componentes que permiten a este sistema funcionar correctamente. A continuación se mencionan y explican cada uno de ellos:

- **Hardware**: En el contexto de este proyecto, se considera como hardware al equipo móvil o computadora de bolsillo que ejecuta la aplicación en desarrollo. Otro hardware presente, es aquel que ejecuta el sistema computacional de la base de datos, este puede variar dependiendo del entorno y pueden ser desde una computadora personal hasta equipos servidores.

- **Aplicación móvil**: Es el componente esencial de este proyecto, ya que permite mediante un conjunto de funciones e interacciones con los usuarios, generar los datos que se manipulan, almacenan y representan a los usuarios de la aplicación en desarrollo mediante reportes o notificaciones. Otro software presente en este proyecto, es el encargado de realizar las transacciones de datos entre la aplicación móvil y la base de datos.

- **Datos o Información**: Es generado por el componente de aplicación móvil. Este permite la interacción entre los usuarios de la aplicación y le dan sentido a la aplicación móvil.

- **Usuarios**: Este componente mediante el uso de la aplicación móvil permite crear los datos de la aplicación móvil. Estos no deben contener conocimientos avanzados para interactuar con la aplicación móvil.

2.3. Elementos de dibujo en un mapa de Google

Los elementos de dibujo básicos permiten personalizar el mapa indicando ubicaciones, las cuales operan como puntos de referencias para el usuario. Estos elementos se describen a continuación:

- **Marcador**: Se utiliza para indicar un punto único en el mapa mediante una coordenada geográfica[9]. Un punto describe una entidad geográfica pequeña como por ejemplo la dirección de una casa o la ubicación de un semáforo.

- **Polilínea**: Está compuesta por una serie de puntos unidos por segmentos de líneas. Cada punto puede representar el inicio, un cambio de dirección o el fin de la línea y pueden ser utilizada para dibujar trazos y rutas en el mapa.
- **Polígono**: En un mapa, un polígono es una porción del mapa delimitada y encerrada por una línea poligonal. Está compuesta por una serie de puntos de coordenadas al igual que una polilínea, sin embargo, en lugar de ser abiertos, un polígono está diseñado para definir regiones como por ejemplo una ciudad, una parcela o el recorrido de un autobús desde que sale del terminal hasta que vuelve.
- **Círculo**: Se define un circulo como todos los puntos de la superficie terrestre que están a ***X*** **radios metros** del punto del centro. Puede ser utilizado por ejemplo para indicar el área de expansión de un temblor.

2.4. Sistema de coordenadas geográficas

Para poder trabajar con un mapa, como agregar elementos o detectar una ubicación, es necesario utilizar un sistema que permita identificar un punto en la Tierra de manera inequívoco. Con este propósito, se creó el sistema de coordenadas geográficas (GCS del inglés, Geographic Coordinate System) que consiste en utilizar una superficie esférica de tres dimensiones para poder definir ubicaciones en la Tierra. Un GCS incluye una unidad angular de medida, un meridiano base y un datum (basado en un esferoide).

Para poder identificar un punto, se utilizan los valores de latitud y longitud, que son ángulos medidos desde el centro de la Tierra hasta la superficie. La linea de latitud que se encuentra en el centro entre los polos norte y sur se llama ecuador y define la latitud cero. Por lo tanto, los valores de latitud se miden respecto al ecuador y fluctúan entre los -90 grados en el polo sur y hasta +90 grados en el polo norte. La linea de longitud cero se llama meridiano base y es la longitud que atraviesa Greenwich, Inglaterra. Sus valores van desde -180 hasta 180 grados cuando se viaja hacia el este.

Existe una tercer elemento para localizar un punto en la superficie terrestre y este es la altitud. En los sistemas de posicionamiento global (GPS del inglés, Global

Positioning System) esta componente toma importancia ya que permite determinar con mayor exactitud un objeto (persona, vehículo, etc).

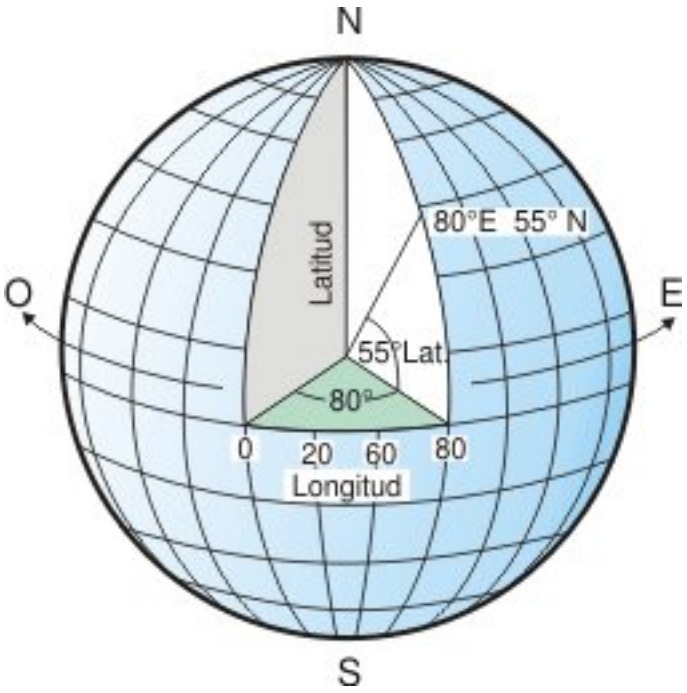

Figura 2.1: Sistema de coordenadas geográficas.

2.5. Arquitectura Cliente-Servidor

La arquitectura cliente-servidor, en un sistema computacional, consiste en tener la aplicación y servidor de datos separados en dos partes. La aplicación, alojada y ejecutada en el cliente interactúa con el servidor mediante solicitudes o envíos de datos a través de una red previamente configurada. El servidor, procesa la solicitud o el envío de datos y responde al cliente con los datos pedidos para ser presentada al usuario. Algunas de las ventajas de implementar este tipo de arquitectura se describen a continuación[4]:

- La aplicación alojada en el cliente no es responsable del procesamiento de los datos, solo deben concentrase en solicitar la información y después de la respuesta del servidor, dada las capacidades del cliente, mostrar la información al usuario.

- La aplicación del cliente se puede diseñar sin depender de la ubicación física del servidor de datos. Si estos datos se distribuyen en otros servidores, el cliente puede seguir funcionando, en algunos casos con una mínima modificación o ninguna.
- El cliente puede optimizarse para la presentación de los datos y el servidor para el almacenamiento y procesamiento de los mismos, por ejemplo, teniendo grandes cantidades de memoria y espacio en disco.

2.6. API

Una API (acrónimo de Application Programming Interface) es una interfaz que facilita la interacción y comunicación entre dos aplicaciones para el intercambio de datos. Cada API está diseñada con un lenguaje de programación concreto y con especificaciones y lógica de negocio que la definen. Por lo tanto, una API proporciona un biblioteca de funciones para que otra aplicación la utilice, creando una capa de abstracción entre el cliente y el servidor.

2.6.1. APIs de servicios web

Son las interfaces de comunicación que permiten el intercambio de datos entre una aplicación y un servicio a través de una URL. Los cuatro tipos de API de servicio web mas habituales se nombran y explican a continuación:

- **SOAP**: Del acrónimo Simple Object Access Protocol, es un tipo de servicio web con un protocolo estándar de intercambio de datos entre dos objetos. El protocolo del mensaje usado en SOAP es XML (acrónimo de eXtensible Markup Language).
- **XML-RPC**: Este servicio funciona en base a llamadas a procedimientos remotos, usando XML como protocolo de datos. El cliente tiene a su disposición métodos o procedimientos que se ejecutan en el servicio, así no tiene que preocuparse de la comunicación entre ambas.
- **JSON-RPC**: Mismo funcionamiento que el servicio anterior, pero usa JSON (acrónimo de JavaScript Object Notation) como formato de datos.

- **REST**: Del acrónimo Representational State Transfer, funciona bajo solicitudes de datos usando el protocolo HTTP. Esta arquitectura está orientada a sistemas hipermedia en World Wide Web.

2.6.2. APIs basadas en bibliotecas

Estas APIs están orientadas a aquellas aplicaciones que necesitan importar una biblioteca de métodos o procedimientos de otra aplicación para ser utilizada en su entorno. La aplicación que importa una biblioteca no tiene la necesidad de implementar tales procedimientos

2.6.3. APIs basadas en clases

Este tipo de interfaces permiten el intercambio de datos en base al uso de las clases. Un ejemplo de esta interfaz se encuentra presente en el lenguaje de programación orientada a objetos como Java. Java usa clases abstractas para crear una aplicación, cada una de estas clases proporcionan todo lo necesario para trabajar con los datos que se utilizan en ese contexto de desarrollo.

2.7. Resumen

En este capítulo se definieron los temas más relevantes para el desarrollo del proyecto. Se abordaron temas como la definición de la aplicación móvil para el reporte de objetivos perdidos y encontrados, así como sus componentes. Posteriormente se interiorizó en cómo se realizaron las técnicas de elementos y sistemas de coordenadas geográficas con los mapas de Google. También se explicaron algunos problemas identificados y cada uno de los componentes de Hardware y Software que se utilizaron. En el siguiente capítulo se enfatizará en el análisis del problema, el estado del arte, la definición del problema y la solución que se planteó.

3. Análisis del problema

En este capítulo se profundizarán los conceptos analíticos de la solución de este proyecto, a través de la definición de la problemática establecida en el reporte de objetos perdidos o encontrados, con un detalle y forma concreta de funcionamiento. A partir de lo anterior, se define la solución, funciones, servicios y alternativas previstas al problema presentado.

3.1. Definición del problema

En la actualidad, estar registrado en al menos una red social es casi obligatorio. En la universidad, en el trabajo, nuestros familiares y amigos nos preguntan cual es tu dirección de correo o el número para agregarte a Wathsapp. El sistema comercial nos ofrece cada día nuevos teléfonos inteligentes con nuevas funcionalidades, ya resulta una tarea difícil encontrar un equipo telefónico convencional solo para realizar llamadas y recibir mensajes de texto. Por lo tanto, es un hecho que tener un teléfono inteligente es una necesidad básica para estar intercomunicados.

Pero, ¿Qué tan bien pueden estas redes sociales adaptarse a una necesidad especifica? Podemos decir con seguridad que Facebook es la red social más completa, ya que permite enviar mensajes, publicar fotos, videos, crear grupos y publicar artículos para su venta, entre otras muchas funcionales. Es aquí donde nace la idea de crear una aplicación móvil para reportar objetos perdidos o encontrados, con gran detalle en la ubicación del reporte. Si bien, las redes sociales actuales permiten poder publicar un estado o una foto indicando que algo se ha perdido, esta no entrega mayor detalle sobre la ubicación, tampoco permite notificar a un usuario cuando este está cerca de la zona del reporte. El segundo motivo que ha llevado a cabo realizar este

proyecto, es que todavía es una opción valida pegar un afiche con la información del reporte en lugares públicos como murales de supermercados, postes o terminales de buses. Esta aplicación no busca eliminar este método, mas bien es entregar una segunda opción para que algún usuario pueda ayudar en la búsqueda. Por último, el tercer motivo, es la experiencia personal del desarrollador del proyecto con el extravío de objetos.

3.2. Estado del arte

Existen muchas aplicaciones desarrolladas para dispositivos móviles que ayudan en la búsqueda de objetos perdidos, algunas dependen de accesorios que se deben adherir a una pertenencia y otras que tan solo debes ingresar una palabra para buscar en la base de datos si es que existe alguna coincidencia. Algunas de las aplicaciones que tratan de resolver este problema se listan y describen a continuación:

- **Lost & Found:** Es una aplicación móvil desarrollada por la empresa Snappii, disponible para los sistemas Android y IOS. En esta se puede realizar un reporte de un objeto perdido o encontrado. El formulario de reportes contiene datos básicos de contacto y descripción del producto, así como también el punto en el mapa donde se realiza el reporte. Estos reportes pueden ser vistos como una lista o como pines en el mapa, sin diferenciar la ubicación actual del usuario.

- **Find it - LaF:** Es una aplicación móvil desarrollada por la empresa Arra Applications, disponible para los sistemas Android y IOS. En ella se puede realizar un reporte o hallazgo con un formulario muy completo que se diferencia de cada categoría del producto en cuestión. Esta aplicación da la opción de pagar un valor por imágenes adicionales para ser incluidas en el reporte, así como también permite ofrecer una recompensa a la persona que encuentre el elemento. Por último, esta aplicación no cuenta con direcciones georreferenciadas, esta solo se agrega como un texto en el reporte.

- **Tile:** Es una aplicación móvil que funciona en conjunto a un pequeño dispositivo bluetooth que se debe adherir a la pertenencia que se desea rastrear, en la aplicación se puede observar en el mapa el lugar donde está ubicado, así como también hacer sonar una alarma para poder ubicarlo fácilmente cuando se encuentre cerca.

- **MissingX:** Es una aplicación móvil disponible para Android y IOS. También tiene una versión disponible para acceder mediante la Web. En ella se puede buscar elementos perdidos ingresando, el país, la fecha y una palabra para poder buscar una coincidencia. También permite registrar un objeto encontrado ingresando el nombre, una descripción del objeto, la fecha, el país y ubicación donde se encontró. No posee interacción con un mapa.

- **LostRFound:** Aplicación móvil para Android disponible en la PlayStore de Google. Esta aplicación permite crear reportes de objetos encontrados y perdidos mediante un formulario, la ubicación del reporte se agrega al reporte ingresando el texto la ubicación. También permite hacer búsquedas rápidas de objetos ingresando una palabra.

- **Traista:** Aplicación disponible para Android y IOS, esta aplicación permite realizar reportes de objetos perdidos, encontrados y ofertas. Para crear un reporte, se debe agregar un título, una descripción y una ubicación mediante la elección de un punto en el mapa, opcionalmente puedes agregar una imagen.

A continuación se muestra una tabla comparativa de todas las aplicaciones investigadas junto a la aplicación en desarrollo llamada en esta tabla MiApp. Mas adelante y como se muestra en algunas capturadas de pantalla esta tiene el nombre de *Find it*.

Cuadro 3.1: Tabla comparativa del estado del arte

Funcionalidad/ Aplicación	Lost & Found	Find it-LaF	Tile	MissingX	LostRFound	Traista	MiApp
Reportar objetos perdidos	Si	Si	No	Si	Si	Si	Si
Reportar objetos encontrados	Si	Si	No	Si	Si	Si	Si
Agregar imagen a reporte	Si	Si	No	No	Si	Si	Si
Agregar ubicación con mapa	Si	No	No	No	Si	Si	Si
Búsqueda colaborativa	Si	Si	No	Si	Si	Si	Si
Buscar objetos	No	Si	No	Si	Si	No	No
Necesidad accesorios	No	No	Si	No	No	No	No
Sugerir ruta para reportes	No	No	No	No	No	No	Si
Visualización según ubicación actual	No	No	Si	No	Si	Si	Si
Ruta para una ubicación	No	No	No	No	No	No	Si
Área para una ubicación	No	No	No	No	No	No	Si
Rastreo de ubicación	No	No	No	No	No	No	Si
Notificación por proximidad para reportes	No	No	No	No	No	No	Si
Visualización de reportes en Mapa	Si	No	No	No	No	Si	Si
Persistencia de datos	Si	Si	Si	Si	Si	Si	Si

3.3. Solución

El sistema a desarrollar, es un prototipo funcional de una aplicación móvil para el sistema operativo Android. La aplicación móvil permite a usuarios registrados, reportar sus pertenencias perdidas, para así, con la ayuda de otros usuarios también registrados, poder trabajar colaborativamente en la búsqueda. Un reporte de un objeto perdido, consta de características que permite a un usuario poder identificarlo con mayor precisión, como: título, descripción, fecha, hora, una ruta posible o un área en el mapa donde podría estar y opcionalmente una imagen. Cuando un usuario encuentra un objeto que esté reportado como perdido, este podrá ponerse en contacto con el creador del reporte, para coordinar la devolución del objeto. Un usuario también puede realizar reportes de objetos que no han sido reportados como perdidos, quedando a la espera para algún usuario lo reclame.

La idea principal es aprovechar las características y autonomía de los teléfonos inteligentes para poder crear una aplicación que esté en constante procesamiento de los datos y ubicación del dispositivo en el mapa.

3.3.1. Funcionalidades de la aplicación móvil

A continuación se presenta el listado de funcionalidades básicas que deben implementarse para llevar a cabo el propósito del prototipo funcional de la aplicación móvil:

- Registrar nuevos usuarios.
- Iniciar sesión.
- Cerrar sesión.
- Registrar reportes de objetos perdidos.
- Registrar reportes de objetos encontrados.
- Eliminar un reporte.
- Visualización de un reporte en el mapa.
- Registrar ruta recorrida por el usuario.

- Filtrar ruta recorrida por el usuario según fecha y hora de inicio y fin.
- Notificar al usuario cuando está cerca de un objeto perdido o encontrado.
- Establecer comunicación con el usuario creador del reporte mediante el número telefónico o email del reporte.
- Persistencia de datos.
- Solicitud de permisos en tiempo de ejecución dependiendo de la versión del sistema Android.

3.3.2. Servicio web

Para poder realizar el registro de los usuarios y de los reportes de la aplicación y que estos estén disponibles para todos los usuarios, en necesario implementar una API de Servicio Web tipo REST para responder a todos los tipos de consultas y envíos de datos. La elección de este tipo de API no es arbitraria, según las funcionalidades de la aplicación y el tiempo de desarrollo del proyecto, una API REST puede ser implementada en poco tiempo.

Como se explica en el capítulo 2, sección 2.6, una API REST se apoya totalmente del estándar HTTP, por lo tanto, puede ser usada por cualquier dispositivo que entienda dicho estándar. Una correcta implementación de una API REST requiere que esta cumpla con 3 niveles de calidad[6] que se mencionan a continuación:

- Uso correcto de URIs
- Uso correcto de HTTP
- Implementar Hypermedia

Cuando se trabaja con una API REST, la información que se consulta, se envía, edita o elimina, es llamada recurso. Así, para acceder a un recurso, se debe hacer por medio de una URL, que es un tipo de URI, que además de permitir identificar de forma única el recurso, permite localizarlo para poder acceder a él o compartir su ubicación.

3.4. Solución alternativa

Un enfoque diferente, orientado a la plataforma web, es el propuesto por el exprofesor de Facultad de Ingeniería de la Universidad de Talca, Ben Ingram. Consta en utilizar distintos componentes de la manipulación de datos sobre mapas, y que al trabajar en conjunto logran el resultado esperado en el desarrollo de este proyecto. Estos componentes se listan y explican a continuación:

- **Base de datos espacial:** Estas bases de datos tienen un tipo de dato "Geometry"que permite almacenar puntos, lineas y polígonos. Un ejemplo de base de datos espacial es PostGIS que es una extensión de Postgresql.
- **Servidor de mapas:** La utilización de GeoServer es una buena opción, debido a que desde la base de datos se puede generar un WMS (acrónimo de Web Map Service) para la representación de la información geográfica en el lado del navegador o cliente.
- **Navegador:** Desde el lado del cliente existen dos opciones para visualizar mapas, OpenLayers o Leaflet, ambos están escritos en JavaScript y permiten conectarse a los servicios web como WMS.

3.5. Resumen

En este capítulo se definieron los temas más relevantes para el análisis del problema. Se abordaron temas como la definición del problema analizado en este proyecto, objetos perdidos y encontrados. Se abordaron temas como la propuesta de solución y sus funcionalidades, para la creación de la aplicación móvil, además del servicio web desarrollado.
En el siguiente capítulo se expondrá la metodología de desarrollo que se utilizó a lo largo del proyecto.

4. Metodología de desarrollo

En este capítulo se profundizará en la forma de trabajo y la planificación realizada a lo largo del proyecto. Además de esto se tratan los temas de requisitos, diseño y base de datos, que son temas previos a la etapa de desarrollo del proyecto. En este proyecto se utilizó una metodología ágil, que permitió avanzar paulatinamente de acuerdo a las necesidades que surgieron en el proyecto, por lo que estos puntos fueron de vital importancia, debido a que tuvieron efecto a lo largo de todo el proceso.

4.1. Metodología

4.1.1. Scrum

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente en equipo y obtener el mejor resultado posible de un proyecto. En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto[10].

Existen tres fases con Scrum. La primera es la planificación del bosquejo, donde se establecen los objetivos generales del proyecto y el diseño de la arquitectura de software. A esto le sigue una serie de ciclos *sprint*, donde cada ciclo se desarrolla un incremento del sistema. Finalmente, la fase de cierre del proyecto concluye el proyecto, completa la documentación requerida, como los manuales del usuario y se valora las lecciones aprendidas en el proyecto[18]. En la figura 4.1 se puede apreciar un diagrama de administración de Scrum.

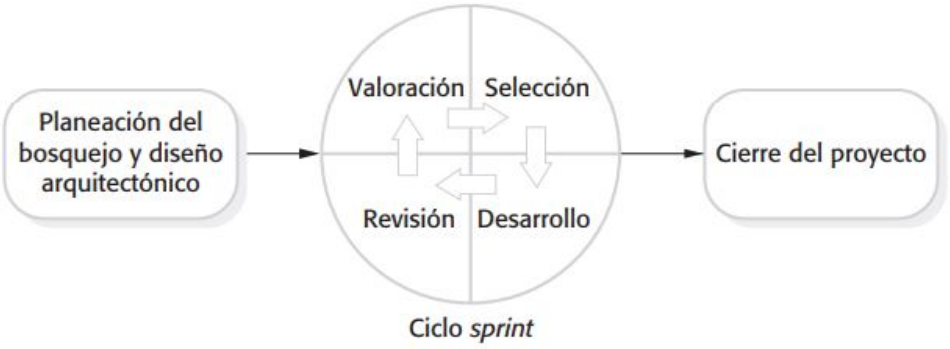

Figura 4.1: El proceso de Scrum.

Un sprint de Scrum es una unidad de planeación en la que se valora el trabajo que se va a realizar, se seleccionan las particularidades por desarrollar y se implementa el software. Al final de un sprint, la funcionalidad completa se entrega a los participantes. Las características clave de este proceso son las siguientes:

- Los sprints tiene una longitud fija, por lo general de dos a cuatro semanas.
- El punto de partida para la planeación es la cartera del producto, que es la lista de trabajo por realizar en el proyecto. Durante la fase de valoración del sprint, esto se revisa, y se asignan prioridades y riesgos. El cliente interviene estrechamente en este proceso y al comienzo de cada sprint puede introducir nuevos requerimientos o tareas.
- La fase de selección incluye a todo el equipo del proyecto que trabaja con el cliente, con la finalidad de seleccionar las características y la funcionalidad a desarrollar durante el sprint.

- Una vez acordado, el equipo se organiza para desarrollar el software. Con el objetivo de revisar el progreso y, si es necesario, volver a asignar prioridades al trabajo, se realizan reuniones diarias breves con todos los miembros del equipo.
- Al final del sprint, el trabajo hecho se revisa y se presenta a los participantes. Luego comienza el siguiente ciclo de sprint.

La idea detrás de Scrum es que debe autorizarse a todo el equipo para tomar decisiones, de modo que se evita deliberadamente el término "administrador del proyecto". En lugar de ello, el "maestro de Scrum", es el facilitador que ordena las reuniones diarias, rastrea el atraso del trabajo a realizar, registra las decisiones, mide el progreso del atraso y se comunica con los clientes y administradores fuera del equipo.

4.1.2. Scrum modificado

Debido al tipo de proyecto, se ha decidido modificar o adaptar algunas de las características de Scrum para poder llevarlo a cabo y obtener los mismos resultados. Como es un proyecto para optar al título de carrera, se han realizado las siguientes modificaciones:

- Los sprints tiene una longitud fija de dos semanas.
- El punto de partida para la planeación es la cartera del producto, que es la lista de trabajo por realizar en el proyecto. Durante la fase de valoración del sprint, esto se revisa, y se asignan prioridades y riesgos. Los profesores guías pueden intervenir estrechamente en este proceso y al comienzo de cada sprint pueden introducir nuevos requerimientos o tareas.
- La fase de selección incluye al único desarrollador del proyecto que trabaja con los profesores guías, con la finalidad de seleccionar las características y la funcionalidad a desarrollar durante el sprint.
- Una vez acordado, el desarrollador se organiza para desarrollar el software. Debido a que solo existe un codificador, no se realizan reuniones diarias, solo una reunión semanal para revisar el progreso del proyecto.

- Al final del sprint, el trabajo hecho se revisa y se presenta a los profesores guías que cumplen el rol de maestros de Scrum y clientes. Luego comienza el siguiente ciclo de sprint.

4.1.3. Planificación

Utilizando como base Scrum, se elaboró una planificación para poder llevar a cabo el desarrollo del prototipo funcional de la aplicación móvil. Esta planificación consta de 14 sprint de 2 semanas de duración cada una. Los primeros sprints tuvieron relación con los requisitos iniciales, definición de la arquitectura física y lógica de la aplicación móvil. Se siguió por la definición de un web service. Los siguientes sprints fueron netamente de implementación de funcionales y solo un caso puntual la reestructuración de los requisitos según los tiempos de entrega del proyecto.

4.2. Requisitos

En esta sección, se presentan los requisitos de usuario y sistema que permiten implementar las funcionalidades del prototipo funcional de la aplicación móvil en desarrollo. Estos fueron capturados mediante entrevistas con el cliente, que en nuestro caso fueron los profesores guiás, Federico Meza y Daniel Moreno. A continuación se muestra un listado de tablas con atributos que se definen en el cuadro 4.1. El listado completo de requisitos de sistema se pueden encontrar en el anexo A.1.

Cuadro 4.1: Tabla Atributo-Descripción para requisitos.

Atributo	**Descripción**
Identificador	Este es un código único que sirve para identificar o reconocer el requisito.Para los requisitos de usuarios se utilizará el formato RUXXXX y para los de software SXXXX.
Nombre	Nombre en lenguaje normal del requisito.
Descripción	Descripción del requisito. Qué aspectos involucra, en qué consiste, etc.
Prioridad	Prioridad asociada al requisito, esta puede ser crítica, deseable o innecesaria. Un requisito es crítico si afecta una operación crítica del negocio. Si existe algún proceso que se quiera incluir para mejorar los procesos actuales, estamos ante un requisito deseable y si se trata de un requisito informativo o que puede esperar para fases posteriores, el requisito es catalogado como innecesario.
Fuente	Documento o persona desde la cual surgió el requisito.
Estabilidad	Este campo tiene como propósito señalar si el requisito puede o no puede estar sujeto a cambio durante el ciclo de vida del software (tranzable o intranzable). El estándar de la ESA lo define como estable o no estable.
Estado	Estado actual del requisito dentro del desarrollo (Cumple, No Cumple, Ambiguo)
Listado de Usuarios	Son los tipos de usuarios que están asociados al requisito
Caso de Prueba	Caso con el cual se probará si se cumple o no con el requisito en el sistema.

Cuadro 4.2: Requisito de Usuario RU01

RU01	Crear cuenta para inicio de sesión	Prioridad: Alta
Los usuarios deben poder crear una cuenta mediante la aplicación móvil para poder hacer uso de todas las funcionalidades.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro 4.3: Requisito de Usuario RU02

RU02	Crear reporte de objeto perdido o encontrado	Prioridad: Alta
La aplicación debe permitir crear un reporte de un objeto perdido o de un objeto encontrado, visualizarlo en el mapa y en el historial de reportes para poder eliminarlo si se desea.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro 4.4: Requisito de Usuario RU03

RU03	Crear rutas de reporte	Prioridad: Alta
La aplicación debe permitir sugerir una ruta para el reporte de un objeto perdido o crear una manualmente. Para un reporte de un objeto encontrado, la ruta se trasforma en un punto.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro 4.5: Requisito de Usuario RU04

RU04	Crear puntos con radio	Prioridad: Alta
La aplicación debe permitir crear puntos con radio para un reporte de objeto perdido		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro 4.6: Requisito de Usuario RU05

RU05	Notificación de reporte	Prioridad: Alta
La aplicación debe permitir notificar a un usuario cuando está cerca de un objeto perdido, permitiendo interactuar con el reporte para indicar que ha sido encontrado.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro 4.7: Requisito de Usuario RU06

RU06	Facilidad de uso	Prioridad: Baja
La aplicación debe ser de fácil y que además funcione en modo retrato.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Transable	Estado: Cumplido	CPXXX

4.3. Diseño

En esta sección se exponen el diseño lógico, físico y la interfaz de usuario que se definieron para implementar las funcionalidades del prototipo funcional del proyecto en desarrollo.

4.3.1. Diagrama de Clases

El diagrama de clases, como se muestra en la Figura 4.2, permite visualizar las relaciones entre las clases que involucran el sistema en desarrollo. En él se puede apreciar cada clase con sus atributos, métodos y visibilidad, así como también relaciones de herencia, composición y agregación[16].

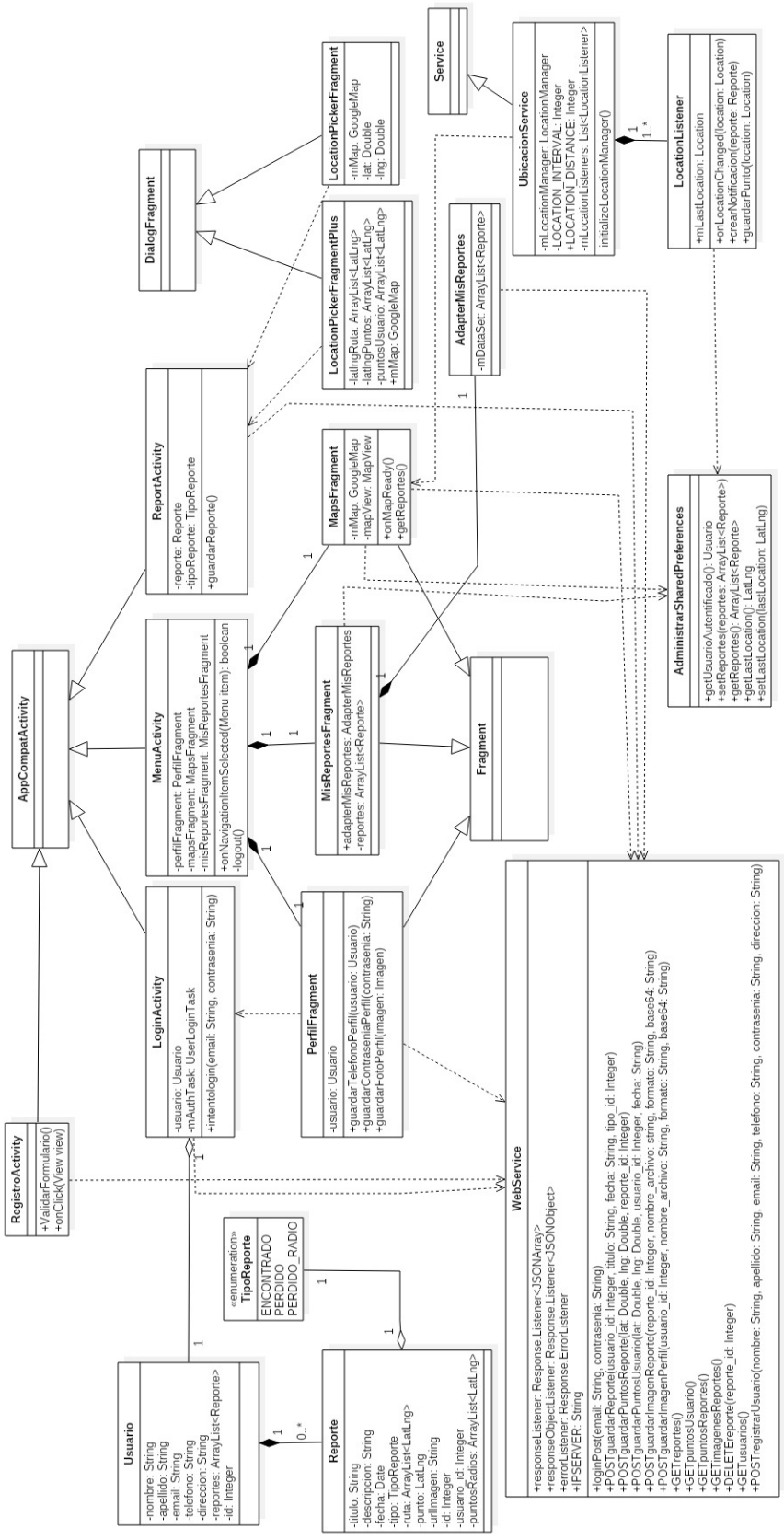

Figura 4.2: Diagrama de Clases.

4.3.2. Casos de Uso

La descripción de las actividades que se deben ejecutar para llevar a cabo un proceso es representado mediante casos de usos que se muestran en las figuras 4.3, 4.4, 4.5, 4.6 y 4.7 . Mediante estos se puede apreciar la interacción que se desarrolla entre la aplicación móvil en desarrollo y un usuario como un actor[16].

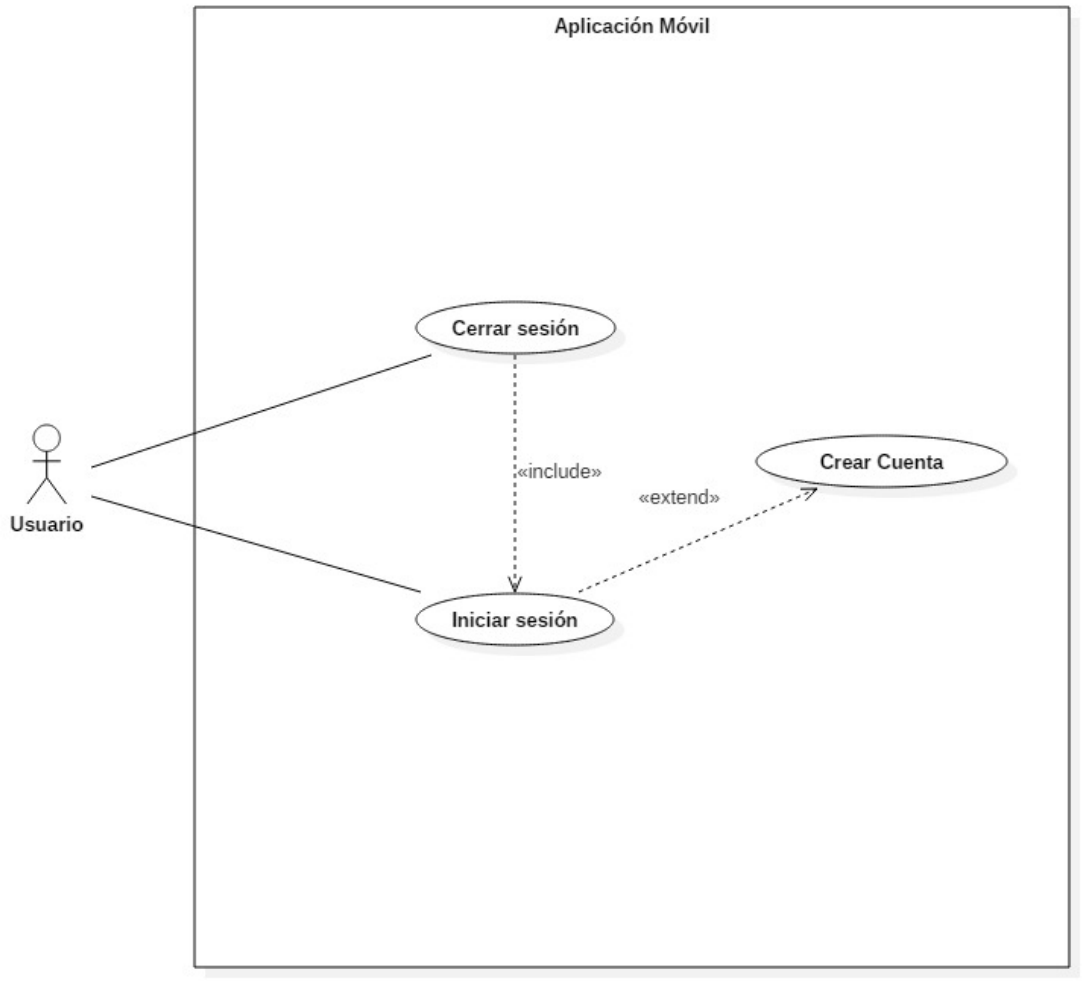

Figura 4.3: Caso de Uso: CU01, Inicio de Sesión.

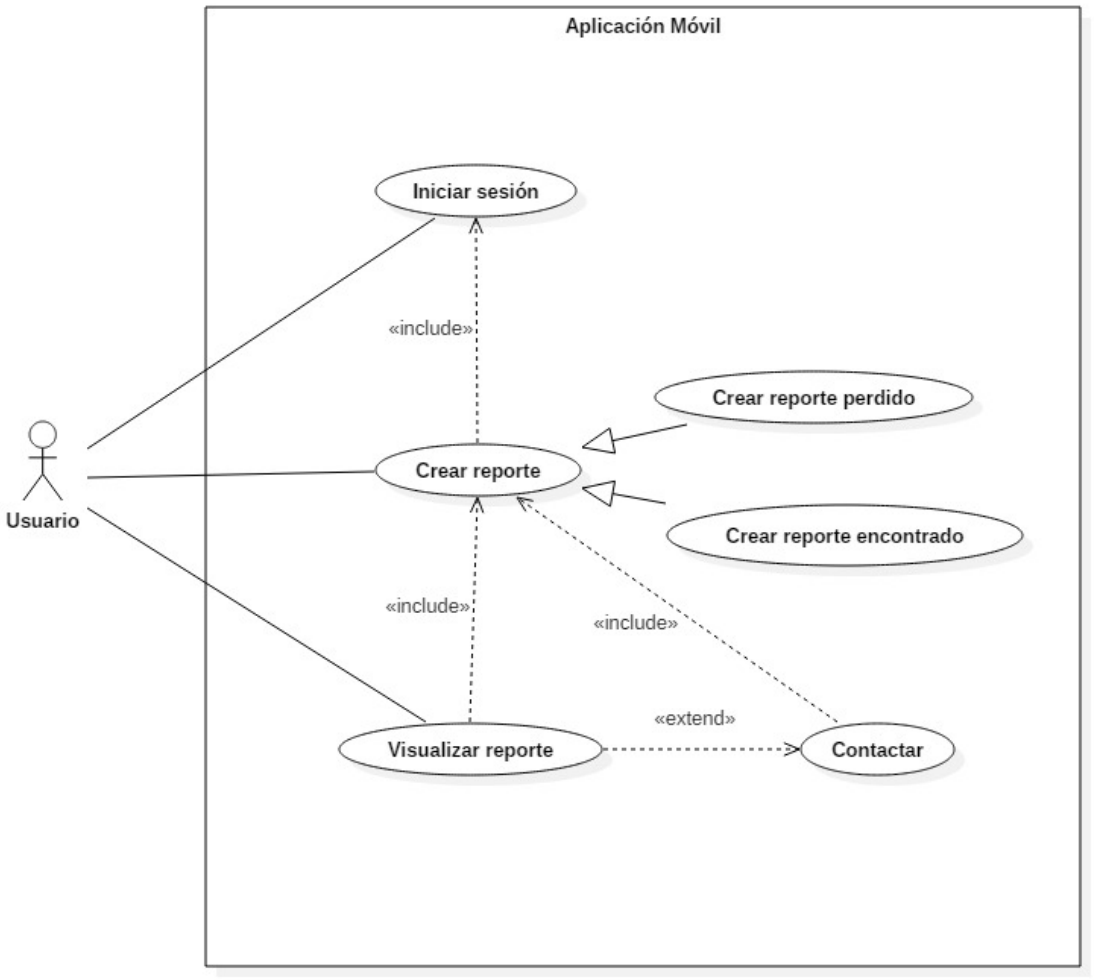

Figura 4.4: Caso de Uso: CU02, Crear reporte.

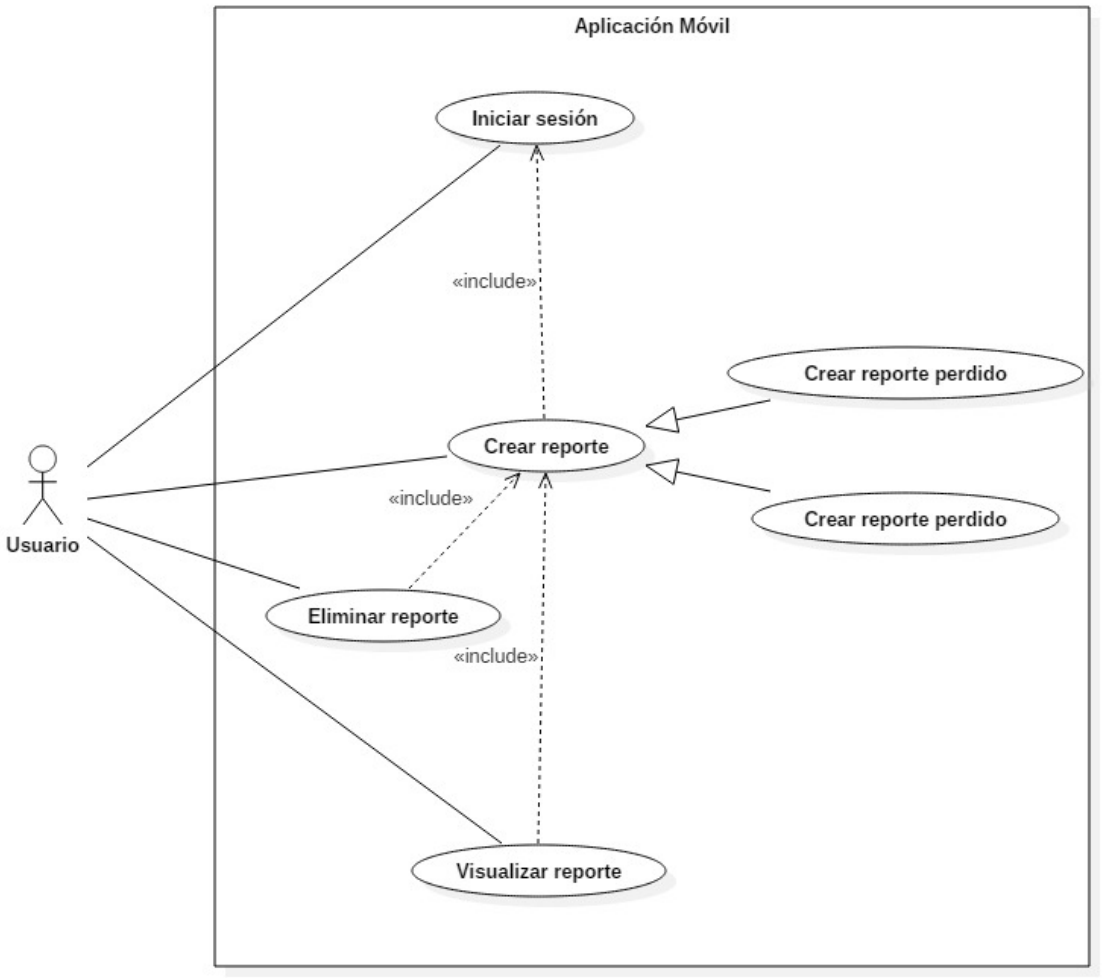

Figura 4.5: Caso de Uso: CU03, Eliminar reporte.

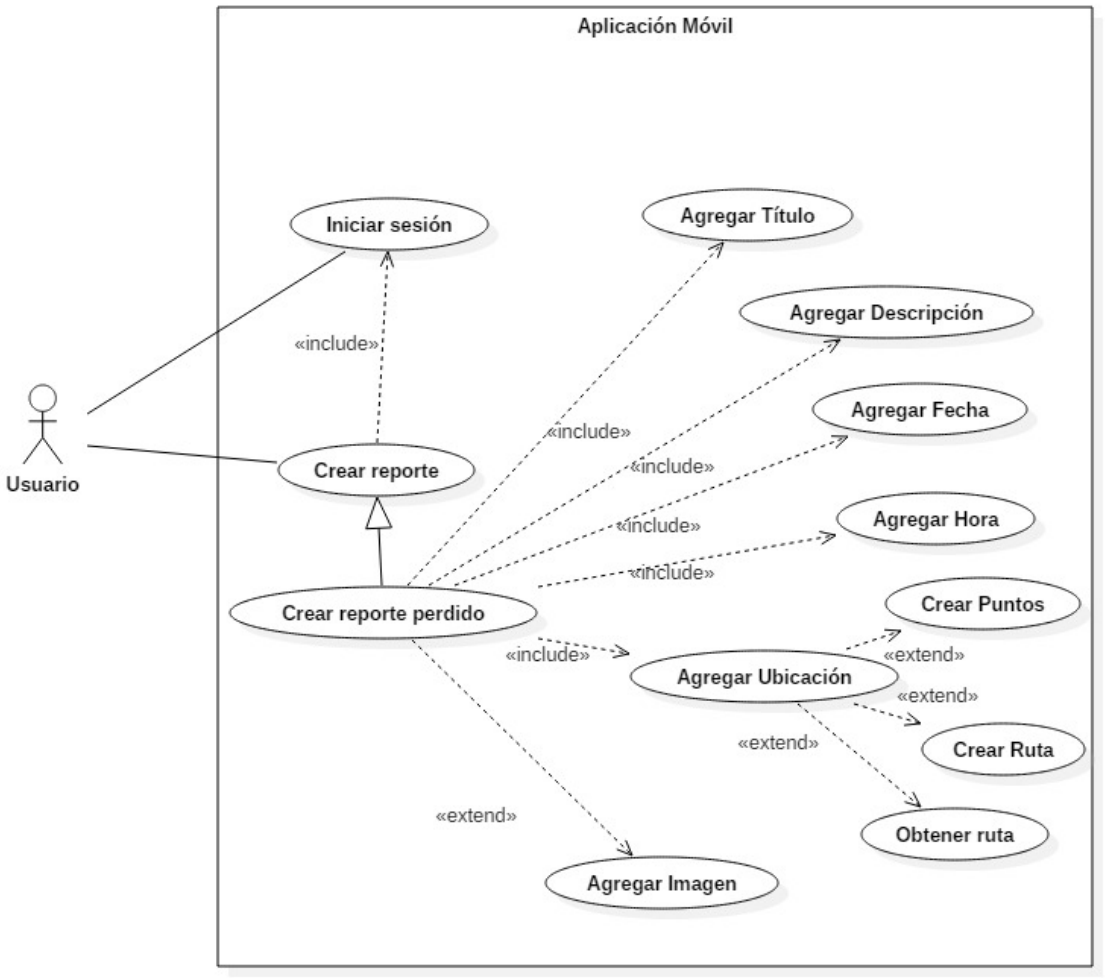

Figura 4.6: Caso de Uso: CU04, Crear reporte perdido.

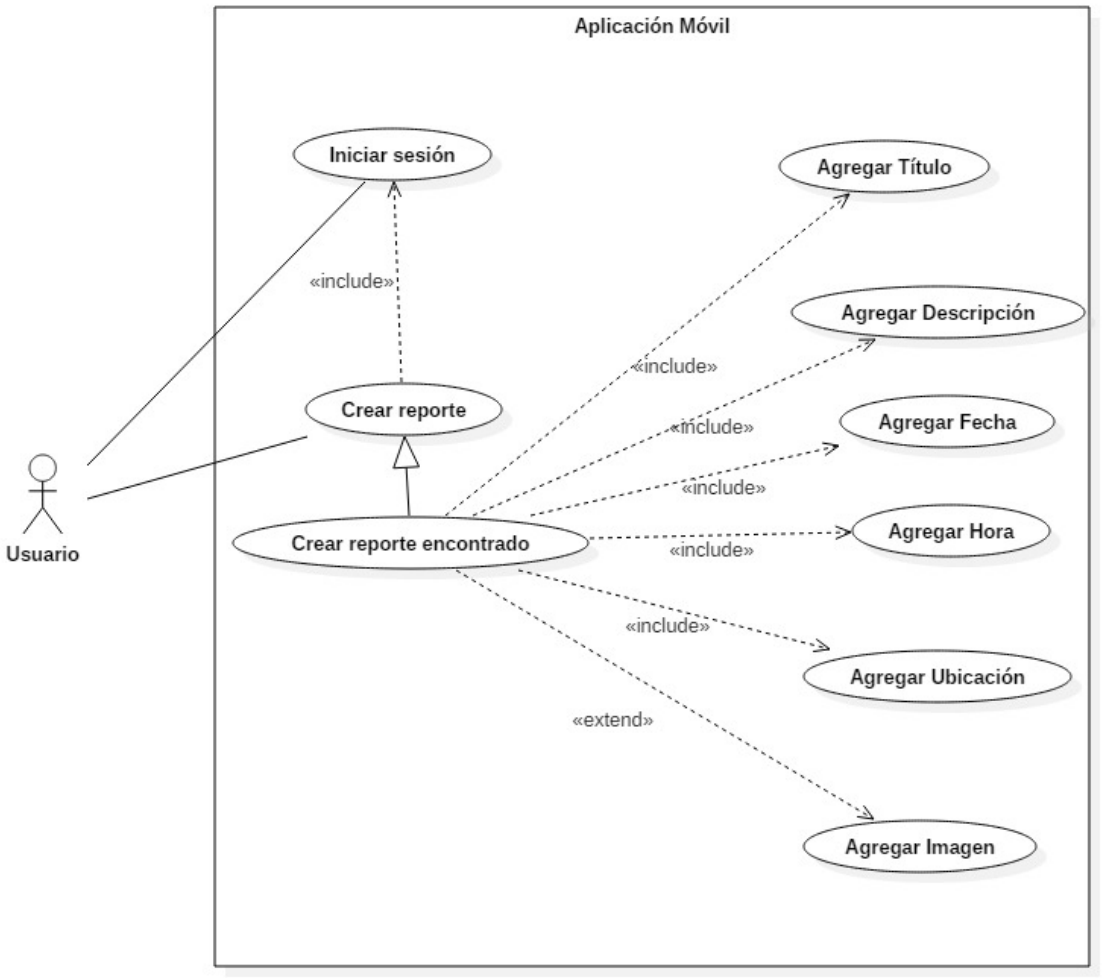

Figura 4.7: Caso de Uso: CU05, Crear reporte encontrado.

4.3.3. Diseño de la Base de Datos

En esta sección se presenta el diseño de la base de datos del sistema, para la implementación de este proyecto, mediante un modelo relacional, junto con una explicación de las entidades que la componen y cómo estas se relacionan entre si.

Modelo Relacional

En la figura 4.8 se puede observar el diagrama del modelo relacional, donde se puede apreciar cada una de las entidades juntos a sus atributos obtenidos del análisis

del diagrama de clases expuesto en la figura 4.2.

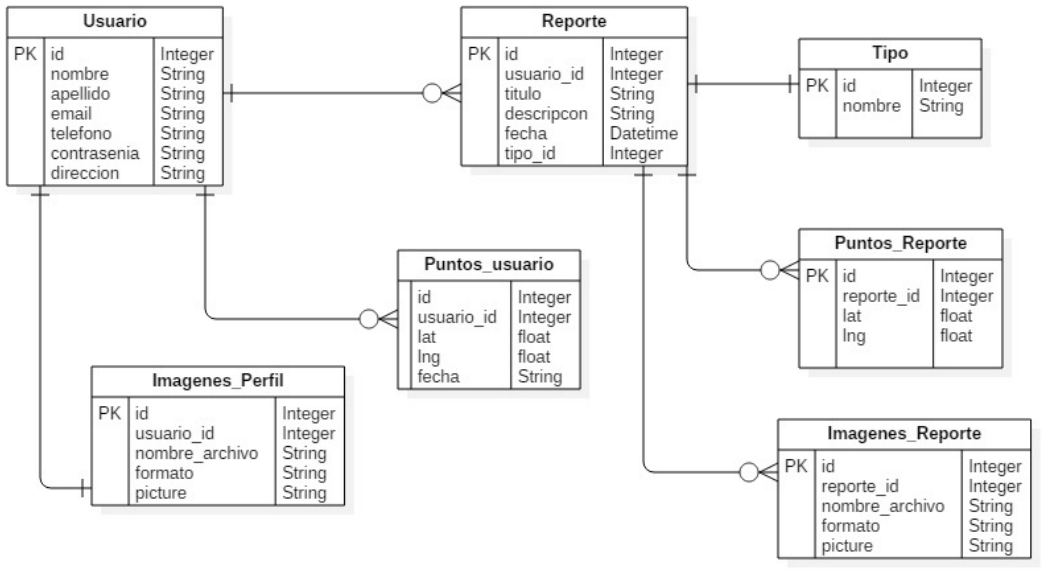

Figura 4.8: Modelo relacional.

Entidades

- **Usuario**: Entidad que representa a un usuario registrado y se encarga de gestionar el acceso a la aplicación móvil.
  - id: Identificador de usuario.
  - nombre: Nombre de un usuario.
  - apellido: Apellido de un usuario.
  - email: Email de un usuario.
  - telefono: Teléfono de un usuario.
  - contrasenia: Contraseña de un usuario.
  - direccion: Dirección de un usuario.
- **Imagenes_Perfil**: Entidad que representa la imagen de un Perfil.
  - id: Identificador de Imágenes Perfil.
  - usuario_id: Clave foránea del Usuario al que representa Imágenes Perfil.

- nombre_archivo: Nombre de Imágenes Perfil.
- formato: Formato de Imágenes Perfil.
- picture: Ruta del Servidor donde se almacena la imagen de Imágenes Perfil.

- **Reporte**: Entidad que representa a un reporte de objeto perdido o encontrado, esta información es generada por un usuario. Esta entidad se encarga de proveer la información necesaria de un reporte para ser visualizado por los demás usuarios.
  - id: Identificador de reporte.
  - usuario_id: Clave foránea del usuario al que pertenece el reporte.
  - titulo: Título de reporte.
  - descripcion: Descripción de reporte.
  - fecha: Fecha de reporte.
  - tipo_id: Clave foránea del tipo al que pertenece el reporte.
- **Tipo**: Entidad que representa a un tipo de reporte, los cuales pueden ser Perdido, Encontrado y Perdido Radio.
  - id: Identificador de tipo.
  - nombre: Nombre de tipo.
- **Puntos_usuario**: Entidad que representa la ubicación exacta de un usuario en un momento específico.
  - id: Identificador de Puntos Usuario.
  - usuario_id: Clave foránea del usuario al que representa Puntos Usuario.
  - lat: Latitud de la ubicación de Puntos Usuario.
  - lng: Longitud de la ubicación de Puntos Usuario.
  - fecha: Fecha y hora de Puntos Usuario.
- **Puntos_Reporte**: Entidad que representa la ubicación exacta de un Reporte.
  - id: Identificador de Puntos Reporte.

- reporte_id: Clave foránea del Reporte al que representa Puntos Reporte.
- lat: Latitud de la ubicación de Puntos Reporte.
- lng: Longitud de la ubicación de Puntos Reporte.

- **Imagenes_Reporte**: Entidad que representa la imagen de un Reporte.
  - id: Identificador de Imágenes Reporte.
  - reporte_id: Clave foránea del Reporte al que representa Imágenes Reporte.
  - nombre_archivo: Nombre de Imágenes Reporte.
  - formato: Formato de Imágenes Reporte.
  - picture: Ruta del Servidor donde se almacena la imagen de Imágenes Reporte.

4.3.4. Arquitectura física

La arquitectura física utilizada para este tipo de proyecto es Cliente-Servidor, ya que nos permite tener separada la ejecución de la aplicación y la datos en equipos distintos obteniendo varios ventajas, como se explica en el capítulo 2, sección 2.5.

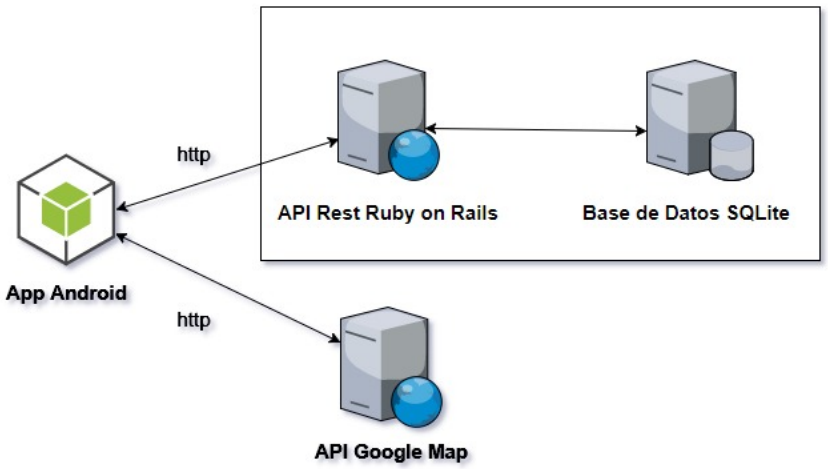

Figura 4.9: Arquitectura física Cliente-Servidor.

4.3.5. Diseño de Interfaz

En esta sección se exponen algunas imágenes de la interfaz de usuario que permiten a un usuario interactuar con la aplicación móvil y utilizar las funcionalidades. Es importante destacar que este diseño se ha ido mejorando durante el desarrollo del proyecto. A continuación se muestra la Figura 4.10 que corresponde a la visualización de reportes en el mapa y un botón que permite crear los dos tipos de reportes, perdido y encontrado. La Figura 4.11 muestra el diseño de la interfaz que se utiliza para ingresar título, descripción, fecha, hora, ubicación, e imagen de un reporte. El resto de los bosquejos se pueden encontrar en el anexo B.

Figura 4.10: Bosquejo de Visualización y creación de Reportes.

Figura 4.11: Bosquejo de crear reporte.

4.4. Resumen

En este capítulo se describió parte de la metodología y planificación utilizada para completar el problema de este proyecto. En primera instancia se explica la metodología a través de sprints de Scrum. Se abordaron además, los requisitos encontrados

y necesarios para el proyecto, en conjunto al diseño para componer el ciclo de vida de la aplicación móvil analizando los requisitos anteriormente presentados, para obtener un producto de calidad.

En el siguiente capítulo se expondrá el desarrollo o creación de la aplicación móvil paso a paso para completar el diseño metodológico presentado.

5. Desarrollo

En este capítulo se mostrarán los pasos necesarios para la construcción y funcionamiento de la aplicación móvil y servicios web para reportar objetos perdidos o encontrados. Para comenzar se analizarán los componentes, además de la instalación y configuración de Android Studio, para el desarrollo de la aplicación móvil y Ruby on Rails, para el desarrollo del servicio web. Adicionalmente se mostrará y explicará ejemplificadamente el desarrollo de estos.

5.1. Ambiente operacional

En el Cuadro 5.1 y 5.2 se exponen todas las herramientas utilizadas para llevar a cabo el desarrollo del proyecto.

Cuadro 5.1: Hardware utilizado

Hardware	Función
Computador Lenovo Z410, Windows 10	Utilizado para la codificación y como servidor para la base de datos
Celular Sony Xperia C4 E5353, Android 6.0	Utilizado para probar la aplicación móvil.
Celular Motorola XT1040, Android 5.1	Utilizado para probar la aplicación móvil.
Celular Lenovo K53b36, Android 7.0	Utilizado para probar la aplicación móvil.

Cuadro 5.2: Software utilizado

Software	Función
Android Studio 2.3	Utilizado para codificar el proyecto.
Postman	Utilizado para probar consultas al servidor web.
Ruby on Rails 3.3	Utilizado para desarrollar el servidor web.
Sublime Text 3	Utilizado para codificar la lógica del servidor web.
Ngrok stable	Utilizado para permitir el acceso desde cualquier red al servidor de la base de datos.

5.2. Instalación y configuración

En esta sección se detalla la instalación y configuración del software utilizado en el desarrollo del proyecto. Estas configuraciones apuntan a un entorno de desarrollo local, por lo tanto, pueden no ser aptas para un entorno de producción.

5.2.1. Android Studio

Android Studio es el IDE oficial para desarrollar aplicaciones Android, trae integrado distintas herramientas que permiten al desarrollador codificar, depurar, compilar y probar de una forma muy cómoda y rápida. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Mediante este entorno se pueden crear aplicaciones escritas en el lenguaje de programación Java orientado a objetos.

Instalación

Para la instalación de Android Studio se debe descargar un instalador gráfico desde `https://developer.android.com/studio/index.html?hl=es-419`. Para este desarrollo, se utilizó la versión 2.3. Al utilizar un instalador gráfico, todo se resume en seguir las instrucciones de instalación, aceptar términos y servicios y esperar que el instalador descargue los paquetes necesarios.

Configuración

Una vez instalada la aplicación, esta se debe ejecutar para dar inicio a la creación de un nuevo proyecto. La creación de un nuevo proyecto, consiste en definir un nombre para la aplicación, definir el dominio de la compañia y seleccionar la carpeta de destino del proyecto. La siguiente configuración tiene relación con la plataforma y número de la versión API a la que el proyecto apunta.

Para el desarrollo del proyecto se seleccionó la plataforma de Teléfono y Tablet, mínima versión de API número 22 correspondiente a los sistemas Android 5.1 llamado Lollipop. La última configuración consiste en agregar opcionalmente una ventana para la aplicación. El proceso de configuración termina presionado en botón Finalizar.

5.2.2. Ruby on Rails

Ruby on Rails es un framework para aplicaciones web de código abierto, que utiliza el lenguaje de programación Ruby. Sigue el patrón de arquitectura Modelo Vista Controlador (MVC). Este framework se distribuye a través de *RubyGems*, que es el formato oficial de paquete y canal de distribución de bibliotecas y aplicaciones Ruby[11].

Instalación

Para la instalación del framework Ruby on Rails, se debe descargar un instalador gráfico desde `http://railsinstaller.org/en`. Para este desarrollo se utilizó la versión 2.3, para la plataforma Windows. Al utilizar un instalador gráfico, todo se resume en seguir las instrucciones de instalación, aceptar términos y servicios y esperar que el instalador descargue los paquetes necesarios.

Configuración

Una vez instalada la aplicación, esta se debe ejecutar a través de una consola dedicada para este framework. La creación de un nuevo servidor web, consiste en generar un proyecto nuevo en este framework, a través de la consola utilizando el siguiente comando `rails new nombre_proyecto`, el cual nos generará el directorio de archivos básicos del framework.

Una vez completada la creación del servidor web, este se encuentra en condiciones

de crear la lógica necesaria para el modelo de negocio del sistema en desarrollo. Para el desarrollo de este proyecto se seleccionaron *gemas*, que permiten agregar funcionalidades extras al entorno del Ruby on Rails.

5.2.3. Ngrok

Ngrok es un programa ejecutable que permite exponer un servidor local al acceso público desde Internet a través de túneles seguros. Existen versiones disponible para las plataformas Microsoft Windows, macOS y Linux.

Instalación

Ngrok no necesita ser instalado para poder hacer uso de él, sólo es necesario ejecutar el archivo ngrok.exe el cual abre un ventana de comandos configurada para el caso de Microsoft Windows. La descarga del programa se realiza de `https://ngrok.com/download`.

Configuración

Existen muchas opciones de uso con Ngrok que se pueden encontrar en su documentación disponible en `https://ngrok.com/docs`. Para nuestro proyecto, solo necesitamos crear un túnel hacia nuestro puerto 3000, que es donde el servidor web creado con Ruby on Rails está escuchando por ejecutar una operación.

Para exponer el puerto 3000 de nuestro servidor local sobre el protocolo http, se debe ejecutar el siguiente comando: `ngrok http 3000` Una vez ejecutado el comando, la consola muestra la nueva Url a la cual nos debemos conectar para acceder a nuestro servidor local. Un ejemplo se puede apreciar en la Figura 5.1.

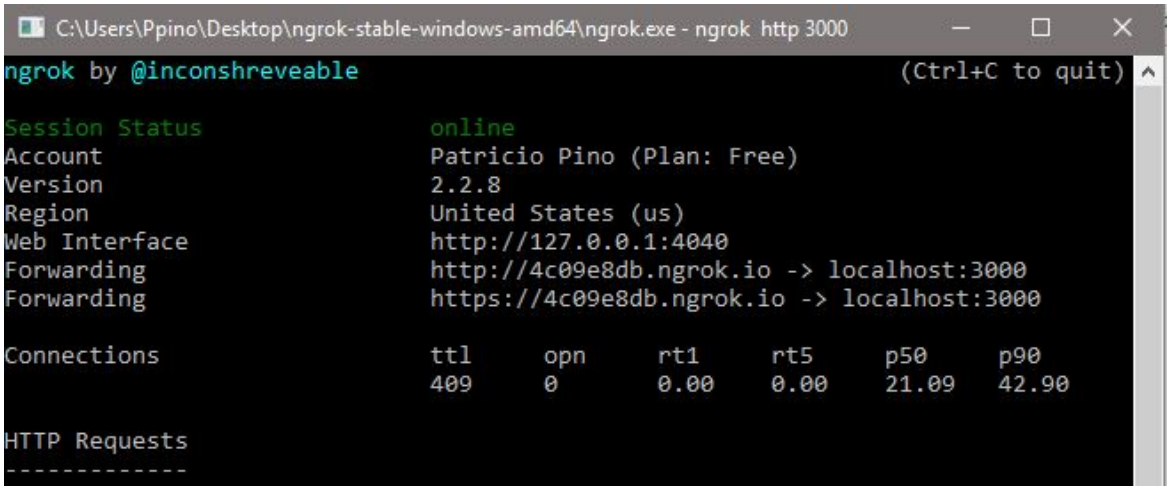

Figura 5.1: Túnel con Ngrok.

5.3. Implementación de la aplicación móvil

Aqui debo realizar la introducción de todo lo que explico en esta sección.

5.3.1. Integración con la API de Google Map

Como se ha mencionado en capítulos anteriores, la aplicación móvil hace uso de un mapa de Google para poder mostrar al usuario la ubicación de un reporte perdido o encontrado. Por lo tanto, es importante habilitar el uso del mapa en el proyecto. Para habilitar el uso del mapa es necesario realizar los siguientes pasos:

- **Añadir Google Play Services al proyecto**: En el directorio de archivos del proyecto, en el IDE de Android Studio, se debe expandir la sección de Gradle Scripts, después se debe abrir el archivo de configuración de dependencias llamado *build.gradle (Module:app)*. Posteriormente se debe agregar la siguiente regla de compilación dentro del espacio de dependencias:
  `compile 'com.google.android.gms:play-services-maps:11.0.4'`
  Se finaliza el proceso guardando los cambios en el archivo y presionando el botón de *Sync Now* para aplicar los cambios al proyecto. El archivo *build.gradle*

debería quedar como la Figura 5.2 que contiene además otras dependencias que no son necesarias para poder integrar la visualización de un mapa.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.3.1'
    compile 'com.android.support:design:25.3.1'
    compile 'com.android.support.constraint:constraint-layout:1.0.2'
    compile 'com.google.android.gms:play-services-maps:11.0.4'
    compile 'com.android.volley:volley:1.0.0'
    compile 'com.getbase:floatingactionbutton:1.10.1'
    compile 'com.kbeanie:multipicker:1.1.31@aar'
    compile 'com.google.maps.android:android-maps-utils:0.5'
    compile 'com.google.code.gson:gson:2.8.2'
    compile 'com.android.support:cardview-v7:25.3.1'
    compile 'com.android.support:recyclerview-v7:25.3.1'
    compile 'com.squareup.picasso:picasso:2.5.2'
    testCompile 'junit:junit:4.12'
}
```

Figura 5.2: Dependencias de compilación para el proyecto en Android Studio.

- **Obtener una clave de Google Maps API**: Para obtener una clave se debe acceder a la siguiente url `https://developers.google.com/maps/documentation/android-api/signup?hl=es-419#key` y presionar el botón *OBTENER UNA CLAVE*. En este paso, el sitio solicita que se inicie sesión con una cuenta de Google, para agregar y asociar el proyecto con la clave generada.

- **Agregar clave al proyecto**: En el directorio de archivos del proyecto, en el IDE de Android Studio, de debe expandir la sección de *res* →*values* y abrir el archivo *google maps api.xml* para agregar la clave recién generada. El archivo debe quedar como la Figura 5.3 con una clave del mismo estilo.

```
<resources>
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
        AIzaSyCIJv5NJ6AA4H_5M5JUWWYaZf-FDuUfIs8
    </string>
</resources>
```

Figura 5.3: Archivo xml con clave de Google Maps API.

Siguiendo todos los pasos anteriormente descritos, el proyecto se encuentra en condiciones para poder visualizar un mapa en una Actividad previamente creado. Al ejecutar la aplicación el mapa se debería ver como la Figura 5.4.

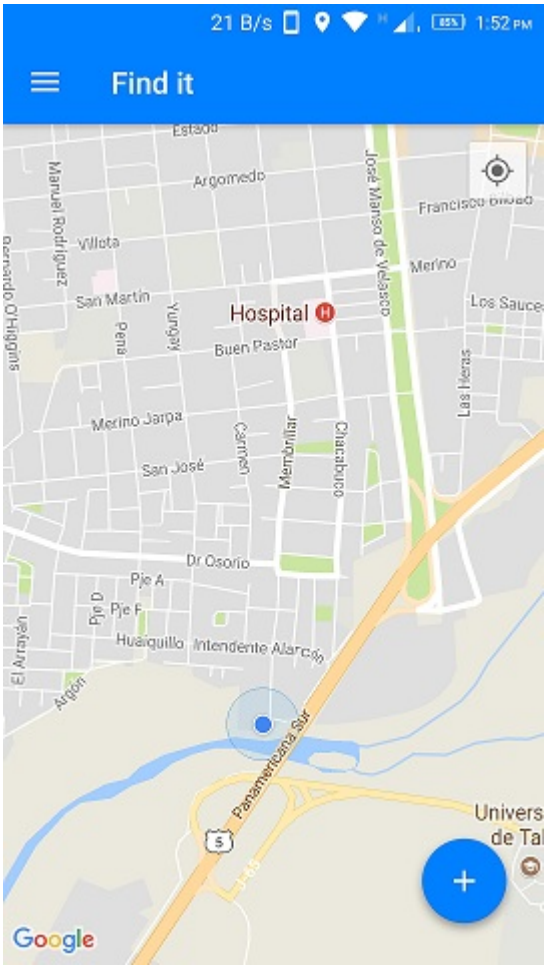

Figura 5.4: Visualización del mapa de Google en un proyecto Android.

5.3.2. Agregar un marcador al Mapa

Un marcador en el proyecto representa una ubicación en el mapa, para el caso del proyecto, este se utiliza para indicar el lugar donde un objeto ha sido encontrado. Por lo tanto, esta lógica de sebe aplicar para cada uno de los reportes de tipo encontrado.

Para agregar un elemento a un mapa, se debe crear una instancia de la clase *LatLng* que representa el par de coordenadas latitud y longitud en grados. Esta clase se puede utilizar realizando la importación que se muestra en Listing 5.1 en la definición de la clase donde se desea utilizar.

Listing 5.1: Importación de la clase LatLng.

```
import com.google.android.gms.maps.model.LatLng;
```

Una vez se tiene la instancia de la clase *LatLng* creada, se debe atribuir estas coordenadas a un elemento gráfico para que pueda ser visible en el mapa. Para nuestro caso utilizamos un marcador, que es representado por la clase *MarkerOptions* que está disponible realizando la siguiente importación que se muestra en el Listing 5.2.

Listing 5.2: Importación de la clase MarkerOptions.

```
import com.google.android.gms.maps.model.MarkerOptions;
```

El código que permite agregar un elemento gráfico al atributo mapa es el que se muestra en el Listing 5.3.

Listing 5.3: Código para agregar un marcador a un mapa.

```
LatLng latLng = new LatLng(-34.994791, -71.236007);
MarkerOptions markerOptions = new MarkerOptions();
markerOptions.position(latLng);
mMap.addMarker(markerOptions);
```

El resultado de agregar un marcador a un mapa se puede apreciar en la Figura 5.5.

Figura 5.5: Marcador creado mediante una coordenada de latitud y longitud.

5.3.3. Agregar una polilínea al Mapa

Una polilínea es utilizada para representar una ruta en el mapa. En el contexto de los reportes de tipo perdido, una ruta indica que un usuario ha recorrido tal ruta y cree haber perdido un objeto. La clase utilizada para representar una polilínea es *PolylineOptions* y permite su uso realizando la importación que se muestra en el Listing 5.4.

Listing 5.4: Importación de la clase PolylineOptions.

```
import com.google.android.gms.maps.model.PolylineOptions;
```

Una instancia de la clase *PolylineOptions* necesita que se le agreguen instancias de la clase *LatLng* para así ir formando una línea siguiendo el orden de agregación. Una vez se agregan todos las instancias de *LatLng*, el objeto *PolylineOptions* se agrega al objeto mapa mediante el método *addPolyline*. El código se puede apreciar en el Listing 5.5.

Listing 5.5: Código para agregar una polilínea a un mapa.

```
PolylineOptions rectOptions = new PolylineOptions();
for(int i = 0; i<this.markersRutaArrayList.size(); i++){
   rectOptions.add(new
      LatLng(this.markersRutaArrayList.get(i).getPosition().latitude,
```

```
        this.markersRutaArrayList.get(i).getPosition().longitude));
}
mMap.addPolyline(rectOptions);
```

El resultado de agregar una polilínea a un mapa se puede apreciar en la Figura 5.6.

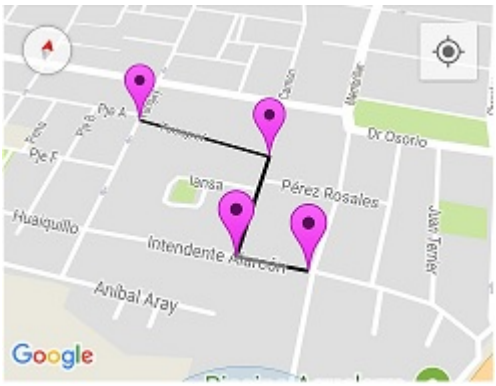

Figura 5.6: Polilínea creada mediante coordenadas de latitud y longitud.

5.3.4. Agregar un círculo al Mapa

Un circulo es utilizado para representar una área posible donde un objeto podría estar perdido. La clase que permite representar un círculo es *CircleOptions*, que se encuentra disponible para su uso mediante la importación que se muestra en el Listing 5.6.

Listing 5.6: Importación de la clase CircleOptions.

```
import com.google.android.gms.maps.model.CircleOptions;
```

Para agregar un objeto *CircleOptions* a un mapa es necesario indicarle al círculo cual es el centro mediante un objeto *LatLng* y el largo de su radio en metros. El código se puede apreciar en el Listing 5.7.

Listing 5.7: Código para agregar un círculo a un mapa.

```
LatLng latLng = new LatLng(-34.994791, -71.236007);
CircleOptions circleOptions = new CircleOptions();
```

```
circleOptions.center(latLng);
circleOptions.radius(75);
mMap.addCircle(circleOptions);
```

El resultado de agregar un círculo a un mapa se puede observar en la Figura 5.7.

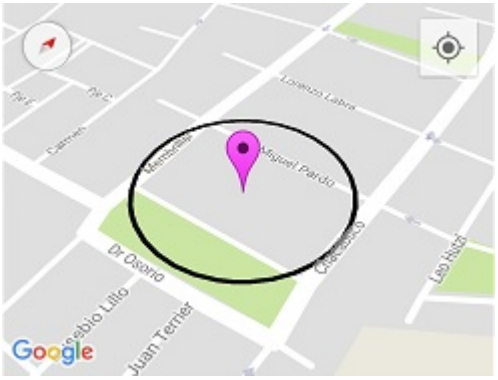

Figura 5.7: Círculo creado mediante una coordenada de latitud, longitud y radio de 75 metros.

5.3.5. Servicio y solicitud de ubicación

Servicio

Como se ha mencionado anteriormente, uno de los requisitos de la aplicación móvil es sugerir un ruta al usuario cuando se está creando un reporte de un objeto perdido. Este requisito consiste en que la aplicación pueda guardar la ubicación del dispositivo móvil en todo momento, independiente si la aplicación se está ejecutando, está cerrada o está en segundo plano. Posteriormente, el usuario puede consultar y visualizar los lugares que ha recorrido ingresando una fecha y hora, de inicio y fin. La aplicación debería ser capaz de mostrar a usuario un ruta mediante una polilínea dibujada en el mapa cuando existen suficientes ubicaciones guardas, en caso contrario se muestra un mensaje que no existen datos suficientes y el usuario tiene la posibilidad de crear una ruta manual.

Para poder dar solución a este requisito, se decidió crear un *Servicio iniciado*.

Un servicio es un componente de una aplicación que puede realizar operaciones de larga ejecución en segundo plano y que no proporciona una interfaz de usuario. Una vez un componente de la aplicación ejecuta el servicio, este continuará ejecutándose en segundo plano aunque el usuario cambie a otra aplicación[5]. Existen dos tipos de servicios que dependen de su funcionamiento, estos se nombran y describen a continuación:

- **Servicio iniciado**: Una vez iniciado, este tipo de servicio puede ejecutarse de manera indefinida en segundo plano, incluso si se destruye el componente que lo inició. Por lo general este tipo de servicio no interactúa con el emisor y realizan una sola operación.
- **Servicio de enlace**: Un servicio de enlace ofrece una interfaz cliente-servidor permitiendo que los componentes de la aplicación interactúen enviando y solicitando datos. Cuando los componentes se desenlazan, el servicio se destruye.

La implementación de un *Servicio iniciado* se realizó mediante una subclase de la clase *Service* llamada *UbicacionService*, que implementa los métodos *onStartCommand()* que solicita que se inicie el servicio, *onCreate()* que realiza los procedimientos de configuración por única vez, *onDestroy()* que destruye el servicio liberando todos los recursos utilizados y *initializeLocationManager()* que inicia los componentes para solicitar las actualización de ubicación. Este último método se explica con mayor detalle mas adelante.
Para iniciar el servicio, se debe crear una instancia de la clase *Intent* y pasar el nombre de la clase del servicio como parámetro. Se continúa llamando al método *startService()* pasando como parámetro la instancia de la clase *Intent* recién creada. El código de esta operación encuentra en el Listing 5.8.

Listing 5.8: Código para crear e iniciar un Servicio.

```
Intent intentService = new Intent(this, UbicacionService.class);
startService(intentService);
```

Para detener la ejecución del servicio, se debe escribir el código del Listing 5.9.

Listing 5.9: Código para detener la ejecución de un Servicio.

```
Intent intent = new Intent();
```

```
intent.setClass(getApplicationContext(), UbicacionService.class);
stopService(intent);
```

Solicitud de ubicación

Para completar con el desarrollo del requisito descrito anteriormente, es necesario que la aplicación guarde constantemente la ubicación del dispositivo. Para esto, la clase *UbicacionService* posee como atributo un objeto llamado *mLocationManager* que es de tipo *LocationManager*. Esta clase posee un método que permite realizar solicitudes de actualización de ubicación y según ciertas condiciones, si estas se cumplen, la clase retorna nuevos valores de ubicación. Para poder realizar la solicitud, primero debemos inicializar el objeto *mLocationManager* como se muestra en el Listing 5.10.

Listing 5.10: Método para inicializar una instancia de *LocationManager*.

```
private void initializeLocationManager() {
   if (mLocationManager == null) {
      mLocationManager = (LocationManager)
          getApplicationContext().getSystemService(Context.LOCATION_SERVICE);
   }
}
```

Este método se ejecuta cuando se crea el Servicio por primera vez llamando al método *onCreate()*. La siguiente operación es realizar la configuración para solicitar actualizaciones de ubicación, el código se muestra en el Listing 5.11.

Listing 5.11: Método onCreate() de la clase *UbicacionService*.

```
// Intervalo de actualización en milisegundos para solicitar una
    actualización
private static final int LOCATION_INTERVAL = 1000 * 10;
// Mínima distancia en metros para recibir una actualización
private static final float LOCATION_DISTANCE = 10;

@Override
public void onCreate(){
   initializeLocationManager();
```

```
    try {
        mLocationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
            LOCATIONINTERVAL, LOCATIONDISTANCE, mLocationListeners[0]);
    } catch (java.lang.SecurityException ex) {
        Log.i(TAG, "fail to request location update, ignore", ex);
    } catch (IllegalArgumentException ex) {
        Log.d(TAG, "network provider does not exist, " + ex.getMessage());
    }
}
```

El método *requestLocationUpdates()* recibe 4 parámetros. El primer parámetro *LocationManager.GPS_PROVIDER* indica el proveedor utilizado para calcular la ubicación, en nuestro caso utilizamos el GPS del dispositivo, el segundo parámetro *LOCATION_INTERVAL* define el tiempo que transcurre entre cada solicitud de actualización, para nuestro caso utilizamos un valor de 10 segundos, el tercer parámetro *LOCATION_DISTANCE* define el mínimo cambio de distancia para recibir una actualización y el último parámetro define la clase que estará escuchando cuando *LocationManager* retorne una nueva actualización[12].

Los parámetros *LOCATION_INTERVAL* y *LOCATION_DISTANCE* funcionan como una condición de tipo *AND*, esto quiere decir que si nos movemos 10 metros y transcurren al menos 10 segundos, se llamará al método *onLocationChanged()* implementado en la clase que se pasó como cuarto parámetro. En el método *onLocationChanged()* se debe implementar la lógica que deseamos para guardar la nueva ubicación en la base de datos y revisar los reportes a nuestro alrededor para crear una notificación[12].

5.3.6. Distancia a un reporte

Uno de los requisitos de la aplicación, es notificar al usuario cuando está cerca de un reporte. Para solucionar este problema, es necesario conocer la ubicación del dispositivo y calcular la distancia a un reporte. Para el caso de los reportes de objetos perdidos con el tipo de ubicación puntos con radio, la notificación se debe crear solo cuando la distancia del dispositivo sea menor o igual a 75 metros, que es la distancia del radio del círculo para este tipo de reportes. El Listing 5.12 muestra el código para calcular la distancia.

Listing 5.12: Fragmento de código para calcular la distancia entre dos puntos

```
Location locationReporte = new Location("locationPunto");
locationReporte.setLatitude(latLng.latitude);
locationReporte.setLongitude(latLng.longitude);
float distancia = locationDispositivo.distanceTo(locationReporte);
if(distancia <= 75){
   crearNotificacion(reporte);
}
```

El fragmento de código anterior muestra la variable de tipo *Location*, *locationDispositivo*, esta variable es creada cuando el método *onLocationChanged()* es ejecutado en la clase *LocationListener*. Para calcular la distancia entre dos puntos se utiliza el método *ditanceTo()* de la clase *Location*, este retorna la distancia entre los puntos en metros.

Para el caso de los reportes de objetos perdidos con el tipo de ubicación ruta, se debe crear la notificación cuando la distancia del dispositivo está cerca o en la polilínea que representa la ruta. Para poder calcular dicha distancia se utilizó una biblioteca de utilidades de la Google Maps Android API la cual contiene la clase *PolyUtil* y esta a su vez un método llamado *isLocationOnPath()* que retorna un objeto tipo *Boolean* que indica si la ubicación está en la polilínea o no[1]. Para utilizar esta biblioteca se debe agregar la siguiente dependencia en el archivo de compilación *build.gradle (Module: app)*.

```
dependencies {
   compile 'com.google.maps.android:android-maps-utils:0.5'
}
```

El fragmento de código para determinar si un punto está cerca o en un ruta se muestra en el Listing 5.13.

Listing 5.13: Fragmento de código para determinar si un punto está cerca o en la ruta.

```
LatLng latlngDispositivo = new LatLng(location.getLatitude(),
    location.getLongitude());
ArrayList<LatLng> ruta = reporte.getRuta();
double tolerancia = 2;
```

```
Boolean isLocationOnPath = PolyUtil.isLocationOnPath(latlngDispositivo,
    ruta, false, tolerancia);
if(isLocationOnPath){
   crearNotificacion(reporte);
}
```

El fragmento de código anterior, muestra la función *isLocationOnPath()* que recibe como primer parámetro el punto del dispositivo y como segundo parámetro una lista de puntos que representa la polilínea. El último parámetro es la tolerancia en metros. Así, la función puede calcular si el punto dado, se encuentra sobre o cerca de una polilínea[3]. Para conocer el método *crearNotificacion()* que se menciona en esta sección, revisar el anexo C, sección C.1.

5.3.7. Envío y Recepción de datos

La arquitectura física definida en el sistema es Cliente-Servidor, el envío de datos entre el dispositivo móvil, actuando como cliente y el Servicio Web, como servidor, es mediante el protocolo *HTTP*. Por lo tanto, para poder solicitar y enviar datos al Servicio Web, se utilizó la librería Volley.

Volley es una librería *HTTP* que hace que las operaciones de redes para las aplicaciones de Android sean más fáciles y, lo que es más importante, más rápidas[14]. Para utilizar esta biblioteca se debe agregar la siguiente dependencia en el archivo de compilación *build.gradle (Module: app)*.

```
dependencies {
   compile 'com.android.volley:volley:1.0.0'
}
```

El envío y recepción de datos en Volley está definida por los verbos HTTP: *POST*, *GET*, *PUT*, *PATCH* y *DELETE*, que corresponden a las operación de *create*, *read*, *update* y *delete* (CRUD) respectivamente[15].

Para realizar una solicitud de operación al Servicio Web usando Volley se definió un método distinto para cada recurso dependiendo si se desea obtener, enviar, actualizar o eliminar. En el Listing 5.14 se muestra el método que permite obtener todos los reportes existentes en la base de datos.

Listing 5.14: Método que obtiene todos los reportes de la base de datos.

```
public void getReportes(Context context) throws JSONException {
   RequestQueue requestQueue = Volley.newRequestQueue(context);
   String URL = "http://"+ IPSERVER +"/reportes";
   JSONObject object = new JSONObject();
   final String requestBody = object.toString();

   JsonArrayRequest jsonArrayRequest = new
       JsonArrayRequest(Request.Method.GET, URL, null, responseListener,
       errorListener) {
      @Override
      public String getBodyContentType() {
         return "application/json; charset=utf-8";
      }
      @Override
      public byte[] getBody() {
         try {
            return requestBody == null ? null :
                requestBody.getBytes("utf-8");
         } catch (UnsupportedEncodingException uee) {
            VolleyLog.wtf("Unsupported Encoding while trying to get the
                bytes of %s using %s", requestBody, "utf-8");
            return null;
         }
      }
};
requestQueue.add(jsonArrayRequest);
}
```

En el método anterior existen 7 elementos importantes que se deben considerar para poder realizar una solicitud. Estos se explican a continuación:

- **Crear una cola de solicitudes**: Esto se realiza al inicio del método creando una instancia de la clase *RequestQueue*.
- **Definir la URL**: Mediante una instancia de la clase *String* y asignando a este objeto la dirección del recurso. Como en este caso queremos obtener todos los

reportes, la URL termina con */reportes*.

- **Tipo de respuesta**: El resultado de obtener todos los reportes será una lista de objetos JSON, por lo tanto, se debe crear una instancia de la clase *JsonArrayRequest*. Si el resultado de la solicitud retorna un objeto JSON, se debe crear una instancia de la clase *JsonObjectRequest*.
- **Verbo HTTP**: Se debe pasar como parámetro el tipo de solicitud, en este caso necesitamos obtener todos los recursos sin realizar modificaciones, utilizamos el verbo *GET*.
- **Listener para la respuesta**: Se pasa como parámetro el listener llamado *responseListener* que es la clase que implementa el método que se ejecutará cuando el servidor responda.
- **Listener para el error**: Al igual que el parámetro anterior, se pasa el listener llamado *errorListener* que es la clase que implementa el método que se ejecutará cuando exista un error en la solicitud o en el Servicio Web.
- **Tipo de solicitud**: Se debe sobrescribir el método *getBodyContentType()* para que retorne el formato de datos que se está utilizando para la solicitud. En este caso, se está trabajando con objetos *JSON*. Existen muchos otros formatos disponibles según sea el caso como: *application/pdf* para trabajar con documentos en formato pdf o *image/jpeg* para imágenes en formatos jpeg[8].
- **Envío de datos**: Para poder enviar datos al servidor mediante un objeto *Json*, se crea una instancia de la clase *JSONObject* y se agregan propiedades indicando la llave y el valor como se muestra en el Listing 5.15.

Listing 5.15: Creación de una instancia de la clase *JSONObject*.

```
JSONObject jsonBody = new JSONObject();
jsonBody.put("lat", lat);
jsonBody.put("lng", lng);
jsonBody.put("usuario_id", usuario_id);
jsonBody.put("fecha", fecha);
```

5.3.8. Transformación de una imagen

En la subsección anterior se explicó como se envían y reciben datos usando la librería Volley, además se mencionó que el tipo de datos utilizado era *JSON*. Por lo tanto, para enviar una imagen como propiedad de un objeto *JSON* es necesario convertirla a una cadena de caracteres. Para realizar esta operación, existe la clase *Base64* que tiene un método llamado *encodeToString()* el cual recibe como parámetro una lista de *byte* y retorna un cadena de caracteres. El fragmento de código se puede observar el el Listing 5.16.

Listing 5.16: Fragmento de código para transformar una imagen en una cadena de caracteres.

```
BitmapFactory.Options options = new BitmapFactory.Options();
options.inJustDecodeBounds = false;
options.inPreferredConfig = Bitmap.Config.RGB_565;
Bitmap myBitmap = BitmapFactory.decodeFile(imagen.getOriginalPath(),
    options);
String format = imagen.getFileExtensionFromMimeTypeWithoutDot();
ByteArrayOutputStream baos = new ByteArrayOutputStream();
myBitmap.compress(Bitmap.CompressFormat.JPEG, 40, baos);
byte[] imageBytes = baos.toByteArray();
String imageBase64 =
    ""+Base64.encodeToString(imageBytes,
    Base64.DEFAULT);
```

El proceso del fragmento de código anterior, comienza por decodificar el archivo que representa la imagen a un mapa de bits, mediante el método *decodeFile()* de la clase *BitmapFactory*. Posteriormente, se debe comprimir el mapa de bits a un objeto de la instancia *ByteArrayOutputStream*. Para obtener la lista de *byte*, el objeto de la clase *ByteArrayOutputStream* posee un método que retorna la lista necesitada. Y por último, se pasa como parámetro el objeto *imageBytes* al método *encodeToString()* que retornará la cadena de caracteres. La variable *imageBase64* contiene además unos valores necesarios para que el Servicio Web entienda que está recibiendo una imagen codificada en base64 y puede realizar el proceso inverso para almacenarla. Cabe destacar que realizar esta codificación a base64 aumenta el tamaño final del archivo en aproximadamente un 30 %.

5.4. Implementación del Servicio Web

Como se ha mencionado anteriormente, la arquitectura física definida para este sistema es Cliente-Servidor, dado que los datos de la aplicación móvil deben ser persistentes y compartidos entre todos los usuarios. Lo anterior provoca que un servidor que almacene los datos, puede compartir de manera más efectiva la información entre los dispositivos, mediante respuesta a solicitudes de datos.

5.4.1. Configuración de un nuevo servidor web

Como mencionamos anteriormente en este documento, el servidor web estará creado con el framework *Ruby on Rails*.
Para la creación de un nuevo proyecto en Ruby on rails es necesario abrir la consola de Ruby on Rails para Windows, y generar el siguiente código.

```
$ rails new <Nombre Proyecto>
```

Este código generará una aplicación Rails con el nombre del proyecto, que contiene directorios e instala las gemas de dependencias del framework. Los directorios contienen una estructura organizada de una aplicación Rails.

5.4.2. Modelos de negocio

Siguiendo con el Modelo Relacional, explicado en la sección 4.4.3, del Capítulo 4, los datos necesarios para esta aplicación deben ser guardadas de acuerdo a una estructura definida en tablas que almacenen la información enviada por la aplicación móvil y la que esta misma recibirá del servidor. Para la creación de un Modelo en el framework Ruby on Rails utilizamos el estándar *scaffold*, que permite crear un conjunto del Modelo, la migración del modelo hacia la base de datos, el controlador y las vistas para la entrega de datos (que no se utilizarán en este proyecto).

Listing 5.17: Formato de código para la creación de un Scaffold de Rails

```
$ generate scaffold <Nombre del Modelo> <atributo>:<tipo atributo>
```

El formato de código anterior, nos permite crear automáticamente toda la estructura respecto al Modelo que se está creando, implementando las acciones básicas de un controlador. Por lo tanto se debe realizar esta instrucción para cada uno de los

modelos vistos en la sección 4.4.3, del Capítulo 4. Una vez realizado el proceso de creación de los Modelos, se debe generar las migraciones de la base de datos del servicio web. Estas pueden ser encontradas en la carpeta *app →db →migrate*, las migraciones son instrucciones *SQL* para la creación de tablas en la base de datos, en esta también debemos asegurarnos de agregar las referencias entre tablas a través de los índices. Una migración, es por ejemplo, la que se muestra en el Listing 5.18.

Listing 5.18: Ejemplo de código de la migración de crear la tabla Usuario en la base de datos.

```
class CreateUsuarios < ActiveRecord::Migration[5.0]
  def change
    create_table :usuarios do |t|
      t.string :nombre
      t.string :apellido
      t.string :email
      t.string :telefono
      t.string :contrasenia
      t.string :direccion

      t.timestamps
    end
  end
end
```

Las migraciones y sus funciones de creación se ejecutan a través del siguiente código.

Listing 5.19: Instrucción para ejecutar los archivos de migración en la base de datos.

```
rails db:migrate
```

5.4.3. Acciones del Controlador

Como vimos en la sección anterior, se utilizó el estándar *scaffold*, creándose el modelo básico del controlador con sus acciones, tales como *Index, New, Create, Show, Edit y Destroy*. Los controladores se encuentran en la carpeta *app →controllers*, un

ejemplo de controlador puede ser visto en el anexo C.2.

En este proyecto, se definió como respuesta a las acciones del controlador, el formato de intercambio de datos *JSON* para facilitar el envío y recepción en la aplicación móvil. Un ejemplo de esto se muestra en el Listing 5.20.

Listing 5.20: Fragmento de código de la acción crear del controlador Reporte

```
# POST /reportes
# POST /reportes.json
def create
  @reporte = Reporte.new(reporte_params)

  if @reporte.save
    render json: @reporte, status: :created
  else
    render json: @reporte.errors, status: :unprocessable_entity
  end
end
```

El fragmento de código anterior, nos muestra la acción de crear un Reporte, en esta obtenemos una respuesta del servidor, con el Reporte recién creado a la aplicación a través de JSON, en el caso de que el Reporte sea creado o no. Además de esto, el controlador envía un código de respuesta HTTP[7].

En el Listing 5.21 se observa la respuesta del servicio web en formato *JSON*, cuando se realiza la creación de un nuevo Reporte.

Listing 5.21: Respuesta JSON del servicio web en la creación de un Reporte.

```
{
  "id": 117,
  "usuario_id": 13,
  "titulo": "Perdí mis audífonos el 2013",
  "descripcion": "Los de la foto pero negros",
  "fecha": "2013-10-16T11:30:00.000Z",
  "tipo_id": 1,
}
```

5.5. Resumen

En este capítulo se ahondó en el desarrollo e implementación de la solución del proyecto. En primera instancia se exponen los hardwares y softwares utilizados en el sistema. Posteriormente se especificó el desarrollo de la aplicación móvil, a través de la herramienta de desarrollo Android Studio. La anterior se comunicará con el Servicio Web que fue desarrollada en base al framework Ruby on Rails, exponiendo la forma de abordar los datos almacenados y el formato de comunicación con la aplicación móvil.

En el siguiente capítulo se abordarán las pruebas realizadas del prototipo de la aplicación móvil, para validar su correcto funcionamiento.

6. Pruebas y Resultados

En este capítulo se exponen las pruebas para evaluar las funcionalidades de la aplicación móvil y la conexión con el servidor, mediante un listado de preguntas para usuarios y pruebas de caja negra. Además de lo anterior, se exponen los resultados de cada una de las pruebas mencionadas, terminando con una discusión de los resultados.

6.1. Pruebas

El aseguramiento de la calidad de software (ACS) es un conjunto de actividades que filtran los errores de un software con el objetivo de tratarlos y obtener un producto de calidad. El ACS incluye un rango amplio de preocupaciones y actividades que se centran en la administración de la calidad del software. Éstas actividades son: Estándares, Revisiones y Auditorias, Pruebas, Colección y análisis de los errores, Administración del cambio y Administración del riesgo, entre otros[17].

Para la aplicación móvil en desarrollo, se utilizaron dos pruebas: Cuestionario de preguntas a usuarios y pruebas de caja negra.

- El cuestionario de preguntas busca obtener la mayor cantidad de información respecto a la experiencia del usuario y calidad del diseño con la interfaz de la aplicación móvil. Las preguntas están compuestas de respuestas de selección única, para facilitar el análisis. Del total de los encuestados, se espera que el porcentaje de aprobación sea igual o mayor a 80 % para considerar el tema de la pregunta como aprobado, en caso contrario, reprobada y se debe tratar esa falencia.

  El cuestionario, busca medir en detalle 3 atributos de la aplicación móvil como

lo es la Usabilidad en función de la interfaz de usuario, donde se utilizan las preguntas 3, 4, 5 y 6 de la subsección 6.1.1. La funcionalidad de la aplicación se busca medir con las preguntas 7, 8, y 10. Por último, la utilidad de la aplicación con las preguntas 9 y 11.

- Las pruebas de caja negra, son pruebas que buscan encontrar la mayor cantidad de errores, antes de que el producto sea entregado a los usuarios. Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. Este tipo de prueba, intentan encontrar errores en las categorías siguientes: funciones incorrectas o faltantes, errores de interfaz, errores en las estructuras de datos o en el acceso a bases de datos externas, errores de comportamiento o rendimiento y errores de inicialización y terminación[17].

6.1.1. Cuestionario de preguntas de usuario

Para la experiencia de los usuarios con la aplicación móvil y calidad del diseño, se generó un cuestionario, que busca obtener información sobre la aplicación móvil en relación a qué tan fácil o intuitiva resulta realizar una operación, en conjunto a la información que se muestra. Estas preguntas fueron aplicadas a un grupo de usuarios avanzados y sin experiencia en el área informática. Las preguntas mencionadas, fueron las siguientes:

1. Ingrese su nombre y apellido (opcional).
2. Seleccione su nivel de habilidad con el uso de aplicaciones móviles.
   - Básico.
   - Intermedio.
   - Avanzado.
3. ¿La paleta de colores utilizados es agradable y no desconcentra, mientras se utiliza la aplicación?
   - Agradable.

- Desagradable.

4. ¿El tamaño del texto utilizado, permite una lectura cómoda?
   - Si.
   - No.
5. ¿Los verbos utilizados en la interfaz de usuario, representan las acciones que estos realizan? Por ejemplo: ¿el botón guardar, efectivamente guarda un recurso?
   - No representan las acciones que realizan.
   - Si representan las acciones que realizan.
   - Los verbos son ambiguos.
6. ¿Los íconos utilizados en la interfaz de usuario, representan las acciones que estos realizan? Por ejemplo: ¿el ícono del bote de basura, representa efectivamente la acción de eliminar un recurso?
   - No representan las acciones que realizan.
   - Si representan las acciones que realizan.
   - Los íconos son ambiguos.
7. Cree un reporte de un objeto perdido con una ubicación tipo ruta; mantenga presionado el mapa para agregar un punto de la ruta; luego presione el botón guardar. ¿Qué tan fácil le ha resultado crear el reporte? En la escala de 1 a 5, siendo 1 muy fácil y 5 muy difícil, elija una opción.
   - 1 - Muy fácil
   - 2 - Fácil
   - 3 - Intermedio
   - 4 - Difícil
   - 5 - Muy difícil

8. Ahora, actualice los reportes en el Mapa; busque su reporte recién creado; toque el marcador para ver su información. ¿Qué tan fácil le ha resultado realizar estas operaciones? En la escala de 1 a 5, siendo 1 muy fácil y 5 muy difícil, elija una opción.
    - 1 - Muy fácil
    - 2 - Fácil
    - 3 - Intermedio
    - 4 - Difícil
    - 5 - Muy difícil
9. ¿Las funcionalidades implementadas en la aplicación, son suficientes para aumentar las posibilidades de recuperar un objeto perdido?
    - Suficiente.
    - Insuficiente.
10. ¿La navegación entre las distintas funcionalidades, resulta fácil de recordar?
    - Si.
    - No.
11. ¿Qué tan relevante considera usted que es esta aplicación para la búsqueda de objetos perdidos?
    - Relevante.
    - Irrelevante.

6.1.2. Pruebas de Caja Negra

Cuadro 6.1: Caso de prueba 1

Caso de uso	CU01
Caso de Prueba	CP01
Entrada	Iniciar sesión con un correo electrónico inválido.
Salida	La aplicación notifica mediante un mensaje que las credenciales son incorrectas.
Condiciones	El usuario debe ingresar valores en los campos de inicio de sesión.

Cuadro 6.2: Caso de prueba 2

Caso de uso	CU01
Caso de Prueba	CP02
Entrada	Iniciar sesión con una contraseña inválido.
Salida	La aplicación notifica mediante un mensaje que las credenciales son incorrectas.
Condiciones	El usuario debe ingresar valores en los campos de inicio de sesión.

Cuadro 6.3: Caso de prueba 3

Caso de uso	CU01
Caso de Prueba	CP03
Entrada	Iniciar sesión con un correo electrónico y contraseña válidos.
Salida	La aplicación inicia mostrando la pantalla principal compuesta por un mapa.
Condiciones	El usuario debe ingresar valores en los campos de inicio de sesión.

Cuadro 6.4: Caso de prueba 4

Caso de uso	CU02
Caso de Prueba	CP04
Entrada	Crear un reporte de un objeto perdido o encontrado, sin ingresar una ruta o un punto con radio.
Salida	La aplicación notifica que faltan campos y evita la creación del reporte.
Condiciones	Seleccionar creación de un nuevo reporte.

Cuadro 6.5: Caso de prueba 5

Caso de uso	CU02
Caso de Prueba	CP05
Entrada	Crear un reporte de un objeto perdido o encontrado, ingresando todos los valores necesarios.
Salida	La notificación muestra el reporte creado en el mapa.
Condiciones	Seleccionar creación de un nuevo reporte.

Cuadro 6.6: Caso de prueba 6

Caso de uso	CU02
Caso de Prueba	CP06
Entrada	Tocar un reporte en el mapa.
Salida	La aplicación muestra el detalle del reporte seleccionado.
Condiciones	Tocar el marcador visible en el mapa.

Cuadro 6.7: Caso de prueba 7

Caso de uso	CU04, CU05
Caso de Prueba	CP07
Entrada	Archivo distinto a una imagen.
Salida	La aplicación notifica el error y evita guardar los cambios.
Condiciones	El usuario debe seleccionar un archivo distinto a una imagen.

Cuadro 6.8: Caso de prueba 8

Caso de uso	CU04, CU05
Caso de Prueba	CP08
Entrada	Agregar un archivo de imagen a un reporte.
Salida	La aplicación guarda los cambios y muestra la imagen en el reporte.
Condiciones	El usuario debe seleccionar una imagen.

Cuadro 6.9: Caso de prueba 9

Caso de uso	CU01
Caso de Prueba	CP09
Entrada	Ingresar valores válidos en los campos de registro de usuario.
Salida	La aplicación muestra un mensaje de registro exitoso y redirecciona al usuario a la pantalla de inicio de sesión.
Condiciones	El usuario debe ingresar todos los campos del registro de usuario.

Cuadro 6.10: Caso de prueba 10

Caso de uso	CU01
Caso de Prueba	CP10
Entrada	Ingresar valores válidos en los campos de registro de usuario y un correo ya registrado.
Salida	La aplicación muestra un mensaje de advirtiendo que ya existe una cuenta creada con el correo ingresado.
Condiciones	El usuario debe ingresar todos los campos del registro.

Cuadro 6.11: Caso de prueba 11

Caso de uso	CU01
Caso de Prueba	CP11
Entrada	Ingresar valores en los campos excepto en uno.
Salida	La aplicación muestra un mensaje, indicando que faltan campos para el registro de usuario.
Condiciones	El usuario debe dejar al menos un campo sin datos.

Cuadro 6.12: Caso de prueba 12

Caso de uso	CU03
Caso de Prueba	CP12
Entrada	Eliminar un reporte.
Salida	La aplicación elimina el reporte y desaparecen el o los marcadores del mapa.
Condiciones	El usuario debe seleccionar la opción eliminar del reporte.

Cuadro 6.13: Caso de prueba 13

Caso de uso	CU04
Caso de Prueba	CP13
Entrada	El usuario debe eliminar un punto cuando se está creando un reporte.
Salida	La aplicación elimina el punto y desaparece del mapa.
Condiciones	El usuario debe seleccionar la opción eliminar un punto, tocando el marcador que desea eliminar.

Cuadro 6.14: Caso de prueba 14

Caso de uso	CU02
Caso de Prueba	CP14
Entrada	El usuario ve el detalle de un reporte y presiona el número de teléfono para contactar al usuario que creó el reporte.
Salida	La aplicación realiza una llamada telefónica con el número seleccionado .
Condiciones	El usuario debe seleccionar un marcador en el mapa y deslizar el panel hacía arriba para ver el detalle del reporte y tocar el número telefónico.

Cuadro 6.15: Caso de prueba 15

Caso de uso	CU04
Caso de Prueba	CP15
Entrada	El usuario recorre una ruta.
Salida	La aplicación debe sugerir según filtro de fecha y hora de inicio y fin la ruta recién recorrida.
Condiciones	El usuario debe recorrer una ruta con el dispositivo encendido y el GPS activado.

Cuadro 6.16: Caso de prueba 16

Caso de uso	CU02
Caso de Prueba	CP16
Entrada	El usuario se acerca en dirección a un reporte de tipo perdido o encontrado.
Salida	La aplicación debe crear una notificación en el dispositivo del usuario
Condiciones	El usuario acercarse al reporte con el dispositivo encendido y el GPS activado.

6.2. Resultados

En esta sección se muestran todos los resultados obtenidos de acuerdo a las diferentes formas de probar la aplicación móvil y el funcionamiento en la conexión con el servidor web.

6.2.1. Resultados de Experiencias de Usuario

En esta sección se muestra en forma de gráficos las respuestas obtenidas al aplicar el cuestionario a 28 usuarios que interactuaron con la aplicación.

- La Figura 6.1 muestra el resultado de las respuestas de la pregunta 2 expuesta en la sección 6.1.1.

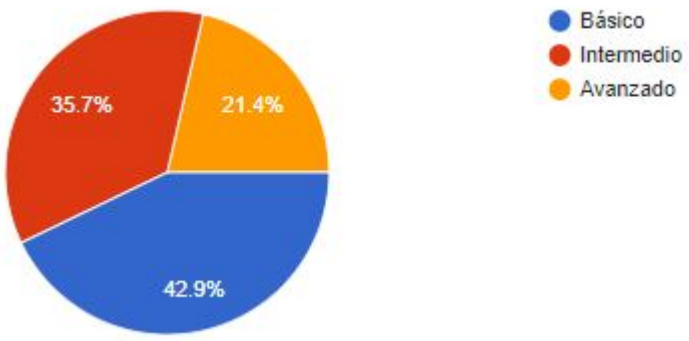

Figura 6.1: Gráfico con porcentajes de respuestas de la pregunta 2.

- La Figura 6.2 muestra el resultado de las respuestas de la pregunta 3 expuesta en la sección 6.1.1.

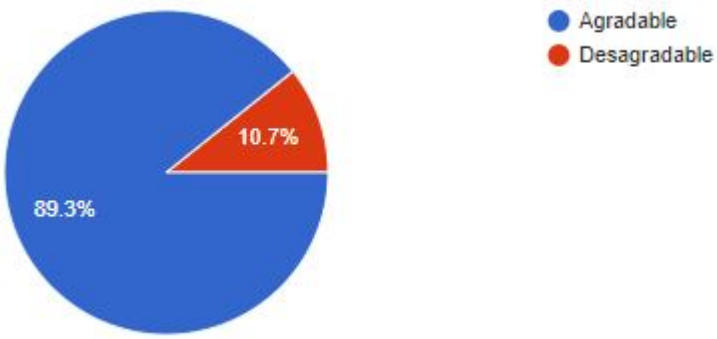

Figura 6.2: Gráfico con porcentajes de respuestas de la pregunta 3.

- La Figura 6.3 muestra el resultado de las respuestas de la pregunta 4 expuesta en la sección 6.1.1.

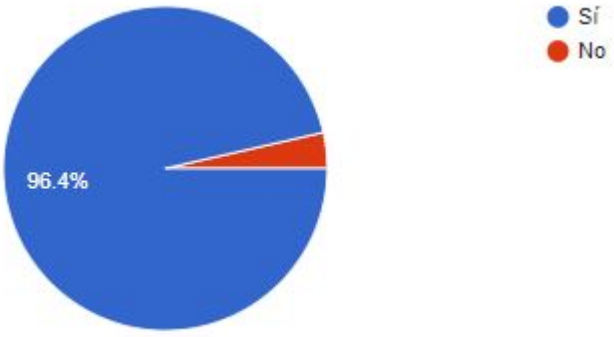

Figura 6.3: Gráfico con porcentajes de respuestas de la pregunta 4.

- La Figura 6.4 muestra el resultado de las respuestas de la pregunta 5 expuesta en la sección 6.1.1.

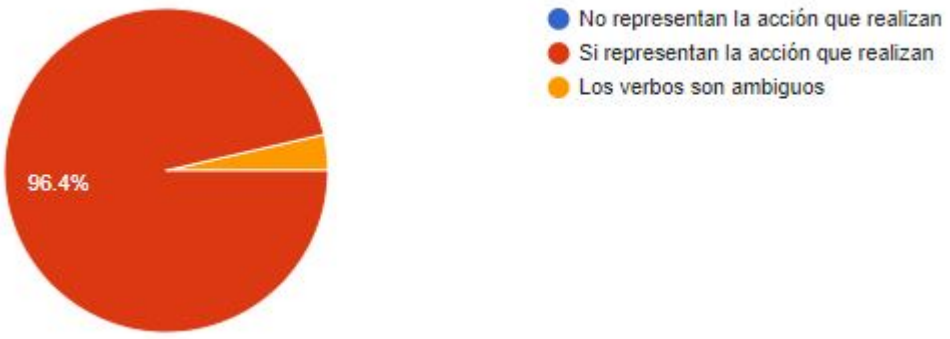

Figura 6.4: Gráfico con porcentajes de respuestas de la pregunta 5.

- La Figura 6.5 muestra el resultado de las respuestas de la pregunta 6 expuesta en la sección 6.1.1.

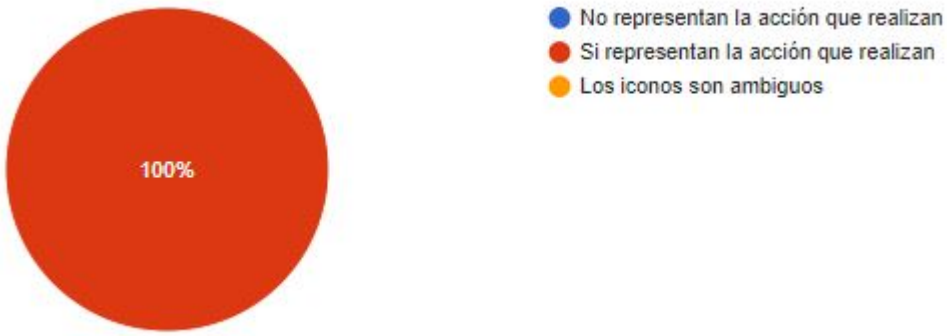

Figura 6.5: Gráfico con porcentajes de respuestas de la pregunta 6.

- La Figura 6.6 muestra el resultado de las respuestas de la pregunta 7 expuesta en la sección 6.1.1.

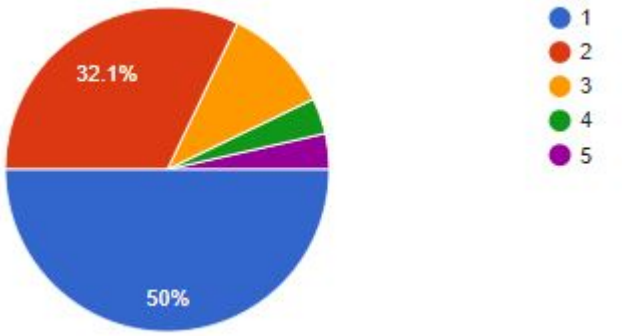

Figura 6.6: Gráfico con porcentajes de respuestas de la pregunta 7.

- La Figura 6.7 muestra el resultado de las respuestas de la pregunta 8 expuesta en la sección 6.1.1.

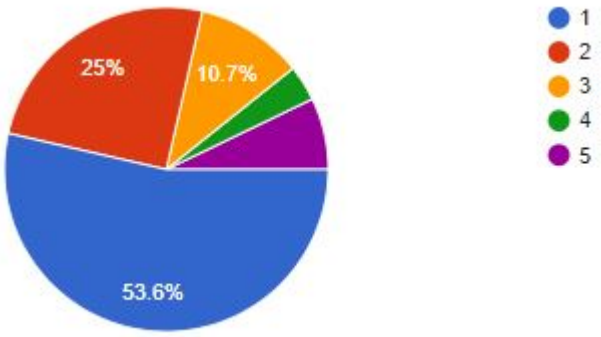

Figura 6.7: Gráfico con porcentajes de respuestas de la pregunta 8.

- La Figura 6.8 muestra el resultado de las respuestas de la pregunta 9 expuesta en la sección 6.1.1.

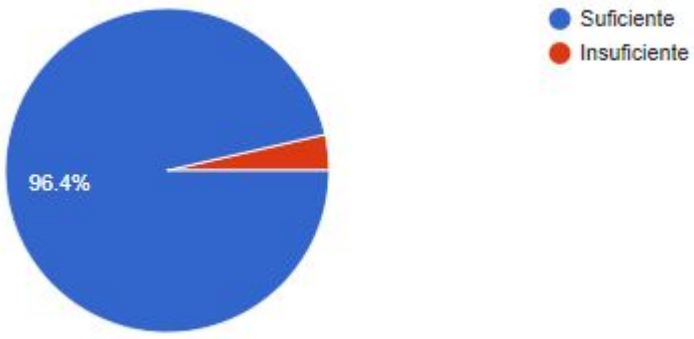

Figura 6.8: Gráfico con porcentajes de respuestas de la pregunta 9.

- La Figura 6.9 muestra el resultado de las respuestas de la pregunta 10 expuesta en la sección 6.1.1.

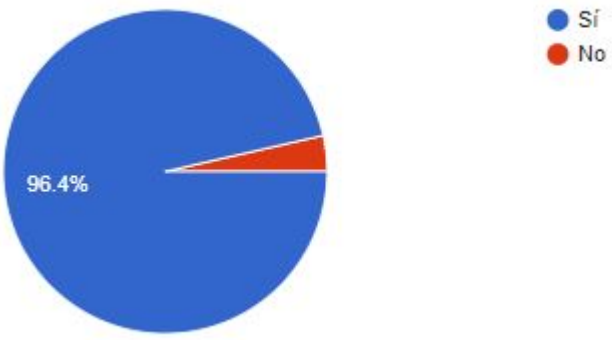

Figura 6.9: Gráfico con porcentajes de respuestas de la pregunta 10.

- La Figura 6.10 muestra el resultado de las respuestas de la pregunta 11 expuesta en la sección 6.1.1.

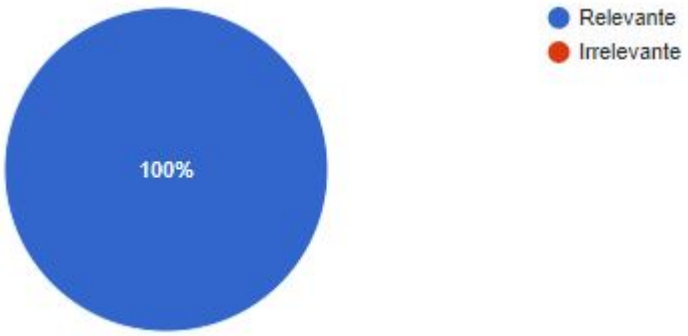

Figura 6.10: Gráfico con porcentajes de respuestas de la pregunta 11.

6.2.2. Resultados de Casos de Prueba

En esta sección se muestra una figura de la aplicación móvil como evidencia por cada uno de los casos de pruebas presentados en la subsección 6.1.3.

- Evidencia de Caso de Prueba CP01 en la Figura 6.11.

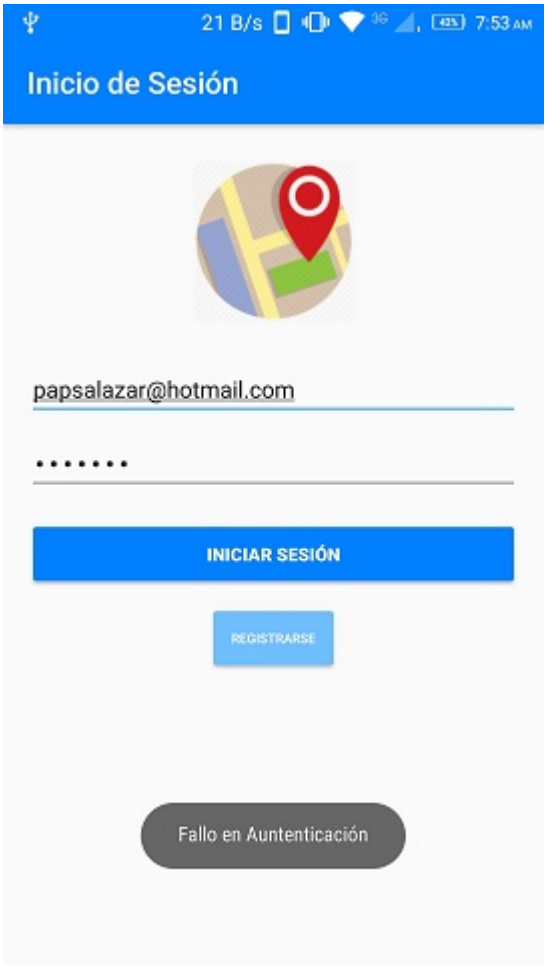

Figura 6.11: Mensaje de fallo de auntenticación.

- Evidencia de Caso de Prueba CP02 en la Figura 6.11.
- Evidencia de Caso de Prueba CP03 en la Figura 6.12.

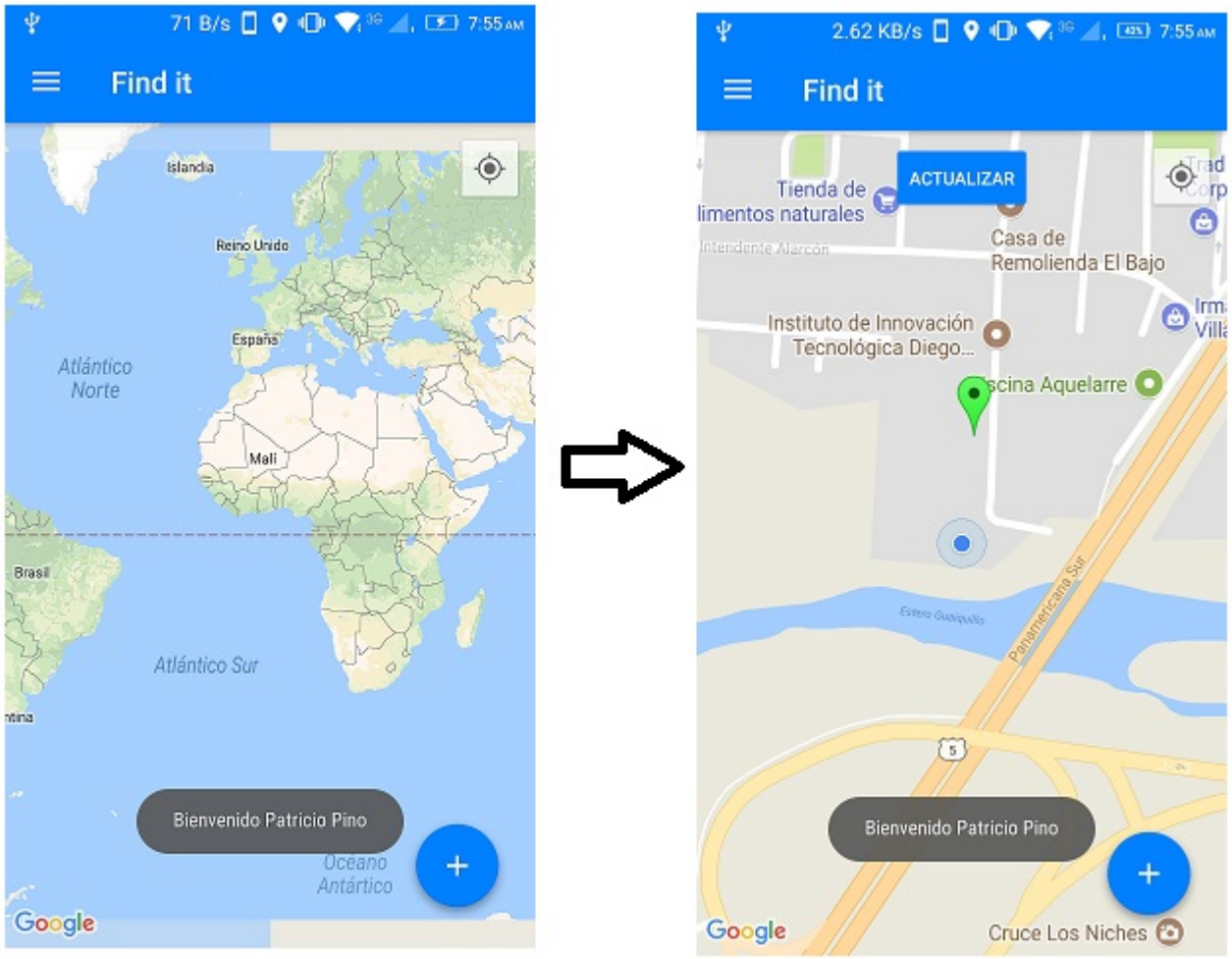

Figura 6.12: Transición desde el inicio de sesión hacia la pantalla principal.

- Evidencia de Caso de Prueba CP04 en la Figura 6.13.

Figura 6.13: Mensaje de fallo al crear un reporte perdido o encontrado cuando faltan valores requeridos.

- Evidencia de Caso de Prueba CP05 en la Figura 6.14.

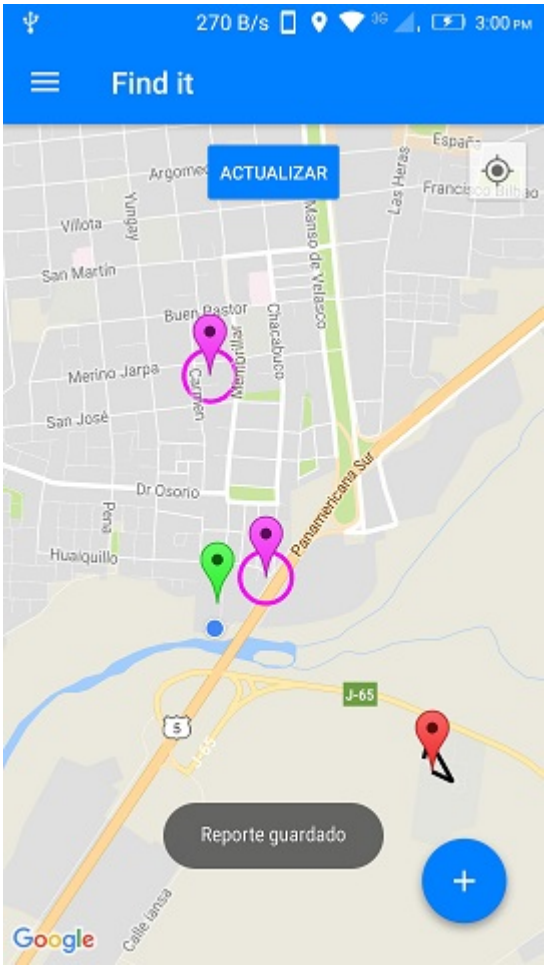

Figura 6.14: Mensaje de éxito cuando guarda un reporte en el Servicio Web.

- Evidencia de Caso de Prueba CP06 en la Figura 6.15.

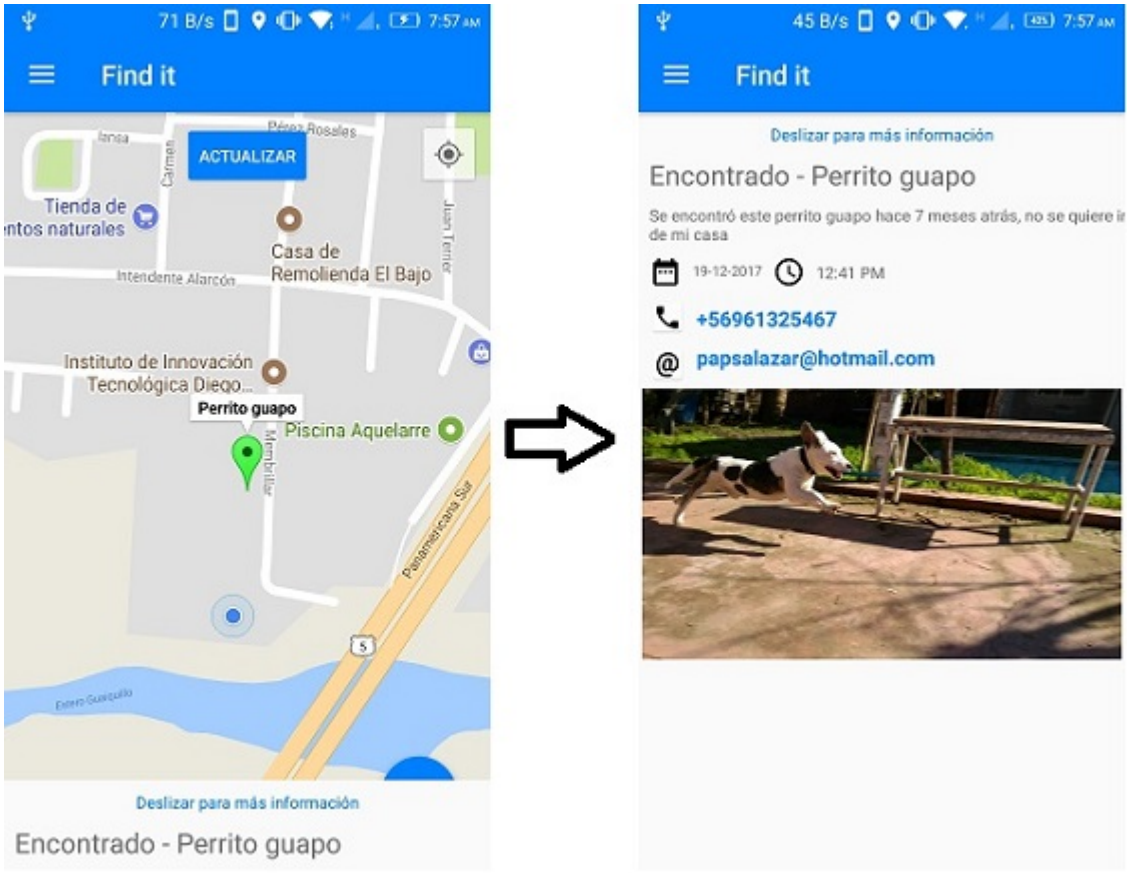

Figura 6.15: Información de un reporte al tocar el marcador e información detallada cuando se desliza el panel hacia arriba.

- Caso de Prueba CP07 no se puede recrear debido a se utilizó un método que evita seleccionar archivos que no sean imágenes.
- Evidencia de Caso de Prueba CP08 en la Figura 6.16.

Figura 6.16: Imagen agregada a la creación de un reporte.

- Evidencia de Caso de Prueba CP09 en la Figura 6.17.

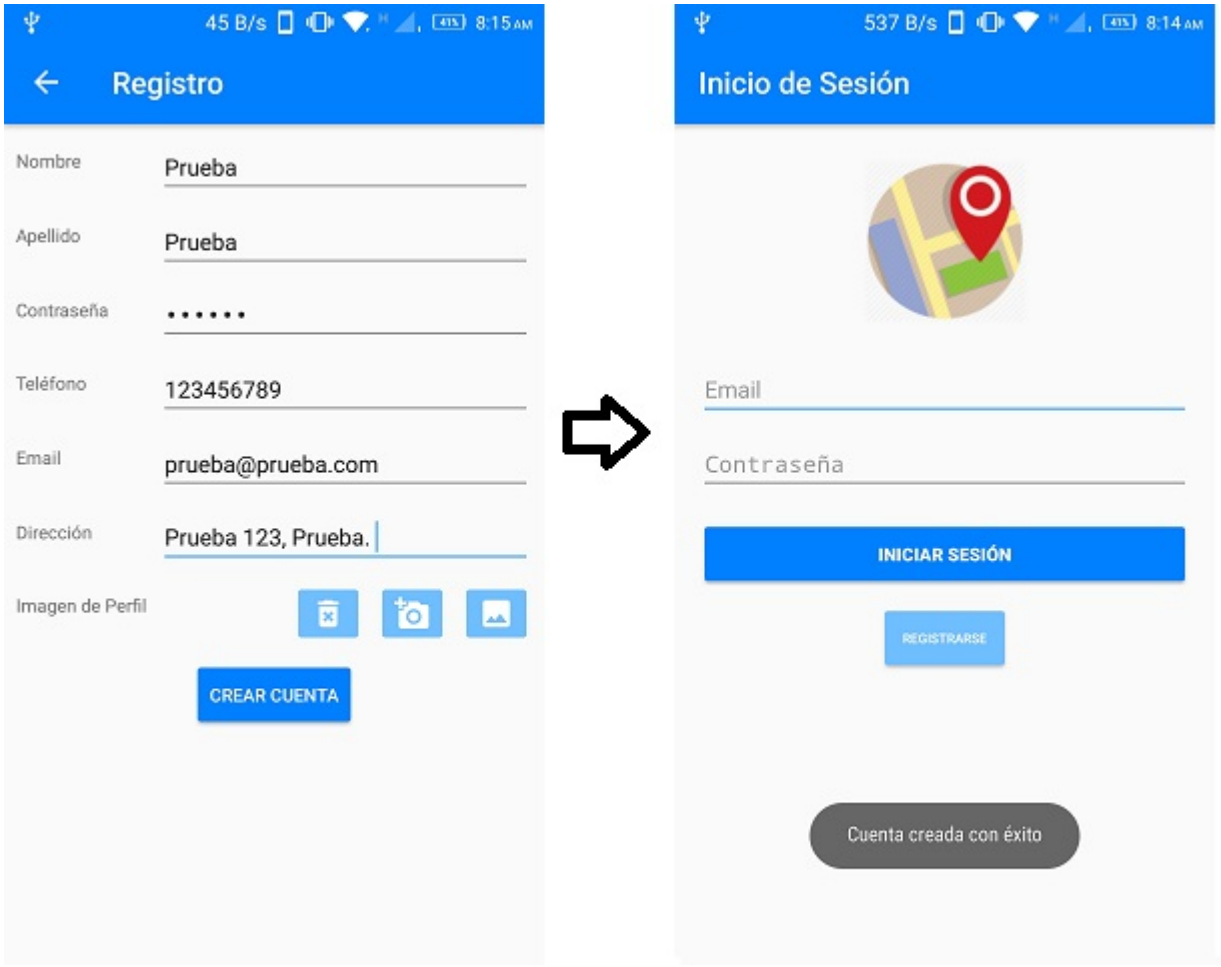

Figura 6.17: Mensaje de registro de usuario exitoso y redirección a la pantalla de Inicio se Sesión.

- Evidencia de Caso de Prueba CP10 en la Figura 6.18.

Figura 6.18: Mensaje de fallo cuando se intenta registrar un usuario con un email existente en la base de datos.

- Evidencia de Caso de Prueba CP11 en la Figura 6.19.

Figura 6.19: Mensaje de fallo cuando se intenta registrar un usuario con campos requeridos faltantes.

- Evidencia de Caso de Prueba CP12 en la Figura 6.20.

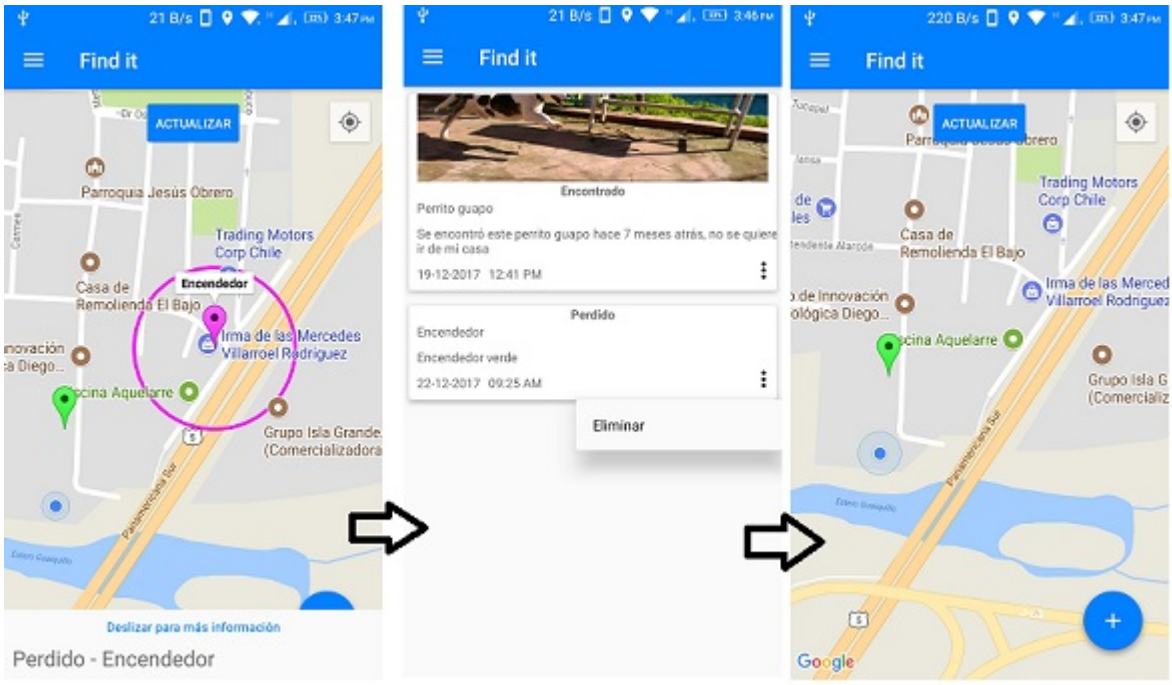

Figura 6.20: Eliminar un reporte desde el menú Mis Reportes y eliminación del reporte en el mapa.

- Evidencia de Caso de Prueba CP13 en la Figura 6.21.

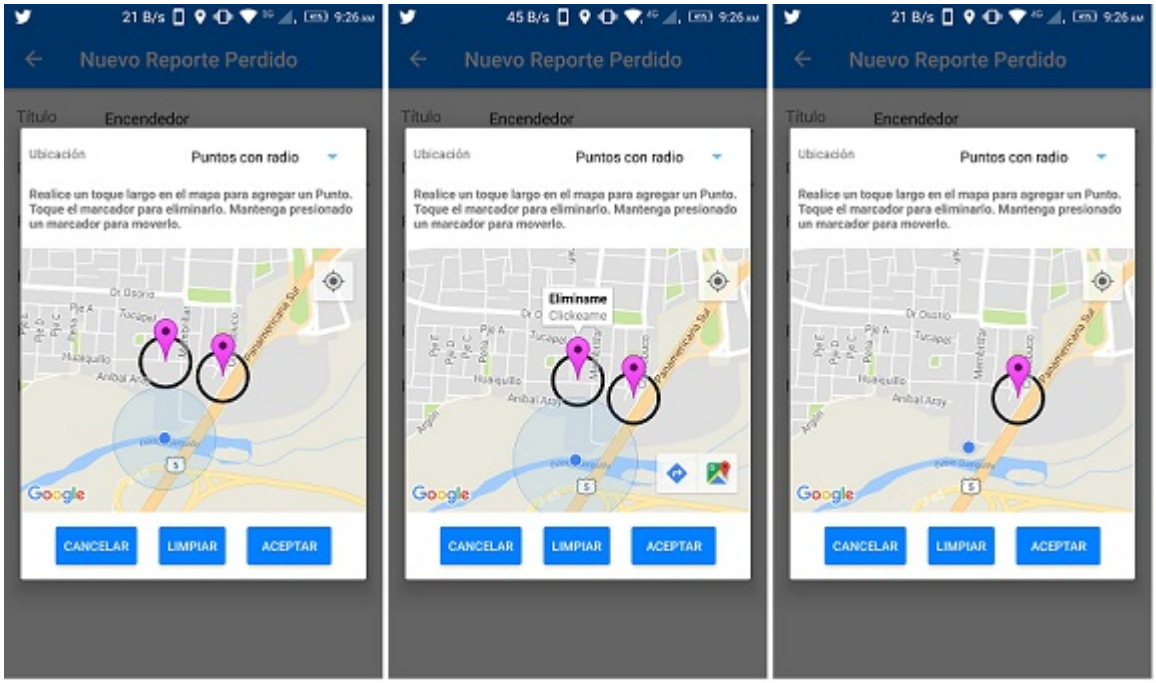

Figura 6.21: Eliminar un punto del mapa cuando se crea la ubicación para el reporte.

- Evidencia de Caso de Prueba CP14 en la Figura 6.22.

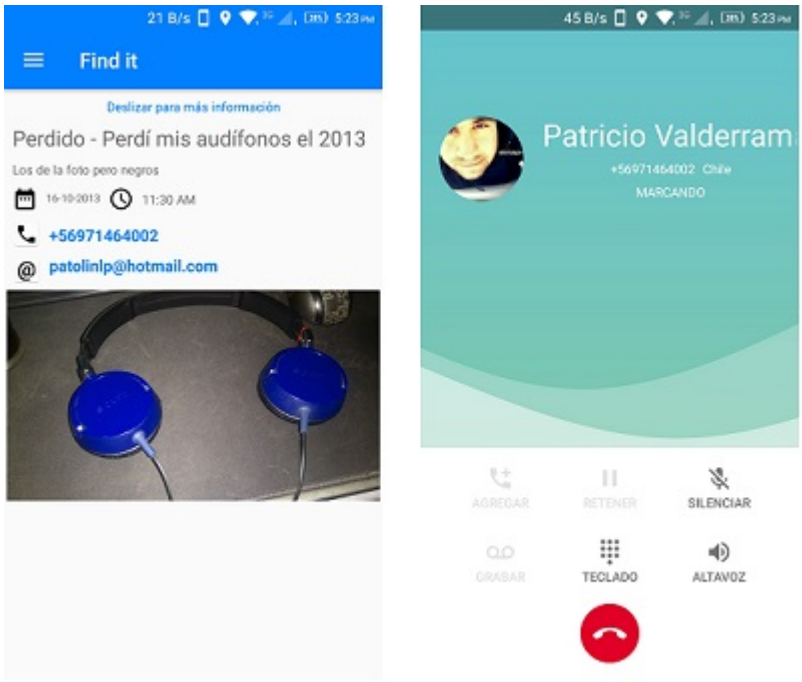

Figura 6.22: Seleccionar número telefónico y realización de llamada al número.

- Evidencia de Caso de Prueba CP15 en la Figura 6.23.

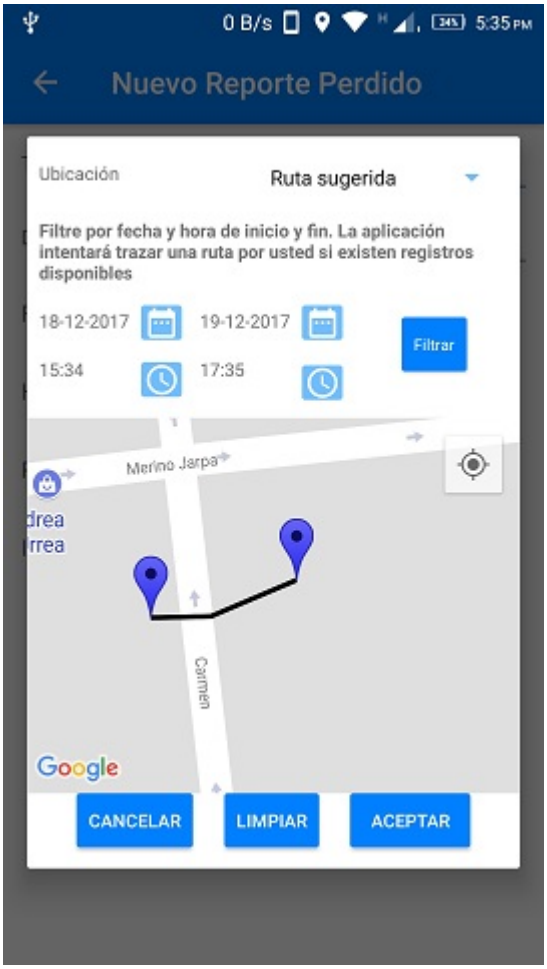

Figura 6.23: Seleccionar número telefónico y realización de llamada al número.

- Evidencia de Caso de Prueba CP16 en la Figura 6.24.

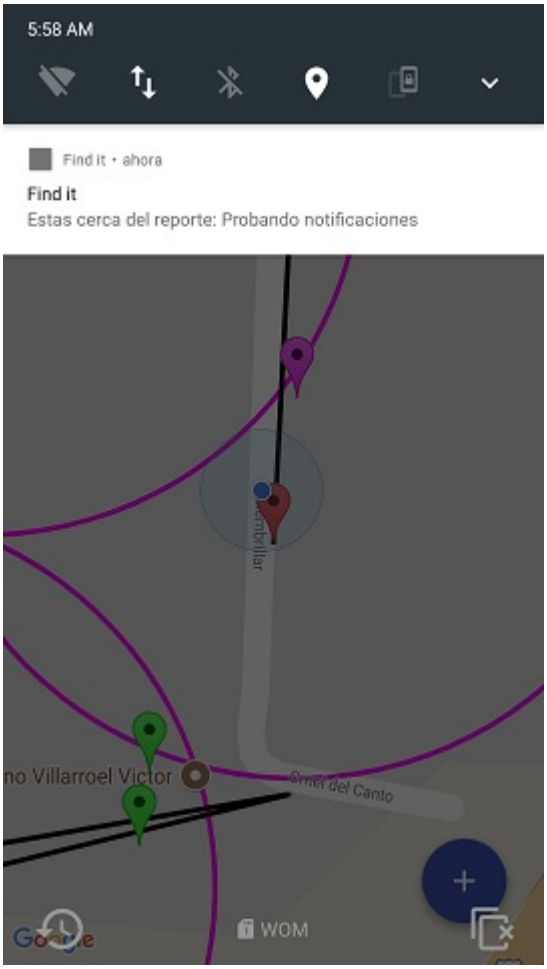

Figura 6.24: Notificación creada cuando un usuario se acerca a un reporte.

6.3. Análisis de los resultados

En esta sección se generará un análisis de las pruebas establecidas para el proyecto desarrollado en este documento. Se analizaron a través de un cuestionario y en base

a los distintos caso de uso realizados.

6.3.1. Resultado de los gráficos

Los resultados del cuestionario, realizado a 28 personas, fueron graficados para agrupar la información obtenida. En esta sección se analizaron estos datos, discutiendo los puntos más importantes y necesarios que se buscaban con este cuestionario.

Tipos de Usuario

Los usuarios de la aplicación móvil, fueron divididos de acuerdo a su nivel de habilidad con el uso de aplicaciones móviles. La necesidad de esta característica, es lograr que la aplicación sea probada por usuarios con diferente experiencia en este ámbito. Tal como se muestra en la Figura 6.1, el 42,9 % de los encuestados, se considera un usuario de aplicaciones móviles de nivel básico, donde se espera que estos usuarios, tengan un teléfono inteligente al cual acceden a diario, utilizando aplicaciones preinstaladas en el dispositivo, tales como *Cámara, Mensajería instantanea, Llamadas, etc.*. El 35.7 % se considera con un nivel intermedio, que comprende a usuarios que en su teléfono inteligente agregan aplicaciones según sus necesidades. Por último solo el 21,4 % se considera de nivel avanzados, quienes utilizan aplicaciones constantemente, considerando el dispositivo móvil como una tecnología necesaria e indispensable.

Paleta de Colores

La paleta de colores de la interfaz de la aplicación móvil, fue un proceso creado antes del desarrollo de esta, seleccionando 2 tonalidades de azul y un contraste blanco, esta decisión fue basada en la experiencia del desarrollador con el uso de aplicaciones móviles. En general, los usuarios respondieron conformes a la paleta de colores, con un 89.3 % de los encuestados, con una opinión agradable acerca de esta combinación, tal como se muestra en la Figura 6.2.

Tamaño del texto

El tamaño del texto, fue desarrollado de acuerdo a los estándares de Android. En base a la experiencia, el desarrollo fue probado en tres diferentes dispositivos,

mencionados en Cuadro 5.1. Tal como se puede apreciar en la Figura 6.3, el tamaño del texto fue aprobado con un 96,4 % de los encuestados.

Verbos en la interfaz

Los verbos utilizados en la interfaz de la aplicación móvil, son cortos y sencillos de acuerdo a su funcionalidad específica, un ejemplo de esto es la opción *Eliminar*, que no se representa de otra forma, descartando ambigüedad en su accionar. Como se apreció en la Figura 6.4, el 96,4 % de los encuestados, considera que los verbos representan la acción que realizan.

Íconos en la interfaz

Al igual que los verbos, los íconos en la interfaz de usuario buscan sustituir palabras para realizar una acción determinada, por lo que estos, deben ser característicos de la acción correspondiente. Los encuestados coinciden en que los íconos utilizados en toda la aplicación, no representan ambigüedad, como se muestra en la Figura 6.5, el 100 % está de acuerdo en que estos iconos representan la acción que realizan.

Facilidad de funcionalidades

El cuestionario indica una instrucción para la creación de un reporte y actualizar el mapa en la aplicación móvil, los resultados de la facilidad de los usuarios para llevar a cabo estas instrucciones se ven reflejados en la Figura 6.6 y la Figura 6.7. En estos gráfico podemos apreciar que el 82.1 % considera que la operación de crear un reporte les fue fácil de completar y un 78.6 % considera fácil la opción de actualizar el mapa.
Como se tenía previsto, el 100 % de los usuarios con habilidades avanzadas, les fue *fácil* llevar a cabo estas operaciones. Así como también, el 100 % de los usuarios de nivel intermedio encontraron *muy fácil* la operación. En cuanto a los usuarios básicos, las respuestas fueron dispersas, pero el 41,6 % de estos, tuvieron inconvenientes en el proceso.

Percepción del objetivo de la aplicación

En el cuestionario, se realizaron preguntas para evaluar la opinión de los usuarios en cuento al objetivo principal de la aplicación, aumentar las posibilidades de

encontrar un objeto perdido. Tal como se ve en la Figura 6.8 y la Figura 6.10, la percepción de los usuarios en aumentar la posibilidad y la relevancia esta aplicación para encontrar un objeto perdido tiene un porcentaje muy cercano al 100 %, lo que indica que los usuarios ven una alternativa útil y convincente para el propósito que se busca cumplir.

6.3.2. Resultado de los casos de prueba

Los resultados de los casos de prueba, fueron expuestos como evidencia en la sección 6.2.2. En esta se presentan los diferentes casos presentados en la sección 6.1.2, a través de capturas de pantalla que evidencian la salida de los casos de prueba mencionados.
Como se puede apreciar, en la sección 6.2.2, todos los casos de prueba expuestos cumplen las salidas establecidas de acuerdo a las diferentes operaciones de los casos de prueba. Por lo tanto la aplicación móvil está preparada para las distintas situaciones de éxito, error o advertencia que se pueden provocar en la interacción con los usuarios.

6.4. Resumen

En este capítulo se expusieron las pruebas bajo las cuales la aplicación móvil fue evaluada. En una primera instancia se presentaron los dos tipos de prueba, exponiendo las diferentes características y forma de evaluación. Como resultado a los diferentes tipos de prueba, se analizaron de acuerdo a gráficos y capturas de pantalla de la aplicación móvil, bajo la utilización de esta por usuarios y generando las diferentes salidas que se espera de las operaciones de la aplicación. En el siguiente capítulo se expondrán las conclusiones generadas a partir de la realización de este proyecto y de las pruebas. Posteriormente pasaremos a revisar los trabajos futuros que podrían ser de gran ayuda en el proyecto.

7. Conclusiones

El proyecto desarrollado permitió aplicar todos los conocimientos del área de la computación al entorno de los dispositivos móviles, que día a día se transforman en aparatos multifuncionales con gran capacidad de computo y autonomía, siendo muy capaces para ejecutar una aplicación que está en constante procesamiento de datos como lo es la aplicación que se desarrolló en este proyecto.

Extraviar una pertenencia, que tiene un valor económico alto, o simplemente un aprecio personal, se transforma de un momento a otro en una labor ardua para el afectado y por lo general, recuperar la pertenencia muchas veces es una acción fortuita. Es por este motivo, que se desarrolló este proyecto, buscando crear una aplicación móvil que permita a usuarios realizar reportes de un objeto perdido o simplemente un hallazgo, con el objetivo de aumentar las posibilidades de que otro usuario lo encuentre y lo retorne a su dueño.

En cuanto al proyecto, se pudo demostrar que es posible implementar una aplicación móvil distribuida de uso colaborativo donde la información generada por medio de reportes georreferenciados, puede ser fácil y amigablemente visualizada en un mapa. Las funciones implementadas logran alcanzar el objetivo de que un usuario mediante el uso de esta aplicación móvil tiene a disposición todo lo necesario para recuperar o devolver una pertenencia. La aplicación por si sola, permite conectar a usuarios que no necesariamente tiene que conocerse entre si. Además, se puede considerar que trabaja de forma autónoma, generando notificaciones cuando un usuario está cerca de un reporte, característica que desliga al usuario de estar ejecutando la aplicación cada vez que quiera saber si existe un reporte cerca de él.

La tecnología de este proyecto, abarca la comunicación de datos livianos a través de la nube, para adaptarse a los distintos sectores en los que los usuarios se encuentren. La aplicación móvil funciona bajo características de teléfonos inteligentes

de última generación, relegando funcionalidades y hardwares que estos poseen, tales como GPS, cámara fotográfica, accesos a internet, realizar llamadas, enviar correos electrónicos, etc. Estas funcionalidades permiten que los usuarios no se apoyen en otro dispositivo más, provocando una simplicidad a la hora de utilizar esta aplicación.

Los objetivos descritos en este documento, se cumplieron de la mejor manera. El trabajo en Android nativo permitió que no se generaran errores visuales o funcionales en los dispositivos. De acuerdo a las pruebas, se tiene manifiesto de que la aplicación móvil tiene un impacto en aumentar las posibilidades de encontrar un objeto perdido, además de ser sencilla de utilizar, en cuanto a sus procesos y funcionalidades.

Como se dijo anteriormente, de acuerdo a las pruebas, la aplicación cumplió con los distintos casos de prueba expuestos, entregando siempre una respuesta óptima a los casos establecidos. Los usuarios que fueron partícipes de las pruebas, comprendían niveles de habilidad distintos, con dispositivos móviles y aplicaciones de estos mismos, además de ser usuarios de diferentes edades, situación económica, así como la prueba en cada uno de sus diferentes dispositivos móviles. A pesar de lo anterior, expusieron mediante el cuestionario su opinión de las principales características de uso de esta aplicación, y un porcentaje significativo de estas respuestas fueron positivos, concluyendo que la aplicación móvil no significó un reto para ninguno de ellos.

De acuerdo a las pruebas, queda claro que la aplicación móvil en cuestión, es una idea que fue llevada a un prototipo, que soluciona un gran problema que presentan los usuarios.

En cuanto al futuro de esta aplicación, existen muchas características que quedaron fuera de alcance, debido al corto tiempo de desarrollo. Algunas de las principales características que podrían incluir funcionalidades provechosas para los usuarios, serían el registro e inicio de sesión con alguna de las redes sociales actuales, además de poder compartir la información de la aplicación en estos mismos. Otra característica importante sería visualizar de mejor manera los datos en los mapas, filtrando los reportes de usuarios que sean importantes, o incluir evaluaciones y agradecimientos para usuarios que encuentren un objeto perdido.

Por último y a modo personal, señalo que fue un agrado y un gran desafío llevar los conocimientos computacionales a la plataforma móvil, incluyendo una gran cantidad de conocimientos abordados, como también tecnologías computacionales. Además, crear un sistema que pueda ayudar a las personas a recuperar objetos per-

didos que no solamente tengan un valor monetario, sino también emocional. Por lo anterior, estoy muy orgulloso de poder contribuir en este proceso para las personas, ya que según mi experiencia personal, tengo el conocimiento del sentimiento al perder un objeto que tiene una gran importancia para mi.

7.1. Trabajo futuro

A continuación se presenta un listado con funcionalidades que debido al corto tiempo de este proyecto y al tratarse de un prototipo no fueron incluidas, pero que a futuro agregarían un valor sustancial a las funcionalidades de la aplicación *Find it*.

- Una mejora importante sería agregar características de conexión con redes sociales, tales como Facebook, Twitter o Google+. Estas características funcionarían perfectamente en el registro e inicio de las sesiones de usuario, además de compartir en estas, la información de los reportes registrados.
- Otra mejora importante, tiene relación con el alcance de los dispositivos móviles del proyecto. Es necesario para este proyecto incluir dispositivos móviles Android con diferentes API. Complementando con una aplicación para teléfonos inteligentes que utilicen otras plataformas, tales como IOS o Windows Phone.
- En cuanto a la seguridad, se podría ahondar más en cuanto al cifrado de los datos, entre el cliente y el servidor web.
- Desarrollar una versión web de este proyecto, utilizando los mismos parámetros de la aplicación móvil actual, permitiendo que incluso, en el caso de no tener la disponibilidad de un dispositivo móvil, se puedan recuperar o reportar objetos perdidos.
- Optimización de la aplicación móvil, en cuanto al consumo de energía, para que no impacte la autonomía del equipo.
- Implementar un sistema de recompensa para personas que desean incentivar la búsqueda de sus objetos perdidos.
- Analizar situaciones contraproducentes, para que no se le de un mal uso a la aplicación. Por ejemplo, que usuarios ocupen la aplicación solamente para buscar objetos y no reportarlos como encontrados.

Bibliografía

[1] Biblioteca de utilidades de la google maps android api. https://developers.google.com/maps/documentation/android-api/utility/?hl=es-419. Acceso: 10-09-2017.

[2] Chile es líder en latinoamérica en uso de internet y smartphones según estudio. http://www.emol.com/noticias/Tecnologia/2016/02/22/789497/Crece-el-uso-de-Internet-y-smarphones-en-paises-emergentes.html. Acceso: 12-04-2017.

[3] Class polyutil. http://googlemaps.github.io/android-maps-utils/javadoc/com/google/maps/android/PolyUtil.html. Acceso: 10-09-2017.

[4] Client/server architecture. https://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch20.htm. Acceso: 03-11-2017.

[5] Componentes de la app: Servicios. https://developer.android.com/guide/components/services.html?hl=es-419. Acceso: 02-11-2017.

[6] Conceptos sobre apis rest. http:/http://asiermarques.com/2013/conceptos-sobre-apis-rest//. Acceso: 10-07-2017.

[7] Http status code symbols for rails. https://gist.github.com/mlanett/a31c340b132ddefa9cca. Acceso: 26-10-2017.

[8] Lista completa de tipos mime. https://developer.mozilla.org/es/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Lista_completa_de_tipos_MIME. Acceso: 22-10-2017.

[9] Marcadores. `https://developers.google.com/maps/documentation/android-api/marker?hl=es-419`. Acceso: 11-06-2017.

[10] Qué es scrum. `https://proyectosagiles.org/que-es-scrum/`. Acceso: 15-09-2017.

[11] The rails doctrine. `http://rubyonrails.org/doctrine/`. Acceso: 25-10-2017.

[12] requestlocationupdates. `https://developer.android.com/reference/android/location/LocationManager.html#requestLocationUpdates(java.lang.String,long,float,android.location.LocationListener)`. Acceso: 02-11-2017.

[13] Subtel: 77 % de las conexiones a internet son a través del celular. `http://www.latercera.com/noticia/subtel-77-de-las-conexiones-a-internet-son-a-traves-del-celular/`. Acceso: 12-04-2017.

[14] Transmitting network data using volley. `https://developer.android.com/training/volley/index.html`. Acceso: 15-10-2017.

[15] Using http methods for restful services. `http://www.restapitutorial.com/lessons/httpmethods.html`. Acceso: 20-10-2017.

[16] Martin Fowler. *UML gota a gota*. Addison Wesley Longman de MÃ©xico, 1 edition, 1999.

[17] Roger S. Pressman. *Ingeniería del software, un enfoque práctico*. McGraw-Hill, 7 edition, 2010.

[18] Ian Sommerville. *Ingeniería de software*. Pearson, 9 edition, 2011.

ANEXOS

A. Tablas de requisitos y Matriz de trazado

En este anexo se listan los requisitos de sistema y la matriz de trazado de los requisitos de usuarios versus requisitos de sistema.

A.1. Requisitos de Sistema

Cuadro A.1: Requisito de Sistema RS01

RS01	Registrar usuario	Prioridad: Alta
La aplicación móvil debe permitir registrar,un usuario en la base de datos del sistema, mediante la solicitud de: nombre,,apellido, correo, teléfono, dirección y contraseña. Además, debe indicar si el,registro ha sido exitoso o erróneo, dependiendo de la situación.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.2: Requisito de Sistema RS02

RS02	Iniciar sesión	Prioridad: Alta
La aplicación debe permitir proveer una interfaz para que un usuario inicie sesión mediante sus credenciales creadas en el Registro de usuario (RS01).		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.3: Requisito de Sistema RS03

RS03	Validar credenciales	Prioridad: Alta
La aplicación debe permitir validar las credenciales ingresadas por el usuario.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.4: Requisito de Sistema RS04

RS04	Agregar imagen	Prioridad: Alta
La aplicación debe permitir agregar una imagen al perfil del usuario y a un reporte de un objeto perdido o encontrado		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.5: Requisito de Sistema RS05

RS05	Crear reporte objeto perdido	Prioridad: Alta
La aplicación debe permitir proveer una interfaz para que un usuario cree un reporte de un objeto perdido.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.6: Requisito de Sistema RS06

RS06	Crear reporte objeto encontrado	Prioridad: Alta
La aplicación debe permitir proveer una interfaz para que un usuario cree un reporte de un objeto encontrado.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.7: Requisito de Sistema RS07

RS07	Crear ruta	Prioridad: Alta
La aplicación debe permitir crear una ruta en un mapa para un reporte de un objeto perdido.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.8: Requisito de Sistema RS08

RS08	Sugerir ruta	Prioridad: Alta
La aplicación debe permitir sugerir una ruta para un mapa según filtros de fecha y hora de inicio y fin respectivamente.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.9: Requisito de Sistema RS09

RS09	Crear punto	Prioridad: Alta
La aplicación debe permitir crear un punto en un mapa para un reporte de un objeto encontrado.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.10: Requisito de Sistema RS10

RS10	Crear punto con radio	Prioridad: Alta
La aplicación debe permitir crear un punto con radio en un mapa para un reporte de un objeto perdido.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.11: Requisito de Sistema RS11

RS11	Guardar reporte	Prioridad: Alta
La aplicación debe permitir a un usuario guardar un reporte en la base de datos.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.12: Requisito de Sistema RS12

RS12	Eliminar reporte	Prioridad: Alta
La aplicación debe permitir a un usuario eliminar un reporte de la base de datos.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.13: Requisito de Sistema RS13

RS13	Notificación de reporte	Prioridad: Alta
La aplicación debe crear una notificación de un reporte para todos los usuarios que esten cerca de un reporte previamente creado por otro usuario.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Intranzable	Estado: Cumplido	CPXXX

Cuadro A.14: Requisito de Sistema RS14

RS14	Facilidad de uso	Prioridad: Baja
La aplicación debe ser fácil uso e intuitiva para un usuario.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Transable	Estado: Cumplido	CPXXX

Cuadro A.15: Requisito de Sistema RS15

RS15	Modo retrato	Prioridad: Baja
La aplicación debe permitir trabajar en modo retrato.		
Fuente: Entrevista con el cliente	Usuarios: Usuario final	
Estabilidad: Transable	Estado: Cumplido	CPXXX

A.2. Matriz de trazado

A continuación se muestra la matriz de trazado que relaciona cada uno de los requisitos de sistema con los requisitos de usuario capturados de los profesores guías.

Cuadro A.16: Matriz de trazado: requisitos de usuario vs requisitos de sistema

	RU01	RU02	RU03	RU04	RU05	RU06
RS01	X					
RS02	X					
RS03	X					
RS04		X				
RS05		X				
RS06		X				
RS07			X			
RS08			X			
RS09			X			
RS10				X		
RS11		X				
RS12		X				
RS13					X	
RS14						X
RS15						X

B. Bosquejos de interfaz de usuario

En este anexo se puede encontrar los bosquejos que no fueron expuestos en la subsección 4.4.5 del Capitulo 4 de Diseño.

Figura B.1: Inicio de Sesión.

Figura B.2: Registro de Usuario

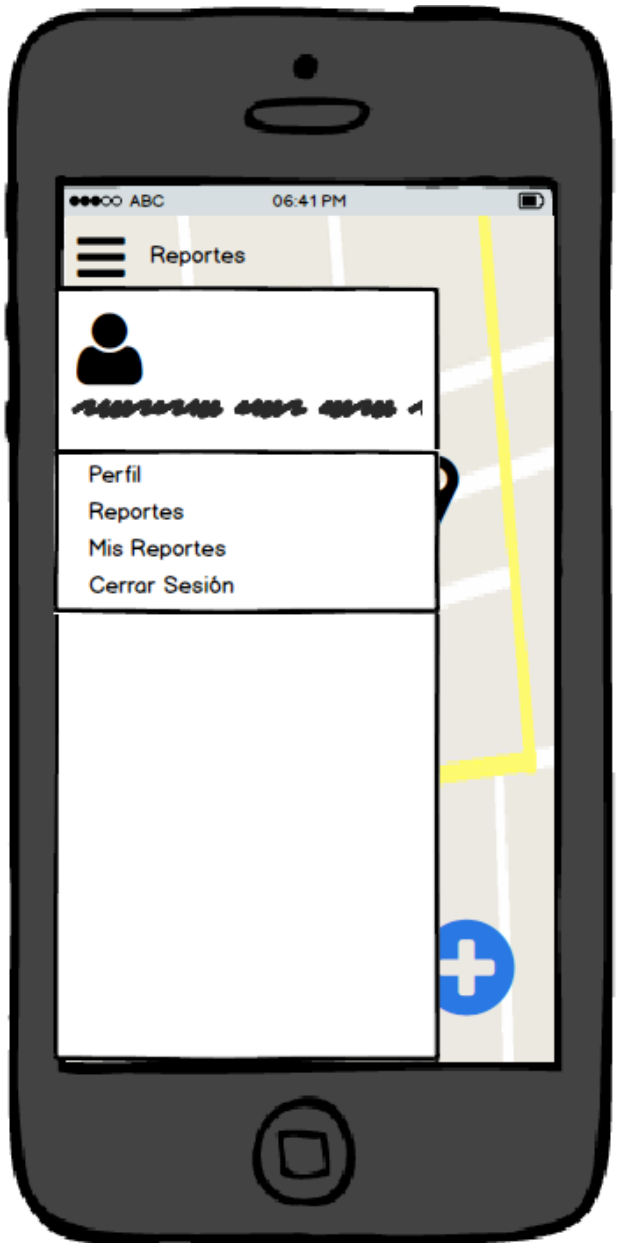

Figura B.3: Menú lateral.

Figura B.4: Perfil de Usuario.

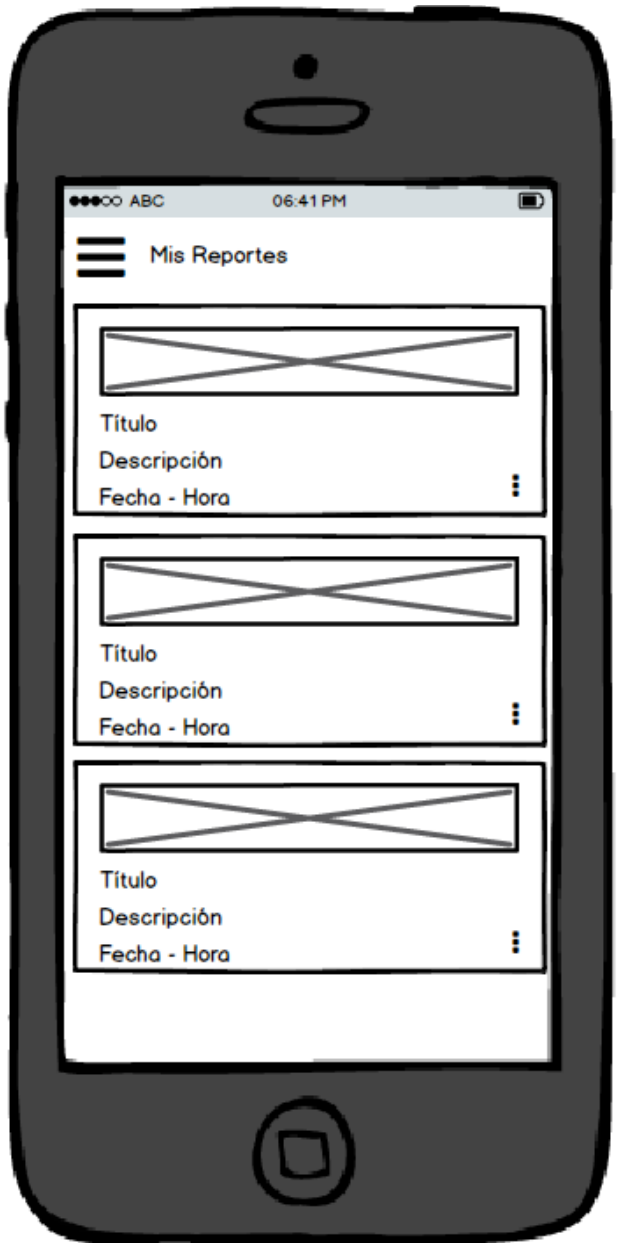

Figura B.5: Mis Reportes.

Figura B.6: Agregar ubicación a un reporte encontrado.

Figura B.7: Agregar ubicación a un reporte perdido.

C. Código fuente

En este anexo se pueden encontrar métodos o extractos de códigos para realizar operaciones que no fueron presentadas en el cuerpo del documento.

C.1. Crear notificaciones en Android

Para crear una notificación en Android se definió la siguiente función.

Listing C.1: Método *crearNotificacion(Reporte reporte)* de la clase *LocationListener*.

```
public void crearNotificacion(Reporte reporte){
// Crear un Intent para poder abrir la aplicación cuando se presiona la
    notificación.
Intent notifyIntent = new Intent(getApplicationContext(),
    MenuActivity.class);
// Configurar la actividad para que se inicie como una nueva tarea en el
    sistema Android.
notifyIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK|
    Intent.FLAG_ACTIVITY_CLEAR_TASK);

PendingIntent notifyPendingIntent = PendingIntent.getActivity(
getApplicationContext(),
0,
notifyIntent,
PendingIntent.FLAG_UPDATE_CURRENT
);
```

```
// Se crear una instancia de la clase NotificationCompat y se configura
    valores como el texto que se mostrará
NotificationCompat.Builder mBuilder = new
    NotificationCompat.Builder(getApplicationContext());
mBuilder.setSmallIcon(R.drawable.common_google_signin_btn_icon_dark);
mBuilder.setContentTitle("Find it");
mBuilder.setContentText("Estas cerca del reporte: "+reporte.getTitulo());
mBuilder.setDefaults(Notification.DEFAULT_ALL);
mBuilder.setContentIntent(notifyPendingIntent);
mBuilder.setAutoCancel(true);

// Se configura un identificador para la notificación
int mNotificationId = 001;
// Se obtiene una instancia de NotificationManager para mostrar la
    notificación en el sistema.
NotificationManager mNotifyMgr =
    (NotificationManager)getSystemService(NOTIFICATION_SERVICE);
// Se crear la notificación y avisa al sistema para que muestre la
    notificación.
mNotifyMgr.notify(mNotificationId, mBuilder.build());
}
```

C.2. Controlador de Reportes del Web Service

Un ejemplo de la estructura de un controlador con sus acciones puede ser apreciado en el Listing C.2.

Listing C.2: Contralador para el recurso Reportes del Servicio Web

```
class ReportesController < ApplicationController
before_action :set_reporte, only: [:show, :edit, :update, :destroy]

# GET /reportes
# GET /reportes.json
def index
  @reportes = Reporte.all
```

```
end

# GET /reportes/1
# GET /reportes/1.json
def show
end

# GET /reportes/new
def new
   @reporte = Reporte.new
end

# GET /reportes/1/edit
def edit
end

# POST /reportes
# POST /reportes.json
def create
   @reporte = Reporte.new(reporte_params)

   if @reporte.save
      render json: @reporte, status: :created
   else
      render json: @reporte.errors, status: :unprocessable_entity
   end
end

# PATCH/PUT /reportes/1
# PATCH/PUT /reportes/1.json
def update
   if @reporte.update(reporte_params)
      render json: @reporte, status: :ok
   else
      render json: {}, status: :unprocessable_entity
   end
```

```
end

# DELETE /reportes/1
# DELETE /reportes/1.json
def destroy
   if @reporte.destroy
      render json: {:status => "no_content"}
   else
      render json: {:status => "internal_server_error"}
   end
end

private
# Use callbacks to share common setup or constraints between actions.
def set_reporte
   @reporte = Reporte.find(params[:id])
end

# Never trust parameters from the scary internet, only allow the white
    list through.
def reporte_params
   params.require(:reporte).permit(:usuario_id, :titulo, :descripcion,
       :fecha, :tipo_id)
   end
end
```

Glosario

hipermedia : Término con que se designa al conjunto de métodos o procedimientos para escribir, diseñar, o componer contenidos que tengan texto, video, audio, mapas u otros medios, y que además tenga la posibilidad de interactuar con los usuarios.

URI : Del inglés *Uniform Resource Identifier*, es una cadena de caracteres que identifica los recursos de una red en forma unívoca.

URL : Del inglés *Uniform Resource Locator*, es un identificador de recursos uniforme (URI) cuyos recurso referidos pueden apuntar a recursos variables en el tiempo.

IDE : Del ingles *Integrated Development Environment*, es un entorno de desarrollo integrado.

Actividad : Una *Actividad* es un componente de la aplicación android que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción.

Fragmento : Un *Fragmento* representa un comportamiento o una parte de la interfaz de usuario en una Actividad.