



# ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS  
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA**

**CARRERA DE ELECTRÓNICA, REDES Y COMUNICACIÓN DE  
DATOS**

**TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL  
TÍTULO DE INGENIERÍA**

**TEMA:** DESARROLLO E IMPLEMENTACIÓN DE UNA  
APLICACIÓN BASADA EN REALIDAD AUMENTADA PARA  
VISUALIZAR ZONAS DE RIESGO Y RUTAS DE ESCAPE EN EL  
SECTOR DEL VALLE DE LOS CHILLOS EN CASO DE UNA  
POSIBLE ERUPCIÓN DEL VOLCÁN COTOPAXI.

**AUTORES:** CAMPAÑA MEJÍA MARIO FERNANDO  
PALACIOS SUÁREZ DARWIN ALEXÁNDER

**DIRECTOR:** MSC. ING. AGUILAR SALAZAR DARWIN LEONIDAS

**SANGOLQUÍ**

**2016**



## DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

### CARRERA DE ELECTRÓNICA, REDES Y COMUNICACIÓN DE DATOS

#### CERTIFICACIÓN

Certifico que el trabajo de titulación, “*DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN BASADA EN REALIDAD AUMENTADA PARA VISUALIZAR ZONAS DE RIESGO Y RUTAS DE ESCAPE EN EL SECTOR DEL VALLE DE LOS CHILLOS EN CASO DE UNA POSIBLE ERUPCIÓN DEL VOLCÁN COTOPAXI*” realizado por los señores *Mario Fernando Campaña Mejía* y *Darwin Alexander Palacios Suárez*, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar a los señores *Mario Fernando Campaña Mejía* y *Darwin Alexander Palacios Suárez*, para que lo sustente públicamente.

Quito, 15 de abril del 2016

  
-----  
MSC. ING. DARWIN LEONIDAS AGUILAR SALAZAR  
DIRECTOR



## DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

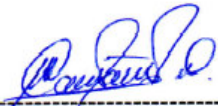
### CARRERA DE ELECTRÓNICA, REDES Y COMUNICACIÓN DE DATOS

#### AUTORÍA DE RESPONSABILIDAD

Nosotros, *Mario Fernando Campaña Mejía*, con cédula de identidad N° 1716814197 y *Darwin Alexánder Palacios Suárez*, con cédula de identidad N° 1720250685 declaramos que este trabajo de titulación “*DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN BASADA EN REALIDAD AUMENTADA PARA VISUALIZAR ZONAS DE RIESGO Y RUTAS DE ESCAPE EN EL SECTOR DEL VALLE DE LOS CHILLOS EN CASO DE UNA POSIBLE ERUPCIÓN DEL VOLCÁN COTOPAXI*” ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaramos que este trabajo es de nuestra autoría, en virtud de ello nos declaramos responsables del contenido, veracidad y alcance de la investigación mencionada.

Quito, 15 de abril del 2016

  
-----  
MARIO FERNANDO CAMPAÑA MEJÍA  
C.C: 1716814197

  
-----  
DARWIN ALEXÁNDER PALACIOS SUÁREZ  
C.C: 1720250685



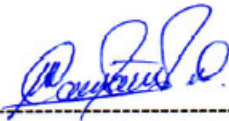
## DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

### CARRERA DE ELECTRÓNICA, REDES Y COMUNICACIÓN DE DATOS

#### AUTORIZACIÓN

Nosotros, *Mario Fernando Campaña Mejía* y *Darwin Alexánder Palacios Suárez*, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación “*DESARROLLO E IMPLEMENTACIÓN DE UNA APLICACIÓN BASADA EN REALIDAD AUMENTADA PARA VISUALIZAR ZONAS DE RIESGO Y RUTAS DE ESCAPE EN EL SECTOR DEL VALLE DE LOS CHILLOS EN CASO DE UNA POSIBLE ERUPCIÓN DEL VOLCÁN COTOPAXI*” cuyo contenido, ideas y criterios son de nuestra autoría y responsabilidad.

Quito, 15 de abril del 2016

  
-----  
MARIO FERNANDO CAMPAÑA MEJÍA  
C.C: 1716814197

  
-----  
DARWIN ALEXÁNDER PALACIOS SUÁREZ  
C.C: 1720250685



## DEDICATORIA

*A Dios, quien ha estado conmigo en cada paso que doy, cuidándome y  
dándome fuerza para continuar.*

*A mis padres, quienes han sido la guía y camino para poder llegar a este punto,  
con su ejemplo, dedicación, palabras de aliento, por su comprensión y amor.  
Me han enseñado todo lo que soy como persona, mis valores, mis principios, mi  
empeño y perseverancia. Ellos nunca bajaron los brazos para que yo tampoco  
lo haga, este logro es para ustedes.*

*A mis hermanas quienes me animaban cada día, por su paciencia ante mi  
cansancio y mal genio. Por acompañarme en este camino.*

*A toda mi familia que ha puesto toda su confianza en mi capacidad para  
afrentar los retos que se han presentado en mi vida.*

*Es por ustedes que he llegado a ser lo que soy y les estaré eternamente  
agradecido.*

Mario Fernando Campaña Mejía

*Dedico este proyecto a Dios por ser mi principal guía y fortaleza para  
seguir adelante con mis objetivos*

*A mi mamá por siempre aconsejarme a seguir el camino correcto y  
apoyarme en cada decisión que haya tomado.*

*A las personas que estuvieron ahí en los momentos difíciles y quienes me  
dieron fuerzas para continuar.*

*Finalmente dedico este trabajo a mi padre quien en vida lucho por verme  
convertido en un gran profesional. En donde se encuentra, estoy seguro que  
está orgulloso porque él sabe que su legado está conmigo.*

Darwin Alexander Palacios Suárez

## **AGRADECIMIENTO**

Agradezco a Dios por darme salud y fuerzas para superar obstáculos y dificultades que se han presentado a lo largo de este camino, de su mano fue posible llegar a culminar esta etapa de mi vida.

A mis padres, Mónica y Mario, quienes siempre me han brindado su apoyo incondicional, por sus consejos sabios brindados en el momento exacto para no dejarme caer y superar momentos difíciles, por darme ese ejemplo de lucha y esfuerzo a diario, por ser la mayor inspiración para formarme profesionalmente, gracias por su educación y enseñanzas para tomar las mejores decisiones y lo más importante gracias por el amor tan grande que me dan. Espero que este paso que doy en mi vida sea un orgullo para ustedes.

A mis hermanas, Cris y Margarita, aunque la mayoría de veces parece que estuviésemos en una batalla, saben que el amor que les tengo es infinito. Gracias por ayudarme en gran manera a concluir esta etapa. Les agradezco por estar en este momento tan importante de mi vida.

A toda mi familia que siempre estuvo a mi lado, preocupándose, ayudándome en lo que sea necesario. Todos intervinieron de cierta forma a que llegue a concluir con esta etapa de mi vida.

A mi director Ing. Darwin Aguilar con quien me encuentro profundamente agradecido por la orientación y supervisión continua, los conocimientos aportados, la confianza depositada para el desarrollo y por todo el tiempo prestado para la realización del presente proyecto de titulación.

A Darwin Palacios, por haber sido un excelente compañero de tesis y amigo, por su perseverancia y dedicación a pesar de los obstáculos y dificultades que se presentaron a lo largo de la realización del proyecto de tesis. Gracias a todas las personas que ayudaron directa e indirectamente en la realización de este proyecto.

Mario Fernando Campaña Mejía

Agradezco a Dios por darme las fuerzas necesarias para poder seguir adelante pese a todas las dificultades. Gracias a Él todavía sigo de pie, luchando para sobresalir ante cualquier dificultad.

Agradezco a mi familia por su paciencia, por siempre darme ánimos y apoyarme hasta el final. En especial a mi mamá, María Dolores, quien siempre tuvo la esperanza de que yo me convierta en un gran profesional y me dio todo su apoyo frente a todas las adversidades. También agradezco a mi papá Washington quien lo dio todo para que nosotros podamos salir adelante y esperó pacientemente hasta el último de sus días para que yo logre cumplir con esta meta.

Agradezco a mis hermanos, Jorge y Miguel, quienes a pesar de todos los conflictos, siempre me consideraron como una persona responsable y perseverante.

Agradezco a Mario Campaña por su gran empeño para poder sacar adelante este proyecto, su capacidad fue de gran ayuda para cumplir con los objetivos establecidos. Estoy muy agradecido con él por aún creer en mí y confiar plenamente en mis conocimientos.

Agradezco a mi tutor Darwin Aguilar quien fue muy comprensivo y nos guió de forma eficiente en el desarrollo de este proyecto. Agradezco además a todos los ingenieros que se han involucrado en mi formación para lograr convertirme en un profesional competente.

Por último agradezco a mis compañeros que me apoyaron para continuar peleando por la carrera y siguieron confiando en mis capacidades a pesar de todas mis fallas.

Darwin Alexander Palacios Suárez

## ÍNDICE

CERTIFICACIÓN.....	ii
AUTORÍA DE RESPONSABILIDAD .....	iii
AUTORIZACIÓN.....	iv
DEDICATORIA .....	v
AGRADECIMIENTO .....	vi
ÍNDICE .....	viii
ÍNDICE DE FIGURAS.....	x
ÍNDICE DE TABLAS.....	xiii
RESUMEN.....	xiv
ABSTRACT .....	xv
CAPÍTULO I.....	1
DESCRIPCIÓN DEL PROYECTO .....	1
1.1 Introducción.....	1
1.2 Antecedentes .....	2
1.3 Justificación e importancia.....	3
1.4 Alcance .....	4
1.5 Objetivos .....	5
1.5.1 Objetivo general .....	5
1.5.2 Objetivos específicos.....	5
CAPÍTULO II.....	7
MARCO TEÓRICO .....	7
2.1 Android.....	7
2.1.1 Descripción .....	7
2.1.2 Características .....	8
2.1.3 Android Studio.....	10
2.1.4 Emulador de Android Studio.....	13
2.2 Google Maps .....	14
2.2.1 Descripción.....	14
2.2.2 Aplicaciones de Google Maps para Android .....	16
2.3 Geolocalización .....	18
2.3.1 Descripción .....	18
2.3.2 Sistema de posicionamiento global .....	18

2.3.2 Algoritmo para identificar un punto dentro de un polígono .....	21
2.4 Análisis y evaluación del riesgo en el sector del Valle de los Chillos en caso de una posible erupción del volcán Cotopaxi .....	22
2.4.1 Zonas de riesgo en el cantón Rumiñahui.....	22
2.6.2 Medidas preventivas ante una posible erupción .....	30
2.6.3. Lista de refugios y sitios seguros en el Valle de los Chillos .....	37
CAPÍTULO III.....	40
CÓDIGO DE LA APLICACIÓN .....	40
3.1 Descripción del programa en Android.....	40
3.1.1 Menú principal .....	40
3.1.2 Implementación de Google Maps en Android .....	42
3.1.2.1 Configuración de la apariencia del mapa.....	43
3.1.2.3 Generación de Polígonos .....	46
3.1.2.4 Generación de Polilíneas .....	47
3.1.2.5 Listas desplegadas sobre el layout de Google Maps .....	48
3.1.2.6 Zoom y posicionamiento en Google Maps.....	51
3.1.2.7 Visibilidad de objetos generados sobre Google Maps .....	53
3.1.2.8 Encontrar un punto dentro de un polígono .....	54
3.1.2.9 Determinación del refugio más cercano .....	57
3.1.2.10 Trazado de rutas hacia el refugio más cercano .....	60
3.1.2.11 Alerta de voz .....	64
3.1.3 Implementación de menús de la aplicación .....	65
3.1.3.1 Llamadas a contactos telefónicos y acceso a páginas web .....	66
3.2 Diagramas de flujo.....	68
3.2.1 Algoritmo de punto en el polígono .....	68
3.2.2 Comparación de zonas.....	69
3.2.3 Cálculo del punto más cercano .....	70
3.2.4 Determinación de la distancia entre dos puntos .....	71
3.2.5 Trazado de ruta al punto más cercano .....	73
3.2.6 Alerta de voz para recomendar el refugio más cercano .....	74
CAPÍTULO IV .....	75
RESULTADOS .....	75
4.1. Análisis de resultados.....	75

4.1.1 Resultados de la interfaz de la aplicación.....	75
4.1.1.1 Resultados en la actividad del mapa .....	79
4.1.1.2 Pruebas preliminares en la actividad del mapa .....	88
4.2. Requerimientos de la aplicación .....	97
4.3. Consumo de recursos de la aplicación .....	99
4.4. Compatibilidad de la aplicación .....	103
CAPÍTULO V .....	107
CONCLUSIONES Y RECOMENDACIONES .....	107
5.1 Conclusiones.....	107
5.2 Recomendaciones.....	110
BIBLIOGRAFÍA.....	113

## ÍNDICE DE FIGURAS

<b>Figura 1.</b> Arquitectura de Android .....	9
<b>Figura 2.</b> Interfaz de programación de Android Studio .....	10
<b>Figura 3.</b> Interfaz multipantalla de Android Studio .....	13
<b>Figura 4.</b> Interfaz del emulador de Android Studio .....	14
<b>Figura 5.</b> Vista Satelital desde Google Maps.....	15
<b>Figura 6.</b> Polígono generado por medio de Google My Maps .....	16
<b>Figura 7.</b> Ruta dibujada sobre Google Maps en una aplicación Android .....	17
<b>Figura 8.</b> Recepción de las señales satelitales del GPS .....	20
<b>Figura 9.</b> Trazado de rayos sobre el polígono .....	22
<b>Figura 10.</b> Zonas afectadas por los lahares en el Valle de los Chillos .....	23
<b>Figura 11.</b> Zona 1 en el Cantón Rumiñahui.....	24
<b>Figura 12.</b> Zona 2 en el Cantón Rumiñahui.....	25
<b>Figura 13.</b> Zona 3 en el Cantón Rumiñahui.....	26
<b>Figura 14.</b> Zona 4 en el Cantón Rumiñahui.....	27
<b>Figura 15.</b> Zona 5 en el Cantón Rumiñahui.....	28
<b>Figura 16.</b> Zona 6 en el Cantón Rumiñahui.....	29
<b>Figura 17.</b> Zona 7 en el Cantón Rumiñahui.....	30
<b>Figura 18.</b> Informe diario de la actividad del volcán brindado por el Instituto Geofísico.....	31
<b>Figura 19.</b> Charlas con los ciudadanos acerca de los posibles riesgos .....	32
<b>Figura 20.</b> Señalización de rutas de escape sobre las calles del sector .....	32
<b>Figura 21.</b> Simulacro de evacuación de estudiantes .....	33

<b>Figura 22.</b> Afiche describiendo la ruta de evacuación en el Centro Comercial San Luis.....	33
<b>Figura 23.</b> Descripción de las acciones a seguir durante las Alertas .....	36
<b>Figura 24.</b> Lista de opciones ubicados en el menú principal .....	40
<b>Figura 25.</b> Creación del menú principal en la Clase MainActivity .....	41
<b>Figura 26.</b> Adaptador del menú principal.....	41
<b>Figura 27.</b> Código de selección de la función de la interface al seleccionar una opción .....	42
<b>Figura 28.</b> Clase MapsActivity.....	43
<b>Figura 29.</b> Método que permite desplegar una lista para cambiar la apariencia del mapa.....	44
<b>Figura 30.</b> Inicialización de puntos de coordenadas.....	45
<b>Figura 31.</b> Generación de marcadores sobre el mapa .....	45
<b>Figura 32.</b> Generación del polígono sobre el mapa.....	47
<b>Figura 33.</b> Generación de una polilínea sobre el mapa .....	48
<b>Figura 34.</b> Implementación de la lista desplegable en el layout de la aplicación.....	49
<b>Figura 35.</b> Ingreso de datos en los headers de la lista .....	49
<b>Figura 36.</b> Ingreso de los headers en la lista desplegable.....	50
<b>Figura 37.</b> Arreglo conformado por los valores de los subitems de la lista desplegable .....	50
<b>Figura 38.</b> Clase del Adapter de la lista desplegable.....	51
<b>Figura 39.</b> Código para la obtención de la posición actual .....	52
<b>Figura 40.</b> Selección de un punto sobre el mapa por medio de la pantalla táctil .....	52
<b>Figura 41.</b> Movimiento de cámara hacia la posición del refugio seleccionado .	53
<b>Figura 42.</b> Visibilidad de las polilíneas cuando se selecciona el checkbox .....	54
<b>Figura 43.</b> Método que maneja ray-casting para determinar un punto dentro de un polígono .....	55
<b>Figura 44.</b> Comparación de las zonas de acuerdo a la posición actual .....	57
<b>Figura 45.</b> Método puntoCercano el cual permite obtener el punto más cercano .....	58
<b>Figura 46.</b> Método getDistance el cual permite obtener la distancia entre dos puntos .....	59
<b>Figura 47.</b> Generación del URL que permite acceder a los datos almacenados en Google Directions .....	60
<b>Figura 48.</b> Método que permite acceder a los valores almacenados desde la URL .....	61
<b>Figura 49.</b> Método que permite descargar los datos desde la URL .....	61
<b>Figura 50.</b> Obtención de los valores de la URL los cuales se encuentran en formato JSON .....	62
<b>Figura 51.</b> Clase para pasar los datos de coordenadas desde JSON hasta un arreglo.....	63
<b>Figura 52.</b> Decodificación de los puntos de la polilínea.....	63



<b>Figura 53.</b> Trazo de la polilínea a partir de los puntos de coordenadas obtenidos .....	64
<b>Figura 54.</b> Inicialización de la variable TextToSpeech.....	65
<b>Figura 55.</b> Asignación del texto que se debe escuchar en la aplicación por medio de TextToSpeech .....	65
<b>Figura 56.</b> Generación del menú de la actividad <i>MedidasPreventivas</i> .....	66
<b>Figura 57.</b> Generación de los contactos telefónicos .....	67
<b>Figura 58.</b> Generación de los links de las páginas web.....	68
<b>Figura 59.</b> Diagrama de flujo del método PointinPolygon para determinar el punto dentro de un polígono .....	69
<b>Figura 60.</b> Diagrama de flujo de algoritmo de comparación de zonas .....	70
<b>Figura 61.</b> Diagrama de flujo del método puntoCercano para determinar el refugio más cercano .....	71
<b>Figura 62.</b> Diagrama de flujo del método getDistance para determinar la distancia entre dos puntos .....	72
<b>Figura 63.</b> Diagrama de flujo del método onPostExecute para trazar la ruta entre dos puntos .....	73
<b>Figura 64.</b> Diagrama de flujo del método voz para emitir una alerta de voz ...	74
<b>Figura 65.</b> Pantalla inicial de la aplicación.....	75
<b>Figura 66.</b> Vista del menú Principal.....	76
<b>Figura 67.</b> Diagrama de las alertas volcánicas .....	76
<b>Figura 68.</b> Listado de las medidas preventivas .....	77
<b>Figura 69.</b> Menú de los enlaces informativos .....	77
<b>Figura 70.</b> Listado del kit de emergencias .....	78
<b>Figura 71.</b> Listado de los objetos del Botiquín de Primeros Auxilios.....	78
<b>Figura 72.</b> Listado de los contactos telefónicos .....	79
<b>Figura 73.</b> Mensaje de recomendación para activar el GPS del móvil.....	80
<b>Figura 74.</b> Vista inicial del mapa .....	80
<b>Figura 75.</b> Menú de apariencia del mapa .....	81
<b>Figura 76.</b> Vista de la zona total de análisis sobre el mapa.....	82
<b>Figura 77.</b> Menú de la lista de refugios .....	83
<b>Figura 78.</b> Vista del refugio seleccionado.....	83
<b>Figura 79.</b> Vista de las rutas de escape .....	84
<b>Figura 80.</b> Ubicación actual en el mapa .....	85
<b>Figura 81.</b> Mensaje de alerta indicando que se encuentra en una zona de riesgo.....	85
<b>Figura 82.</b> Mensaje de alerta indicando que se encuentra en una zona cercana al riesgo.....	86
<b>Figura 83.</b> Comparación de la ruta que se forma al caminar con la ruta que se forma por medio de un vehículo .....	86
<b>Figura 84.</b> Generación de ruta a refugio más cercano .....	87
<b>Figura 85.</b> Mensaje indicando que el dispositivo tiene conexión wifi .....	88
<b>Figura 86.</b> Ruta generada hacia la entrada principal del Liceo del Valle .....	89
<b>Figura 87.</b> Ruta generada hacia la entrada alterna del Liceo del Valle.....	90

<b>Figura 88.</b> Generación de posibles rutas hacia las entradas de la Universidad de las Fuerzas Armadas.....	91
<b>Figura 89.</b> Determinación del refugio más cercano calculando la distancia por línea recta.....	92
<b>Figura 90.</b> Determinación del refugio más cercano calculando la distancia de la ruta.....	92
<b>Figura 91.</b> Generación de rutas hacia el refugio cercano dentro de una sección en la Zona 1 .....	93
<b>Figura 92.</b> Generación de rutas hacia el refugio cercano pasando por el lahar.....	94
<b>Figura 93.</b> Generación de rutas hacia el refugio cercano por un camino seguro.....	95
<b>Figura 94.</b> Rutas generadas hacia refugios ubicados en otras zonas .....	95
<b>Figura 95.</b> Rutas generadas que obligatoriamente pasan por lahares .....	96
<b>Figura 96.</b> Rutas generadas que obligatoriamente pasan por lahares .....	97
<b>Figura 97.</b> Almacenamiento de la aplicación en el dispositivo móvil.....	98
<b>Figura 98.</b> Almacenamiento de la memoria caché de la aplicación .....	99
<b>Figura 99.</b> Consumo de batería de la aplicación .....	99
<b>Figura 100.</b> Consumo de pila de la aplicación.....	100
<b>Figura 101.</b> Procesamiento de CPU de la aplicación .....	101
<b>Figura 102.</b> Uso de memoria de la aplicación .....	101
<b>Figura 103.</b> Estadísticas de Red .....	102
<b>Figura 104.</b> Uso de datos al conectarse la aplicación a la red wi-fi .....	103
<b>Figura 105.</b> Configuración de las versiones de Android en el SDK manager	104
<b>Figura 106.</b> Comparación del menú principal en un teléfono <i>Alcatel One Touch</i> con un Samsung Galaxy S6 .....	104
<b>Figura 107.</b> Comparación de la actividad del mapa en un teléfono <i>Alcatel One Touch</i> con un Samsung Galaxy S6 .....	105
<b>Figura 108.</b> Aumento de la capacidad de la pila del Dalvik .....	106

## ÍNDICE DE TABLAS

<b>Tabla 1:</b> Lista de refugios en el Valle de los Chillos.....	37
---	----

## RESUMEN

Se desarrolló una aplicación de Android por medio de Google Maps que permite visualizar el paso de lahares en el Valle de los Chillos que pueden ser provocados durante una posible erupción del Volcán Cotopaxi. La aplicación permite identificar si un usuario se encuentra dentro de una zona de peligro. Si ese es el caso, se alerta al usuario que se encuentra un lugar de riesgo. Inmediatamente se determina el refugio más cercano y se traza una ruta hacia el mismo por las calles de la zona. Adicionalmente, por medio de un mensaje de voz se recomienda al usuario el sitio seguro más cercano. Sobre el mapa se puede apreciar cómo se divide la zona de análisis en distintas subzonas. La aplicación permite que se hagan pruebas en distintos lugares de la zona de análisis utilizando la pantalla táctil sobre el mapa. Durante la implementación de esta aplicación se realizaron pruebas preliminares para verificar si las rutas dibujadas automáticamente sobre el mapa son seguras y si su recorrido se lo realiza en el menor tiempo posible. En el presente documento se explican las consideraciones que se tomaron para desarrollar la aplicación. Se detallan los comandos que se implementaron en Android para dibujar figuras sobre el mapa y trazar las rutas hacia el punto cercano por medio de Google Directions. También se explican las dificultades que se han presentado durante el desarrollo y las soluciones que se han implementado. Por último se hace un análisis del funcionamiento de la aplicación en dispositivos móviles.

### Palabras Clave

- **ANDROID**
- **GOOGLE MAPS**
- **LAHAR**
- **ZONAS DE RIESGO**
- **GOOGLE DIRECTIONS**

## ABSTRACT

An Android application was developed through Google Maps to visualize the passage of lahars in *Valle de los Chillos* that could be caused by a possible eruption of the Cotopaxi Volcano. The application identifies whether a user is within a danger zone. If that is the case, the app alerts the user that he/she is inside a place of risk. Immediately, the nearest refuge is determined and a route to that place is traced through the streets of the zone. Additionally, the app recommends to the user the nearest secure site by an audio message. The test area is divided in different sub-areas over the map. The application allows testing different locations using the touch screen over the map. During the implementation of this application, preliminary tests were performed to check whether the drawn routes are safe and if the user can transit over them as soon as possible.

In this document, the considerations that were taken to develop the application are explained. The commands that were implemented into Android to draw shapes over the map and to trace routes to the closest point through Google Directions are detailed. It explains the difficulties that have been encountered during the development and the solutions that have been implemented. Finally, an analysis of the application performance on mobile devices was made.

### Keywords

- **ANDROID**
- **GOOGLE MAPS**
- **LAHAR**
- **RISK ZONE**
- **SAFE PLACE**
- **GOOGLE DIRECTIONS**

## **CAPÍTULO I**

### **DESCRIPCIÓN DEL PROYECTO**

#### **1.1 Introducción**

En los últimos años se han desarrollado diversas aplicaciones para teléfonos móviles. El uso de estos dispositivos ha ido creciendo últimamente y se han convertido en herramientas útiles para distintas actividades en la vida cotidiana. Se pueden encontrar aplicaciones dedicadas al entretenimiento, para actividades deportivas, comunicación, redes sociales, informativas, entre otros. Adicionalmente, se puede utilizar este tipo de herramientas para cualquier tipo de emergencias, ya sean de menor o de gran magnitud. Últimamente se han desarrollado aplicaciones que ayudan a la evacuación de edificios en caso de incendios, como también existen aplicaciones que brindan ayuda en caso de que se presente un fenómeno natural.

Durante el transcurso del año 2015, el Ecuador ha tenido que hacer frente ante dos posibles eventos naturales, la posible erupción del Volcán Cotopaxi y la llegada del fenómeno de El Niño. En ambos casos el gobierno ha tomado medidas para prevenir a la gente por medio de capacitación y simulacros.

Para poder mejorar la capacitación y mantener informada a la gente en caso de un posible fenómeno natural, se deben hacer uso del mayor número de herramientas posibles. Se puede desarrollar una aplicación que ayude al usuario a identificar si se encuentra en zona de peligro, a donde debe dirigirse si ese es el caso y qué medidas se deben tomar durante una emergencia. Todo esto se lo debe realizar de una forma más personalizada para evitar confusión y se pueda realizar la evacuación de forma más organizada dependiendo de la zona en la que se vive.

## 1.2 Antecedentes

A partir del mes de Junio del 2015, se reinició la actividad en el Volcán Cotopaxi, provocando que los organismos del Sistema de Gestión de Riesgos declaren alerta amarilla. Esto ha generado preocupación por parte del gobierno y del Instituto Geofísico debido a los posibles riesgos que se pueden presentar en caso de pasar a alerta naranja o roja.

En el caso de los sectores que conforman el sector del Valle de los Chillos en el cantón Rumiñahui se pueden presentar lahares que pueden ser causados por el derretimiento de los glaciares del volcán. Este fenómeno provocaría que los ríos que nacen de dicho volcán incrementen notablemente su caudal y acumulen lodo en su trayectoria. Los ríos Santa Clara, Pita y San Pedro pasan por la zona del Valle de los Chillos lo que causaría grandes desastres en las zonas de mayor riesgo e inclusive muchas pérdidas humanas.

Tanto el gobierno como las instituciones encargadas de prevenir y tomar las medidas necesarias en caso de un desastre, han estado informando acerca de los posibles riesgos. Se han presentado mapas de los posibles lugares afectados por los lahares. También se han realizado simulacros, se han instalado alarmas sonoras y se han dado avisos de prueba por medio de mensajes de texto en colaboración con las principales compañías de telefonía del país.

A pesar de que todas estas medidas están dando buenos resultados, existe incertidumbre y desconocimiento por gran parte de la población que vive en estas zonas. Muchas personas no saben qué se debe hacer en caso de que ocurra una erupción o no están seguras si están realmente a salvo en el lugar en donde viven. Debido a que el último evento similar ocurrió siglos atrás, no se sabe con certeza cómo el fenómeno puede afectar realmente a esta zona.

Es por este motivo que se debe aprovechar la tecnología móvil actual, no sólo para monitorear y estudiar el comportamiento del volcán, sino también

para prevenir y alertar a la población de forma personalizada de acuerdo a su ubicación.

### **1.3 Justificación e importancia**

Es necesario mantener a la población informada y prevenida antes y durante una emergencia. Ahora que la tecnología lo permite, se pueden implementar distintas aplicaciones para distintos tipos de emergencia. El manejo de los dispositivos móviles puede ser de gran utilidad a lo hora de informar a la población que se encuentra en zonas de riesgo. Por medio del sistema GPS de los teléfonos actuales se puede ayudar al usuario a conocer su ubicación exacta y la distancia que pueden recorrer. Esto puede ser de gran ayuda en caso de que una inminente evacuación.

La realidad aumentada es una nueva tendencia tecnológica que se está utilizando en la actualidad para poder visualizar en tiempo real parámetros virtuales que brindan información del entorno en donde se encuentra la persona. Esto se utiliza principalmente de forma turística, donde el usuario puede visualizar de forma interactiva información importante del lugar que se está visitando.

En caso de que ocurra una emergencia al pasar de alerta amarilla a naranja o roja, por medio de un mapa virtual se puede brindar al usuario información acerca de las zonas que corren riesgo y de las rutas de escape que se pueden tomar. Por medio del manejo de herramientas de geolocalización y de los sensores del teléfono se puede indicar al usuario su ubicación y el camino que se necesita tomar para llegar a una zona segura o al refugio más cercano.

El manejo de una aplicación de esta magnitud puede ser de gran importancia para evitar el pánico durante un evento como este. Además, el



usuario de forma anticipada podrá hacer uso de esta aplicación para conocer de antemano la ruta de escape más confiable.

Sin embargo, para desarrollar este tipo de herramienta se debe tomar en cuenta la información oficial confiable brindada por los principales organismos que están a cargo de esta situación. Por este motivo, se debe recolectar información de las zonas de riesgo por medio de reportes y mapas oficiales. La información brindada por esta aplicación deberá ser precisa y con un bajo margen de error ya que de eso dependerían vidas humanas.

Esta aplicación puede servir de base para el desarrollo de otro tipo de aplicaciones que ayuden a la prevención en caso de que exista algún tipo de emergencia similar.

#### **1.4 Alcance**

Se hará un estudio de cómo desarrollar una aplicación de realidad aumentada basada en posicionamiento que se pueda ejecutar en dispositivos móviles con la plataforma Android. Este programa deberá utilizar la funcionalidad y los sensores de los dispositivos móviles como GPS, wifi entre otros, a fin de poder mostrar la información a partir de la ubicación del usuario.

Posteriormente se desarrollará un programa piloto y se realizarán pruebas con el mismo en zonas importantes del sector del Valle de los Chillos. A partir de estas pruebas se añadirán más funcionalidades al programa y se seguirán haciendo pruebas en más lugares residenciales.

Este programa inicialmente identificará la ubicación del dispositivo, el usuario será notificado si se encuentra en zona de riesgo, en una zona cercana o en una zona segura. Se recomendará el refugio más cercano y se indicará la ruta en el mapa. Al programa se le añadirán funciones adicionales para el usuario como la posibilidad de acceder a la información oficial publicada por los organismos encargados, acceder al mapa donde se pueden visualizar las zonas

de riesgo por medio de Google Maps y se brindará información que guíe al usuario en caso de que se presente una emergencia.

Se busca que el programa funcione principalmente en zonas residenciales o de gran afluencia en el Valle de los Chillos. Sin embargo, se hará lo posible por abarcar gran parte de los lugares pertenecientes a las zonas de riesgo dentro del área de cobertura y alcance del cantón Rumiñahui. Para poder brindar información más personalizada al usuario según el lugar, se dividirá el área de cobertura en zonas.

## **1.5 Objetivos**

### **1.5.1 Objetivo general**

Desarrollar e implementar una aplicación para dispositivos móviles basada en geolocalización y posicionamiento a fin de que sirva como una herramienta que ayude a la prevención y pueda brindar información de las zonas de riesgo y rutas de evacuación en el Valle de los Chillos en caso de una posible erupción del volcán Cotopaxi.

### **1.5.2 Objetivos específicos**

- Determinar las ventajas del desarrollo de herramientas de geolocalización en teléfonos móviles y cómo esto puede ayudar en casos de emergencia
- Desarrollar un sistema que permita identificar si el usuario se encuentra en zona de mayor riesgo por medio del receptor GPS del dispositivo móvil.
- Implementar una herramienta que permita determinar el mejor camino para llegar a un sitio seguro sobre las principales calles de la zona de forma automática en caso de una emergencia.

- Desarrollar un sistema de prevención a partir de los dispositivos móviles, donde se tenga acceso a la información oficial del estado del volcán, los sitios seguros, las rutas de evacuación y un mapa donde se visualicen las zonas de riesgo.

## **CAPÍTULO II**

### **MARCO TEÓRICO**

#### **2.1 Android**

##### **2.1.1 Descripción**

Android es un sistema operativo basado en Linux el cual funciona para dispositivos móviles como tablets o teléfonos inteligentes donde se hace uso de la pantalla táctil para el manejo de distintas funcionalidades. La última versión de Android desarrollada en la actualidad es la 6.0

Android fue desarrollado inicialmente por Rich Miner, Chris White, Andy Rubin y Nick Sears quienes fundaron la compañía Android inc. en el año 2003 en la localidad de Palo Alto. La compañía Google, la cual financió gran parte del desarrollo del proyecto, terminó comprando la compañía en julio de 2005 y presentó el nuevo sistema operativo para dispositivos móviles el 5 de noviembre del 2007. A partir del año 2010, Android se consolida como el sistema operativo más utilizado para dispositivos móviles. (Hernandez, 2013)

Este sistema operativo es de código abierto por lo que en la actualidad existe una comunidad inmensa de desarrolladores quienes buscan añadir más funcionalidades a los dispositivos móviles sin necesidad de pagar licencias. En la actualidad se han desarrollado más de 1.000.000 de aplicaciones que están disponibles en Google Play Store, el cual es el portal oficial de Google para adquirir este tipo de herramientas.

### 2.1.2 Características

Para que el sistema operativo de Android sea de código abierto, Google liberó gran parte del código de Android bajo la licencia de Apache v2.0. Android soporta Java por lo que la gran mayoría de aplicaciones están escritas en este lenguaje de programación. Android utiliza el framework de Java a partir de aplicaciones orientadas a objetos las cuales se ejecutan sobre las librerías de Java que corren sobre la máquina virtual Dalvik. (Tomás Girones, 2015)

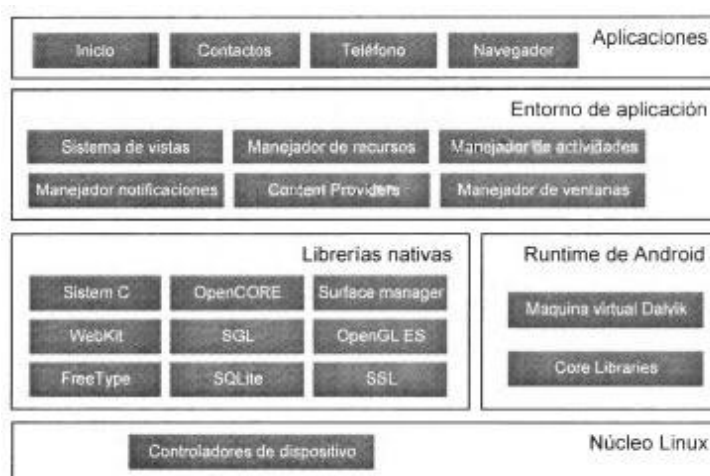
La máquina Virtual Dalvik (DVM) permite la ejecución de programas desarrollados en Java en dispositivos móviles donde la capacidad de energía y memoria es bastante limitada. Dalvik permite ejecutar varias instancias de forma simultánea, para así optimizar el manejo de gestión de memoria, hilos y aislamiento de procesos. (Tomás Girones, 2015)

Android utiliza librerías gráficas en 2D y 3D basadas en OPENGL ES 2.0 permitiendo que las aplicaciones se adapten a distintas resoluciones de pantallas. Android permite el almacenamiento de datos a partir de sqlLite y soporta tecnologías y estándares de comunicación como lo son wi-fi, WIMAX, CDMA bluetooth, HSPA+, entre otros. (Ribas Lequerica, 2014)

El sistema operativo de Android está estructurado en una arquitectura conformada por capas que ofrecen las herramientas necesarias para crear aplicaciones. Cada capa de la arquitectura presta servicios a los niveles superiores, comportándose como una pila. (Tomás Girones, 2015). El orden de los niveles que conforman la arquitectura de Android son las siguientes:

- Las aplicaciones base escritas en Java;
- El entorno de trabajo de Java donde los desarrolladores pueden acceder directamente a las API del framework de las aplicaciones. Esta capa permite simplificar la reutilización de componentes. Las aplicaciones pueden publicar sus capacidades para que puedan ser utilizadas por otras, permitiendo que los usuarios reemplacen componentes.

- Un conjunto de librerías base escritas en C/C++ y compiladas en código nativo las cuales son utilizadas por varios componentes del sistema.
- El runtime de Android el cual está conformado por es un set de librerías escritas en Java que proporcionan las principales funciones disponibles para facilitar la programación. Las aplicaciones corren su propio proceso a través de la máquina virtual Dalvik .
- El núcleo Linux conformado por el sistema operativo Linux v2.6 que proporciona servicios base como seguridad, gestión de memoria y procesos, pila de red y controladores y soporte de drivers. (Tomás Girones, 2015)

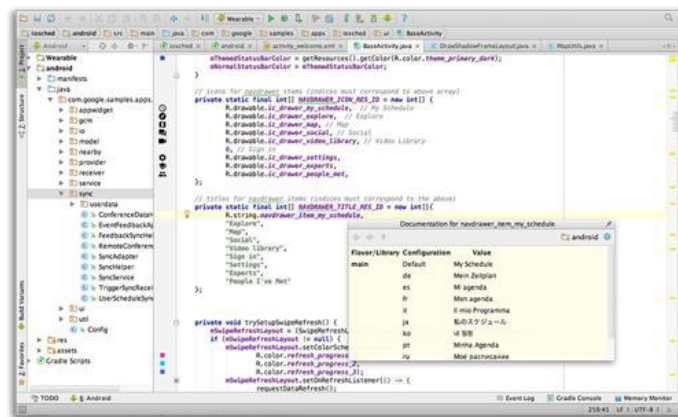


**Figura 1.** Arquitectura de Android  
(Tomás Girones, 2015).

Las aplicaciones se desarrollan en Java por medio de Android Software Development Kit (SDK) el cual es provisto gratuitamente por Google. Las aplicaciones también pueden ser desarrolladas en C/C++ por medio de Android NDK (Native Development Kit). Las aplicaciones se comprimen en archivos APK (Application Package File) los cuales son variantes de los archivos JAR de Java y permiten la distribución e instalación de aplicaciones en dispositivos móviles. (Ribas Lequerica, 2014)

### 2.1.3 Android Studio

Android Studio es un entorno de desarrollo integrado gratuito para Android diseñado por IntelliJ. Es un entorno de desarrollo de Java que trabaja bajo la licencia Apache 2.0 y funciona bajo los sistemas operativos Windows, Linux y MAC OS X. (Ribas Lequerica, 2014)



**Figura 2.** Interfaz de programación de Android Studio (Android Developers)

La versión de prueba fue lanzada en mayor del 2013 y la primera versión estable el 8 de diciembre del 2014. Eventualmente, Android Studio reemplazo a Eclipse ADT (Android Development Tools) como IDE oficial para el desarrollo de aplicaciones móviles debido a que ofrece una solución más cercana al momento de programar aplicaciones para Android (Ribas Lequerica, 2014).

Entre las principales características de la versión actual de Android Studio están (Android Developers) :

- Soporte basado en Gradle para construcción de aplicaciones.
- Arreglos rápidos y refactorización específica de Android.
- Herramientas Lint para detectar problemas de rendimiento, manejo y compatibilidad de versiones.



- Integración ProGuard y capacidades app-signing.
- Asistentes basados en plantillas para crear diseños y componentes comunes en Android.
- Un editor completo para layouts que permite arrastrar y soltar componentes UI y permite la visualización anticipada de layouts sobre múltiples configuraciones de pantalla.
- Soporte para construir aplicaciones para Android Wear (dispositivos corporales).
- Soporte para la plataforma Google Cloud, permitiendo la integración con Google Cloud Messaging y App Engine.
- Renderización en tiempo Real.
- Maneja una Consola de Desarrollador que permite dar consejos de optimización, ayuda para la traducción y estadísticas de uso.

Android Studio viene integrado con el *SDK Manager* que permite gestionar drivers, herramientas de diseño, documentación, imágenes del sistema entre otras herramientas que facilitan el desarrollo de la aplicación. También gestiona el manejo de las distintas versiones de Android dentro del entorno de desarrollo. (Ribas Lequerica, 2014)

Android Studio incorpora *Gradle* la cual es una herramienta que permite procesar la construcción de aplicaciones. Gradle es considerado un lenguaje de programación basado en Groovy. Esta herramienta permite automatizar procesos para poder actualizar y modificar aplicaciones (Ribas Lequerica, 2014).

El núcleo de Android Studio es un editor inteligente que permite la edición, refactorización y análisis del código del programa de forma avanzada. Además se facilita el desarrollo de la aplicación por medio de la vista multi-pantalla. (Android Developers)

Android Studio despliega los archivos del proyecto en *Android Project View* el cual muestra la estructura del proyecto de forma simplificada donde se accede inmediatamente a los códigos fuente del proyecto. Esto facilita el

manejo de los proyectos y de los recursos de la aplicación. (Android Developers)

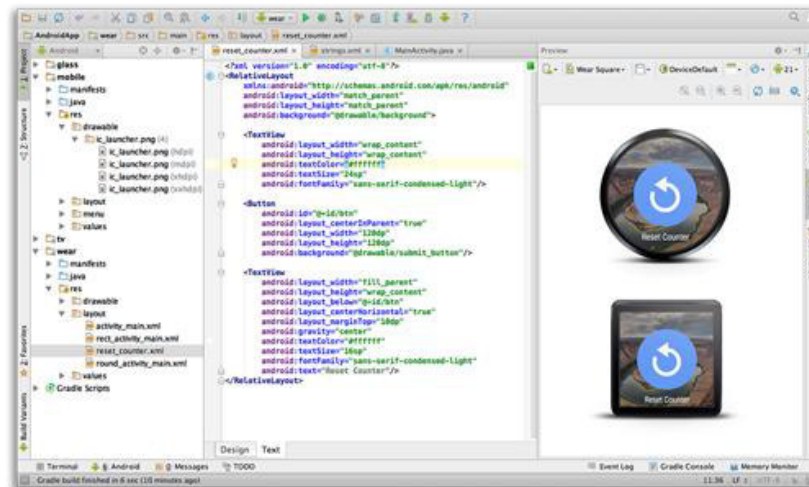
El sistema de Construcción de Android es un conjunto de herramientas utilizado por Android Studio que permite construir, probar, correr y empaquetar las aplicaciones. Se utiliza para personalizar, configurar y extender el proceso de construcción; permite crear múltiples APKs para la misma aplicación, manejando el mismo proyecto y el mismo módulo; también permite reusar códigos y recursos. Todo esto se logra sin necesidad de modificar el código fuente de la aplicación (Android Developers).

Android Studio posee distintas herramientas para poder depurar el programa y mejorar el funcionamiento del código como un administrador de dispositivo virtual, el *inline debugging* y herramientas propias de análisis de funcionamiento (Android Developers).

El Administrador del Dispositivo Virtual de Android (AVD) ayuda al usuario a seleccionar las mejores configuraciones del dispositivo, los tamaños de pantalla y las resoluciones para visualizar la aplicación. Viene con emuladores para dispositivos con Nexus 6 y Nexus 9 y soporta la creación de apariencias para los dispositivos Android, pudiendo asignar esas apariencias a perfiles de hardware reales (Android Developers).

Android Studio posee una herramienta que permite monitorear el consumo de memoria y CPU de la aplicación. Mientras se monitorea la aplicación se puede recolectar la información basura para así detectar posibles fugas de memoria y encontrar soluciones para optimizar el rendimiento de la aplicación . (Android Developers)

Android Studio permite visualizar los layouts por medio del *Design View* donde se puede arrastrar y soltar elementos y componentes desde la paleta de programación hacia el el panel *Preview* o hacia el árbol de Componentes. (Android Developers)

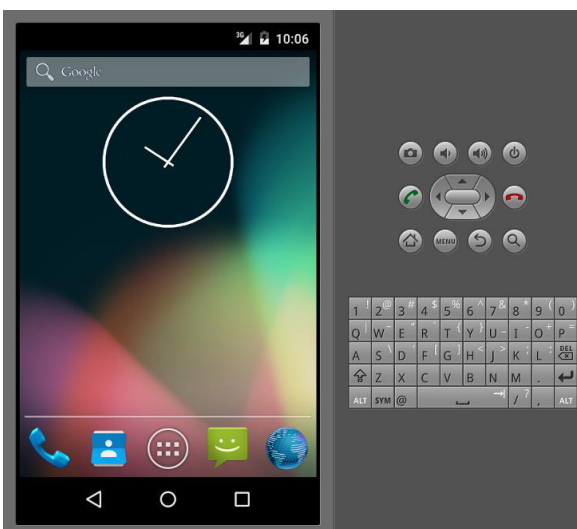


**Figura 3.** Interfaz multipantalla de Android Studio (Android Developers)

La versión 2.0 de Android Studio implementa la función Instant Run la cual permite verificar los cambios realizados en el código al instante en el emulador. En esta versión el emulador funciona de forma más rápida, facilitando realizar pruebas de forma sencilla e inmediata. (Perez, 2015)

#### 2.1.4 Emulador de Android Studio

Android Studio incluye un emulador virtual optimizado que muestra el comportamiento del dispositivo móvil. Este emulador se puede ejecutar en el ordenador y permite desarrollar pruebas del prototipo de la aplicación en Android. El emulador utiliza el Dispositivo Virtual de Android (AVD) para definir ciertos aspectos de hardware y permite crear varias configuraciones para probar las distintas plataformas de Android. (Android Developers)



**Figura 4.** Interfaz del emulador de Android Studio (Android Developers)

Cuando la aplicación se esté ejecutando en el emulador, se pueden utilizar servicios de la plataforma Android para poder invocar otras aplicaciones, acceder a la red, acceder a archivos de audio y video, adquirir información y realizar transiciones gráficas y de temas. (Android Developers)

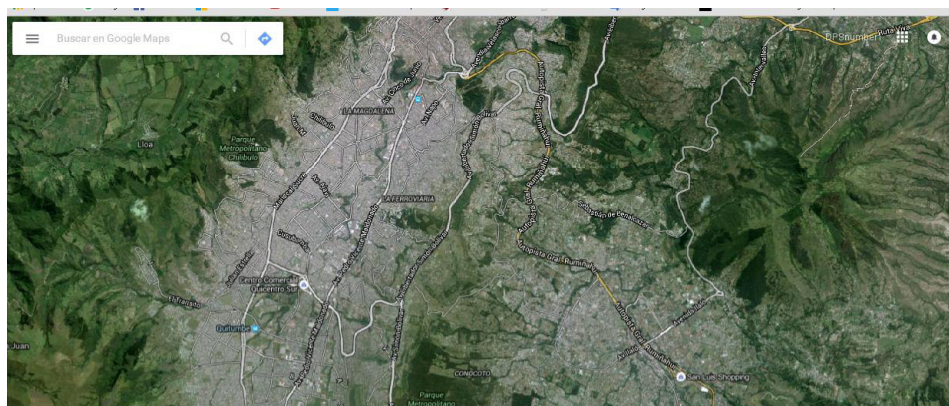
El emulador posee herramientas de depuración para simular interrupciones de aplicaciones, simular efectos de latencia e identificar pérdidas de datos. (Android Developers)

## 2.2 Google Maps

### 2.2.1 Descripción

Google Maps es una aplicación web que permite al usuario acceder a mapas desplazables que permiten visualizar las zonas de cualquier parte del mundo. Este servicio permite visualizar fotografías satelitales de los lugares seleccionados y se pueden identificar las rutas y principales avenidas de las ciudades más importantes del mundo. (Google, 2016)

Fue lanzado en febrero del 2005 de forma limitada para algunos navegadores y desde entonces ha tenido bastante demanda para identificar zonas y localizar la dirección de lugares importantes, ganando importancia a nivel turístico e inclusive empresarial. (Moya, 2015)



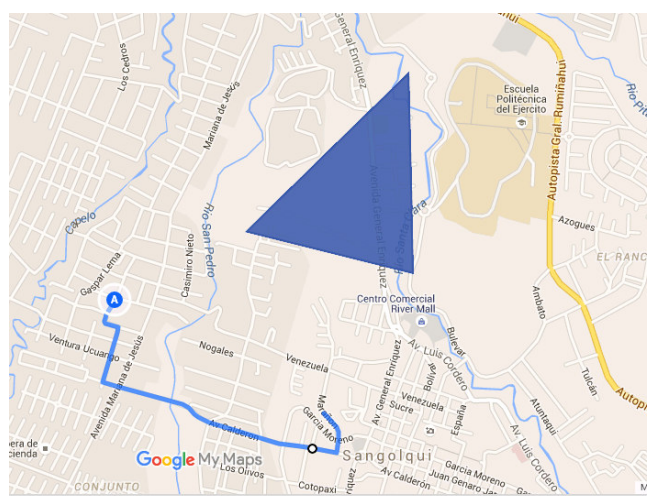
**Figura 5.** Vista Satelital desde Google Maps

(Google, 2016)

Google Maps permite al usuario localizar su propia ubicación por medio del sistema GPS o por medio de la red por la cual el usuario se conecta a través de su navegador. Por medio del mouse o del teclado, el usuario puede desplazarse y cambiar el nivel de zoom del mapa para ver la zona con mayor detalle. Los usuarios además pueden localizar una ubicación específica al ingresar las coordenadas, la dirección de un sitio o el nombre de una organización en el buscador de Google Maps. Además, se pueden obtener las coordenadas de un punto fijo y se puede calcular la distancia en metros entre dos puntos. (Google, 2016)

Google Maps funciona a través de una gran cantidad de archivos JavaScript y XML. Mientras el usuario navega por el mapa, las imágenes satelitales se van descargando desde el servidor principal de Google. (Google Developers, 2016). Google Maps utiliza coordenadas geográficas CGS en el sistema WGS84 para poder representar la latitud y longitud de un punto específico. (Obemapa, 2011)

Por medio de la herramienta *My Maps*, se pueden crear mapas personalizados tomando como base los mapas suministrados por Google Maps. Por medio de esta herramienta se pueden añadir íconos personalizados en puntos específicos del mapa, se pueden sobreponer polígonos con distintas gamas de colores y se pueden añadir pollíneas para formar rutas, entre otras cosas. Esto es de gran utilidad a la hora de diseñar mapas con fines turísticos o para explicar fenómenos naturales dentro de una zona. (Google, 2016)



**Figura 6.** Polígono generado por medio de Google My Maps (Google, 2016)

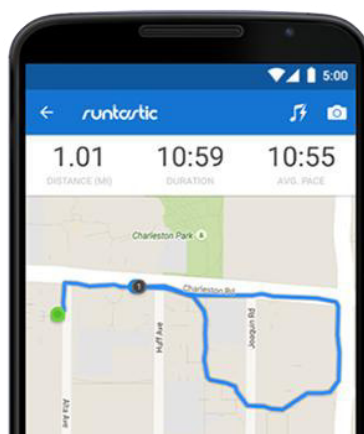
Los mapas creados en My Maps se pueden guardar dentro de la plataforma en la nube Google Drive, o se pueden almacenar dentro de un archivo kml el cual es un archivo xml que almacena las coordenadas de las capa, rutas e íconos. Los archivos kml se pueden exportar a otros mapas. (Google, 2016)

### 2.2.2 Aplicaciones de Google Maps para Android

Google Maps se puede utilizar también en dispositivos móviles que funcionan con Android. El usuario puede utilizar la pantalla táctil para desplazar y ampliar el mapa. El sistema GPS del dispositivo móvil permite que el usuario

localice su ubicación de forma inmediata en el mapa. Después de que Google haya lanzado la API de Google Maps, la interfaz original ha podido ser completamente modificable, siendo libre para su desarrollo en aplicaciones para dispositivos móviles. (Google Developers, 2016)

Para poder realizar una aplicación en Android se debe descargar y configurar los servicios de Google Play. Posteriormente se deberá conseguir y añadir una API key de Google Maps a la aplicación, integrando de esta manera los permisos necesarios para visualizar y modificar los mapas. (Google Developers, 2016)



**Figura 7.** Ruta dibujada sobre Google Maps en una aplicación Android (Google Developers, 2016)

Por medio de comandos de Java creados específicamente para Google Maps, se pueden dibujar polígonos, polilíneas y marcadores en puntos específicos sobre el mapa. Además se pueden agregar capas a partir de puntos de coordenadas almacenados en archivos *kml* o *geojson*. Esto facilita la creación de mapas personalizables para aplicaciones específicas. (Google Developers, 2016)



## **2.3 Geolocalización**

### **2.3.1 Descripción**

La geolocalización es la capacidad de obtener la ubicación exacta de un objeto como un dispositivo móvil. Esto está relacionado con el manejo de sistemas de posicionamiento a fin de determinar la ubicación precisa, permitiendo obtener no solo las coordenadas del dispositivo, sino también la zona o dirección del lugar donde se encuentra.

La localización de un dispositivo se puede determinar por medio de la red de redes a través de la dirección IP asignada al dispositivo. Por medio de este direccionamiento se puede asociar la ubicación geográfica del dispositivo conectado a internet con la dirección asignada por el proveedor ISP. Esto es posible debido a que los rangos de direcciones IP se asignan dependiendo de la zona de acuerdo a lo propuesto por la IANA (Internet Assigned Number's Authority). (King, 2009)

El posicionamiento de un dispositivo también se puede obtener por medio de sensores GPS integrados en el dispositivo o al conectarse a puntos de red inalámbricos. Los proveedores de internet pueden visualizar la ubicación exacta de sus clientes por medio de la red. De igual forma, las empresas que brindan servicios web pueden verificar la ubicación de los clientes que visitan sus páginas web, de esta forma pueden brindar información más personalizada de acuerdo a la región en la que el usuario se encuentra. (King, 2009)

### **2.3.2 Sistema de posicionamiento global**

El Sistema de Posicionamiento Global (GPS) es un sistema que permite determinar la posición exacta de un objeto en cualquier parte del mundo. Este sistema permite identificar la posición a través de la recepción de señales emitidas por una red conformada por 24 satélites que cubren toda la superficie de la tierra. (GPS.gov, 2016)

Para determinar una posición, el receptor localiza al menos cuatro satélites ubicados aproximadamente a 20200 km de altura (GPS.gov, 2016). Los satélites envían una señal que contiene una identificación y la hora de cada uno de ellos. El receptor sincroniza el reloj de GPS y realiza el cálculo del tiempo que tardan las señales en transmitirse. Por medio del método de trilateración inversa, se determina la distancia entre cada satélite y el punto de recepción. Finalmente se determina la posición del receptor respecto a la ubicación de los satélites. El receptor obtiene los valores de latitud, longitud y altitud con respecto a su ubicación actual. (GPS, 2013)

La señal GPS es emitida continuamente por los satélites a 50 bps en la frecuencia de microondas de 1600 Mhz. La señal satelital envía la hora exacta, la identificación de la semana y un reporte del estado del satélite. Se transmite una señal de 1500 bits de datos codificados durante 30 segundos a través de la codificación con secuencia pseudoaleatoria (PRN). El receptor GPS decodifica la señal, pudiendo identificar las distintas señales de los satélites. (Mio, 2010)

Se han integrado receptores GPS en dispositivos móviles como tablets o teléfonos inteligentes. Esto ha facilitado encontrar la localización exacta de dichos dispositivos, permitiendo utilizar esta información para distintas aplicaciones. La posición de un usuario puede ser de gran utilidad para dar información turística del lugar en el que se encuentra. También puede servir para medir distancias y trazar rutas para fines turísticos o deportivos.

Los dispositivos móviles disponen de un sistema de triangulación llamado A-GPS (Sistema de Posicionamiento Global Asistido) el cual ha sido desarrollado para mejorar el funcionamiento del sistema de posicionamiento y la velocidad de procesamiento de los dispositivos. Esto fue implementado inicialmente para identificar la posición de un teléfono inmediatamente en caso de una llamada de emergencia. (GPS World, 2002)



**Figura 8.** Recepción de las señales satelitales del GPS  
(Bioadrian, 2015)

Cuando se activa el GPS del dispositivo móvil, se obtiene una posición aproximada a partir de la información brindada por la red móvil cuando se conecta a internet. El dispositivo se conecta a un servidor externo el cual permite definir la ubicación. A partir de esta posición inicial, se pueden identificar los satélites más cercanos y se triangulará la posición exacta a partir de la señal de los mismos. De esta forma se obtiene la posición del dispositivo de forma más precisa y exacta. (GPS World, 2002)

Cuando el teléfono no está conectado a una red móvil (off-line), el GPS asistido utilizará información de la ubicación que ha sido almacenada previamente. A partir de esta información, se determina la ubicación de los satélites y posteriormente la ubicación exacta del dispositivo. (GPS World, 2002)

En la actualidad se utiliza el receptor GPS de un dispositivo móvil junto con la brújula interna y el acelerómetro para aplicaciones de realidad aumentada basadas en posición, donde por medio de la posición exacta del dispositivo se pueden generar capas de información sobre puntos de interés cercanos. (aumentame, 2011)

Mientras el usuario se mueve con su dispositivo por el entorno, la aplicación utiliza un mapa integrado para determinar los puntos de interés

dependiendo de su posición geográfica y la del usuario. Los puntos de interés son un conjunto de puntos de coordenadas que permiten identificar ubicaciones sobre el espacio físico terrestre. En cada punto de interés se pueden generar mensajes con información que describan las ubicaciones de dichos puntos. Una capa de información es un conjunto de puntos de interés identificados sobre un mapa virtual con sus respectivos valores de longitud y latitud. (aumentame, 2011)

### **2.3.2 Algoritmo para identificar un punto dentro de un polígono**

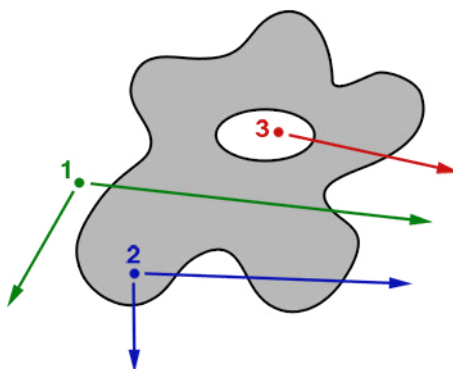
Por medio de la geolocalización se puede obtener la ubicación exacta de un dispositivo móvil. Dicha ubicación se puede visualizar por medio de un mapa generado en Google Maps. Por medio de esta herramienta también es posible dibujar figuras poligonales sobre el mapa. En muchas aplicaciones es bastante útil que se pueda identificar si el punto de coordenadas se encuentra dentro del área conformada por el polígono dibujado.

Uno de los algoritmos más utilizados para encontrar un punto dentro de un polígono sin importar su forma o el número de vértices que lo conforma es el algoritmo de ray-casting. Este algoritmo fue definido inicialmente por Scott Roth en 1982 para poder resolver distintos problemas en geometría computacional y gráficos por ordenador. (Roth, 1982)

Ray-casting maneja el trazado de rayos que intersecan una superficie. Los rayos se trazan desde un observador hacia un plano, pudiendo de esta forma determinar las superficies visibles en la escena. (Roth, 1982)

Se puede manejar el trazado de rayos para determinar si un punto se encuentra dentro de un polígono irregular. En primer lugar, se debe trazar una línea desde el punto de coordenadas que se va a analizar. El rayo debe ser paralelo al eje X o Y. La línea trazada se interseca con los lados del polígono. Si el número de intersecciones es impar, el punto está dentro del polígono. Caso

contrario, si el número de intersecciones es par, entonces el punto se encuentra fuera del polígono. (AssemblySys dataServices, 2013)



**Figura 9.** Trazado de rayos sobre el polígono  
(AssemblySys dataServices, 2013)

## 2.4 Análisis y evaluación del riesgo en el sector del Valle de los Chillos en caso de una posible erupción del volcán Cotopaxi

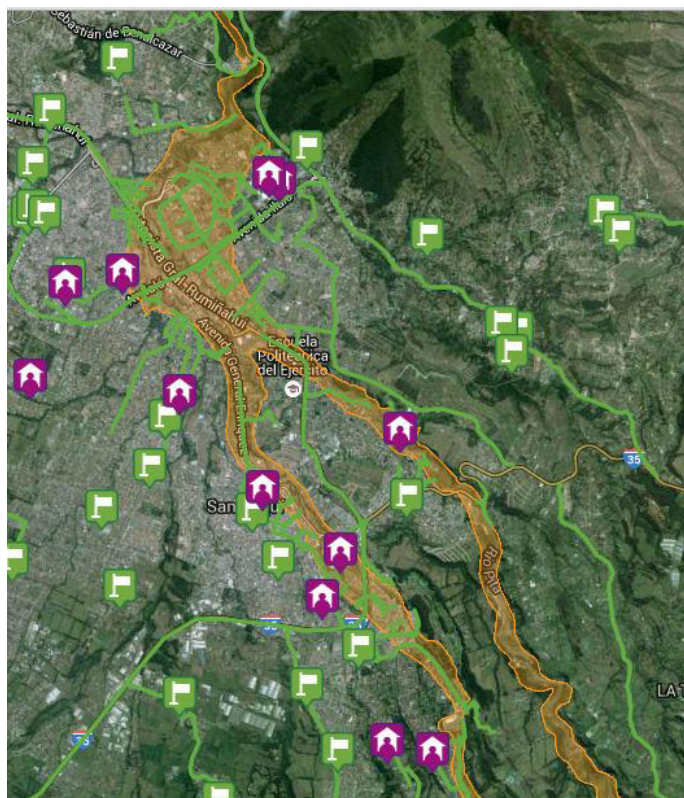
### 2.4.1 Zonas de riesgo en el cantón Rumiñahui

Una posible erupción del volcán Cotopaxi no solo afectaría a las zonas cercanas en la provincia de Latacunga sino que podría provocar el aumento de caudales de los ríos Santa Clara y Pita los cuales nacen del volcán y su trayecto pasa por varias zonas dentro de la Provincia de Pichincha.

El paso de lahares podría afectar varias zonas residenciales dentro de los cantones Mejía, Rumiñahui y del Distrito Metropolitano de Quito. El Ministerio Coordinador de Seguridad y la Secretaría de gestión de Riesgos han realizado un análisis de las posibles zonas que podrían ser afectadas a partir de las condiciones geográficas y de los reportes históricos que describen lo ocurrido durante erupciones pasadas (Ministerio Coordinador de Seguridad, 2015).

El Valle de los Chillos es una de las zonas más afectadas por el paso de lahares debido a que justamente en esta zona se unen los ríos Santa Clara,

Pita y San Pedro, generando un considerable aumento de caudales que pueden llegar a aproximadamente 15 metros de altura según los estudios realizados (Secretaría de Gestión de Riesgos, 2015).

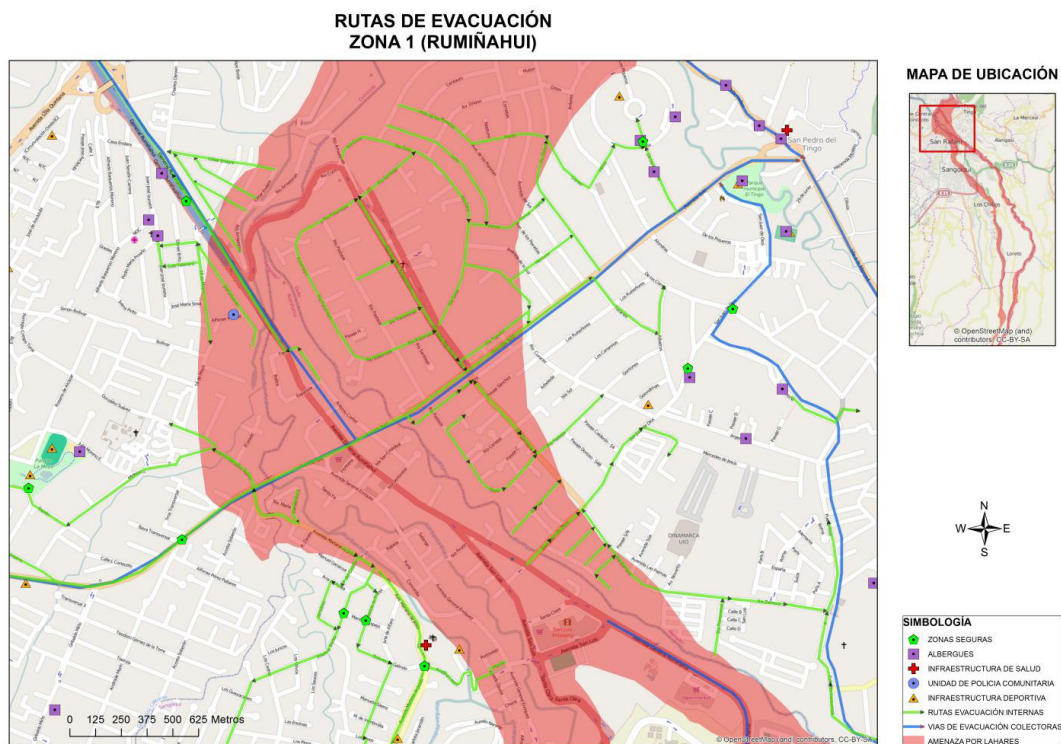


**Figura 10.** Zonas afectadas por los lahares en el Valle de los Chillos (Secretaría de Gestión de Riesgos, 2015)

Debido a que el paso de lahares cubre una zona extensa dentro del cantón Rumiñahui, el municipio y la central de gestión de riesgos han dividido el cantón en subzonas para poder manejar la evacuación de los residentes de forma más organizada (Ministerio Coordinador de Seguridad, 2015). Las zonas más afectadas en el cantón Rumiñahui son las siguientes:

- **Zona 1:** Esta zona es la más afectada debido a que aquí se juntan los ríos Santa Clara y Pita, y más adelante se junta el río San Pedro. Los lugares más afectados que pertenecen a esta zona son los siguientes: Armenia, Armenia 1. Colegio Farina, el Triángulo, Urb. Roble Antigo, Urb. Chiriboga,

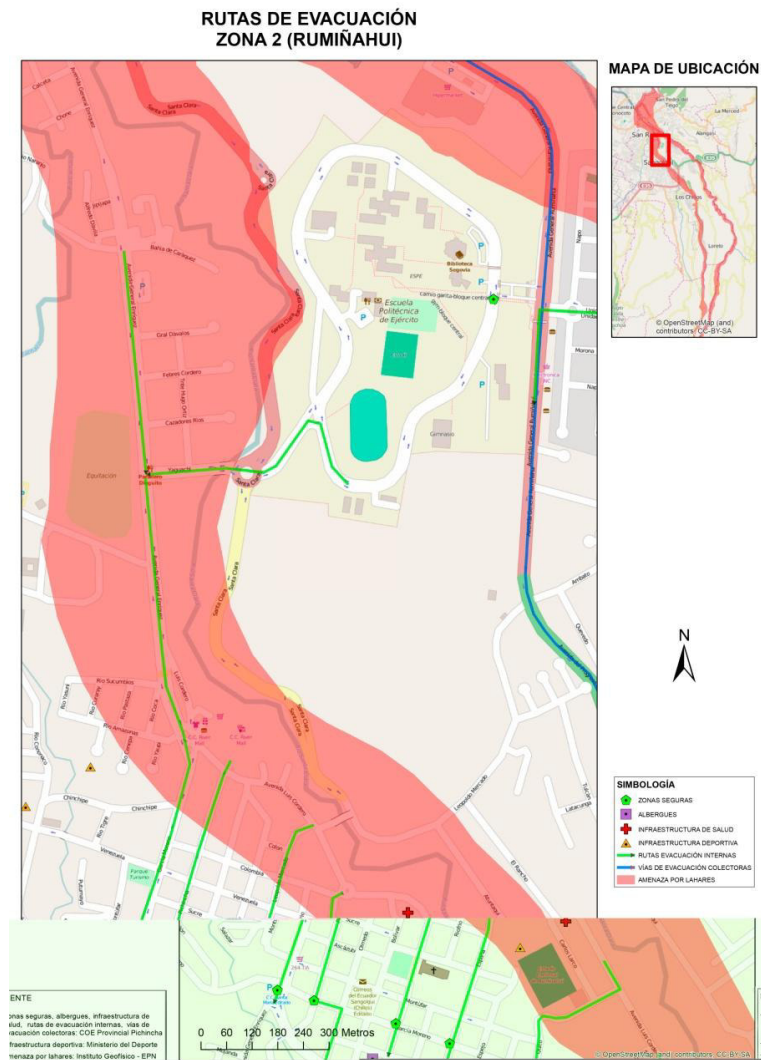
Condominios San Rafael, Conjunto el Remanso, San Rafael, Hogar de Ancianos "San Rafael", Conjunto Valle Verde, Playa Chica, Centro Comercial San Luis. Los refugios más cercanos dentro de esta zona son los siguientes: Sain Dominic School, Colegio Franz Schubert, Hogar Juvenil Gonzales, Parque la Moya, Hogar Juvenil Domingo, Unidad Educativa las Américas del Valle, Fundación Sagrado Corazón de Jesús, jardín Gereátrico Los Años de Oro, Hostería Mirasierra, APCH (Colegio Ángel Polibio) Centro Recreacional Los Cactus, Centro de la Unidad de Justicia Ilalo, Comité Central Urbanización Capelo, Complejo de la Empresa Omnibus, Centro San Juan de Dios, Casa Comunal San Carlos, Unidad Educativa San Esteban del Valle (Secretaría de Gestión de Riesgos, 2015), ( Secretaría de Seguridad y Gobernabilidad - Municipio de Quito, 2015),



**Figura 11.** Zona 1 en el Cantón Rumiñahui.  
(Ministerio Coordinador de Seguridad, 2015)



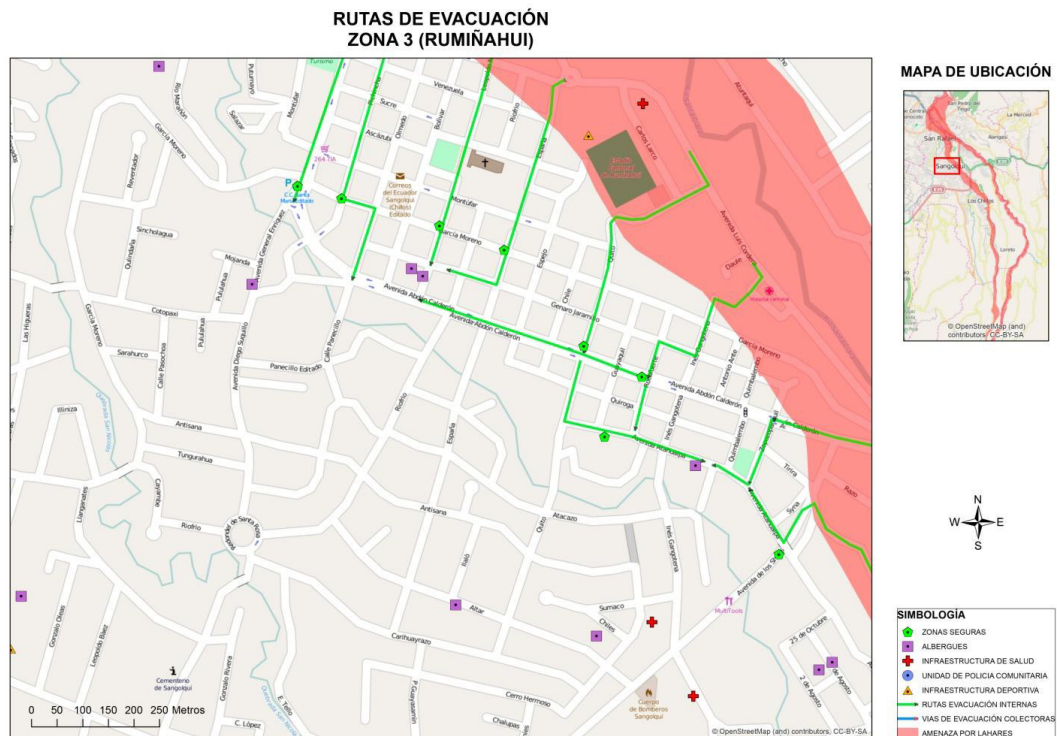
- Zona 2:** Por esta zona pasa el río Santa Clara. Los lugares más afectados dentro de esta zona son los siguientes: Urbanización San Luis, Urbanización Yahuachi, Conjunto Alborada, Barrio Santa Bárbara, River Mall, Av. Luis Cordero, Redondel del Aguacate. Los refugios más cercanos dentro de esta zona son los siguientes: Escuela Vicente Aguirre, Unidad Multiusos de la Policía Metropolitana, Comité Central urbanización Capelo, Universidad de las Fuerzas Armadas (ESPE). (Secretaría de Gestión de Riesgos, 2015),.



**Figura 12.** Zona 2 en el Cantón Rumiñahui  
(Ministerio Coordinador de Seguridad, 2015)



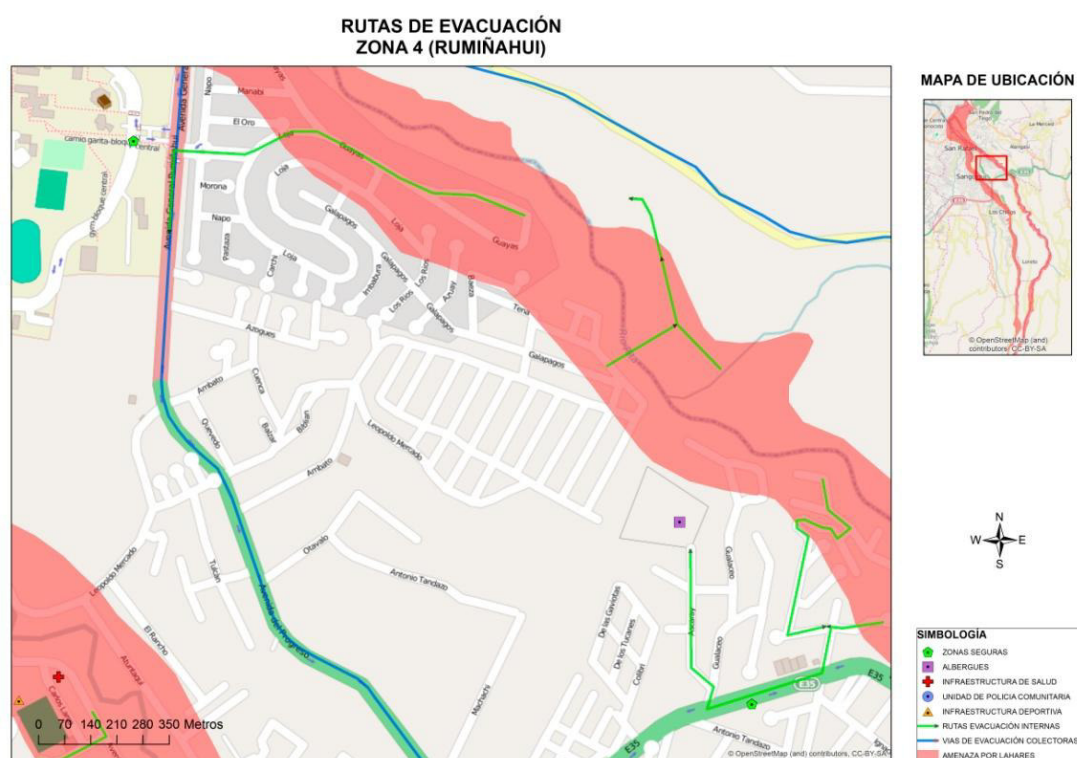
- Zona 3:** Por esta zona continúa pasando el río Santa Clara. Los lugares más afectados dentro de esta zona son los siguientes: Barrio El Progreso, Barrio Luis Cordero, Empresa Eléctrica, Hospital Sangolquí, Destacamento de la Policía, Av. Luis Cordero. Los refugios más cercanos dentro de esta zona son los siguientes: Escuela Juan Montalvo sección primaria, Unidad Educativa Juan Montalvo. (Secretaría de Gestión de Riesgos, 2015), (Ministerio Coordinador de Seguridad, 2015). .



**Figura 13.** Zona 3 en el Cantón Rumiñahui  
(Ministerio Coordinador de Seguridad, 2015)

- Zona 4:** Por esta zona pasa el río Pita. Los lugares más afectados dentro de esta zona son los siguientes: Conjunto san Nicolás, Conjunto el valle, Conjunto Aguirre Ayala, Urbanización la Colina (parte Baja), Ciudadela El

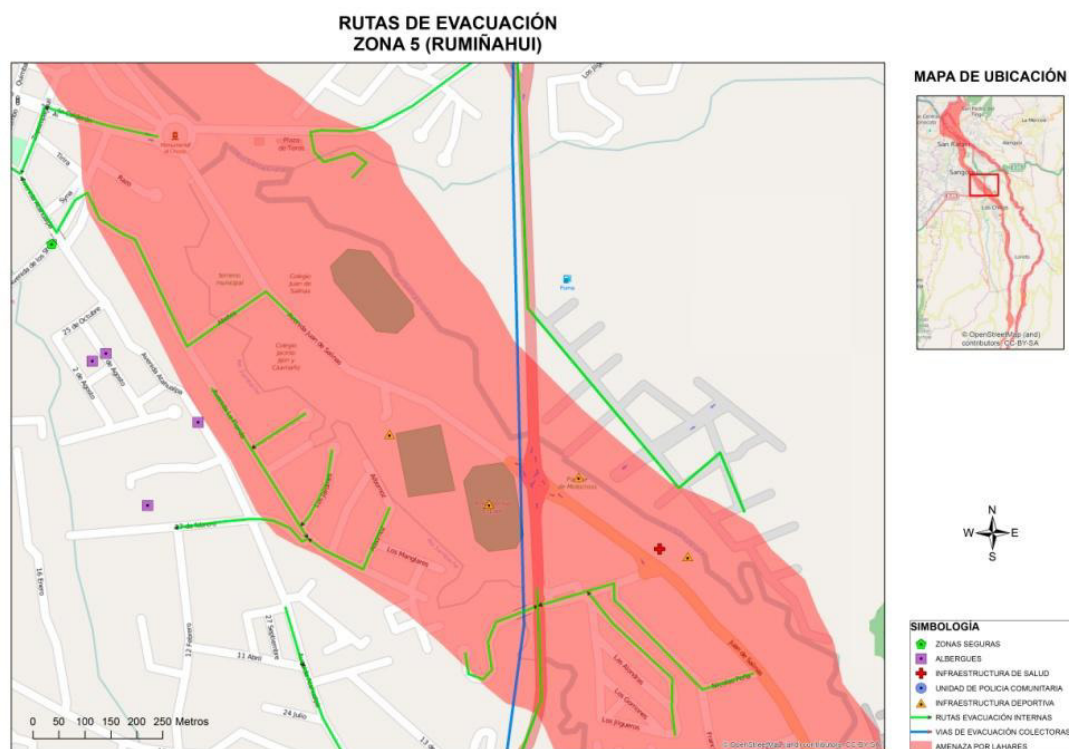
Ejército, Comuna Cashapamba. Los refugios más cercanos dentro de esta zona son los siguientes: Liceo del valle, Centro Educativo Cotogchoa, Universidad de las Fuerzas Armadas ESPE, Complejo Ministerio del Deporte. (Secretaría de Gestión de Riesgos, 2015), (Ministerio Coordinador de Seguridad, 2015).



**Figura 14.** Zona 4 en el Cantón Rumiñahui  
(Ministerio Coordinador de Seguridad, 2015)

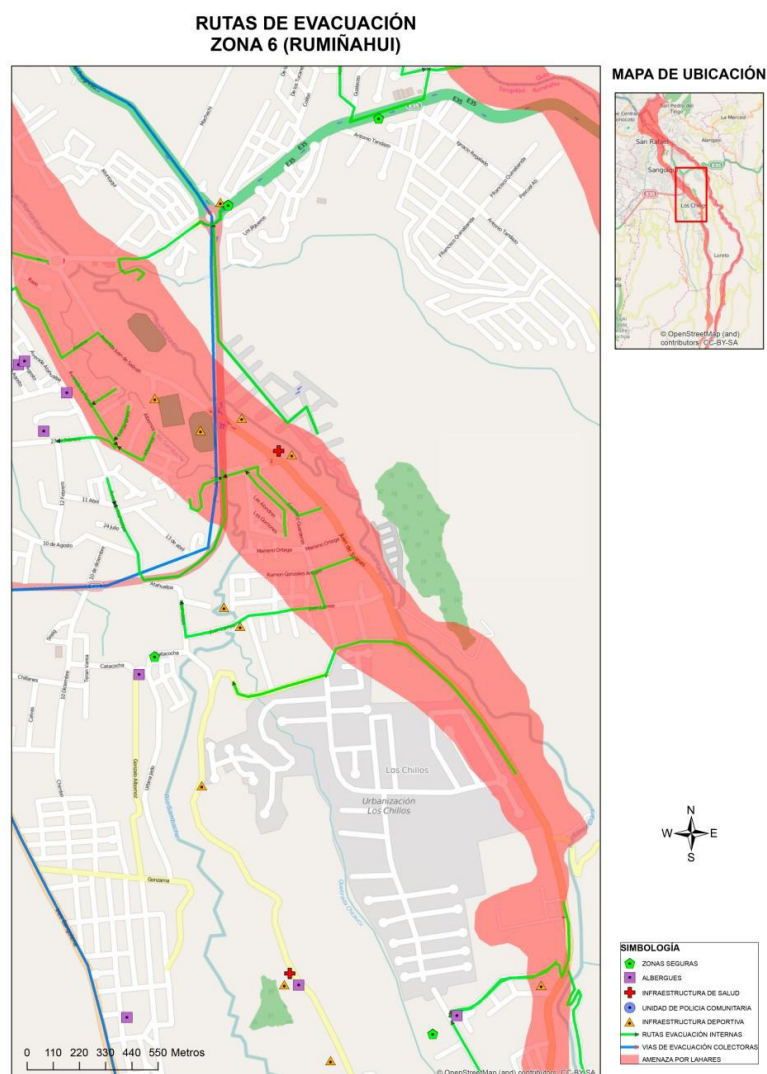
- **Zona 5:** Por esta zona continúa pasando el río Santa Clara. Los lugares más afectados dentro de esta zona son los siguientes: Redondel el Choclo, Barrio la Florida, Barrio Los pinos, Urbanización Olmedo Andrade, Urbanización Los jardines, Urbanización Mag. Los refugios más cercanos dentro de esta zona son los siguientes: Instituto Técnico Rumiñahui, Colegio

Juan de Salinas. (Secretaría de Gestión de Riesgos, 2015), (Ministerio Coordinador de Seguridad, 2015).



**Figura 15.** Zona 5 en el Cantón Rumiñahui  
(Ministerio Coordinador de Seguridad, 2015)

- **Zona 6:** Por esta zona continúa pasando el río Santa Clara. Los lugares más afectados dentro de esta zona son los siguientes: Urbanización Cedapac, barrio Selva Alegre, Conjunto Alcantar, Capilla Chillo Compañía, Enkador. Los refugios más cercanos dentro de esta zona son los siguientes: Unidad Educativa Marquez de Selva Alegre, Fundación General Ecuatoriana, Centro Educativo Leonidas García, Hacienda San Francisco. (Secretaría de Gestión de Riesgos, 2015), (Ministerio Coordinador de Seguridad, 2015).

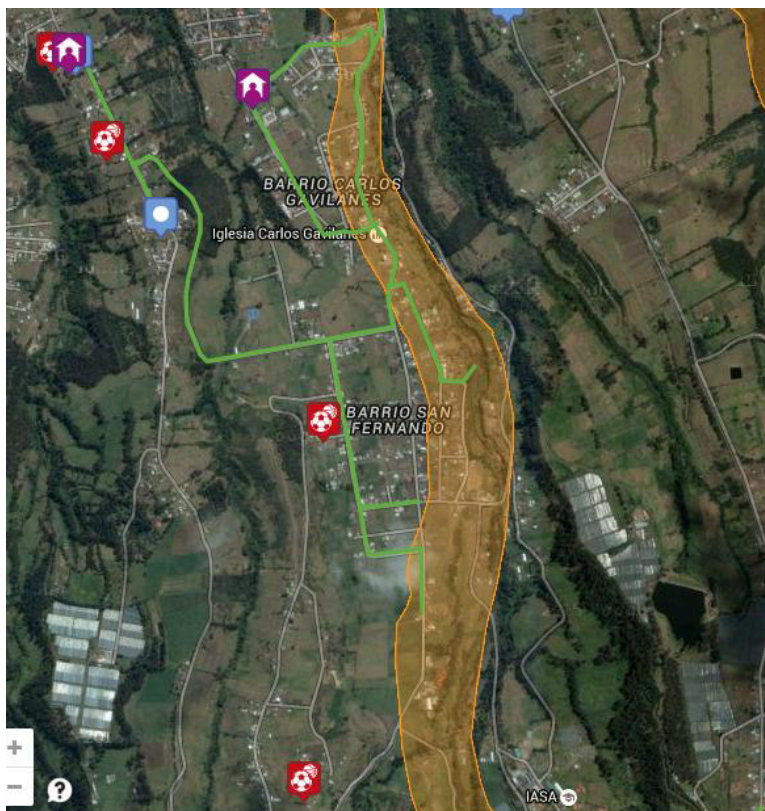


**Figura 16.** Zona 6 en el Cantón Rumiñahui  
(Ministerio Coordinador de Seguridad, 2015)

- **Zona 7:** Por esta zona continúa pasando el río Santa Clara. Los lugares más afectados dentro de esta zona son los siguientes: Barrio Carlos Gavilanes, barrio Luz de América, Barrio san Fernando, Central Hidroeléctrica Los Chillos. Los refugios más cercanos dentro de esta zona son los siguientes: Unidad educativa Galileo Galilei, Escuela José María



Larco, Junta Administrativa de Agua Potable, Casa Comunal Jatapumgo.  
(Secretaría de Gestión de Riesgos, 2015).



**Figura 17.** Zona 7 en el Cantón Rumiñahui  
(Secretaría de Gestión de Riesgos, 2015)

### 2.6.2 Medidas preventivas ante una posible erupción

Tanto el Ministerio Coordinador de Seguridad, como los Municipios de Rumiñahui y del Distrito Metropolitano de Quito, han realizado diversas actividades para prevenir a los ciudadanos en caso de una emergencia dentro del Valle de los Chillos ( Secretaría de Seguridad y Gobernabilidad - Municipio de Quito, 2015). Desde que aumentó la actividad del volcán Cotopaxi ha

existido preocupación por parte de los residentes en las zonas de mayor riesgo. Es por eso que se han tomado las siguientes medidas:

- En primer lugar se ha puesto a disposición una página web donde se presentan informes de la actividad del volcán por parte del Instituto Geofísico de la Escuela Politécnica Nacional. También se brinda información de mapas de las zonas más afectadas, se da recomendaciones de lo que se debe realizar en caso de emergencias y de cómo se debe armar un kit de emergencias. Se accede a la página por medio del enlace [www.volcancotopaxi.com](http://www.volcancotopaxi.com). (Ministerio Coordinador de Seguridad, 2015).



**Figura 18.** Informe diario de la actividad del volcán brindado por el Instituto Geofísico

(Volcán Cotopaxi Pase o no Pase, 2015)

- Se han realizado charlas y ferias de prevención en distintos puntos de encuentro donde se ha informado acerca de los planes familiares de emergencia, puntos de encuentro, refugios y rutas de evacuación. También

se han coordinado los simulacros que están por realizarse. En estas charlas se han involucrado tanto las principales autoridades de los organismos encargados como los ciudadanos que se encuentran en las zonas de peligro. ( Secretaría de Seguridad y Gobernabilidad - Municipio de Quito, 2015).



**Figura 19.** Charlas con los ciudadanos acerca de los posibles riesgos  
(El Comercio, 2015)

- El municipio de Quito ha instalado sirenas en varios lugares dentro del Valle de los Chillos. Se han dado indicaciones a los residentes de cómo debe sonar la alarma durante simulacros y cómo debe sonar en caso de que se deba realizar una evacuación. En las principales carreteras se han instalado letreros informativos y se han pintado señales indicando qué ruta se debe seguir en caso de una posible emergencia.



**Figura 20.** Señalización de rutas de escape sobre las calles del sector  
(El Comercio, 2015)

- Se han realizado simulacros en cada una de las subzonas dentro del cantón. La participación de los ciudadanos ha sido positiva y todo se lo ha realizado en orden gracias a la colaboración de las brigadas policiales,

cuerpo de bomberos y de las fuerzas armadas. El día 27 de noviembre se realizó un simulacro simultáneo a nivel nacional. (El Comercio, 2015)



**Figura 21.** Simulacro de evacuación de estudiantes  
(El Comercio, 2015)

- En los principales medios de comunicación se han dado recomendaciones de lo que se debe realizar en caso de emergencia y se ha publicado afiches indicando los pasos que se deben seguir, los mapas por donde pasan los lahares y las rutas hacia los refugios cercanos. (Ministerio Coordinador de Seguridad, 2015)



**Figura 22.** Afiche describiendo la ruta de evacuación en el Centro Comercial San Luis

- Los mapas oficiales que indican por donde pasan los lahares se han publicado en la red por medio de la herramienta Google Maps donde además se pueden visualizar las rutas de escape, los refugios y zonas seguras. Se puede acceder a este mapa por medio de ordenadores y también por dispositivos móviles (Secretaría de Gestión de Riesgos, 2015).
- El municipio de Rumiñahui publicó en Google Play una aplicación que permite al usuario visualizar las zonas de paso de lahares de forma general,



se brinda información de las medidas preventivas y se da recomendaciones de cómo armar el kit de emergencias. (Ministerio Coordinador de Seguridad, 2015)

- Se busca realizar obras de mitigación para frenar el impacto de los lahares. Entre esas obras se busca construir muros de contención en los ríos Santa Clara y Pita, sin embargo, existe preocupación por parte de la población porque se cree que estos muros aumentarían el caudal del río y ampliarían el rango de las zonas de alto riesgo. (El Comercio, 2015)
- El Municipio Metropolitano de Quito está ejecutando seis obras de mitigación los cuales son tres pasos elevados y tres estaciones de bombeo los cuales garantizarán el abastecimiento de agua potable ante una posible erupción. (El Comercio, 2016)

Los principales organismos se han encargado de dar recomendaciones en caso de que se presente una emergencia. (Ministerio Coordinador de Seguridad, 2015). Si la persona se encuentra en una zona de riesgo deberá tomar las siguientes medidas preventivas:

- Mantener la calma para poder actuar con prudencia
- Evitar las zonas de riesgo.
- Diseñar un plan de evacuación por medio de simulacros y probando el tiempo de reacción al momento de evacuar.
- Ubicar el sitio seguro más cercano.

Los ciudadanos además deberán armar un kit de emergencias el cual debe estar conformado por:

- Documentos personales
- Alimentos no perecibles
- Aguas o bebidas hidratantes
- Radio a pilas
- Botiquín de primeros auxilios
- Artículos sanitarios e higiénicos

- Fósforos, velas o una linterna portátil
- Silbato o pito
- Ropa adicional
- Dinero en efectivo
- Gafas y mascarillas
- Navaja Multiusos.

El botiquín de primeros auxilios deberá poseer:

- Algodón
- Agua Oxigenada
- Vendas y Curitas
- Tijeras
- Analgésicos
- Medicamentos

De acuerdo a los reportes del Instituto Geofísico, la Secretaría de Gestión de Riesgos está encargada de declarar si existe un cambio de alerta. Dependiendo de esto, los municipios de las zonas afectadas y las principales brigadas se encargarán de tomar las acciones necesarias para resguardar la vida de los habitantes. ( Secretaría de Seguridad y Gobernabilidad - Municipio de Quito, 2015).

Como en cualquier tipo de emergencia por fenómenos naturales, se han presentado tres tipos de alertas (Ministerio Coordinador de Seguridad, 2015):

- **Alerta Amarilla:** Se declara este tipo de alerta cuando se ha determinado la existencia de actividad volcánica irregular. Las personas que se encuentran en zonas de riesgo deben revisar su plan familiar de emergencia y deben identificar las rutas de escape. También deben tener listo su kit de emergencia y deben estar atentos a cualquier cambio de alerta.
- **Alerta Naranja:** Se declara esta alerta cuando se ha detectado la erupción inminente del volcán. En este caso se realiza la evacuación de las personas

ubicadas en zonas de mayor peligro. Las personas deben seguir las rutas de evacuación hacia los sitios seguros con calma.

- **Alerta Roja:** Se declara esta alerta cuando los lahares ya están pasando por las zonas de riesgo. La población evacuada debe permanecer en el sitio seguro con calma, siguiendo las recomendaciones de las entidades de respuesta.

Es necesario que los ciudadanos tengan presente estas medidas preventivas, participen en simulacros, identifiquen los refugios y sitios seguros, y estén completamente preparados a fin de poder evacuar de forma prudente y ordenada en caso de una posible erupción. Se desconoce el momento exacto en que se llegue a presentar una emergencia, sin embargo, los ciudadanos deben convivir con la amenaza del volcán y estar siempre atentos ante cualquier actividad.



**Figura 23.** Descripción de las acciones a seguir durante las Alertas (Ministerio Coordinador de Seguridad, 2015):

### 2.6.3. Lista de refugios y sitios seguros en el Valle de los Chillos

A lo largo de las zonas afectadas, los principales organismo han designado varios lugares hacia dónde se puede evacuar en caso de una posible emergencia. Los lugares designados pueden ser albergues, centros educativos o puntos de encuentro como parques o complejos deportivos ( Secretaría de Seguridad y Gobernabilidad - Municipio de Quito, 2015), (Secretaría de Gestión de Riesgos, 2015). En el Valle de los Chillos se han designado los siguientes refugios mostrados en la tabla 1:

**Tabla 1**

Lista de Refugios en el Valle de los Chillos

<b>NOMBRE</b>	<b>LONGITUD</b>	<b>LATITUD</b>	<b>DESCRIPCIÓN</b>
<b>ZONA 1</b>			
Unidad Educativa Saint Dominic School	-0.2789	-78.4625	Refugio
Colegio Franz Schubert	-0.2778	-78.4659	Refugio
Colegio Gonzaga	-0.2728	-78.4558	Centro Educativo
Hogar Juvenil Savio Gonzales	-0.2993	-78.4661	Refugio
Parque La Moya	-0.3009	-78.4736	Refugio
Hogar Juvenil Domingo	-0.2985	-78.4682	Sitio seguro (refugio temporal)
Colegio Nacional Conocoto	-0.2987	-78.472	Centro Educativo
Unidad educativa Las Américas del Valle	-0.2877	-78.4676	Albergue
Fundación Sagrado Corazón de Jesús	-0.2903	-78,468	Albergue
Jardín Geriátrico Los Años de Oro	-0.2895	-78.4681	Albergue
Hostería Mirrasierra	-0.2862	-78.4467	Refugio
Apch	-0.2873	-78.4455	Refugio

 continúa

Centro Cactus	Recreacional	Los	-0.2855	-78.445	Sitio seguro (refugio temporal)	
Colegio Cuello	Alejandro	Andrade	-0.2831	-78.4418	Centro Educativo	
Centro de la Ilalo	Unidad	Justicia	-0.287	-78.4402	Albergue	
<b>ZONA 2</b>						
Comité Central Urb.	Capelo		-0.3091	-78.4557	Sitio seguro (refugio temporal)	
Complejo Omnibus	de la Empresa		-0.2933	-78,436	Albergue	
Centro San Juan de Dios			-0.2968	-78.4436	Albergue	
Casa comunal San Carlos			-0.2975	-78.4403	Albergue	
Unidad Educativa del Valle	San Esteban		-0.2992	-	Albergue	
				78.44183		
<b>ZONA 2</b>						
Escuela Vicente Aguirre			-0.3153	-78.4586	Refugio	
Unidad Multiuso de la Metropolitana	de la Policía		-0.3123	-78.476	Refugio	
Colegio Carlos Cadena			-0.3185	-78.4606	Centro Educativo	
San Pedro Claver			-0.3252	-78.4626	Centro Educativo	
<b>ZONA 3</b>						
Escuela Montalvo	Primaria	Juan	-0.328	-78.4476	Refugio	
Escuela Juan Montalvo			-0.3309	-78.4489	Centro Educativo	
<b>ZONA 5</b>						
Instituto Rumiñahui	Superior	Técnico	-0.3393423	-	Refugio	
				78.44021		
Colegio Juan de Salinas			-0.3373	-78.4457	Centro Educativo	
<b>ZONA 4</b>						
Liceo del Valle			-0.3242	-78.4305	Refugio	
Colegio Cotogchoa			-0.329	-78.4287	Centro Educativo	
Universidad de las Armadas	de las Fuerzas		-0.3144	-78.4437	Centro Educativo	
Complejo Ministerio del Deporte			-0.3075	-78.435	Albergue	
<b>ZONA 6</b>						
U.E. Marquez de Selva Alegre			-0.3428	-78.4397	Refugio	



continúa

Fundación General Ecuatoriana	-0.350534	-	78.43382	Sitio seguro (refugio temporal)
Colegio Leonidas garcia	-0.3489	-78.435		Centro Educativo

### ZONA 7

Unidad Educativa Galileo Galilei	-0.3627	-78.4252		Refugio
Casa Comunal Jatupumgo	-0.3615	-78.4312		Refugio
Hacienda San Francisco	-0.36	-78.4169		Sitio seguro (refugio temporal)
Escuela José María Larco	-0.3669	-78.4282		Sitio seguro (refugio temporal)
Junta Administrativa Agua Potable	-0.3616	-78.4309		Sitio seguro (refugio temporal)

**Fuentes: (Secretaría de Gestión de Riesgos, 2015), ( Secretaría de Seguridad y Gobernabilidad - Municipio de Quito, 2015)**

## CAPÍTULO III

### CÓDIGO DE LA APLICACIÓN

#### 3.1 Descripción del programa en Android

##### 3.1.1 Menú principal

Al ejecutar la aplicación se muestra un menú principal el cual permite acceder a las distintas funcionalidades del programa. Por medio de la pantalla táctil, al dar clic en cada opción del menú se abre una nueva actividad.

En el código principal se construye la clase *MainActivity* en donde se implementa una lista de opciones en la interfaz inicial del programa. En cada opción de la lista se debe visualizar el título de la actividad, una descripción y un ícono tal como se muestra en la figura 24.



**Figura 24.** Lista de opciones ubicados en el menú principal

Los títulos de las opciones del menú y las descripciones de cada una de las opciones se almacenan en arreglos de Strings. En un arreglo se almacenan los íconos asignados en cada opción del menú. Por medio del método

*R.drawable* se puede obtener una imagen que se asigna en cada ítem del listado.

```
public class MainActivity extends AppCompatActivity {
    private String menuCotopaxi[]=new String[]{"Ir a Mapa","Alertas volcánicas","Medidas Preventivas","Enlaces Informativos",
    ,"kit de Emergencia","Botiquín Primeros Auxilios","Llamadas de Emergencia","Manual de Usuario"};
    private String descrMenu[] = new String[]{"verificar ubicación, Zonas de Riesgo, Refugios y Rutas de Escape"
    ,"Conocer cada Alerta existente", "Conocer las Medidas Preventivas ante una Erupción",
    ,"Ir hacia las páginas oficiales Informativas", "Elementos ante una Emergencia", "Contenido De un Botiquín"
    ,"Se marcan de manera automática los números de Emergencia", "Conocer el uso correcto de la Aplicación"};
    private Integer[] imgId={
        R.drawable.mapa,
        R.drawable.alertas,
        R.drawable.preventivas,
        R.drawable.enlace,
        R.drawable.kitemerg,
        R.drawable.botiquin,
        R.drawable.llamdaemergencia,
        R.drawable.manual
    };
};
private ListView lista;
```

**Figura 25.** Creación del menú principal en la Clase MainActivity

Los valores agrupados en los arreglos son manejados por un adaptador que permite gestionar y ordenar los datos que se mostrarán en la lista de forma vertical sobre el layout de la aplicación.

```
public class AdaptadorMenuInicial extends ArrayAdapter<String>{
    private final Activity context;
    private final String[] itemname;
    private final String[] subitem;
    private final Integer[] integers;

    public AdaptadorMenuInicial(Activity context, String[] itemname, String[] subitem, Integer[] integers) {
        super(context, R.layout.lista_principal, itemname);
        // TODO Auto-generated constructor stub

        this.context=context;
        this.itemname=itemname;
        this.subitem = subitem;
        this.integers=integers;
    }

    public View getView(int posicion,View view, ViewGroup parent){

        LayoutInflater inflater=context.getLayoutInflater();
        View rowview=inflater.inflate(R.layout.lista_principal,null,true);

        TextView txtTitle = (TextView) rowview.findViewById(R.id.texto_principal);
        ImageView imageView = (ImageView) rowview.findViewById(R.id.icon);
        TextView etxDescripcion = (TextView) rowview.findViewById(R.id.texto_secundario);

        txtTitle.setText(itemname[posicion]);
        imageView.setImageResource(integers[posicion]);
        etxDescripcion.setText(subitem[posicion]);
        //etxDescripcion.setText("descripción "+itemname[posicion]);

        return rowview;
    }
}
```

**Figura 26.** Adaptador del menú principal

Por medio del método *OnClick*, al dar clic en cada ítem del menú se genera un nuevo atributo del tipo *intent* el cual permite iniciar una nueva actividad. Por ejemplo, al dar clic en la opción "Ir a Mapa", se inicia la actividad



*MapsActivity* donde se abre la actividad que permite visualizar el mapa en Google Maps.

```

AdaptadorMenuInicial adapter=new AdaptadorMenuInicial(this,menucotopaxi, descrMenu, imgid);
lista=(ListView)findViewById(R.id.mi_lista);
lista.setAdapter(adapter);
lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        String selecteditem = menuCotopaxi[+position];
        Toast.makeText(getApplicationContext(), selecteditem, Toast.LENGTH_SHORT).show();
        if (selecteditem=="Ir a Mapa"){
            Intent i = new Intent(MainActivity.this,MapsActivity.class);
            startActivity(i);
        }else
        if (selecteditem=="Alertas volcánicas"){
            Intent i = new Intent(MainActivity.this,Alertasvolcanicas.class);
            startActivity(i);
        }else
        if (selecteditem=="Medidas Preventivas"){
            Intent i = new Intent(MainActivity.this,MedidasPreventivas.class);
            startActivity(i);
        }else
        if (selecteditem=="Enlaces Informativos"){
            Intent i = new Intent(MainActivity.this,EnlacesInfo.class);
            startActivity(i);
        }else
        if (selecteditem=="Kit de Emergencia"){
            Intent i = new Intent(MainActivity.this,KitEmergencia.class);
            startActivity(i);
        }else
        if (selecteditem=="Botiquín Primeros Auxilios"){
            Intent i = new Intent(MainActivity.this,Botiquin.class);
            startActivity(i);
        }else
        if (selecteditem=="Llamadas de Emergencia"){
            Intent i = new Intent(MainActivity.this,LlamadasEmergencia.class);
            startActivity(i);
        }
    }
});

```

**Figura 27.** Código de selección de la función de la interface al seleccionar una opción

### 3.1.2 Implementación de Google Maps en Android

Android Studio permite el desarrollo de aplicaciones para visualizar mapas por medio de la herramienta Google Maps a través de un entorno de desarrollo más sencillo de manejar.

Para empezar a desarrollar una aplicación Google Maps API V2 por medio de Android Studio se crea un nuevo proyecto. Se ingresa el nombre de la aplicación y el dominio de la compañía (en este caso es ESPE). Luego se selecciona la opción *Google Maps Activity*.

Después de crear el proyecto, se debe generar el API key de Google el cual permite que los mapas funcionen en la aplicación. Para realizar esta generación se debe usar el link proporcionado por el programa en Android

Studio. Se debe copiar el link desde el archivo `google_Maps_api.xml` hacia la barra de direcciones del navegador. En el navegador se abre la Consola de Desarrolladores de Google. Se deben seguir las instrucciones para crear un nuevo proyecto. Luego, se genera un API key para el proyecto de la consola. Por último se copia la API Key generada y se la pega en el elemento `<string>` en el archivo `google_Maps_api.xml` (Google Developers, 2016).

Otra forma para generar el API key de Google es instalando el SDK Manager. Por medio de este programa se debe crear un nuevo proyecto. En la sección servicios, se selecciona la opción “Google Maps API v2”. Al seleccionar la opción *API Access* se debe seleccionar *Create New Android Key*. Para poder generar la llave se debe obtener una clave SHA-1 (Secure Hash Algorithm) brindado por el SDK de Java. (Latorre, 2013).

Al crear el proyecto, se genera el archivo `activity_Maps.xml` que define el layout de la aplicación. También se genera el archivo `MapsActivity.java` el cual define la actividad del mapa en la aplicación.

En el código generado en Java se crea la clase `MapsActivity` la cual implementa la interfaz `OnMapClickListener` que permite realizar acciones cuando se manipula el mapa por medio de la pantalla táctil. Se crea un atributo privado del tipo `GoogleMap` llamado `Mmap`. Este atributo puede ser utilizado para dibujar objetos sobre el mapa o realizar acciones sobre el mismo.

```
public class MapsActivity extends AppCompatActivity implements
    GoogleMap.OnMapClickListener {

    private GoogleMap mMap;
```

**Figura 28.** Clase `MapsActivity`

### 3.1.2.1 Configuración de la apariencia del mapa

Cuando se trabaja sobre un mapa por medio de Google Maps, se tiene la posibilidad de cambiar la apariencia del mismo de acuerdo a los elementos que

se quieren visualizar. El mapa normal permite visualizar por dónde pasan las carreteras y avenidas principales. El mapa satelital permite visualizar el mapa real, mostrando detalladamente las ubicaciones por medio de imágenes tomadas por satélites. El mapa terrestre muestra con mayor detalle las zonas geográficas y las rutas. Por último, el mapa híbrido es una combinación del mapa satelital con el mapa normal, donde se detalla de mejor manera las calles y avenidas principales sobre las imágenes satelitales. Dentro de la aplicación se muestra por defecto la apariencia híbrida del mapa, pero por medio de un menú desplegable se puede seleccionar la apariencia del mapa. Se genera una adaptador para poder gestionar los valores mostrados en la lista desplegable y a través de la pantalla táctil se cambia la apariencia al seleccionar cada opción.

```
String []opciones={"Híbrido","Normal","terrestre","satelital"};
ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,R.layout.spinner_mapa, opciones);
spinner.setAdapter(adapter);
spinner.setOnItemSelectedListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        if (position == 0){
            mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
        } else if (position == 1){
            mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
        } else if (position == 2){
            mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
        } else if (position == 3){
            mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
        }
    }
});
@Override
public void onNothingSelected(AdapterView<?> parent) {
}
});
```

**Figura 29.** Método que permite desplegar una lista para cambiar la apariencia del mapa

### 3.1.2.2 Generación de marcadores

En un mapa generado por Google Maps se pueden colocar marcadores sobre puntos de coordenadas específicos. Sobre una ubicación establecida se inserta un ícono que permite marcar el lugar y dar una descripción del mismo (Google Developers, 2016). Este ícono puede ser una imagen por defecto o se puede exportar de un archivo externo.

Sobre el mapa se añadirán marcadores sobre varios puntos de coordenadas donde se ubican los refugios o sitios seguros en el sector del Valle de los Chillos.

Para poder crear marcadores se debe llamar a la librería `com.google.android.gms.maps.model.Marker`. Posteriormente se deben inicializar los puntos de coordenadas en donde se colocarán los marcadores al principio del código. Estos puntos son atributos privados finales los cuales son del tipo `LatLng`. Estos valores se inicializan con un valor de latitud y longitud como se muestra a continuación:

```
private final LatLng REF1 = new LatLng(-0.2789, -78.4625);
private final LatLng REF2 = new LatLng(-0.2778, -78.4659);
private final LatLng REF3 = new LatLng(-0.2728, -78.4558);
```

**Figura 30.** Inicialización de puntos de coordenadas

Posteriormente se generan los marcadores sobre el mapa por medio de la función `mMap.addMarker` en donde se debe ingresar el punto de coordenadas, el título del marcador y una descripción.

```
mMap.addMarker(new MarkerOptions().position(REF1).title("Saint Dominic
chool").snippet("REFUGIO Centro Educativo")
icon(BitmapDescriptorFactory.fromResource(android.R.drawable.ic_menu_compass)).anchor
0.5f, 0.5f));
```

**Figura 31.** Generación de marcadores sobre el mapa

Por medio de la función `icon` se puede definir la imagen que represente al ícono del marcador. También se puede definir las dimensiones del ícono.

Posteriormente se generan arreglos que contengan todos los puntos de coordenadas de los marcadores creados y estos puedan ser agrupados por zona. Estos arreglos permitirán más adelante realizar comparaciones para determinar de forma automática los posibles refugios cercanos dentro de la zona de análisis.

### 3.1.2.3 Generación de Polígonos

Sobre un mapa generado en Google Maps se puede dibujar polígonos los cuales son figuras geométricas conformadas por un número ilimitado de lados. (Google Developers, 2015) En esta aplicación se dibujan polígonos de forma rectangular a fin de poder delimitar las zonas analizadas dentro del Valle de los Chillos.

También se dibujan polígonos irregulares cuyas formas permiten cubrir lo más cerca posible las zonas de mayor riesgo consideradas por el Ministerio Coordinador de seguridad dentro del cantón Rumiñahui.

Adicionalmente, se dibujan polígonos rectangulares invisibles que permiten subdividir las zonas para poder determinar las rutas de escape hacia los refugios de forma más distribuida y definir los sitios seguros más cercanos a las zonas de riesgo.

Sobre el mapa se pueden adaptar la figuras generadas en cualquier superficie. Se puede además definir el color que rellena el polígono, el color de la línea que contiene el polígono y el grosor de la misma, así como asignar un valor de transparencia del color de relleno del polígono y la visibilidad del mismo. Para poder graficar los polígonos se debe llamar a las librerías:

- *com.google.Android.gms.Maps.model.Polygon*
- *com.google.Android.gms.Maps.model.PolygonOptions*

Luego, se debe crear el atributo del tipo *PolygonOptions* en el cual se deben añadir los puntos de coordenadas de los vértices del polígono en orden. Se pueden definir además en el polígono el grosor de la línea que contiene el polígono, el color de la línea y el color de relleno del polígono como se muestra a continuación.

```

PolygonOptions zona2_riesgo1 = new PolygonOptions()
    .add(new LatLng(-0.3100189,-78.4537125),
        new LatLng(-0.32100509999999993,-78.452704),
        new LatLng(-0.3217776,-78.452307),
        new LatLng(-0.3256667,-78.4503016),
        new LatLng(-0.3256667,-78.4441048),
        new LatLng(-0.32497469999999995,-78.4448183),
        new LatLng(-0.3234512,-78.4473985),
        new LatLng(-0.32128399999999996,-78.4485197),
        new LatLng(-0.31975250000000005,-78.4489033),
        new LatLng(-0.31870780000000004,-78.4482797),
        new LatLng(-0.3178025,-78.4477848),
        new LatLng(-0.316204,-78.4476024),
        new LatLng(-0.3155174,-78.4470123),
        new LatLng(-0.31505600000000006,-78.4469989),
        new LatLng(-0.3144043,-78.4476024),
        new LatLng(-0.3126099,-78.4478009),
        new LatLng(-0.31191250000000004,-78.44830510000001),
        new LatLng(-0.3116202,-78.4482005),
        new LatLng(-0.3100055,-78.4494987),
        new LatLng(-0.3100189,-78.4537125));
zona2_riesgo1.strokeWidth(4);
zona2_riesgo1.strokeColor(Color.RED);
zona2_riesgo1.fillColor(Color.TRANSPARENT);
mMap.addPolygon(zona2_riesgo1);

```

**Figura 32.** Generación del polígono sobre el mapa

Por último, se dibuja el polígono sobre el mapa por medio de la función *mMap.addPolygon*.

### 3.1.2.4 Generación de Polilíneas

Sobre el mapa también se pueden dibujar polilíneas, los cuales son un conjunto de líneas rectas continuas que se trazan a través de distintos puntos de coordenadas (Google Developers, 2016). Las polilíneas se implementan sobre el mapa para poder graficar senderos, siendo una herramienta bastante utilizada para trazar caminos en aplicaciones con fines turísticos o deportivos. En este caso se deben trazar polilíneas para poder visualizar las rutas de escape predefinidas de acuerdo a la información brindada por la Central de Gestión de Riesgos.

Al igual que en los polígonos, se puede determinar las características de la polilínea como el color y el grosor. Para añadir una polilínea sobre el mapa se deben invocar las librerías:

- *com.google.Android.gms.Maps.model.Polyline*
- *com.google.Android.gms.Maps.model.PolylineOptions*

Luego, se debe crear un atributo del tipo Polyline donde se deben añadir los puntos de coordenadas que conforman la ruta por medio de la función *mMap.addPolyline*. Se define además el color de línea y su grosor dentro de esta función como se muestra en la siguiente figura:

```
final Polyline ruta_zona1A1 = mMap.addPolyline(new PolylineOptions()
    .add(new LatLng(-0.27825, -78.45130),
        new LatLng(-0.27930, -78.45270),
        new LatLng(-0.28035, -78.45530),
        new LatLng(-0.28000, -78.45605),
        new LatLng(-0.28035, -78.45715),
        new LatLng(-0.27960, -78.45812),
        new LatLng(-0.27950, -78.45820),
        new LatLng(-0.27940, -78.45860),
        new LatLng(-0.27940, -78.45880),
        new LatLng(-0.27972, -78.45960),
        new LatLng(-0.28068, -78.46470),
        new LatLng(-0.27610, -78.46600)).width(5).color(Color.YELLOW));
```

**Figura 33.** Generación de una polilínea sobre el mapa

### 3.1.2.5 Listas desplegables sobre el layout de Google Maps

En la interfaz de Google Maps, se puede añadir una lista desplegable la cual se implementa en la aplicación como un menú que contiene un listado de opciones que permiten realizar algún tipo de acción sobre el mapa.

En este caso se desplegará una lista sobre la interfaz del mapa la cual permitirá mostrar las zonas y refugios ubicados en dichas zonas. Al seleccionar cada subítem de la lista desplegada, se hace un zoom hacia la ubicación del refugio seleccionado dentro de su respectiva zona. Para crear listas desplegables se deben invocar las librerías:

- *Android.widget.ExpandableListAdapter*
- *Android.app.ExpandableListActivity*
- *Android.widget.ExpandableListView*
- *Android.widget.ListView*

Sobre el layout del mapa en Google Maps se coloca un ícono. Al dar clic sobre dicho ícono se expande la lista desplegable de forma vertical. El método

*SetupDrawer* permite que se visualice la lista expandible sobre el layout al dar click en el ícono.

```
private void setUpDrawer() {
    mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);

    mDrawerLayout.setScrimColor(getResources().getColor(android.R.color.transparent));
    mDrawerLayout.setDrawerListener(mDrawerListener);
    explistView = (ExpandableListView) findViewById(R.id.lvExp);
    prepareListData();
    listAdapter = new ExpandableListAdapter(this, listDataHeader, listDataChild);
    // setting list adapter
    explistView.setAdapter(listAdapter);
    fragment = new MercuryFragment();
    getSupportFragmentManager().beginTransaction().replace(R.id.content_frame,
    fragment).commit();
    mDrawerLayout.closeDrawer(explistView);

    explistView.setOnChildClickListener(new
    ExpandableListView.OnChildClickListener() {
        @Override
        public boolean onChildClick(ExpandableListView parent, View v, int
        groupPosition, int childPosition, long id) {
```

**Figura 34.** Implementación de la lista desplegable en el layout de la aplicación

La lista desplegable contiene items y subitems. Los items contienen las zonas de análisis. Los subitems poseen los nombres de los refugios dentro de cada zona. Los valores de los items los cuales son los nombres de las zonas se almacenan por medio del arreglo *ListDataHeader.add()*.

```
listDataHeader = new ArrayList<String>();
listDataChild = new HashMap<String, List<String>>();

// Adding child data
listDataHeader.add("ZONA 1");
listDataHeader.add("ZONA 2");
listDataHeader.add("ZONA 3");
listDataHeader.add("ZONA 4");
listDataHeader.add("ZONA 5");
listDataHeader.add("ZONA 6");
listDataHeader.add("ZONA 7");
//listDataHeader.add("ZONA 8");
```

**Figura 35.** Ingreso de datos en los headers de la lista

Por medio del método *ListDataChild.put* se añaden los valores almacenados en el arreglo *ListDataHeader.add()*.



```
listDataChild.put(listDataHeader.get(0), zona1); // Header, Child data
listDataChild.put(listDataHeader.get(1), zona2);
listDataChild.put(listDataHeader.get(2), zona3);
listDataChild.put(listDataHeader.get(3), zona4);
listDataChild.put(listDataHeader.get(4), zona5);
listDataChild.put(listDataHeader.get(5), zona6);
listDataChild.put(listDataHeader.get(6), zona7);
```

**Figura 36.** Ingreso de los headers en la lista desplegable

Los valores de los subitems se almacenan dentro de ArrayLists conformados por valores del tipo String.

```
// Adding child data
List<String> zona1 = new ArrayList<String>();
zona1.add("Saint Dominic School");
zona1.add("C. Franz Schubert");
zona1.add("C. Gonzaga");
zona1.add("Hogar Juvenil Savio Gonzales");
zona1.add("Parque La Moya");
zona1.add("Hogar Juvenil Domingo");
zona1.add("Colegio Nacional Conocoto");
zona1.add("U.E. Las Américas Del Valle");
zona1.add("F. Sagrado Corazón De Jesús");
zona1.add("J.G. Los Años De Oro");
zona1.add("Hostería MiraSierra");
zona1.add("APCH");
zona1.add("Colegio Ángel Polibio");
zona1.add("Centro Recreacional Los Cactus");
zona1.add("Alejandro Andrade Cuello");
zona1.add("Centro De La Unidad Justicia Ilaló");
zona1.add("Comité Central U. Capelo");
zona1.add("Complejo De La Empresa Omnibus");
zona1.add("Centro San Juan De Dios");
zona1.add("Casa Comunal San Carlos");
zona1.add("U.E. San Esteban Del Valle");
```

**Figura 37.** Arreglo conformado por los valores de los subitems de la lista desplegable

En la lista desplegable, los nombres de las zonas se agrupan en filas Parents (padres), mientras que los arrays que contienen los nombres de los refugios se almacenan en filas Child (hijos) las cuales son contenidas por los Parents.

Se debe implementar un adaptador el cual permite gestionar el grupo de datos almacenados en los arreglos que se visualizarán en la lista.

```

public class ExpandableListAdapter extends BaseExpandableListAdapter {
    private Context _context;
    private List<String> _listDataHeader; // header titles
    // child data in format of header title, child title
    private HashMap<String, List<String>> _listDataChild;

    public ExpandableListAdapter(Context context, List<String> listDataHeader,
        HashMap<String, List<String>> listChildData) {
        this._context = context;
        this._listDataHeader = listDataHeader;
        this._listDataChild = listChildData;
    }

    @Override
    public Object getChild(int groupPosition, int childPosition) {
        return this._listDataChild.get(this._listDataHeader.get(groupPosition))
            .get(childPosition);
    }

    @Override
    public long getChildId(int groupPosition, int childPosition) {
        return childPosition;
    }
}

```

**Figura 38.** Clase del Adapter de la lista desplegable

### 3.1.2.6 Zoom y posicionamiento en Google Maps

Google Maps permite cambiar la posición de la vista del mapa y hacer un zoom hacia una ubicación específica. A esto se le llama movimiento de cámara y se utiliza para tener una mejor vista del entorno de acuerdo a la ubicación seleccionada (Google Developers, 2016).

El movimiento de cámara se lo puede hacer directamente a partir de la ubicación donde se encuentra el usuario de acuerdo a la posición brindada por el receptor GPS del dispositivo móvil. Se puede realizar un zoom hacia la posición exacta del usuario por medio del método *animateCamera*.

Para obtener la ubicación actual se debe usar el método *getMyLocation* el cual devuelve el valor de la longitud y latitud en la posición actual. Cuando se obtiene las coordenadas de la ubicación, se hace un zoom al lugar indicado y se lo señala con un marcador. Se debe verificar si el GPS del dispositivo móvil se encuentra activado. Si no es así, aparece una ventana de alerta indicando que se debe activar el GPS.

```

}
public void animateCamera(View view) {
    gps = new Gps(MapsActivity.this);
    // check if GPS enabled
    if(gps.canGetLocation()){
        double latitude = gps.getLatitude();
        double longitude = gps.getLongitude();
        MI_UBICACION = new LatLng(mMap.getMyLocation().getLatitude(), mMap.getMyLocation().getLongitude());
        mMap.addMarker(new MarkerOptions().position(MI_UBICACION).draggable(true)
            .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_BLUE)));
        mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(MI_UBICACION, 18));

        // \n is for new line
        Toast.makeText(getApplicationContext(), "Your Location is - \nLat: " + latitude + "\nLong: " + longitude, Toast.LENGTH_LONG).show();
    }else{
        // can't get location
        // GPS or Network is not enabled
        // Ask user to enable GPS/network in settings
        gps.showSettingsAlert();
    }
}
/*MI_UBICACION = new LatLng(mMap.getMyLocation().getLatitude(), mMap.getMyLocation().getLongitude());
mMap.addMarker(new MarkerOptions().position(MI_UBICACION).draggable(true)
    .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_BLUE)));*/

```

**Figura 39.** Código para la obtención de la posición actual

Por medio de la pantalla táctil se puede seleccionar un punto de coordenadas específico sobre el mapa. Esto permite realizar pruebas sobre distintas ubicaciones sin necesidad de que el usuario se encuentre en un lugar preciso. Por medio de la clase *onMapClick* se añade un marcador al pulsar sobre cualquier punto del mapa dentro de la zona de análisis

```

@Override
public void onMapClick(LatLng puntoPulsado) {
    mMap.addMarker(new MarkerOptions().position(puntoPulsado).
        icon(BitmapDescriptorFactory
            .defaultMarker(BitmapDescriptorFactory.HUE_GREEN)));
}

```

**Figura 40.** Selección de un punto sobre el mapa por medio de la pantalla táctil

Adicionalmente, se puede seleccionar el lugar hacia donde se quiere realizar el zoom por medio de la opción seleccionada en la lista desplegable. Esto permite visualizar con mayor detalle la ubicación de los refugios analizados.

Por medio del método *mMap.moveCamera* se realiza el zoom hacia un punto de coordenadas específico. Para lograr esto, se debe ingresar el punto de coordenadas y el valor de zoom.

Por medio de un *switch case*, se hace zoom a un punto de coordenadas específico, dependiendo del valor seleccionado en la lista.

```
switch (childPosition) {  
    case 0:  
        Toast.makeText(MapsActivity.this, "Saint Dominic  
School", Toast.LENGTH_SHORT).show();  
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(REF1, 15));
```

**Figura 41.** Movimiento de cámara hacia la posición del refugio seleccionado

Por medio de la función `Toast.MakeText` se muestra un texto indicando el nombre del refugio sobre la interfaz del mapa.

### 3.1.2.7 Visibilidad de objetos generados sobre Google Maps

En la interfaz de Google Maps, se puede añadir un `Checkbox`, el cual es una casilla de verificación que al ser seleccionada, permite realizar ciertas acciones sobre las figuras dibujadas en el mapa.

En este caso, al seleccionar el `checkbox`, las polilíneas que conforman las rutas de escape predefinidas se harán visibles. Esto se lo ha realizado debido a que existe una gran cantidad de elementos dibujados sobre el mapa que pueden confundir al usuario.

En el código del programa se debe invocar la librería `Android.widget.CheckBox`. Luego se crea un atributo `chk1` del tipo `CheckBox`. Por medio de la función `chk1.isChecked()` se hace una comparación: Si se ha marcado el `checkbox`, las polilíneas de las rutas de escape se hacen visibles.

```

chk1 = (CheckBox) findViewById(R.id.ChkEscape);
chk1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (chk1.isChecked() == true) {

ruta_zonalA1.setVisible(true);ruta_zonalA2.setVisible(true);ruta_zonalA3.setVisible(tr
ue);

ruta_zonalA4.setVisible(true);ruta_zonalA5.setVisible(true);ruta_zonalA6.setVisible(tr
ue);
                ruta_zonalA7.setVisible(true);

ruta_zonalB1.setVisible(true);ruta_zonalB2.setVisible(true);ruta_zonalB3.setVisible(tr
ue);

ruta_zonalB4.setVisible(true);ruta_zonalB5.setVisible(true);ruta_zonalB6.setVisible(tr
ue);
                ruta_zonalB7.setVisible(true);

ruta_zonalC1.setVisible(true);ruta_zonalC2.setVisible(true);ruta_zonalC3.setVisible(tr
ue);

```

**Figura 42.** Visibilidad de las polilíneas cuando se selecciona el checkbox

### 3.1.2.8 Encontrar un punto dentro de un polígono

Los polígonos graficados sobre un mapa en Google Maps pueden ser utilizados para determinar una ubicación específica dentro de una zona. La versión 3 de Google Maps permite implementar métodos que identifican si un punto se encuentra dentro de un polígono. Estos métodos son: *google.Maps.containslocation()* y *google.Maps.containslatlng()*.

Debido a que la versión 2 de Google Maps utilizada en el programa de Android no maneja un método propio que permita identificar si un punto de coordenadas se encuentra dentro de un polígono generado sobre el mapa, se debe implementar un algoritmo que permita realizar dicha acción. En este caso se implementa un algoritmo basado en ray-casting el cual realiza un conteo de las veces que un rayo trazado desde el punto analizado interseca con los lados del polígono (Randolph Franklin, 2014).

Para poder implementar este algoritmo, en el programa se crea un método llamado *PointInPolygon* el cual requiere como atributos el punto de coordenadas a analizarse y el polígono. Los puntos de coordenadas que conforman el polígono se almacenan en un arreglo llamado *points*.

Se crean valores enteros  $i, j$  los cuales son banderas que se manejarán sobre un bucle *for*. También se crea un valor entero llamado *nvert* que tendrá el

valor del tamaño del arreglo. En el bucle for, la bandera  $i$  se inicia desde cero y aumenta en 1 su valor hasta llegar al valor de  $nvert$ . La bandera  $j$  en cambio inicia desde el valor de  $nvert - 1$  y su valor incrementará de tal forma que  $j=i+1$ .

En cada ciclo del bucle se realizará la siguiente comparación: Se verifica si el valor de latitud del vértice almacenado en el arreglo es mayor o igual al valor de latitud del punto de coordenadas. También se compara si el valor de latitud del siguiente vértice es mayor o igual al valor de latitud del punto de coordenadas. Con estas dos comparaciones se verifica si el punto de coordenadas se encuentra entre los vértices del polígono. Además se hace otra comparación por medio de la siguiente ecuación:

$$longitud\_punto \leq \frac{(longitud\_vertice\_2 - longitud\_vertice\_1) * (latitud\_vertice\_2 - latitud\_vertice\_1)}{(longitud\_vertice\_2 - longitud\_vertice\_1) + longitud\_vertice\_1}$$

Donde  $longitud\_punto$  es el valor de longitud del punto analizado,  $longitud\_vértice\_1$  y  $latitud\_vértice\_1$  son los valores de coordenadas del primer vértice y  $longitud\_vértice\_2$  y  $latitud\_vértice\_2$  son los valores de coordenadas del vértice siguiente.

Si el punto de coordenadas se encuentra dentro de los vértices y además su valor de longitud es menor o igual al valor obtenido en la fórmula, entonces se retorna un valor booleano  $c$  como  $true$ . Si el valor del booleano es verdadero, el punto se encuentra dentro del polígono y si es falso, el punto se encuentra fuera del polígono.

```
//punto en poligono algoritmo de rayo
public boolean PointInPolygon(LatLng point, Polygon polygon) {
    List<LatLng> points = polygon.getPoints();
    int i, j, nvert = points.size();
    boolean c = false;

    for(i = 0, j = nvert - 1; i < nvert; j = i++) {
        if( ( (points.get(i).latitude >= point.latitude) != (points.get(j).latitude >= point.latitude) ) &&
            (point.longitude <= (points.get(j).longitude - points.get(i).longitude) * (point.latitude - points.get(i).latitude) /
              (points.get(j).latitude - points.get(i).latitude) + points.get(i).longitude)
            )
        )
            c = !c;
    }

    return c;
}
```

**Figura 43.** Método que maneja ray-casting para determinar un punto dentro de un polígono

Posteriormente se hace una comparación para verificar la zona del punto de coordenadas seleccionado por medio de los polígonos graficados sobre el mapa. El punto de coordenadas puede ser la ubicación actual del usuario o un punto seleccionado sobre el mapa por medio de la pantalla táctil.

Se verifica si el punto de coordenadas se encuentra dentro de un polígono que representa una zona de alto riesgo, si ese no es el caso, se verifica si el punto de coordenadas se encuentra dentro una zona cercana, caso contrario, se verifica si el punto se encuentra en una zona segura. Si no cumple con ninguna de las condiciones se debe verificar si el punto se encuentra en otras zonas o si se encuentra fuera de rango. Por medio de la función *toast* se indica la zona donde el usuario se encuentra y se utiliza el método *AlertDialog.Builder* para notificar por medio de mensaje de alerta si el usuario se encuentra en una zona de alto riesgo o en un lugar cercano.

Al realizar la comparación determinar la zona se invoca al método que determina el refugio más cercano dependiendo si el punto analizado se encuentra dentro de una zona de alto riesgo o en un lugar cercano. Si es así, se llama a los métodos para trazar la ruta hacia el refugio determinado y se invoca al método que permite recomendar al usuario el refugio más cercano por medio de un mensaje auditivo.

Cuando el código identifica si el punto de coordenadas se encuentra dentro de una zona de riesgo, entonces el teléfono vibra por medio de la clase *vibrator*. Por medio de esta clase, se llama al método *vibrate* al cual se le asigna la duración de la vibración en milisegundos (2000).

```

//COMPARACIONES ENTRE ZONAS//
if (PointInPolygon(MI_UBICACION, zona1_total)) {
    if (PointInPolygon(MI_UBICACION, zona1_cercana)) {
        if (PointInPolygon(MI_UBICACION, zona1_riesgo1) || PointInPolygon(MI_UBICACION, zona1_riesgo2)) {
            puntoCercano(MI_UBICACION, pref);
            // getting URL to the Google Directions API
            String url = getDirectionsurl(MI_UBICACION, pCercano);

            DownloadTask downloadTask = new DownloadTask();

            // Start downloading json data from Google Directions API
            downloadTask.execute(url);
            vibrator vibrator;
            vibrator = (vibrator) getSystemService(Context.VIBRATOR_SERVICE);
            vibrator.vibrate(2000);
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setMessage("ZONAL RIESGO. Se encuentra en una zona de Riesgo. Dirijase al Refugio mas cercano. SIGA EL MAPA HACIA EL REFUGIO")
                .setTitle("MENSAJE URGENTE!!!").setCancelable(false)
                .setNegativeButton("Aceptar", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int id) {
                        dialog.cancel();
                    }
                });
            AlertDialog alert = builder.create();
            alert.show();
            voz(pCercano, refugios);//Llamada de voz hacia refugio
        } else {
            puntoCercano(MI_UBICACION, pref);
            // getting URL to the Google Directions API
            String url = getDirectionsurl(MI_UBICACION, pCercano);

            DownloadTask downloadTask = new DownloadTask();

            // Start downloading json data from Google Directions API
            downloadTask.execute(url);
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setMessage("ZONAL SEGURA. Tome las debidas Precauciones ya que se encuentra cerca de la Zona de Riesgo")
                .setTitle("ALERTA!!!").setCancelable(false)
                .setNegativeButton("Aceptar", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int id) {
                        dialog.cancel();
                    }
                });
            AlertDialog alert = builder.create();
            alert.show();
            voz(pCercano, refugios);//Llamada de voz hacia refugio
        }
    }
}
Toast.makeText(this, "Se encuentra en Zona1", Toast.LENGTH_LONG).show();
}

```

**Figura 44.** Comparación de las zonas de acuerdo a la posición actual

### 3.1.2.9 Determinación del refugio más cercano

Al momento de realizar la comparación de zonas, se debe definir el refugio más cercano a partir de la distancia más corta de las posibles rutas entre la posición actual del usuario y los refugios de la zona.

El método *puntoCercano* realiza una comparación de las distancias entre el punto de coordenadas de la ubicación seleccionada y los puntos de coordenadas de los refugios de la zona los cuales han sido almacenados en un arreglo. Por medio de un bucle *for* se determina cuál es el punto de coordenadas con la distancia más corta y se lo almacena en la variable llamada *pCercano*.



```

private void puntoCercano(LatLng p, ArrayList<LatLng> puntos) {
    //LatLng pCercano;
    int nPuntos;
    float d;
    nPuntos = puntos.size();
    pCercano = puntos.get(0);
    if (estaConectado()) {
        d = getDistance(p.latitude, p.longitude, pCercano.latitude, pCercano.longitude);
        for (int k = 1; k < nPuntos; k++) {
            LatLng pCercano1 = puntos.get(k);
            float dNueva = getDistance(p.latitude, p.longitude, pCercano1.latitude, pCercano1.longitude);
            if (dNueva < d) {
                d = dNueva;
                pCercano = pCercano1;
            }
        }
    }
}

```

**Figura 45.** Método puntoCercano el cual permite obtener el punto más cercano

Para determinar la distancia hacia los posibles refugios se implementa el método *getDistance*. Este método permite obtener la distancia que se recorre entre dos puntos a través de las calles y rutas de la zona. Para poder realizar esto, se debe acceder al portal de Google Directions el cual permite generar las posibles rutas hacia el destino.

Para acceder a este portal se debe generar un URL (Uniform Resource Locator) a partir de los valores de los puntos de coordenadas de origen y destino. Se debe definir el valor del sensor cuyo valor debe ser falso y el modo para recorrer la ruta el cual puede ser *walking* (caminando) o *driving* (conduciendo).

Se puede elegir el modo para recorrer el trayecto por medio de un *radio-button* colocado sobre la interfaz del mapa, el cual es un objeto que permite realizar la selección de una opción dentro de un conjunto de opciones. Este objeto está conformado por un atributo del tipo *radiogroup* llamado *rgModes* el cual contiene los atributos *RadioButton* los cuales son *rbWalking* y *rbDriving*.

A partir del URL generado se determinan los posibles puntos de coordenadas por donde debe pasar la ruta. Estos puntos se encuentran almacenados en formato *JSON*. *JSON* (Java Script Object Notation) es un formato ligero que permite el intercambio de datos que se utiliza de forma independiente en cualquier lenguaje de programación. Por medio de este formato se pueden agrupar varios atributos en arreglos.

El método *getDistance* permite obtener los datos almacenados en un objeto *JSON*, permitiendo determinar la distancia entre los posibles puntos

obtenidos que conforman la ruta. El método calcula el valor de la distancia y lo entrega por medio de un valor String el cual debe ser transformado en a un valor flotante que representa la distancia en metros. Si la distancia está en kilómetros, se debe multiplicar el valor obtenido por mil para realizar la comparación de todas las distancias en metros.

```
//MÉTODO PARA CALCULAR LA DISTANCIA
public float getDistance(final double lat1, final double lon1, final double lat2, final double lon2){
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                String sensor = "sensor=false";
                String mode = null;
                if (rBORIVING.isChecked()) {
                    mode = "mode=driving";
                    mMODE = 0;
                } else if (rBWALKING.isChecked()) {
                    mode = "mode=walking";
                    mMODE = 1;
                }
                | URL url = new URL("http://maps.googleapis.com/maps/api/directions/json?origin="+ lat1 + ", " + lon1 + "&destination=" + lat2 + ", " + lon2
                "&" + sensor + "&" + mode);
                final HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setRequestMethod("POST");
                InputStream in = new BufferedInputStream(conn.getInputStream());
                response = org.apache.commons.io.IOUtils.toString(in, "UTF-8");
                JSONObject jsonObject = new JSONObject(response);
                JSONArray array = jsonObject.getJSONArray("routes");
                JSONObject routes = array.getJSONObject(0);
                JSONArray legs = routes.getJSONArray("legs");
                JSONObject steps = legs.getJSONObject(0);
                JSONObject distance = steps.getJSONObject("distance");
                parsedDistance = distance.getString("text");
                dist = Float.valueOf(parsedDistance.replaceAll("[^\\d.]|\\|\\. (?!\\d)", ""));
                if (parsedDistance.contains("k"))
                {
                    dist=dist*1000;
                }
            } catch (ProtocolException e) {
                e.printStackTrace();
            } catch (MalformedURLException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    });
    thread.start();
    try {
        thread.join();
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    return dist;
}
}
```

**Figura 46.** Método getDistance el cual permite obtener la distancia entre dos puntos

Al realizar las comparaciones para determinar el refugio más cercano, se debe tener conexión a internet para obtener acceso a Google Directions y determinar las posibles distancias. Por esta razón, se puede llegar a generar elevado procesamiento al calcular todas las posibles distancias. Es por eso que es recomendable que el arreglo con los puntos de coordenadas que se utilizan en la comparación para determinar la menor distancia no contenga más de diez elementos. Dependiendo de la zona en la que se encuentra el punto de coordenadas seleccionado se asignan los puntos de los refugios que se deben comparar.

### 3.1.2.10 Trazado de rutas hacia el refugio más cercano

A fin de que la aplicación sea una herramienta que guíe al usuario durante una emergencia, es necesario que siempre se recomiende la ruta más cercana para llegar al sitio seguro. En el código se debe implementar una clase que permita dibujar una ruta de forma automática sobre las calles que conducen hacia el refugio determinado a partir de la posición actual del usuario. El trayecto puede variar, dependiendo del modo para llegar al destino, ya sea caminando o por medio de un vehículo. Estas son las dos alternativas que la aplicación ofrece para que el usuario seleccione la ruta hacia el refugio.

Para poder trazar la ruta, se debe nuevamente acceder al portal de *Google Directions* el cual proporciona los posibles puntos por donde la ruta debe pasar. Se debe generar el URL a partir de los valores de los puntos de coordenadas de origen y destino, el valor del sensor el cual debe ser falso y el modo para llegar a la ruta el cual puede ser *walking* o *driving*. De igual manera, por medio de un objeto *radio-button* colocado sobre la interfaz del mapa se puede seleccionar el modo para recorrer la ruta.

```

////PARA TRAZAR RUTAS////
private String getDirectionsurl(LatLng MI_UBICACION, LatLng dest) {
    // origin of route
    String str_origin = "origin=" + MI_UBICACION.latitude + "," + MI_UBICACION.longitude;
    // destination of route
    String str_dest = "destination=" + dest.latitude + "," + dest.longitude;
    // sensor enabled
    String sensor = "sensor=false";
    // Travelling Mode
    String mode = "mode=driving";
    if (rbriving.isChecked()) {
        mode = "mode=driving";
        mMode = 0;
    } else if (rbwalking.isChecked()) {
        mode = "mode=walking";
        mMode = 1;
    }
    // Building the parameters to the web service
    String parameters = str_origin + "&" + str_dest + "&" + sensor + "&" + mode;
    // output format
    String output = "json";
    // building the url to the web service
    String url = "https://maps.googleapis.com/maps/api/directions/" + output + "?" + parameters;
    return url;
}

```

**Figura 47.** Generación del URL que permite acceder a los datos almacenados en Google Directions

Luego de generar el URL se implementa un método que permite descargar los datos entregados por Google Directions.

```

/** A method to download json data from url */
private String downloadurl(String strurl) throws IOException {
    String data = "";
    InputStream istream = null;
    HttpURLConnection urlConnection = null;
    try{
        URL url = new URL(strurl);

        // Creating an http connection to communicate with url
        urlConnection = (HttpURLConnection) url.openConnection();

        // Connecting to url
        urlConnection.connect();

        // Reading data from url
        istream = urlConnection.getInputStream();

        BufferedReader br = new BufferedReader(new InputStreamReader(istream));
        StringBuffer sb = new StringBuffer();

        String line = "";
        while( ( line = br.readLine()) != null){
            sb.append(line);
        }

        data = sb.toString();

        br.close();
    }catch(Exception e){
        Log.d("Exception while downloading url", e.toString());
    }finally{
        istream.close();
        urlConnection.disconnect();
    }
    return data;
}

```

**Figura 48.** Método que permite acceder a los valores almacenados desde la URL

Por medio de la clase DownloadTask se descarga la información obtenida desde Google Directions la cual se debe procesar para trazar la ruta.

```

/** A class to download data from Google Directions URL */
private class DownloadTask extends AsyncTask<String, void, String>{

    // Downloading data in non-ui thread
    @Override
    protected String doInBackground(String... url) {

        // For storing data from web service
        String data = "";

        try{
            // Fetching the data from web service
            data = downloadurl(url[0]);
        }catch(Exception e){
            Log.d("Background Task",e.toString());
        }
        return data;
    }

    // Executes in UI thread, after the execution of
    // doInBackground()
    @Override
    protected void onPostExecute(String result) {
        super.onPostExecute(result);

        ParserTask parserTask = new ParserTask();

        // Invokes the thread for parsing the JSON data
        parserTask.execute(result);
    }
}

```

**Figura 49.** Método que permite descargar los datos desde la URL

Los datos obtenidos desde Google Directions se almacenan en formato *JSON*. El objeto *JSON* generado contiene los puntos de coordenadas por donde debe pasar la ruta.

```

/** A class to parse the Google Directions in JSON format */
private class ParserTask extends AsyncTask<String, Integer, List<List<HashMap<String, String>>> > {

    // Parsing the data in non-ui thread
    @Override
    protected List<List<HashMap<String, String>>> doInBackground(String... jsonData) {

        JSONObject jobject;
        List<List<HashMap<String, String>>> routes = null;

        try{
            jobject = new JSONObject(jsonData[0]);
            DirectionsJSONParser parser = new DirectionsJSONParser();

            // Starts parsing data
            routes = parser.parse(jobject);
        }catch(Exception e){
            e.printStackTrace();
        }
        return routes;
    }
}

```

**Figura 50.** Obtención de los valores de la URL los cuales se encuentran en formato *JSON*

Dentro de la de la clase *DirectionsJsonParser* se implementa un método que recibe un objeto *JSON* y retorna un arreglo con los valores de longitud y latitud de coordenadas de los puntos por donde pasa la ruta.

```

public class DirectionsJSONParser {
    /** Receives a JSONObject and returns a list of lists containing latitude and longitude */
    public List<List<HashMap<String, String>>> parse(JSONObject jsonObject) {
        List<List<HashMap<String, String>>> routes = new ArrayList<List<HashMap<String, String>>>();
        JSONArray jRoutes = null;
        JSONArray jLegs = null;
        JSONArray jSteps = null;
        JSONObject jDistance = null;
        JSONObject jDuration = null;

        try {
            jRoutes = jsonObject.getJSONArray("routes");

            /** Traversing all routes */
            for(int i=0; i<jRoutes.length(); i++){
                jLegs = ((JSONObject)jRoutes.get(i)).getJSONArray("legs");
                List path = new ArrayList<HashMap<String, String>>();

                /** Traversing all legs */
                for(int j=0; j<jLegs.length(); j++){
                    /** Getting distance from the json data */
                    jDistance = ((JSONObject)jLegs.get(j)).getJSONObject("distance");
                    HashMap<String, String> hmdistance = new HashMap<String, String>();
                    hmdistance.put("distance", jDistance.getString("text"));

                    /** Getting duration from the json data */
                    jDuration = ((JSONObject)jLegs.get(j)).getJSONObject("duration");
                    HashMap<String, String> hmduration = new HashMap<String, String>();
                    hmduration.put("duration", jDuration.getString("text"));

                    /** Adding distance object to the path */
                    path.add(hmdistance);

                    /** Adding duration object to the path */
                    path.add(hmduration);

                    jSteps = ((JSONObject)jLegs.get(j)).getJSONArray("steps");

                    /** Traversing all steps */
                    for(int k=0; k<jSteps.length(); k++){
                        String polyline =
                            (String)((JSONObject)((JSONObject)jSteps.get(k)).get("polyline")).get("points");
                        List<LatLng> list = decodePoly(polyline);

                        /** Traversing all points */
                        for(int l=0; l<list.size(); l++){
                            HashMap<String, String> hm = new HashMap<String, String>();
                            hm.put("lat", Double.toString(((LatLng)list.get(l)).latitude));
                            hm.put("lng", Double.toString(((LatLng)list.get(l)).longitude));
                            path.add(hm);
                        }
                    }
                    routes.add(path);
                }
            }
        } catch (JSONException e) {
            e.printStackTrace();
        } catch (Exception e) {}
        return routes;
    }
}

```

**Figura 51.** Clase para pasar los datos de coordenadas desde JSON hasta un arreglo

Para definir los puntos de coordenadas por donde debe pasar la ruta se debe crear un método que permita decodificar dichos puntos.

```

/**
 * Method to decode polyline points
 * Courtesy : jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-api-with-java
 */
private List<LatLng> decodePoly(String encoded) {
    List<LatLng> poly = new ArrayList<LatLng>();
    int index = 0, len = encoded.length();
    int lat = 0, lng = 0;

    while (index < len) {
        int b, shift = 0, result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lat += dlat;

        shift = 0;
        result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lng += dlng;

        LatLng p = new LatLng((((double) lat / 1E5)),
                               (((double) lng / 1E5)));
        poly.add(p);
    }
    return poly;
}

```

**Figura 52.** Decodificación de los puntos de la polilínea

Después de obtener los valores de los puntos de coordenadas, se dibuja la polilínea a través de los puntos obtenidos. Además se calcula la distancia de la ruta y la duración del trayecto al caminar o conducir.

```
// Executes in UI thread, after the parsing process
@Override
protected void onPostExecute(List<List<HashMap<String, String>>> result) {
    ArrayList<LatLng> points = null;
    PolylineOptions lineoptions = null;
    MarkerOptions markeroptions = new MarkerOptions();
    String distance = "";
    String duration = "";

    if(result.size()<1){
        Toast.makeText(getBaseContext(), "No Points", Toast.LENGTH_SHORT).show();
        return;
    }

    // Traversing through all the routes
    for(int i=0;i<result.size();i++){
        points = new ArrayList<LatLng>();
        lineoptions = new PolylineOptions();

        // Fetching i-th route
        List<HashMap<String, String>> path = result.get(i);

        // Fetching all the points in i-th route
        for(int j=0;j<path.size();j++){
            HashMap<String,String> point = path.get(j);

            if(j==0){ // Get distance from the list
                distance = (String)point.get("distance");
                continue;
            }else if(j==1){ // Get duration from the list
                duration = (String)point.get("duration");
                continue;
            }

            double lat = Double.parseDouble(point.get("lat"));
            double lng = Double.parseDouble(point.get("lng"));
            LatLng position = new LatLng(lat, lng);

            points.add(position);

            // Adding all the points in the route to Lineoptions
            lineoptions.addAll(points);
            lineoptions.width(4);
            lineoptions.color(Color.GREEN);
        }
        tvDistanceDuration.setText("Distance:"+distance + ", Duration:"+duration);
        // Drawing polyline in the Google Map for the i-th route
        mMap.addPolyline(lineoptions);
    }
}
```

**Figura 53.** Trazo de la polilínea a partir de los puntos de coordenadas obtenidos

### 3.1.2.11 Alerta de voz

Por medio del método *TextToSpeech* se emite una alerta sonora. Lo que hace este método es transformar un mensaje de texto en un mensaje auditivo. Para poder realizar esto se deben importar las librerías *Android.speech.tts.TextToSpeech* y *Android.speech.tts.TextToSpeechService*. Posteriormente se debe generar un atributo t1 el cual es del tipo *textoSpeech*. Se inicializa esa variable con un idioma predeterminado. En este caso el idioma es el español mexicano.

```

t1 = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
    @Override
    public void onInit(int status) {
        if (status != TextToSpeech.ERROR) {
            Locale locale = new Locale("spa", "MEX");
            t1.setLanguage(locale);

            //t1.setLanguage(Locale.US);
        }
    }
});

```

**Figura 54.** Inicialización de la variable TextToSpeech

En el programa se debe implementar el método *TextToSpeech* para avisar al usuario que se dirija al refugio más cercano en caso de encontrarse en una zona de alto riesgo o en un lugar cercano.

Al determinar la zona, se define cuál es el refugio más cercano y se invoca al método *voz*. Este método utiliza el punto de coordenadas del refugio más cercano y el arreglo de coordenadas que contiene los marcadores de los refugios. Si las coordenadas del punto más cercano coinciden con los valores de uno de los refugios en el arreglo, se alerta al usuario, indicando el nombre del refugio determinado. El mensaje de voz dice lo siguiente: “diríjase a [nombre del refugio]”.

```

//METODO PARA LA VOZ//
private void voz(LatLng p, List<Marker> puntos) {
    int nPuntos1 = puntos.size();

    for (int k = 1; k < nPuntos1; k++) {
        if (puntos.get(k).getPosition().equals(p)) {
            String toSpeak = "Dirijase a" + puntos.get(k).getTitle();
            //Toast.makeText(getApplicationContext(), "Dirijase a" +toSpeak, Toast.LENGTH_SHORT).show();
            t1.speak(toSpeak, TextToSpeech.QUEUE_FLUSH, null);
        }
    }
}
}

```

**Figura 55.** Asignación del texto que se debe escuchar en la aplicación por medio de TextToSpeech

### 3.1.3 Implementación de menús de la aplicación

Dentro de las actividades *Medidas preventivas*, *EnlacesInfo*, *Botiquín*, *KitEmergencia* *LLamadasaEmergencia* se despliega un menú con varias



opciones que permiten brindar información al usuario en caso de una posible emergencia.

Al igual que en el menú principal, se almacena en un arreglo valores string los cuales son los títulos de cada ítem. Se crea otro arreglo de strings los cuales describen cada ítem del menú. En otro arreglo se almacenan los íconos que se deben asignar en cada ítem los cuales se obtienen por medio del método *R.drawable*. Posteriormente se utiliza un Adaptador que permite gestionar y ordenar los valores dentro de la lista.

```
public class MedidasPreventivas extends AppCompatActivity {
    private String menuCotopaxi[] = new String[]{"MANTENGA LA CALMA", "EVITE ZONAS DE RIESGO", "DISEÑE UN PLAN DE EVACUACIÓN",
        "ACUDA A SITIOS SEGUROS"};
    private String descrMenu[] = new String[]{"Estar tranquilo permite actuar con prudencia.",
        "Los ríos y zonas bajas corren el riesgo de ser alcanzados por lahares.",
        "Realice simulacros para probar el tiempo de reacción hacia una zona segura.",
        "Ubique el refugio más cercano a su domicilio."};
    private Integer[] imgId = {
        R.drawable.calma,
        R.drawable.evitar_zona,
        R.drawable.evacuacion,
        R.drawable.encuentro
    };
    private ListView lista;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_medidas_preventivas);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        AdaptadorMenuInicial adapter1 = new AdaptadorMenuInicial(this, menuCotopaxi, descrMenu, imgId);
        lista = (ListView) findViewById(R.id.mi_lista);
        lista.setAdapter(adapter1);
        /* lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
                String selectedItem = menuCotopaxi[position];
                //toast.makeText(getApplicationContext(), selectedItem, Toast.LENGTH_SHORT).show();
            }
        }); */
    }
}
```

**Figura 56.** Generación del menú de la actividad *MedidasPreventivas*

### 3.1.3.1 Llamadas a contactos telefónicos y acceso a páginas web

Dentro de la actividad *LlamadasEmergencia*, se debe generar un menú. La lista del menú se genera de igual forma que en el menú principal. Al dar clic en cada una de las opciones del menú se realiza una llamada telefónica a un contacto predeterminado.

Al dar clic en un ítem de la lista, se genera un objeto *intent* el cual invoca al método *ACTION\_CALL*. Este método permite realizar una llamada a un contacto telefónico. A este atributo se le debe asignar un número telefónico local. Para realizar una llamada de emergencia, se solicita de forma automática los permisos para poder ejecutar dicha acción.

```

@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    String selecteditem = menuCotopaxi[+position];
    Toast.makeText(getApplicationContext(), selecteditem, Toast.LENGTH_SHORT).show();
    if (selecteditem == "Ecu 911") {
        Intent i = new Intent(Intent.ACTION_CALL, Uri.parse("tel:+5959614580005"));
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (checkSelfPermission(Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
                // TODO: Consider calling
                // public void requestPermissions(@NonNull String[] permissions, int requestCode)
                // here to request the missing permissions, and then overriding
                // public void onRequestPermissionsResult(int requestCode, String[] permissions,
                // int[] grantResults)
                // to handle the case where the user grants the permission. See the documentation
                // for Activity#requestPermissions for more details.
                return;
            }
        }
        startActivity(i);
    } else if (selecteditem == "Policía (Sangolquí)") {
        Intent i = new Intent(Intent.ACTION_CALL, Uri.parse("tel: 022330977"));

        startActivity(i);
    } else
    if (selecteditem == "Bomberos (Sangolquí)") {
        Intent i = new Intent(Intent.ACTION_CALL, Uri.parse("tel: 022330402"));

        startActivity(i);
    } else
    if (selecteditem == "Cruz Roja (Sangolquí)") {
        Intent i = new Intent(Intent.ACTION_CALL, Uri.parse("tel: 022333621"));

        startActivity(i);
    }
}
}

```

**Figura 57.** Generación de los contactos telefónicos

Dentro de la actividad *EnlacesInfo*, se genera un menú similar, sin embargo, en este caso al seleccionar cada ítem se abre una página web específica. Al dar clic en una opción se genera un atributo del tipo *intent* que invoca el método `ACTION_VIEW`. Este método permite que se pueda visualizar la página web por medio del navegador del dispositivo. A cada opción se le debe asignar el nombre de dominio de la página web solicitada.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_enlaces_info);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    AdaptadorMenuInicial adapter1 = new AdaptadorMenuInicial(this, menuCotopaxi, descrMenu, imgid);
    lista = (ListView) findViewById(R.id.mi_lista);
    lista.setAdapter(adapter1);
    lista.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
            String slecteditem = menuCotopaxi[position];
            Toast.makeText(getApplicationContext(), slecteditem, Toast.LENGTH_SHORT).show();
            if (slecteditem == "MUNICIPIO DE RUMIÑAHUI") {
                Intent i = new Intent(Intent.ACTION_VIEW);
                i.setData(Uri.parse("https://www.facebook.com/mruminahui"));
                startActivity(i);
            } else
            if (slecteditem == "MRUMINAHUI") {
                Intent i = new Intent(Intent.ACTION_VIEW);
                i.setData(Uri.parse("https://mobile.twitter.com/mruminahui"));
                startActivity(i);
            } else
            if (slecteditem == "#VOLCANCOTOPAXI") {
                Intent i = new Intent(Intent.ACTION_VIEW);
                i.setData(Uri.parse("https://mobile.twitter.com/hashtag/volcancotopaxi"));
                startActivity(i);
            } else
            if (slecteditem == "RADIO ECOS RUMIÑAHUI") {
                Intent i = new Intent(Intent.ACTION_VIEW);
                i.setData(Uri.parse("http://ecosderuminahui.blogspot.com/?m=1"));
                startActivity(i);
            }
        }
    });
}

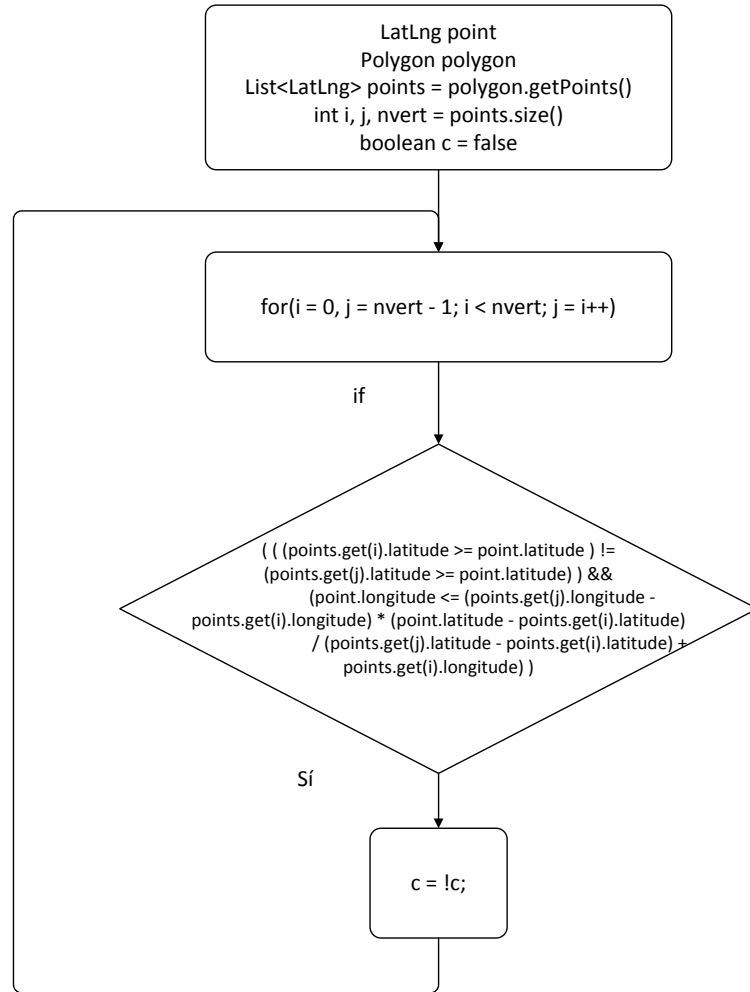
```

**Figura 58.** Generación de los links de las páginas web

## 3.2 Diagramas de flujo

### 3.2.1 Algoritmo de punto en el polígono

Por medio del siguiente diagrama se explica cómo se implementa el método *PointInPolygon* para determinar si un punto de coordenadas se encuentra dentro de un polígono a través de un algoritmo basado en ray-casting (Randolph Franklin, 2014).

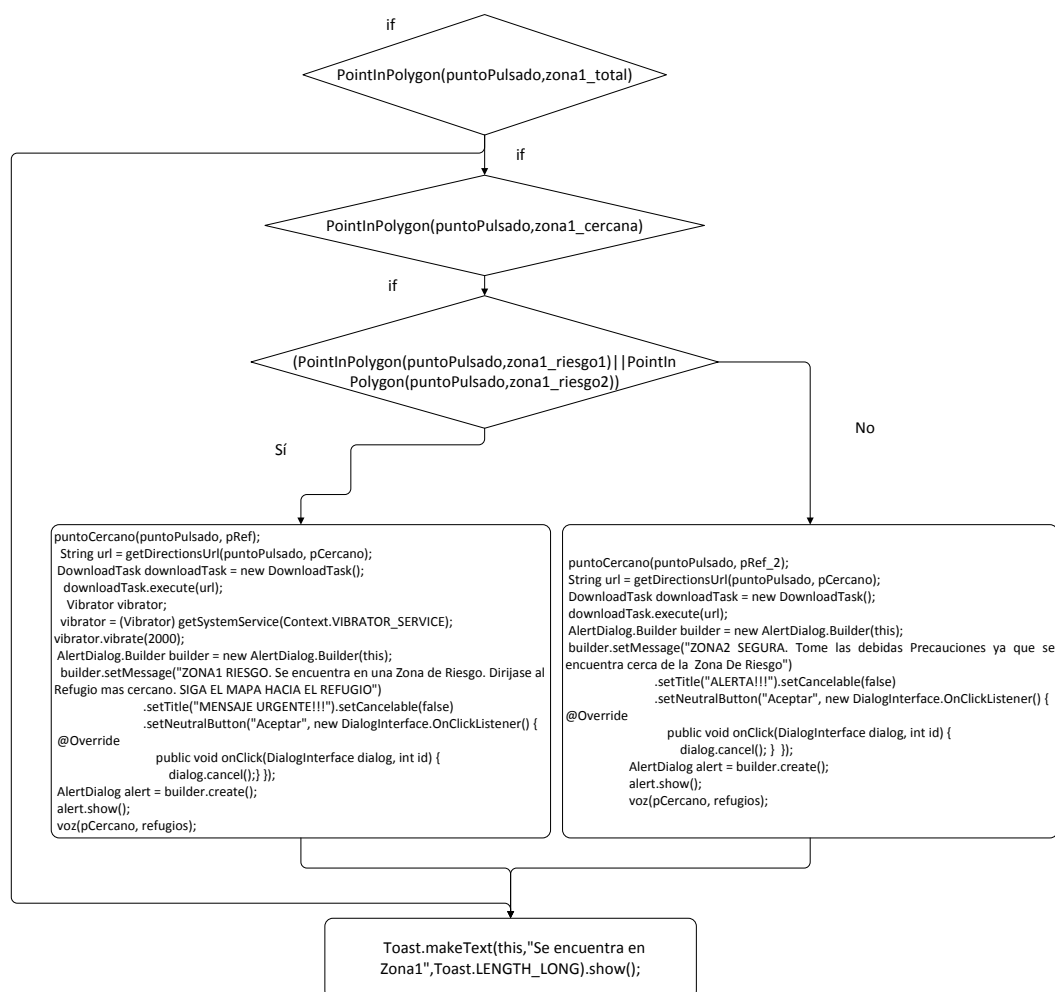


**Figura 59.** Diagrama de flujo del método PointinPolygon para determinar el punto dentro de un polígono

### 3.2.2 Comparación de zonas

El siguiente diagrama explica la forma cómo se identifica en qué zona se encuentra el punto de coordenadas a partir de los polígonos dibujados sobre el mapa. Después de identificar la zona, se debe mostrar un mensaje de alerta, llamar a los métodos para determinar la ruta hacia el refugio más cercano y hacer que el teléfono vibre en caso de encontrarse en zona de alto riesgo.

El diagrama indica cómo se determina la zona cuando se selecciona el punto en el mapa por medio de la pantalla táctil. Se utiliza el mismo algoritmo para determinar la zona cuando se selecciona la ubicación del receptor GPS.

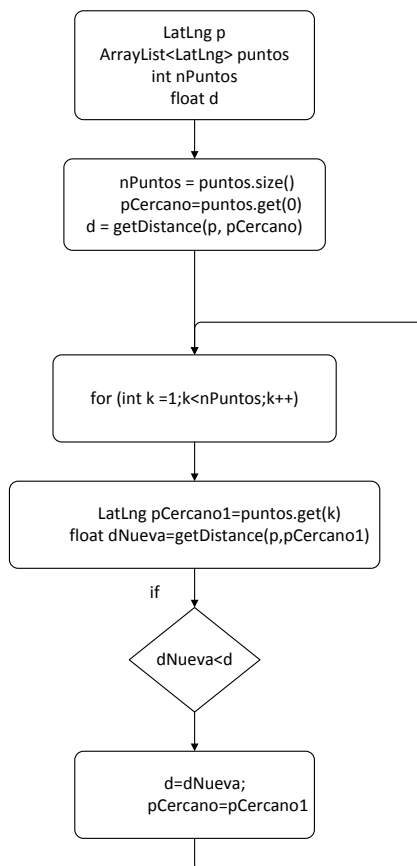


**Figura 60.** Diagrama de flujo de algoritmo de comparación de zonas

### 3.2.3 Cálculo del punto más cercano

En el siguiente diagrama se indica cómo el método *puntoCercano* determina el refugio más cercano. Se debe comparar las posibles distancias entre un punto de coordinas seleccionado y un grupo de refugios almacenados

en un arreglo. Al determinar la distancia más corta, el método retorna el refugio con la menor distancia.

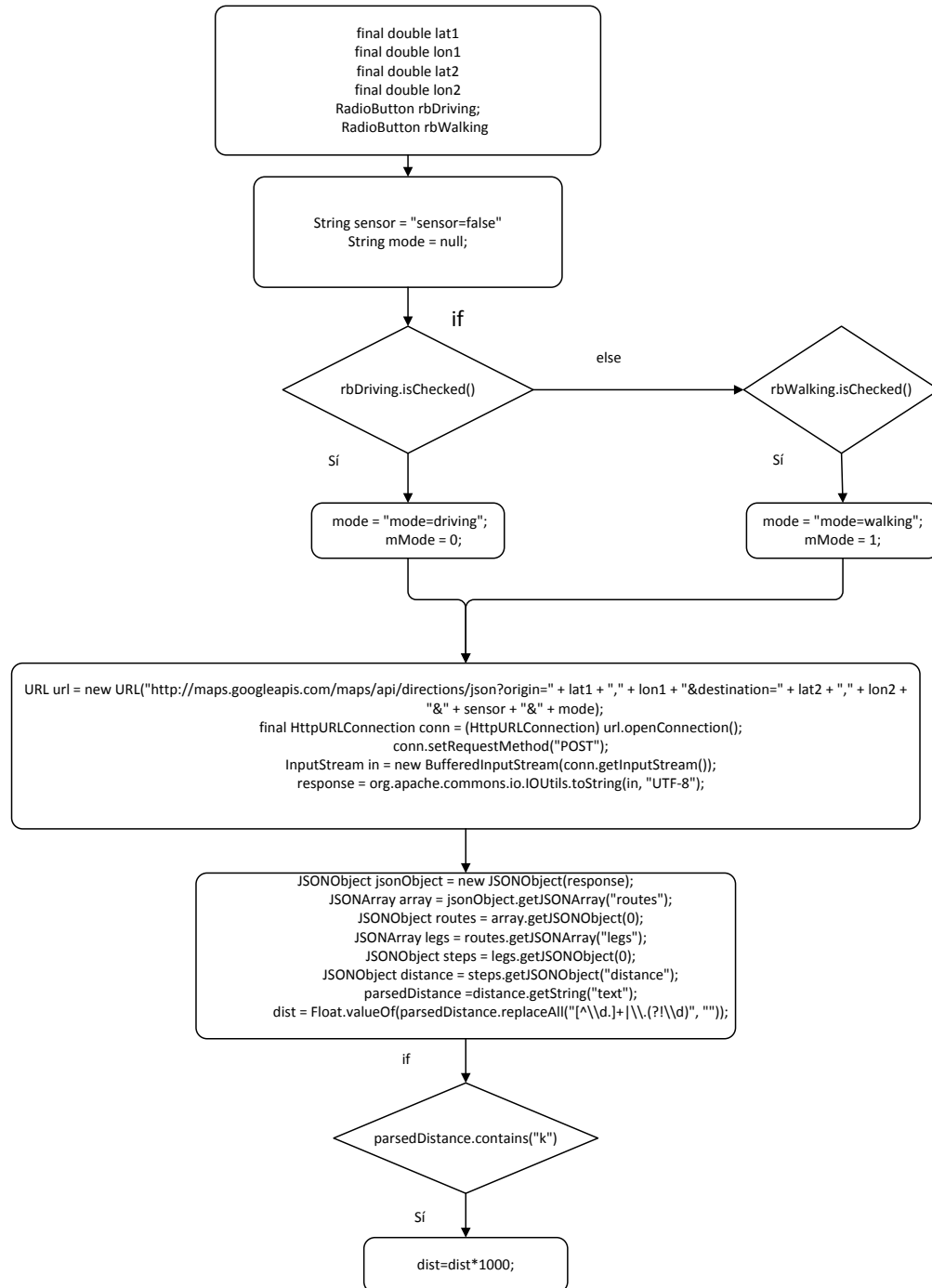


**Figura 61.** Diagrama de flujo del método puntoCercano para determinar el refugio más cercano

### 3.2.4 Determinación de la distancia entre dos puntos

En el siguiente diagrama se explica cómo el método *getDistance* determina la distancia del camino que se forma entre dos puntos de coordenadas. Se genera un URL para poder acceder al portal de Google Directions. Dicho portal determina los puntos de coordenadas que conforman la ruta del trayecto y que son almacenados en formato JSON. El método determina la distancia total en metros entre todos los puntos que forman el

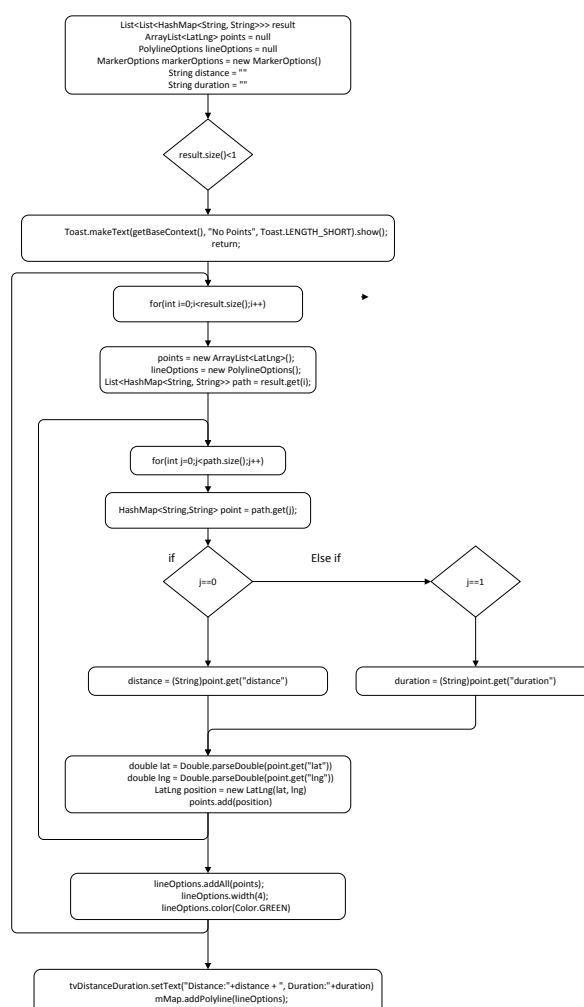
camino. (Draw Which is more accurate method for finding distance between two locations in google maps in android? distanceTo or Distance?, 2015)



**Figura 62.** Diagrama de flujo del método getDistance para determinar la distancia entre dos puntos

### 3.2.5 Trazado de ruta al punto más cercano

En el siguiente diagrama se explica cómo se traza una ruta entre dos puntos de coordenadas a través de las calles presentes en el mapa. Al generar el URL para acceder a Google Directions, se obtienen distintos puntos de coordenadas almacenados en formato JSON que conforman la trayectoria de la ruta. Posteriormente estos puntos se almacenan en un arreglo y se procesan en el método *onPostExecute* para que se dibuje una polilínea entre dichos puntos (Draw path between two points using Google Maps Android API v2, 2013).

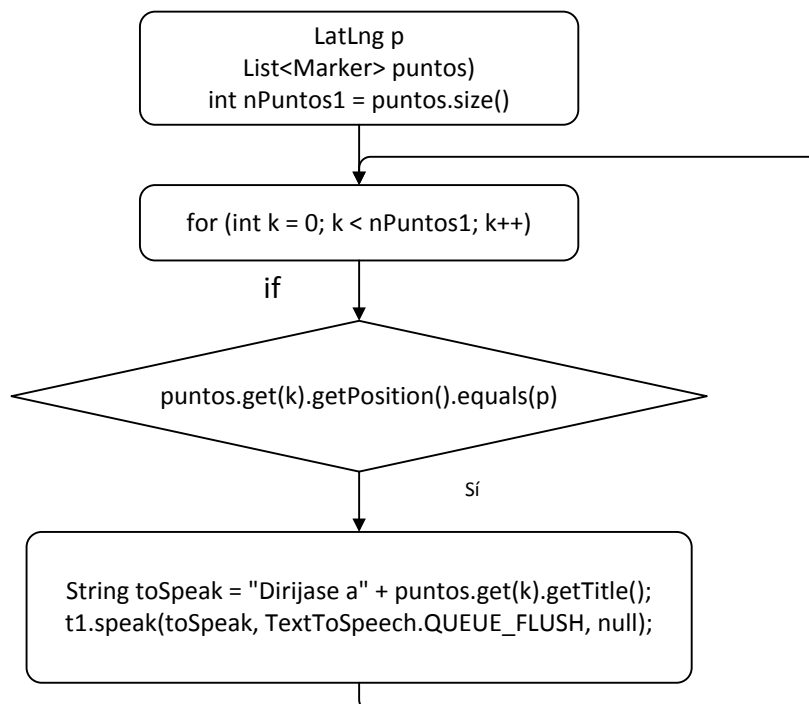


**Figura 63.** Diagrama de flujo del método *onPostExecute* para trazar la ruta entre dos puntos



### 3.2.6 Alerta de voz para recomendar el refugio más cercano

En el siguiente diagrama se indica el funcionamiento del método *voz*. A partir de un punto de coordenadas determinado y un arreglo de marcadores, se compara si el punto seleccionado coincide con las coordenadas de uno de los marcadores. Si es así se transforma el texto del título del marcador a un mensaje de voz.



**Figura 64.** Diagrama de flujo del método *voz* para emitir una alerta de voz

## CAPÍTULO IV

### RESULTADOS

#### 4.1. Análisis de resultados

##### 4.1.1 Resultados de la interfaz de la aplicación

Al iniciar la aplicación se muestra una ventana de inicio con el nombre de la aplicación (App Cotopaxi).



**Figura 65.** Pantalla inicial de la aplicación

Inmediatamente se visualiza el menú principal donde se pueden seleccionar las distintas opciones por medio de la pantalla táctil. Las opciones del menú son: Ver Mapa, Áreas Volcánicas, Medidas Preventivas, Enlaces

Informativos, Kit de Emergencia, Botiquín de Primeros Auxilios, Llamadas de Emergencias.



Figura 66. Vista del menú Principal

Al seleccionar la opción *Alertas Volcánicas* se visualiza un esquema que explica las distintas alertas que se pueden presentar durante la emergencia.



Figura 67. Diagrama de las alertas volcánicas

Al seleccionar la opción *Medidas Preventivas* se muestra un listado indicando las medidas preventivas que se deben tomar durante una emergencia.



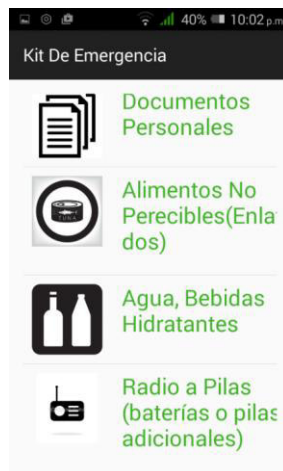
**Figura 68.** Listado de las medidas preventivas

Al seleccionar la opción *Enlaces Informativos* se muestra un menú donde se puede acceder a las páginas de información de los organismos oficiales que están encargados de informar del estado del volcán.



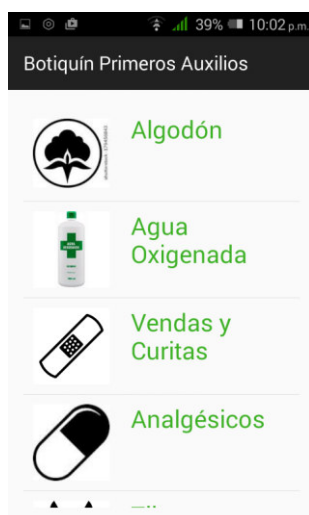
**Figura 69.** Menú de los enlaces informativos

Al seleccionar la opción *Kit de Emergencia* se muestra un listado indicando los objetos que se deben tomar en cuenta la momento de armar un kit de emergencia.



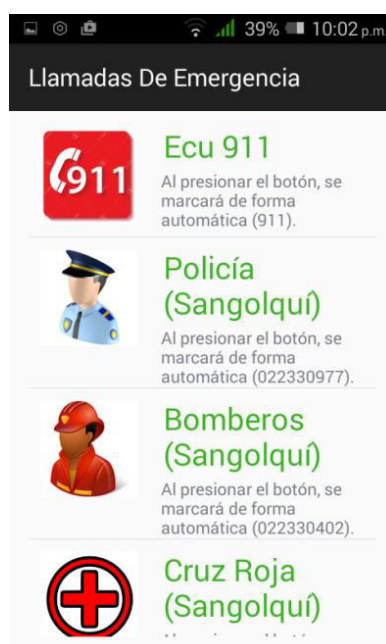
**Figura 70.** Listado del kit de emergencias

Al seleccionar la opción *Botiquín de Primeros Auxilios* se muestra un listado indicando los implementos que se deben llevar en un botiquín de primeros auxilios.



**Figura 71.** Listado de los objetos del Botiquín de Primeros Auxilios

Al seleccionar la opción *Llamadas de Emergencia* se muestra un menú que permite realizar llamadas telefónicas a contactos nacionales de emergencia al seleccionar la opción con la pantalla táctil.

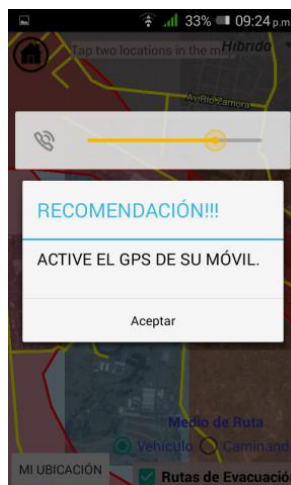


**Figura 72.** Listado de los contactos telefónicos

#### 4.1.1.1 Resultados en la actividad del mapa

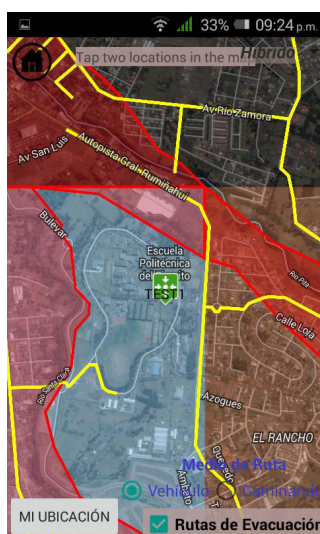
Al seleccionar la opción *Ver Mapa* en el menú principal se abre la actividad del mapa por medio de Google Maps. Si no se tiene instalada o actualizada la aplicación de Google Maps, se muestra un mensaje indicando al usuario que instale la última versión, permitiendo el acceso a la app de Google Play para realizar la descarga del programa. La última versión de Google Maps con la que se comprobó el funcionamiento de la aplicación era la 9.19.1.

Al ingresar a la actividad, se muestra inmediatamente un mensaje notificando al usuario que active el GPS. Si se ejecuta por primera vez la aplicación, el mapa tarda en cargarse en el dispositivo.



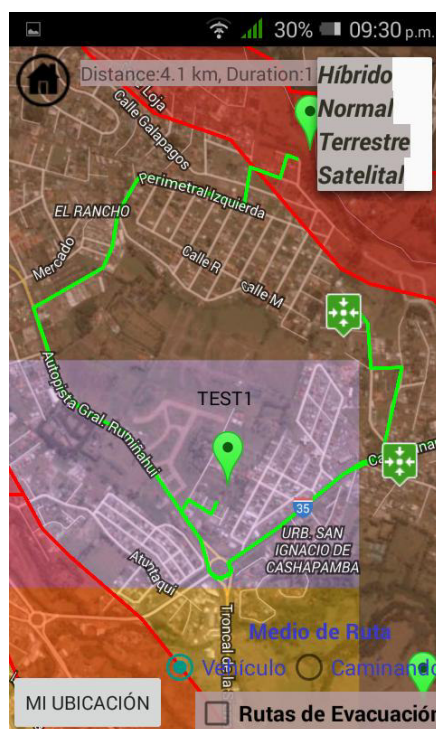
**Figura 73.** Mensaje de recomendación para activar el GPS del móvil

Después de que el mapa se termine de cargar, se puede visualizar la zona inicial a partir del punto de coordenadas predefinido el cual se encuentra ubicado en la Universidad de las Fuerzas Armadas (ESPE).



**Figura 74.** Vista inicial del mapa

La apariencia del mapa por defecto es la híbrida, donde se pueden visualizar las zonas por medio de imágenes satelitales, mostrando las calles de forma más detallada. En la parte superior derecha existe un menú que permite cambiar la apariencia del mapa a híbrido, normal, terrestre o a geográfica.

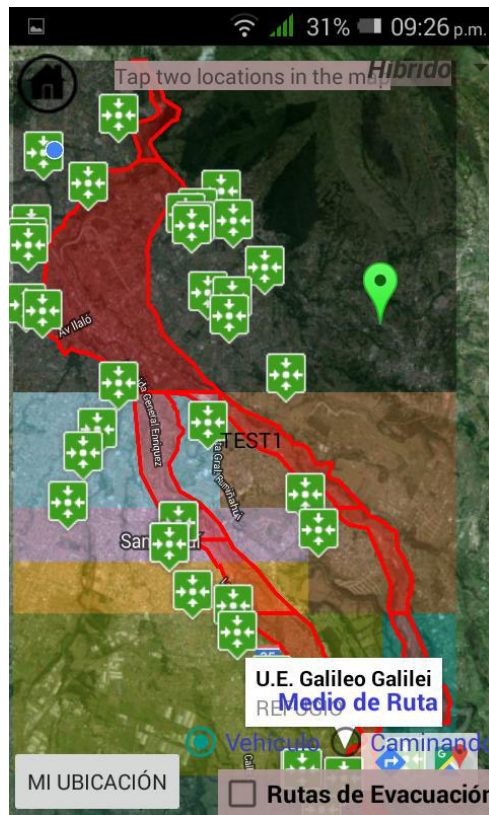


**Figura 75.** Menú de apariencia del mapa

Por medio de la pantalla táctil se puede desplazar el mapa hacia cualquier dirección y además se puede aumentar o reducir el zoom para poder visualizar de mejor manera la zona total de análisis. De esta forma se pueden apreciar claramente cómo están divididas las zonas por medio de figuras rectangulares dibujadas sobre el mapa. A cada zona se la ha asignado un color de fondo distinto. De igual manera se pueden visualizar las zonas de mayor riesgo por donde pasan los lahares. Las figuras que representan las zonas de riesgo tienen formas irregulares y se muestran con un color rojizo. Las figuras graficadas sobre el mapa tienen un cierto grado de transparencia, permitiendo



que se visualicen las principales calles y lugares habitados dentro de los polígonos dibujados.



**Figura 76.** Vista de la zona total de análisis sobre el mapa

Se colocaron marcadores en las ubicaciones exactas de los refugios los cuales pueden ser albergues, unidades educativas o zonas de encuentro como parques, complejos deportivos, etc. Al dar clic con la pantalla táctil en cada marcador se muestra un mensaje con el nombre del refugio. Existe además un menú desplegable que enlista todos los refugios por zona. En la parte superior izquierda hay un ícono con forma de una casa. Al dar clic en ese ícono se despliega de forma vertical el menú. Al dar clic en una zona se abre un submenú con los refugios que pertenecen a dicha zona.



**Figura 77.** Menú de la lista de refugios

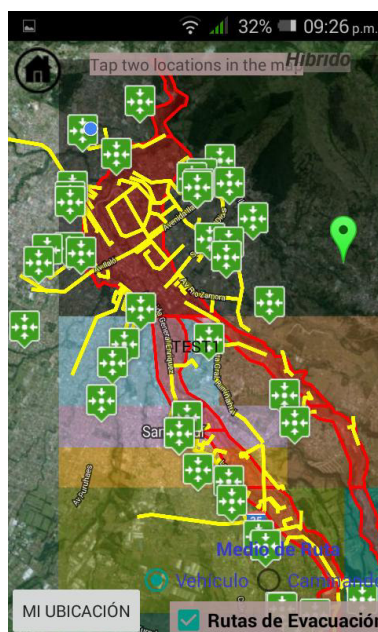
Al dar clic en el nombre de un refugio, el mapa hace un zoom directamente hacia el refugio seleccionado y se hace visible solamente el polígono de la zona al que pertenece el refugio.



**Figura 78.** Vista del refugio seleccionado

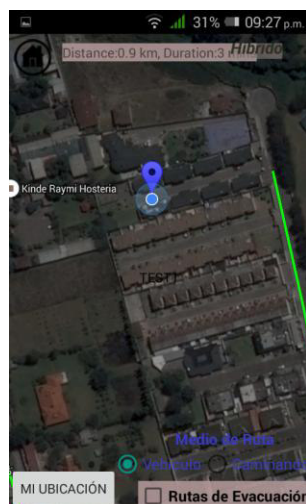
En el mapa se pueden apreciar rutas de escape predefinidas las cuales están dibujadas sobre las vías principales de la zona. Por medio del checkbox ubicado en la parte inferior derecha se pueden hacer visibles o invisibles dichas

rutas de escape. Esto permite que se pueda apreciar de mejor manera el mapa sin tener que visualizar todos los elementos dibujados sobre el mismo.



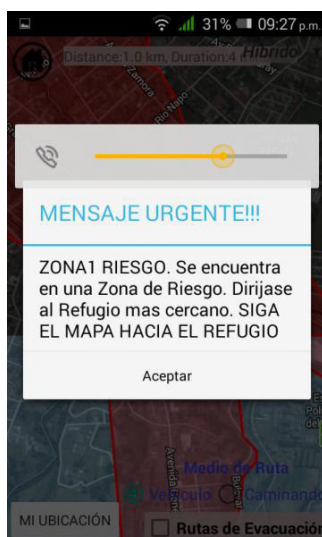
**Figura 79.** Vista de las rutas de escape

Al dar clic sobre el cuadro de texto *Mi ubicación*, el cual se encuentra en la parte inferior izquierda del mapa, se realiza un zoom hacia la ubicación actual del usuario. Es necesario que el receptor GPS del dispositivo esté activado para realizar dicha acción. Si el GPS está desactivado, se muestra un mensaje indicando que se realice su activación. Dependiendo de la conectividad del dispositivo, la ubicación exacta puede demorarse en establecerse.

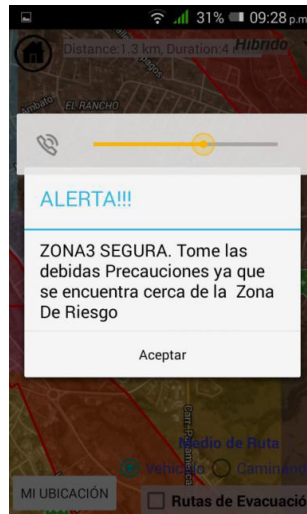


**Figura 80.** Ubicación actual en el mapa

Por medio de la pantalla táctil, se puede seleccionar un lugar específico. Dependiendo de la ubicación seleccionada se muestra un mensaje de alerta indicando si el usuario se encuentra dentro de una zona de alto riesgo o en una zona cercana. Si el usuario se encuentra dentro de un lugar por donde pasa el lahar, el teléfono vibra.

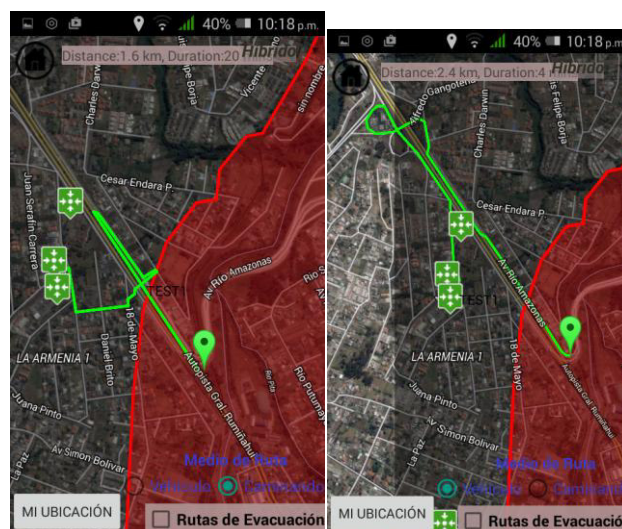


**Figura 81.** Mensaje de alerta indicando que se encuentra en una zona de riesgo



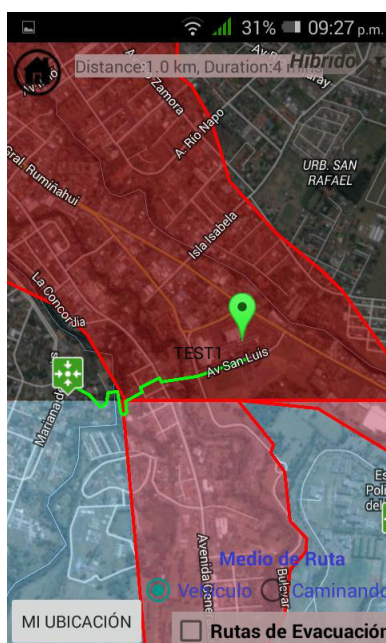
**Figura 82.** Mensaje de alerta indicando que se encuentra en una zona cercana al riesgo

Dependiendo del lugar seleccionado, la aplicación debe recomendar el refugio más cercano por medio de un mensaje de voz. Además, se dibuja sobre el mapa la ruta hacia dicho refugio sobre las carreteras de la zona. El camino puede variar dependiendo si se toma la ruta conduciendo o caminando. Se puede elegir el modo para llegar al destino seleccionando la opción en lado inferior derecho del mapa.



**Figura 83.** Comparación de la ruta que se forma al caminar con la ruta que se forma por medio de un vehículo

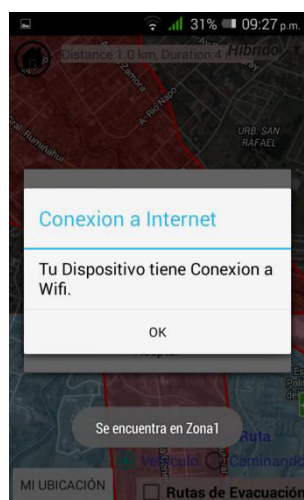
La aplicación realiza la comparación de las posibles distancias hacia los refugios que se encuentran en la zona seleccionada. Después de determinar el refugio con la distancia más corta se procede a dibujar el camino hacia dicho refugio. En la parte superior se muestra la distancia y el tiempo que el usuario se demoraría en recorrer la ruta



**Figura 84.** Generación de ruta a refugio más cercano

Para poder recomendar el refugio más cercano y dibujar la ruta, es necesario que el dispositivo tenga acceso a internet, ya sea por medio de datos móviles como por medio de la conexión a una red inalámbrica. Si el usuario tiene conexión a la red, al seleccionar un sitio, aparece un mensaje indicando que se tiene acceso a internet y se genera la ruta. El tiempo que se demora la aplicación en procesar la ruta dependerá de la velocidad de conexión.





**Figura 85.** Mensaje indicando que el dispositivo tiene conexión wifi

Se ha comprobado que la generación de rutas se realiza de forma inmediata cuando el usuario tiene acceso a internet a través de una red inalámbrica estable. Cuando se realiza la conexión por medio de los datos móviles del dispositivo, existe mayor demora para generar la ruta dependiendo de la velocidad de conexión. En dispositivos con mayor velocidad de conexión se genera la ruta de forma más rápida.

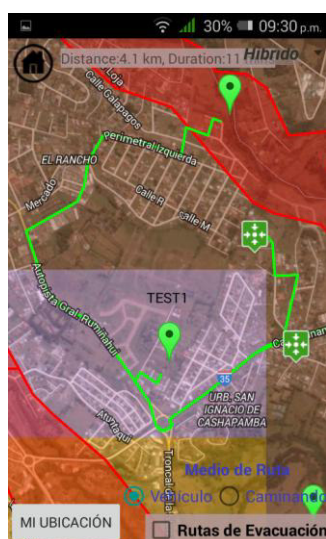
Adicionalmente, en algunos teléfonos se solicita al usuario que se descargue la aplicación de síntesis de voz para que se pueda recomendar el refugio más cercano por medio del mensaje de voz. La voz usada en la aplicación es una voz de mujer que se expresa en español mexicano. La calidad de la voz variará dependiendo de la versión de la aplicación de síntesis de voz descargada.

#### **4.1.1.2 Pruebas preliminares en la actividad del mapa**

Durante el desarrollo de la aplicación se realizaron pruebas preliminares para verificar el comportamiento de la aplicación y realizar mejoras a partir de los resultados obtenidos.

Al realizar pruebas preliminares sobre el mapa en distintos sitios, se pudo verificar en un principio que cuando se genera una ruta al refugio más cercano, a veces se forma un sendero bastante largo para llegar al sitio indicado. Esto se debe a que el portal de Google Directions, el cual permite la generación de estos caminos, no toma en cuenta todas las posibles entradas para ingresar a un destino, por lo que siempre se busca llegar a dicho lugar únicamente a través de la entrada principal. Es por eso que se buscaron mecanismos para poder modificar las direcciones para llegar a un destino y así poder mejorar la generación de rutas. Por medio de la ubicación de marcadores invisibles se pueden generar rutas hacia entradas alternas de un establecimiento.

Por ejemplo, el Liceo del Valle, el cual es un refugio ubicado en la Zona 4, posee una entrada trasera que Google Directions no toma en consideración para trazar el camino, por esa razón se formaba una ruta demasiado larga para llegar a la entrada principal de dicho establecimiento.



**Figura 86.** Ruta generada hacia la entrada principal del Liceo del Valle

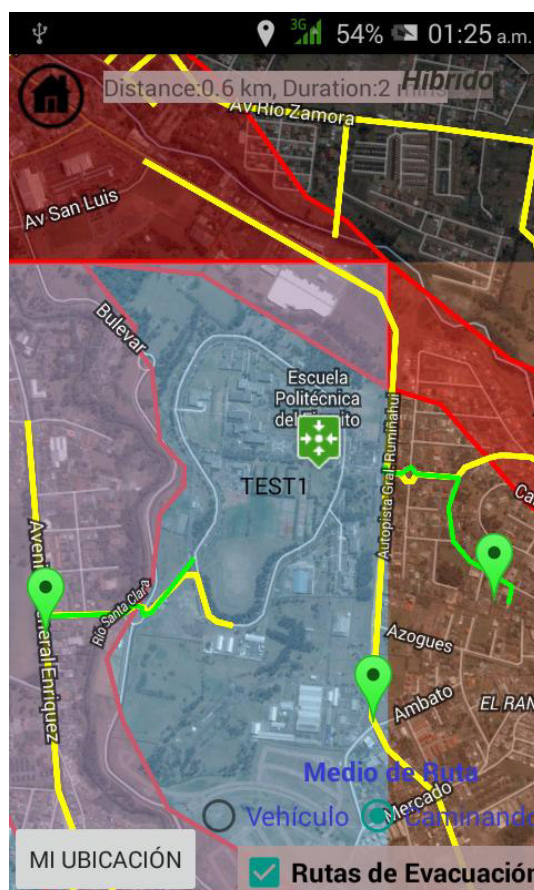


Para solucionar este problema se tuvo que añadir un punto invisible en el extremo de la calle donde se ubica la entrada trasera para que el camino pueda trazarse hacia dicho punto.



**Figura 87.** Ruta generada hacia la entrada alterna del Liceo del Valle

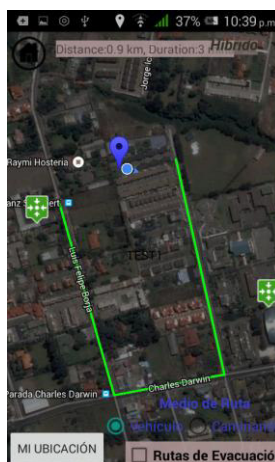
Igualmente existían inconvenientes al trazar caminos hacia la Universidad de la Fuerzas Armadas ya que Google Directions considera que a la entrada principal sólo se puede acceder caminando, mientras que a la entrada trasera solo se puede acceder por medio de vehículos. Se solucionó este problema añadiendo puntos invisibles en las entradas del establecimiento para que se puedan trazar las rutas hacia dichos puntos sin importar el modo.



**Figura 88.** Generación de posibles rutas hacia las entradas de la Universidad de las Fuerzas Armadas

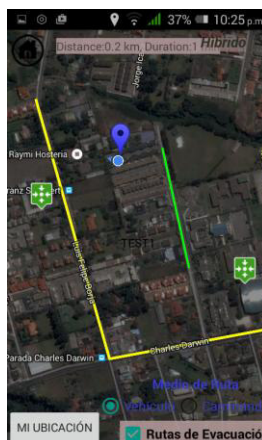
Al realizar pruebas preliminares, se determinó que en algunas ocasiones no se recomendaba el sitio seguro correcto. Esto se debía a que se estaba determinando el refugio más cercano a través del cálculo de distancias en línea recta.

Por ejemplo, dentro de la zona 1 se encuentran los refugios *Franz Schubert* y *Saint Dominic School*. En la ubicación que se muestra en la imagen, el refugio más cercano sería el *Colegio Franz Schubert* debido a que existe una menor distancia en línea recta, sin embargo para llegar hasta dicho refugio se forma un camino mucho más largo que la ruta para llegar al refugio *Saint Dominic School*.



**Figura 89.** Determinación del refugio más cercano calculando la distancia por línea recta

La solución a este inconveniente es determinando el refugio más cercano a partir del cálculo de distancias de los posibles caminos generados automáticamente sobre las calles.

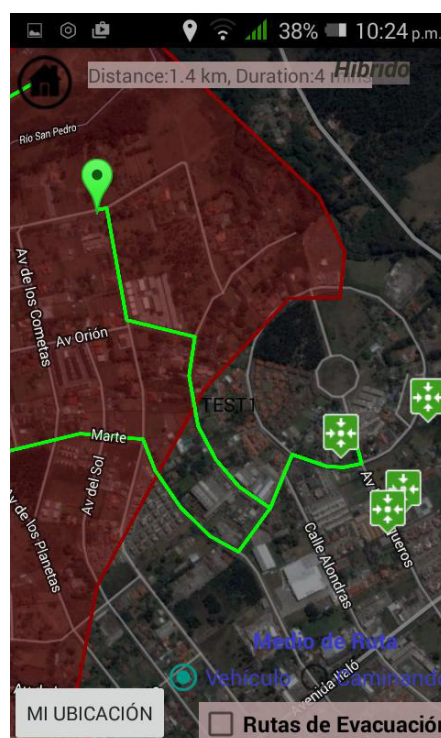


**Figura 90.** Determinación del refugio más cercano calculando la distancia de la ruta

Sin embargo, este procedimiento genera demasiado procesamiento cuando se determina la distancia más corta entre todos los posibles refugios dentro del Valle de los Chillos. Al existir sobrecarga la aplicación se paraliza y no se puede dibujar la ruta. Por esa razón, para determinar cuál es el refugio

más cercano se debe calcular la distancia más corta entre una cantidad menor de refugios ubicados dentro de la zona en la que se encuentra el usuario.

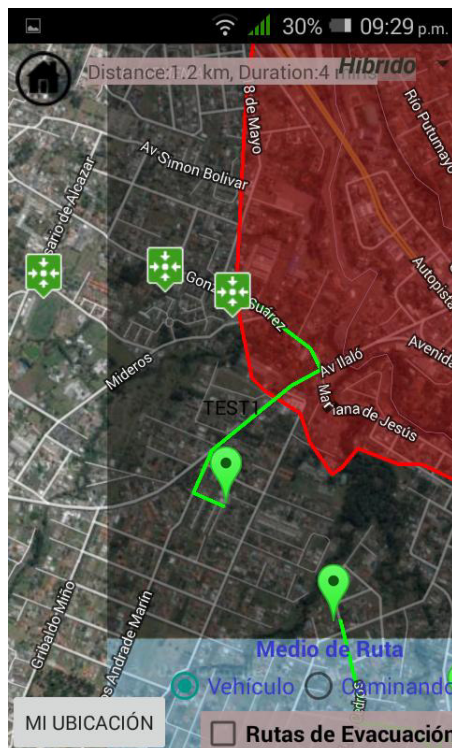
Se ha comprobado que para que no exista sobrecarga y no se presenten errores al generar la ruta, se debe determinar el refugio más cercano entre un número máximo de 10 refugios dentro de cada zona. La zona 1, por ejemplo, es una zona bastante extensa que contiene más de veinte refugios, por esa razón se la ha dividido en 4 subzonas invisibles donde se puede distribuir de mejor forma los refugios y determinar a dónde el usuario debe dirigirse sin generar mucho procesamiento.



**Figura 91.** Generación de rutas hacia el refugio cercano dentro de una sección en la Zona 1

Al generar las rutas hacia los refugios, estas no deben atravesar las zonas por donde pasan los lahares cuando se selecciona una ubicación que se encuentra en una zona segura. Es por esta razón que se debe recomendar ciertos refugios dependiendo de la zona en la que se encuentra el usuario.

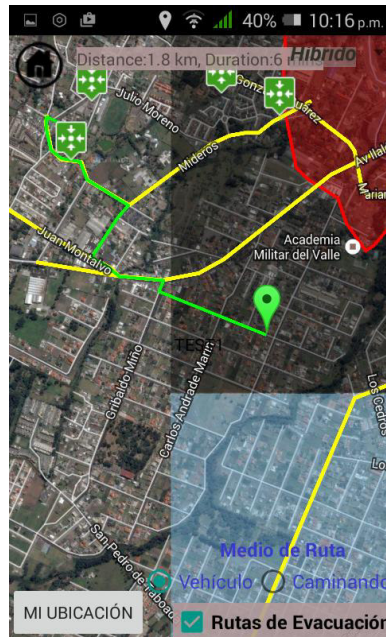
Por ejemplo, las personas que se encuentran en una zona de mayor riesgo en la Zona 1 pueden dirigirse al refugio *Savio Domingo*. Sin embargo, si se encuentra en una zona segura cercana, el usuario no puede dirigirse a dicho refugio ya que la ruta generada pasaría por el lahar.



**Figura 92.** Generación de rutas hacia el refugio cercano pasando por el lahar

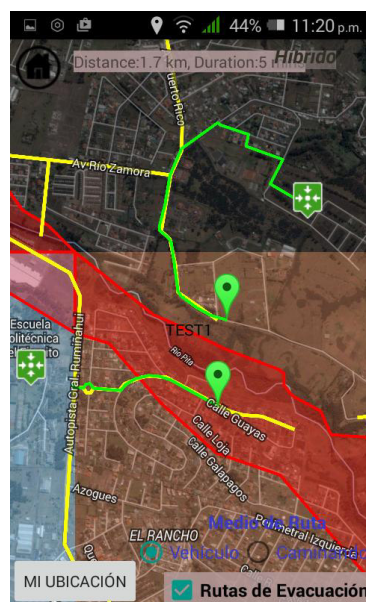
Es por eso que en ese lugar se debe descartar ese refugio y sugerir al usuario que se dirija a otro más seguro como el *Parque La Moya*, evitando cruzar un lugar de alto riesgo.





**Figura 93.** Generación de rutas hacia el refugio cercano por un camino seguro

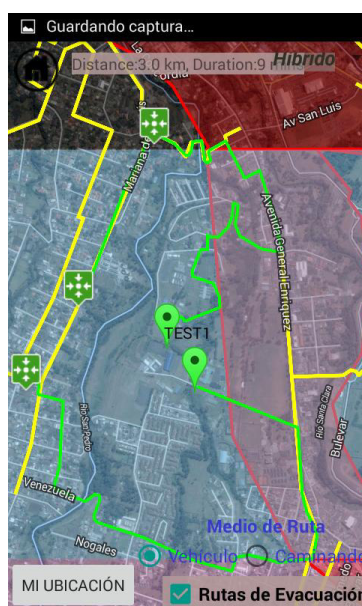
Es posible que sea más factible que un usuario que se encuentra dentro de una zona tenga que dirigirse a un refugio ubicado en otra zona, por lo que se compartieron refugios entre zonas al momento de distribuirlos.



**Figura 94.** Rutas generadas hacia refugios ubicados en otras zonas

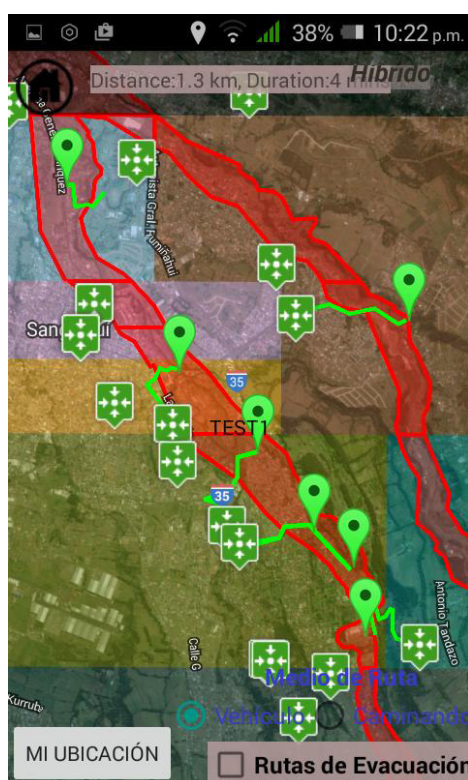
Se debe hacer todo lo posible por evitar que las rutas generadas pasen por los lahares, sin embargo, es posible que en algunos sitios las únicas rutas disponibles de evacuación tengan que pasar obligatoriamente por las zonas de riesgo, por lo que el usuario debería transitar la ruta en el momento preciso antes de que ocurra el desastre.

Un ejemplo de esta situación es lo que se muestra en la figura, donde se selecciona una ubicación dentro de la Zona 2. El camino generado para llegar al refugio más cercano pasa obligatoriamente por los lahares. No se pueden determinar otras rutas alternativas ya que sólo existe un sola ruta de escape en ese lugar. Es por eso que se debería tomar ese camino de forma anticipada.



**Figura 95.** Rutas generadas que obligatoriamente pasan por lahares

Por medio de las mejoras mencionadas, se puede distribuir de mejor manera los refugios dentro de cada zona, permitiendo que se generen las rutas de forma más óptima y evitando en lo mayor posible que se tomen rutas riesgosas. En la figura se visualiza cómo se trazan las rutas a los refugios más cercanos en distintos puntos a lo largo de la zona de riesgo.

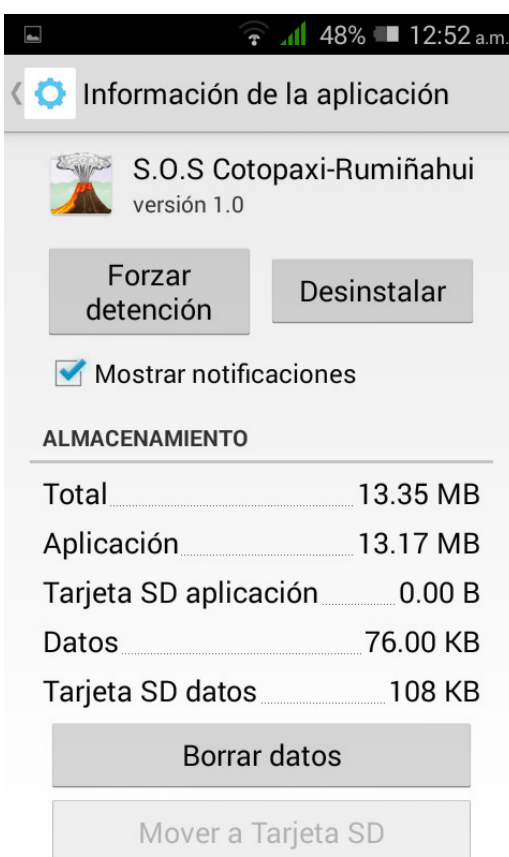


**Figura 96.** Rutas generadas que obligatoriamente pasan por lahares

## 4.2. Requerimientos de la aplicación

Al realizar la instalación el programa, el dispositivo requiere de 13.35 Mb de memoria de almacenamiento, siendo una aplicación bastante liviana. También se requiere que esté instalada la última versión de Google Maps (v9.19.1), que el dispositivo tenga acceso a las coordenadas brindadas por el receptor GPS y que tenga conexión a internet para determinar el camino hacia el refugio más cercano. Estos requerimientos deben ser validados por la aplicación al momento de su instalación.





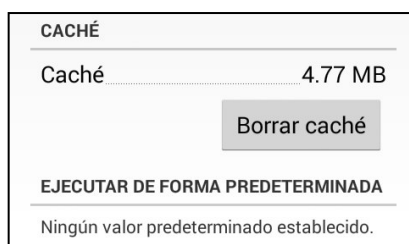
**Figura 97.** Almacenamiento de la aplicación en el dispositivo móvil

La aplicación requiere acceder a los siguientes recursos del teléfono:

- Llamar directamente a números de teléfono.
- Precisar la ubicación del dispositivo (ubicación basada en red).
- Acceso completo a la red: Recibir datos de internet, visualización de conexiones de red de datos o wifi.
- Buscar cuentas del dispositivo. Leer la configuración de los servicios de Google. Utilizar las cuentas del dispositivo.
- Modificar/eliminar contenido de almacenamiento, probar el acceso a un almacenamiento protegido.
- Controlar modo vibrar
- Impedir que el teléfono entre en modo suspensión

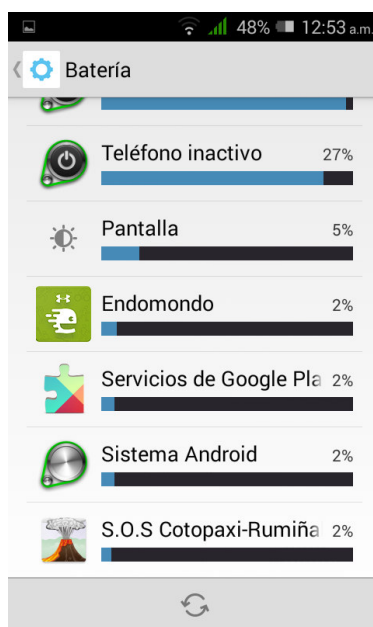
### 4.3. Consumo de recursos de la aplicación

Después de realizar pruebas con la aplicación durante 5 minutos, generando rutas hacia el refugio más cercano en distintos puntos sobre el mapa, se pudo verificar la memoria caché de la aplicación la cual no sobrepasó los 5 Mb durante dicho periodo.



**Figura 98.** Almacenamiento de la memoria caché de la aplicación

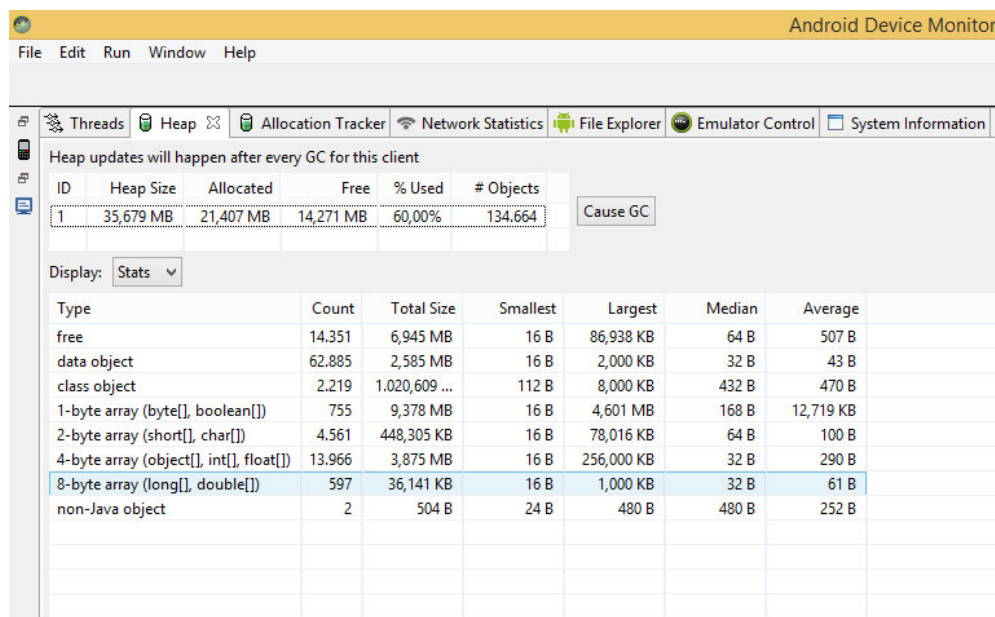
La aplicación utiliza recursos como el receptor GPS o conexión una red inalámbrica, sin embargo, no produce un alto consumo de batería, llegando a generar solamente el 2% del consumo.



**Figura 99.** Consumo de batería de la aplicación

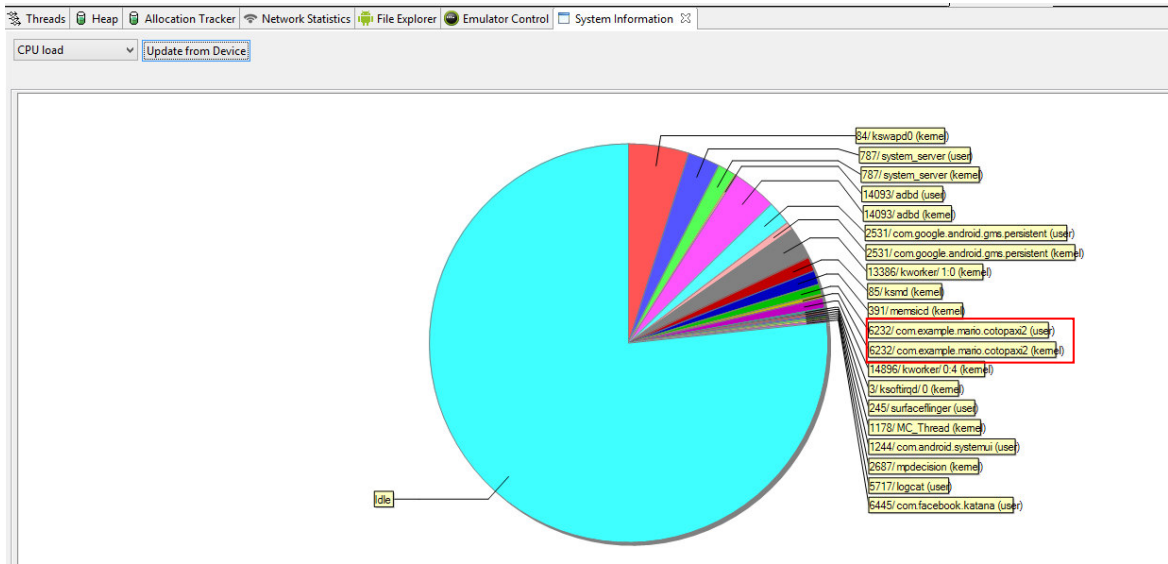
Por medio del Monitor del Dispositivo, el cual es una herramienta integrada en Android Studio, se puede verificar en tiempo real el estado de los recursos utilizados como el uso de la pila, el consumo de memoria y el procesamiento de CPU de la aplicación.

En la siguiente figura se muestra la cantidad de datos que la aplicación genera en la pila. El tamaño total de la pila es de 35.679 MB de la cual se está consumiendo 21.407 MB.



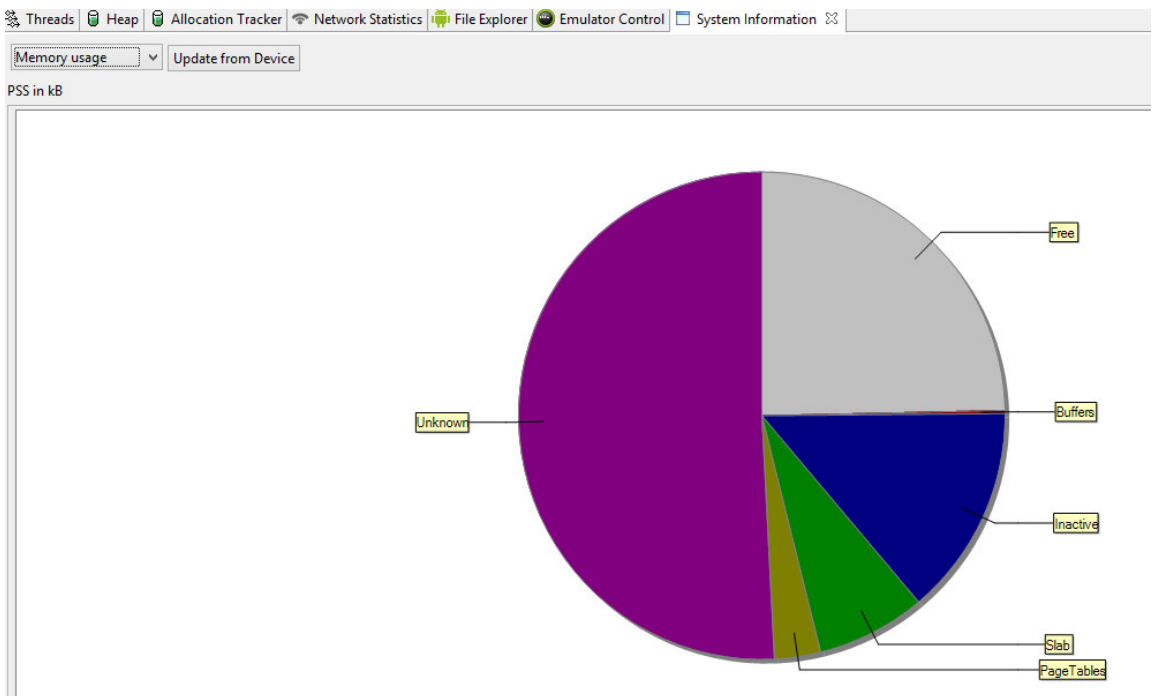
**Figura 100.** Consumo de pila de la aplicación

Al verificar la información del dispositivo, se puede visualizar una gráfica con el consumo del procesamiento de CPU de la aplicación cotopaxi2 (usuario y kernel) con respecto al consumo de otras aplicaciones en el dispositivo. Según la gráfica, la aplicación consume poco procesamiento.



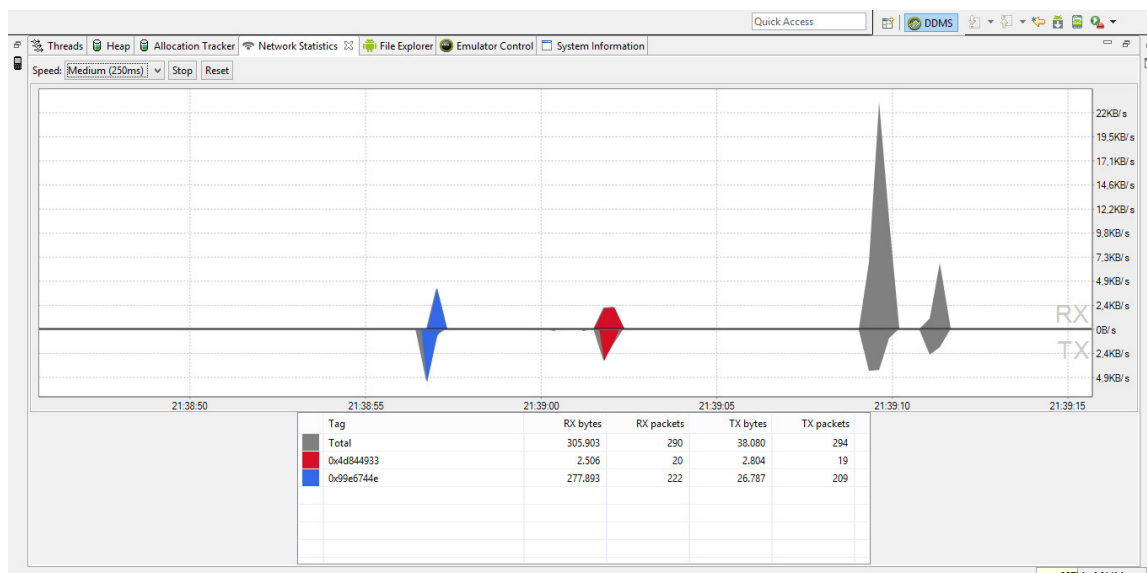
**Figura 101.** Procesamiento de CPU de la aplicación

En la siguiente gráfica se puede visualizar el uso de memoria de los recursos que se manejan en la aplicación.



**Figura 102.** Uso de memoria de la aplicación

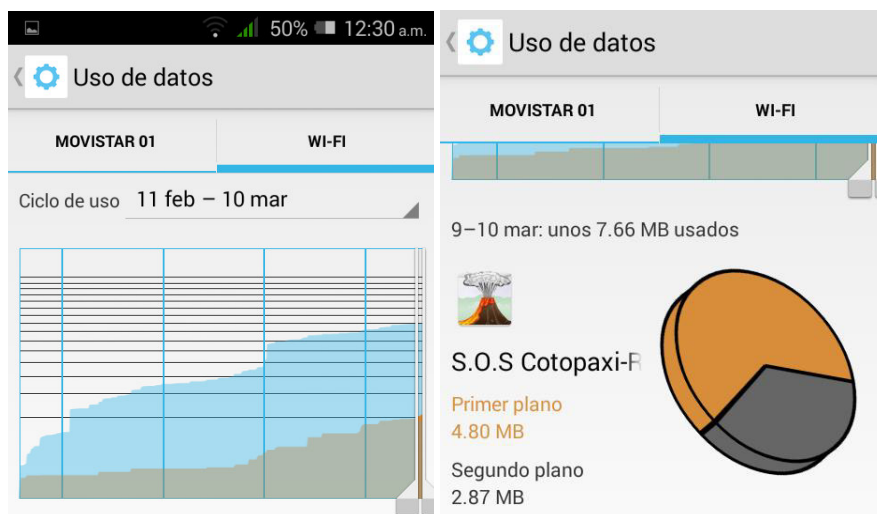
Adicionalmente, por medio del Monitor del Dispositivo se puede visualizar una gráfica con las estadísticas de la red, pudiendo obtener información acerca de la cantidad de datos que se envían y se reciben. Durante la emulación de la aplicación se pudo determinar que se recibieron 305.904 MB a una velocidad máxima de 22KB/s.



**Figura 103.** Estadísticas de Red

La aplicación trabaja correctamente con cualquier dispositivo Android que tenga conexión a internet. Dependiendo de la velocidad de conexión, los resultados de procesamiento para generar las rutas varían. Los teléfonos que funcionan con velocidad 4G LTE permiten que las rutas se generen de forma casi inmediata, mientras que con conexiones menores las rutas tardan más tiempo en generarse. El teléfono debe tener una buena señal para que las rutas puedan dibujarse de forma rápida.

El consumo de datos varía dependiendo de las veces que se requiera determinar el refugio más cercano y dibujar la ruta hacia el mismo sobre el mapa. Durante un periodo de 24 horas se utilizó aproximadamente 7.6 MB después de realizar pruebas con la aplicación.



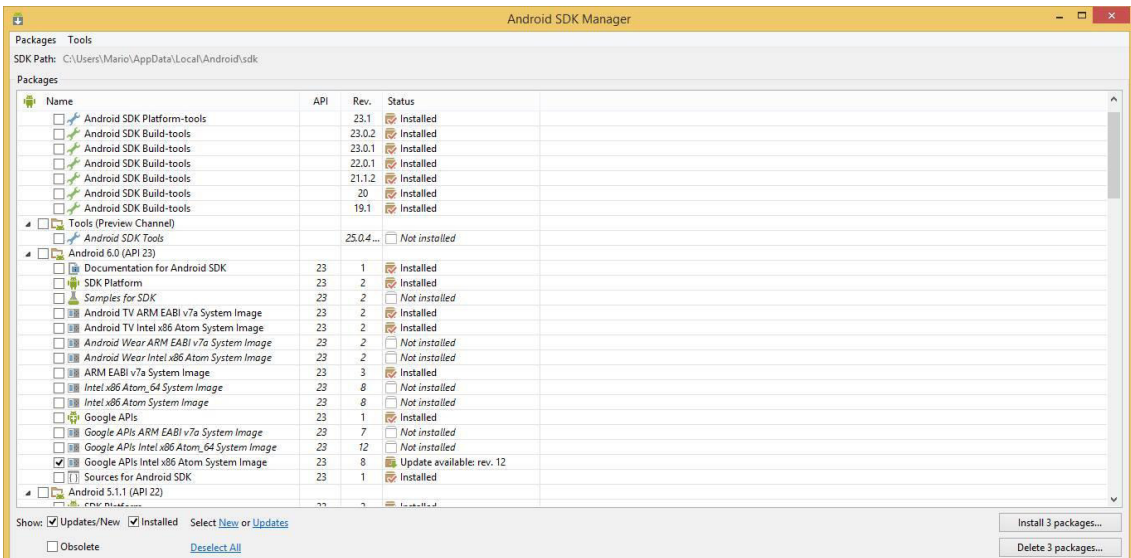
**Figura 104** Uso de datos al conectarse la aplicación a la red wi-fi

La cantidad de datos que la aplicación utiliza al generar una ruta depende de la cantidad de refugios con los que se realiza la comparación dentro de cada zona. Los datos generados varían desde 10 a 100 KB aproximadamente por cada ruta. Adicionalmente, se consumen datos por otros factores como la obtención de la ubicación por medio del receptor A-GPS, también al descargar las imágenes satelitales del mapa por primera vez y al obtener la aplicación de síntesis de voz.

#### 4.4. Compatibilidad de la aplicación

La aplicación trabaja con cualquier dispositivo que funciona con Android. Se han realizado pruebas con distintos modelos de dispositivos Samsung, Sony y Alcatel. Los resultados han sido satisfactorios en estos teléfonos.

En el SKD manager, se configuró la aplicación para que sea compatible con los teléfonos Android desde la versión 2.0 (Eclair) hasta la versión 6.0 (Marshmallow).



**Figura 105.** Configuración de las versiones de Android en el SDK manager

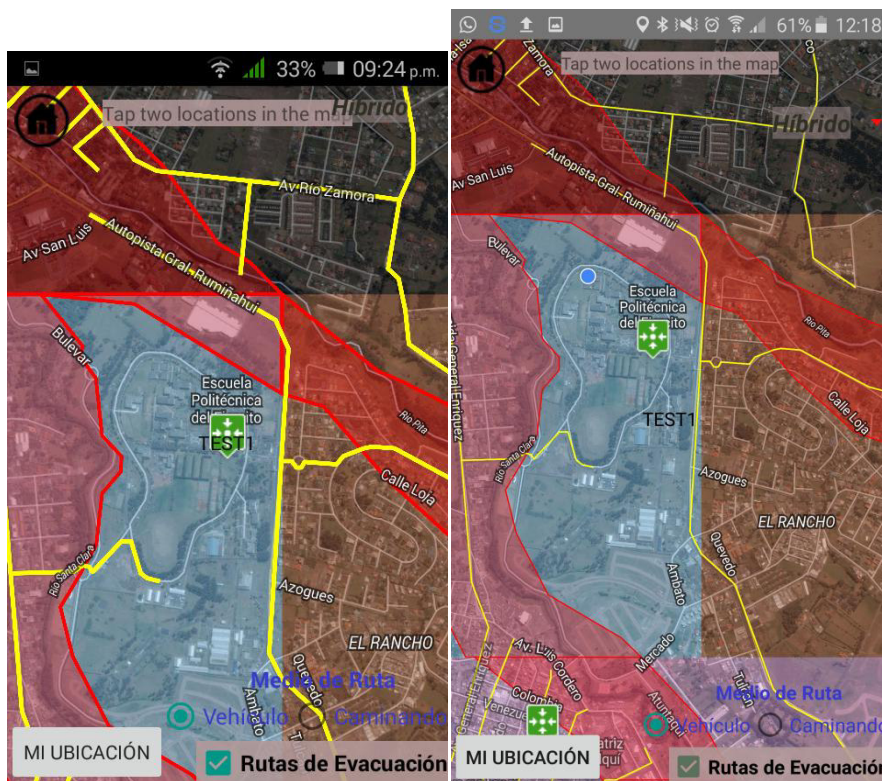
Dependiendo del modelo del dispositivo, la vista de la aplicación cambiará, adaptándose a la pantalla del artefacto el cual puede ser desde un teléfono móvil hasta una tablet. En los menús de la aplicación, los elementos ubicados en las listas se adaptan de mejor manera en dispositivos con mayor resolución y con pantallas más amplias.



**Figura 106.** Comparación del menú principal en un teléfono Alcatel One Touch con un Samsung Galaxy S6



De igual forma, los elementos sobre el mapa se visualizan de mejor forma en teléfonos con mayor resolución.



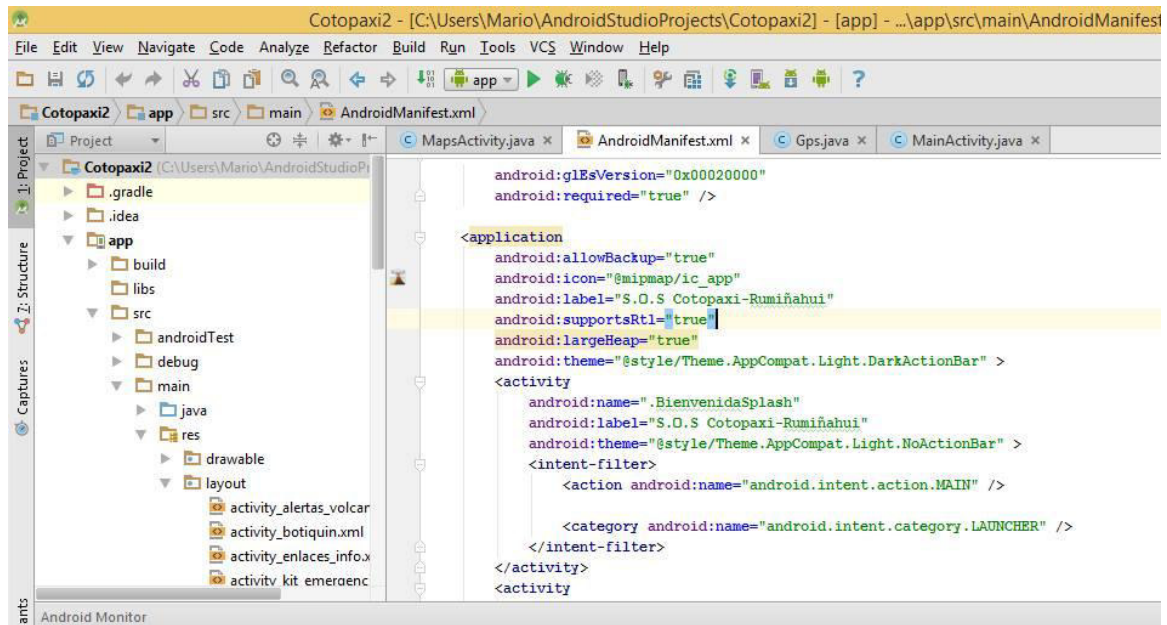
**Figura 107.** Comparación de la actividad del mapa en un teléfono *Alcatel One Touch* con un *Samsung Galaxy S6*

En un inicio, al hacer pruebas en algunos modelos de teléfonos Samsung como los Galaxy S3, S4, S5 y S6, la aplicación se cerraba inmediatamente antes de abrirse el menú principal. Se diagnosticó el problema en estos dispositivos y se determinó que existía falta de memoria al ejecutar el programa. Esto se debía principalmente a que estos modelos manejan una mayor resolución y requieren de mayor espacio de memoria para poder ejecutar las funcionalidades de la aplicación.

Para solucionar este problema se tuvo que aumentar la capacidad de pila de la máquina virtual de la aplicación (Dalvik). En el archivo



*AndroidManifest.xml* de la aplicación se debe asignar el valor “true” en donde dice *Android:largeHeap*.



**Figura 108.** Aumento de la capacidad de la pila del Dalvik

## CAPÍTULO V

### CONCLUSIONES Y RECOMENDACIONES

#### 5.1 Conclusiones

- Por medio del desarrollo de aplicaciones con Google Maps se puede brindar al usuario un esquema interactivo que permita identificar zonas de riesgo y posibles rutas de escape durante una emergencia. Por medio de los polígonos graficados sobre el mapa se logró adaptar las posibles zonas por donde pasarían los lahares resultantes del proceso eruptivo (en caso de llegar a realizarse) dentro de los mapas georeferenciados sobre los cuales se desarrolló la aplicación.
- A pesar de que el dispositivo necesita conectarse a una red de datos y requiere del uso del receptor GPS, la aplicación no consume muchos recursos de memoria y el procesamiento de CPU. La aplicación no genera gran cantidad de datos en la memoria caché, ocupando menos de 10 MB. El consumo de batería también es bajo, ocupando solamente el 2% del consumo total. El consumo de datos de internet es considerablemente bajo aunque varía dependiendo de las veces que se generen las rutas de escape. El procesamiento que se maneja para generar una ruta depende de la cantidad de refugios con los que se determina a dónde debe ir el usuario. El consumo de datos varía entre 10 y 100 KB por ruta. El establecimiento de la ubicación del usuario y el proceso de generación del mapa también consumen recursos de internet.
- Por medio de la API de Google Directions se puede dibujar un camino entre dos puntos siguiendo las rutas sobre el mapa. Se puede además obtener la distancia del trayecto y el tiempo que se demora en recorrerlo, dependiendo si se transita caminando o en un vehículo. En la mayoría de los casos, las rutas generadas son bastante precisas, sin embargo en algunas ocasiones

no se toma en cuenta todas las posibles entradas y salidas de un establecimiento, así como las rutas dentro de entidades privadas, por lo que en algunos casos la ruta generada puede ser demasiado larga debido a que se busca llegar al destino sólo por las entradas principales. Es por eso que se colocaron puntos invisibles en las entradas alternas para que la ruta se pueda generar hacia dichas entradas y de esta forma brindar información real a los usuarios de la aplicación.

- Para determinar la distancia desde un punto dentro de la zona del Valle de los Chillos hasta el refugio más cercano no se ha determinado la distancia hacia el mismo por medio de una línea recta, ya que la distancia que se recorre por las calles para llegar al lugar establecido es distinta. Por tanto se hizo necesario determinar la ubicación y la ruta hacia el refugio más cercano por medio de la distancia que se recorre por las calles, brindando de esta forma información más precisa y real al usuario.
- Se dibujaron zonas y subzonas sobre el mapa para que el usuario pueda identificar cómo está dividida la región. Además, esto permitió que se puedan distribuir los refugios ubicados en cada zona y se logre trazar las rutas de escape hacia los mismos sin que se genere demasiado procesamiento. Se comprobó que se puede determinar el sitio seguro más cercano entre un número máximo de 10 refugios por zona, si se sobrepasa ese valor se genera mayor procesamiento y es posible que el proceso de la aplicación pueda llegar a detenerse.
- La aplicación debe ayudar al usuario a dirigirse al refugio más cercano a través del camino más seguro antes de que se superen los tiempos previstos de llegada de los lahares en caso de una eventual erupción. Es por eso que en cada zona se realiza la comparación entre una cierta cantidad de refugios para evitar que el usuario se dirija a un refugio por medio de un camino el cual este atravesando una zona riesgosa.
- La ruta generada debe atravesar la zona de peligro solamente cuando el usuario se encuentra ubicado en una zona de mayor riesgo. Esta ruta debe

dirigir al usuario en el menor tiempo posible a un refugio ubicado en sitio seguro. Una excepción a esto es cuando existe una única ruta de escape que obligatoriamente pasa por una zona de peligro, en ese caso el usuario debe recorrer esa ruta inmediatamente antes de que lleguen a pasar los lahares.

- Las rutas dibujadas de forma predeterminada sobre las calles en el mapa permiten que el usuario pueda verificar las posibles rutas de escape en su zona y las compare con la ruta generada automáticamente. De esta forma el usuario puede decidir cuál sería el camino más adecuado para poder escapar.
- Existe cierto margen de error al establecer las coordenadas exactas brindadas por el receptor GPS del dispositivo. Esto se debe a que el dispositivo tarda cierto tiempo en obtener la ubicación exacta la cual depende de la señal de los satélites y de la conexión a la red. Inicialmente se establece una ubicación dentro de un determinado radio hasta obtener las coordenadas exactas. La velocidad que se requiere para obtener la ubicación precisa depende de la conexión a la red de internet la cual puede ser una red inalámbrica o la red datos de la compañía telefónica.
- Se realizó la programación de la aplicación por medio de la versión 2 de Google Maps debido a que esta brinda sencillez a la hora de programar y adicionalmente se encontró una gran cantidad de guías donde se manejaba esta versión. La versión 3 de Google Maps es más avanzada y brinda funcionalidades que facilitan el desarrollo de algunos parámetros sobre el mapa, sin embargo existe mayor dificultad para entender la programación en esta versión y el proceso para traspasar la programación de la versión 2 a la versión 3 es bastante compleja.
- Debido a que la programación en Android es de código abierto, existe una gran comunidad de programadores que comparten sus avances para el desarrollo de diversas aplicaciones. Es por eso que gracias a esta comunidad se encontraron las herramientas necesarias que facilitaron el

desarrollo de algunas funcionalidades de esta aplicación como por ejemplo la identificación de un punto dentro de un polígono, o el trazo automático de rutas sobre las calles.

- En un principio se consideró implementar una base de datos la cual almacenaría información de coordenadas y ubicaciones. Además, se la utilizaría para controlar el acceso de los usuarios a la aplicación. Sin embargo, debido a que esta es una aplicación de emergencia, no es necesario que los usuarios tengan que registrarse para poder utilizarla. Además, es más factible que la aplicación acceda a la información de las coordenadas de forma inmediata sin tener que acceder a un servidor externo.
- En teléfonos de alta resolución, como la línea de Samsung Galaxy, se requiere de un mayor uso de memoria para poder ejecutar las funcionalidades de la aplicación. Es por eso que durante el desarrollo del programa se aumentó la pila del Dalvik para que la aplicación soporte la resolución de dichos teléfonos.
- La aplicación funciona correctamente para cualquier modelo actual de dispositivos que manejan Android. Sin embargo, la forma cómo se muestra la interfaz de la aplicación, así como la velocidad de procesamiento dependen de la resolución y de la capacidad del teléfono. En pantallas más amplias los menús se ajustan de mejor forma y las funcionalidades sobre el mapa se ejecutan de forma más rápida si se tiene mayor capacidad de memoria interna.

## **5.2 Recomendaciones**

- Es recomendable para los usuarios que verifiquen con anticipación las rutas generadas y las comparen con la información de los organismos oficiales ya que la API de Google Directions toma en cuenta ciertos criterios para dibujar

la ruta los cuales posiblemente sean erróneos. El usuario puede manejar la aplicación durante simulacro o de forma preliminar para comprobar que la ruta sea segura y confiable.

- Debido a que la aplicación requiere de acceso a internet para generar las rutas de escape, es recomendable que se la utilice como una herramienta de prevención ya que durante un evento de emergencia las redes inalámbricas pueden caerse por apagones o por pérdida de señal. Sin embargo, es posible que la red de datos brindada por las principales compañías telefónicas siga manteniéndose activa.
- Se recomienda que la aplicación sea verificada por los organismos encargados de la emergencia antes de su distribución ya que en esta situación no se puede brindar información a los usuarios que posiblemente sea errónea. Sin embargo, durante el desarrollo de esta aplicación se tomaron en cuenta los diagramas presentados por la Secretaría de Gestión de Riesgo para poder dibujar las zonas de riesgo de la forma precisa.
- Se recomienda hacer uso de esta aplicación para que el municipio y la Secretaría de Gestión de Riesgos puedan identificar qué rutas son las más adecuadas para llegar a un sitio seguro. Basándose en esto, los organismos encargados pueden mejorar los señalamientos y pintar las rutas sobre las calles que no aún no han sido señaladas.
- Para mejorar las rutas generadas automáticamente entre un sitio a otro se podría hacer uso de la herramienta *MapMaker* la cual permite sugerir a Google los posibles caminos para llegar a un establecimiento y notificar posibles entradas alternas. Sin embargo esta es una herramienta que no está disponible todavía para realizar modificaciones de los sitios ubicados en Ecuador. Se recomendaría hacer uso de esa herramienta cuando Google habilite esta opción en el país. (Google, 2016)
- Sería recomendable que en un futuro se mejore la aplicación para que tenga la posibilidad de dirigir al usuario en tiempo real siguiendo la ruta hacia un sitio seguro por medio de mensajes de voz que indiquen hacia donde se

debe girar. Sin embargo, las políticas de Google Maps y Google Earth establecen que no se deben desarrollar aplicaciones que implementen funcionalidades *de navegación en tiempo real*. Se requeriría de algún tipo de licencia para poder implementar dicha funcionalidad en la versión 2 de Google Maps. (Google Developers, 2015)

- La aplicación se desarrolló principalmente para cubrir las principales zonas residenciales del Valle de los Chillos. Se recomienda posteriormente expandir la zona de análisis hacia el resto de lugares afectados y desarrollar una aplicación más extensa y de mayor alcance.
- Esta aplicación puede ser tomada como una base para la prevención ante fenómenos similares, donde los habitantes puedan identificar las mejores rutas para poder evacuar con seguridad y en el menor tiempo posible.

## BIBLIOGRAFÍA

- Secretaría de Seguridad y Gobernabilidad - Municipio de Quito. (2015). *Plan de Contingencia del Volcán Cotopaxi*. Obtenido de [http://www.safi.com.ec/manager/uploads/files/Plan\\_Contingencia%20\\_Volc%C3%A1n\\_Cotopaxi.pdf](http://www.safi.com.ec/manager/uploads/files/Plan_Contingencia%20_Volc%C3%A1n_Cotopaxi.pdf)
- Android Developers. (s.f.). *Android Studio*. Obtenido de <http://developer.android.com/intl/es/sdk/index.html>
- Android Developers. (s.f.). *Using the Emulator*. Obtenido de <http://developer.android.com/intl/es/tools/devices/emulator.html>
- AssemblySys dataServices. (2013). *Algoritmo del punto en un polígono en PHP*. Obtenido de <http://assemblysys.com/es/algoritmo-punto-en-poligono/>
- aumentame. (2011). *Tipos de Realidad Aumentada*. Obtenido de <http://aumenta.me/node/36>
- Bioadrian. (2015). *El GPS y su funcionamiento*. Obtenido de <http://bioadrianllamas.blogspot.com/2015/11/el-gps-y-su-funcionamiento.html>
- Draw path between two points using Google Maps Android API v2*. (2013). Obtenido de <http://stackoverflow.com/questions/14702621/answer-draw-path-between-two-points-using-google-maps-android-api-v2>
- Draw Which is more accurate method for finding distance between two locations in google maps in android? distanceTo or Distance?* (2015). Obtenido de <http://stackoverflow.com/questions/33496128/which-is-more-accurate-method-for-finding-distance-between-two-locations-in-goog>
- El Comercio. (14 de Septiembre de 2015). *24 minutos tomó la evacuación del colegio Giovanni Farina en Los Chillos*. Obtenido de <http://www.elcomercio.com/actualidad/evacuacion-estudiantes-colegio-farina-valle.html>



- El Comercio. (16 de Agosto de 2015). *En la Armenia II se realizó una charla de prevención sobre el volcán Cotopaxi.* Obtenido de <http://www.elcomercio.com/actualidad/armenia-charla-prevencion-volcan-cotopaxi.html>
- El Comercio. (4 de Septiembre de 2015). *En las calles de Latacunga se señalizan las rutas de evacuación por el Cotopaxi.* Obtenido de <http://www.elcomercio.com/actualidad/latacunga-senalizacion-zonasseguras-volcancotopaxi.html>
- El Comercio. (18 de Noviembre de 2015). *Muro de La Caldera para encauzar los lahares despierta interrogantes.* Obtenido de <http://www.elcomercio.com/actualidad/prevencion-lahares-volcancotopaxi-muro.html>
- El Comercio. (2015). *Simulacro Nacional.* Obtenido de <http://www.elcomercio.com/tag/simulacro-nacional>
- El Comercio. (17 de Marzo de 2016). *La construcción de obras de mitigación por el Cotopaxi avanzan en Quito.* Obtenido de <http://www.elcomercio.com/actualidad/construccion-obras-mitigacion-cotopaxi-avanzan.html>
- Google. (2016). *Acerca de Map Maker: Países para los que se están creando mapas.* Obtenido de <https://support.google.com/mapmaker/answer/155415?hl=es>
- Google. (2016). *Ayuda de Google Maps.* Obtenido de <https://support.google.com/maps/?hl=es#topic=3092425>
- Google. (2016). *Google Maps.* Obtenido de <https://www.google.com.ec/maps>
- Google. (2016). *Introducción a Google My Maps.* Obtenido de <https://support.google.com/mymaps/answer/3024396?hl=es>
- Google. (2016). *My Maps.* Obtenido de [https://www.google.com/maps/d/?hl=en\\_US&app=mp](https://www.google.com/maps/d/?hl=en_US&app=mp)

- Google Developers. (2015). *Google Maps Android API: Primeros Pasos*.  
Obtenido de [https://developers.google.com/maps/documentation/android-api/start?hl=es#paso\\_2\\_instalar\\_el\\_sdk\\_de\\_google\\_play\\_services](https://developers.google.com/maps/documentation/android-api/start?hl=es#paso_2_instalar_el_sdk_de_google_play_services)
- Google Developers. (2015). *Google Maps/Google Earth APIs Terms of Service*.  
Obtenido de <https://developers.google.com/maps/terms#10-license-restrictions>
- Google Developers. (2016). *API de Google Maps*. Obtenido de <https://developers.google.com/maps/faq?hl=es#google-maps-api-services>
- Google Developers. (2016). *Google Maps Android API*. Obtenido de Google Maps Android API:  
<https://developers.google.com/maps/documentation/android-api/intro?hl=es-419>
- Google Developers. (2016). *Google Maps Android API: Cámara y Vista*.  
Obtenido de <https://developers.google.com/maps/documentation/android-api/views?hl=es>
- Google Developers. (2016). *Google Maps Android API: Formas*. Obtenido de <https://developers.google.com/maps/documentation/android-api/shapes>
- Google Developers. (2016). *Google Maps Android API: Marcadores*. Obtenido de <https://developers.google.com/maps/documentation/android-api/marker>
- Google Developers. (2016). *Google Maps Android API: Primeros Pasos*.  
Obtenido de [https://developers.google.com/maps/documentation/android-api/start?hl=es#paso\\_2\\_instalar\\_el\\_sdk\\_de\\_google\\_play\\_services](https://developers.google.com/maps/documentation/android-api/start?hl=es#paso_2_instalar_el_sdk_de_google_play_services)
- GPS. (2013). Obtenido de <http://informaticamascomputacion.blogspot.com/2013/03/gps.html>
- GPS World. (2002). *Assisted GPS: A Low-Infrastructure Approach*. Obtenido de <http://web.archive.org/web/20090301054909/http://www.gpsworld.com/gpsworld/article/articleDetail.jsp?id=12287>

- GPS.gov. (2016). *Space Segment*. Obtenido de <http://www.gps.gov/systems/gps/space/2016>
- Hernandez, H. (2013). *La Historia de Android*. Obtenido de <http://www.malavida.com/post/la-historia-de-android>
- King, K. F. (2009). *Geolocation and Federalism on the Internet: Cutting Internet Gambling's Gordian Kno*. Obtenido de [http://papers.ssrn.com/sol3/papers.cfm?abstract\\_id=1433634](http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1433634)
- Latorre, A. (2013). *Desarrollando en Android #2 Google Maps API*. Obtenido de <http://www.elandroidelibre.com/2013/10/desarrollando-en-android-2-google-maps-api.html>
- Ministerio Coordinador de Seguridad. (2015). *Volcán Cotopaxi*. Obtenido de <http://www.seguridad.gob.ec/volcancotopaxi/>
- Mio. (2010). *El GPS a fondo*. Obtenido de [http://webcache.googleusercontent.com/search?q=cache:8YQG6JfGOjQJ:eu.mio.com/es\\_es/sistema-posicionamiento-global\\_4990.htm+&cd=3&hl=es-419&ct=clnk&gl=ec](http://webcache.googleusercontent.com/search?q=cache:8YQG6JfGOjQJ:eu.mio.com/es_es/sistema-posicionamiento-global_4990.htm+&cd=3&hl=es-419&ct=clnk&gl=ec)
- Moya, P. (2015). *Google Maps cumple 10 años: historia y evolución*. Obtenido de El Android Libre: <http://www.elandroidelibre.com/2015/02/google-maps-cumple-10-anos-historia-y-evolucion.html>
- Obemapa. (2011). *Las Coordenadas de Google maps*. Obtenido de <http://www.orbemapa.com/2011/08/las-coordenadas-de-google-maps.html>
- Perez, D. (Noviembre de 2015). *Android Studio 2.0, crea apps el doble de rápido con el entorno de desarrollo de Google*. Obtenido de <http://www.elandroidelibre.com/2015/11/android-studio-2-0-crea-apps-el-doble-de-rapido-con-el-entorno-de-desarrollo-de-google.html>
- Randolph Franklin, W. (21 de Enero de 2014). *Point Inclusion in Polygon Test*. Obtenido de [https://www.ecse.rpi.edu/Homepages/wrf/Research/Short\\_Notes/pnpoly.html](https://www.ecse.rpi.edu/Homepages/wrf/Research/Short_Notes/pnpoly.html)

- Ribas Lequerica, J. (2014). *Manual Imprescindible de Desarrollo de Aplicaciones para Android*. Madrid: Anaya Multimedia.
- Roth, S. (1982). *Ray casting for modeling solids*. . Obtenido de <http://www.sciencedirect.com/science/article/pii/0146664X82901691>
- Secretaría de Gestión de Riesgos. (2015). *Volcán Cotopaxi: Zonas de mayor peligro de lahares*. Obtenido de <https://www.google.com/maps/d/viewer?mid=zGbH7Ti1d4Wo.koA0ESubpzRU>
- Tomás Girones, J. (2015). *El Gran libro de Android*. Barcelona: Marcombo.
- Volcán Cotopaxi Pase o no Pase*. (2015). Obtenido de <https://www.facebook.com/Volc%C3%A1n-Cotopaxi-Pase-o-No-Pase-1720749308161056/>