UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

La Universidad Católica de Loja

ÁREA TÉCNICA

TITULACIÓN DE INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Implementación de una aplicación de activación de emergencia y rescate en dispositivos móviles

TRABAJO DE FIN DE TITULACIÓN

Autor: Sancho Sánchez, Luis Aurelio

Director: Torres Tandazo, Rommel Vicente, Ing PhD

LOJA – ECUADOR

2014

APROBACIÓN DEL DIRECTOR DEL TRABAJO DE FIN DE TITULACIÓN

Ingeniero PhD

Rommel Vicente Torres Tandazo

DOCENTE DE LA TITULACIÓN

De mi consideración:

El presente trabajo de fin de titulación: Implementación de una aplicación de activación de emergencia y rescate en dispositivos móviles realizado por Sancho Sánchez Luis Aurelio, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo.

Loja, abril de 2014

f) . . . . . .

DECLARACIÓN DE AUTORÍA Y CESIÓN DE DERECHOS

“ Yo Sancho Sánchez Luis Aurelio declaro ser autor (a) del presente trabajo de fin de titulación: Implementación de una aplicación de activación de emergencia y rescate en dispositivos móviles, de la Titulación de Ingeniero en Sistemas Informáticos y Computación, siendo Rommel Vicente Torres Tandazo Ing PhD director (a) del presente trabajo; y eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones legales. Además certifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo, son de mi exclusiva responsabilidad.

Adicionalmente declaro conocer y aceptar la disposición del Art. 67 del Estatuto Orgánico de la Universidad Técnica Particular de Loja que en su parte pertinente textualmente dice: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad”

f. .....
Autor: Sancho Sánchez Luis Aurelio
Cédula: 1900560127

DEDICATORIA

El presente trabajo está dedicado especialmente a mis padres Luis Sancho y Rosa Sánchez quienes han sabido brindarme su ayuda en todo momento y por haberme permitido cumplir con mis metas planteadas, a mi hermano y hermanas que día a día fortalecen mi vida en especial en aquellos momentos más difíciles.

A todos mis profesores y amigos con los que he compartido una carrera universitaria, en especial por aquellos que supieron brindarme su apoyo y amistad en todo momento.

Sancho Sánchez Luis Aurelio

AGRADECIMIENTOS

Agradecer a mis padres por los consejos y la ayuda prestada durante la realización de este proyecto y por todo el apoyo recibido durante estos años. Así mismo agradecer a mis hermanos y tíos que desinteresadamente han sabido ser mis guías y compañeros en momentos muy difíciles. En definitiva, agradecer a todas aquellas personas que en mayor o menor medida han ayudado a que este proyecto se desarrollase.

Índice de Contenidos

Índice de figuras

Índice de tablas

RESUMEN

Con el creciente ritmo de desarrollo de aplicaciones para dispositivos móviles se dispone de diversos sistemas de localización, siendo el GPS el de mayor utilización y con lo cual es posible conocer la ubicación actual de una persona en cualquier momento. Además posibilita el desarrollo de soluciones que permitan una interacción más cercana entre dos o más personas independientemente del entorno geográfico en el que se encuentren.

En el presente proyecto se pretende desarrollar una aplicación móvil basada en el sistema operativo android.

Con la finalidad de ayudar a los usuarios en las tareas de evacuación en situaciones de emergencia, presentando a dichos usuarios una ruta de salida más óptima desde la ubicación actual hacia una zona de evacuación más cercana, la cual previamente debe ser establecida en una zona segura y lejos de los posibles obstáculos que hayan sido reportados.

Esta tesis aporta con el desarrollo de una aplicación en una versión inicial, la misma que puede ser actualizada a fin de mejorar sus prestaciones.

PALABRAS CLAVES: Android, GPS, dispositivos móviles, software, aplicaciones

ABSTRACT

With the increasing pace of development of applications for mobile devices have various tracking systems, GPS is the most commonly used, as a result, it is possible to know the current location of a person at any time. Furthermore is possible to develop solutions that enable closer interaction between two or more persons irrespective of the geographical environment in which they are located.
In the present project is to develop an android based operating system that will make use of mobile technology aforementioned application.

This application will have a priority in order to help users in the tasks of evacuation in emergency situations, presenting users a optimal exit route from the current location to a another security location, which must be previously set to a safe and away from any obstacles that have been reported.

This thesis contributes to the development of an application on an initial version, it may be updated in order to improve its performance.

KEYWORDS: Android, GPS, mobile devices, software, applications

INTRODUCCIÓN

Motivación

El importante despliegue que han tenido las tecnologías de comunicación hoy en día supone una visión diferente de la forma en que el ser humano se comunica con su entorno social, a tal punto que la distancia o ubicación geográfica ya no es un impedimento determinante.
Con el continuo avance de las tecnologías y gracias a la masiva colaboración de usuarios es posible que se tenga al alcance cualquier información que quizá en épocas anteriores era imposible.

Así también es de marcada importancia tomar en cuenta que hoy en día el tema de movilidad cada vez tome fuerza y conjuntamente con ello las aplicaciones que permitan interactuar con otro usuario independientemente de su situación (chat, llamadas, ocio, situaciones de emergencia, avisos, mensajería,. . .), manteniendo presentes dos aspectos importantes: la portabilidad y la facilidad de uso.

El tema de movilidad no solo se puede aplicar a situaciones habituales de comunicación sino que también se lo puede hacer en situaciones de emergencia y rescate, siendo en esos momentos de vital importancia contar con elementos que nos ayuden a encontrar un centro de evacuación, facilitando las operaciones de rescate al personal encargado mediante la notificación de la posición exacta de cada víctima existente en el lugar del desastre.

Teniendo clara la idea de la movilidad existente y las diferentes necesidades de comunicación que se presente, es necesario elegir una plataforma de desarrollo adecuada para dar cobertura a situaciones de emergencia y rescate que, como ya se ha comentado es de vital importancia.

Dada la existencia del sistema operativo orientado a dispositivos móviles Android y su rápida expansión tanto en el mercado como en la comunidad de desarrolladores y acogido por numerosas marcas de terminales móviles y un alto número de usuarios, se vio una gran oportunidad para explorar su entorno de desarrollo y las posibilidades que ofrecía. Además, fue un factor importante para tomar esta decisión el hecho de que sea un sistema operativo abierto, lo que permite al desarrollador llevar a cabo sus ideas y poder usarlas de primera mano para su desarrollo.

Antecedentes

Con el creciente ritmo de desarrollo de nuevas aplicaciones para dispositivos móviles, el cual se ha incrementado en los últimos años debido al surgimiento de nuevas tecnología sumado a ello la necesidad de agilitar los tiempos de evacuación de víctimas en situaciones de emergencia ha llevado a una demanda de desarrollo de aplicaciones que permitan agilitar el proceso de evacuación.

Ante ello surge la necesidad de desarrollo una aplicación para dispositivos móviles que permitan a los organismos de socorro ubicar las zonas declaradas como seguras para que sea el aplicativo el cual indique al usuario su mejor ruta de salida.

Todo el funcionamiento se orienta hacia una red wifi de corto alcance.

Situación Problemática

De acuerdo a la problemática de la inexistencia de una aplicación que ayude a los organismos de socorro en sus labores y que sea capaz de funcionar entornos independientes de información proveniente de internet se decidió el desarrollo de una aplicación que se adapte al entorno mencionado.

Las aplicaciones que posteriormente se presentan no cumplen con el requisito de funcionar en una red wifi localmente lo cual constituye un problema de operatividad en entornos pequeños como una intranet.

Objetivos de la Tesis

Objetivo general

Mejorar el tiempo de evacuación de personas en situación de emergencia con la ayuda de una aplicación móvil basada en android.

Objetivos específicos

Que tienes que hacer para llegar a poder realizar la aplicación

Mejorar tiempo de evacuación de personas en situación de emergencia.

- Visualizar todos los dispositivos móviles activos en un mapa.
- Gestionar las transmisiones de datos desde y hacia un dispositivo android.
- Visualizar en un dispositivo android el mapa con el lugar de la evacuación.

- Visualizar en un dispositivo android los obstáculos existentes para llegar al lugar de evacuación.

Hipótesis

La trasmisión de datos entre dispositivos móviles en situaciones de emergencia depende exclusivamente del estado de conectividad existente en el lugar de los hechos, dando como resultado retardo de los tiempos de respuesta a rescates.

Estructura de la Tesis

El presente proyecto se encuentra estructurado en 6 capítulos, los mismos que se describen a continuación:

Capítulo 1. Marco de referencia

En este apartado se presentan aspectos generales a considerar. Tales aspectos hacen mención al entorno de aplicaciones orientadas a situaciones de emergencia utilizando un dispositivo móvil.

Capítulo 2. Solución propuesta

En este capítulo se detalla la solución que se plantea para el desarrollo de una aplicación que permita conocer las posibles rutas de salida hacia un lugar de evacuación desde la ubicación actual del usuario utilizando un dispositivo con sistema operativo android.

Capítulo 3. Especificación de requerimientos de software

Esta parte del desarrollo del proyecto se define las necesidades por las cuales se va a desarrollar una solución que permita la mejora de los tiempos de respuesta de organismos de socorro ante una emergencia.
Así, se definen los requerimientos de usuario o funcionales y los requerimientos de aplicación o no funcionales.

Capítulo 4. Diseño de la solución

En este capítulo se establecen las estrategias que se utilizaron para desarrollar una solución informática que permita reducir los tiempos de respuesta de organismos de socorro

ante una eventual emergencia haciendo uso de los dispositivos móviles existentes en el mercado. Para lo cual se definen aspectos de software como la arquitectura a utilizar para la funcionalidad de la aplicación dentro de un entorno real.

Capítulo 5. Desarrollo de la solución

Este capítulo se enfoca en el desarrollo de una aplicación móvil capaz de cubrir los requerimientos propuestos, la solución a desarrollar está orientada a dispositivos móviles android.

Capítulo 6. Pruebas de funcionalidad

Capítulo que hace referencia a los resultados obtenidos de las pruebas realizadas a la aplicación en un entorno y dispositivo real, arrojando resultados que demuestren la funcionalidad y aplicación de la misma.

Metodología

En el presente proyecto se hace uso de la metodología de desarrollo ágil XP debido a que contempla un desarrollo iterativo e incremental, lo cual nos permite agregar pequeñas mejoras durante el desarrollo del proyecto.

CAPÍTULO 1
MARCO DE REFERENCIA

1.1. Android

En los últimos años, la popularidad de los dispositivos móviles con pantalla táctil fue destacada primeramente por el iPhone el cual ha evolucionado en el mercado en cuanto a la venta y el reciente negocio de las aplicaciones móviles. Los dispositivos móviles en sus últimos modelos se han convertido en los ordenadores del futuro con un gran valor añadido: disponibilidad, comodidad y portabilidad total.

Un estudio publicado por la IDC (International Data Corporation)[1] revela que android está en tres de cada cuatro dispositivos que se vendieron en el primer trimestre del 2013, existiendo 152,3 millones de móviles vendidos a nivel mundial de los cuales el 59% usa el sistema operativo y el otro 23% usa el sistema operativo de Apple. La compra de dispositivos inteligentes (smartphones) se ha elevado exponencialmente al igual que los celulares; mientras que los ordenadores portátiles se han detenido en comparación a los dos primeros en los últimos años como lo muestra la Figura 1.

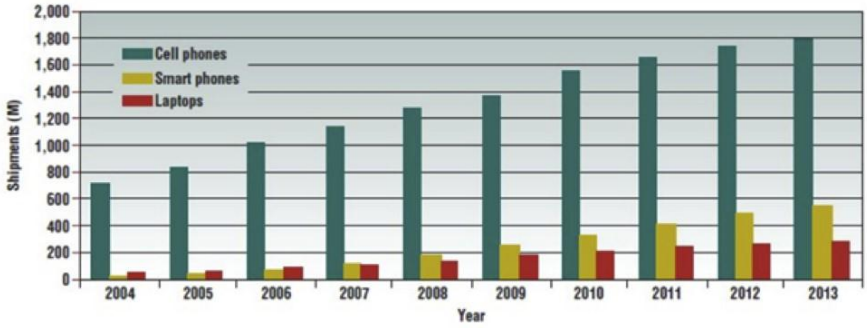

Figura 1. Evolución de los pedidos de móviles, terminales inteligentes y portátiles en el mundo

Fuente: FUMERO, Antonio; WERTERSKI, Adam; RODRÍGUEZ, Pedro; BLANCO, Paco; CAMARERO, Julio. Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y el iPhone.

Por otro lado la firma consultora Research2Guidance[2] especializada en la investigación de tecnologías móviles presentó una investigación especializada también en el 2012 a partir del análisis de una encuesta realizada a los principales actores en el mercado y presenta un informe detallado sobre el mercado de las aplicaciones móviles en Android que revela que se espera un crecimiento del 807 % hasta el 2013, en el cual se pretende mover 17.5 billones de dólares desde el 2012 y llegue hasta 500 millones de usuarios hasta el año 2015.

[1] http://www.idc.com.
[2] http://www.research2guidance.com/.

1.1.1. Historia.

Android es un sistema operativo y una plataforma de software basada en el núcleo de Linux para dispositivos móviles. Sus aplicaciones se escriben en el lenguaje de programación JAVA disponiendo además de una máquina virtual llamada Dalvik. Aranaz (2009) señala que ”Android no solo es un sistema operativo sino que representa toda una pila de software para dispositivos móviles que incluye gran cantidad de controladores, gestor de bases de datos, un completo framework de aplicaciones, y numerosas aplicaciones de usuario.” Herraiz (2012) por otro lado lo define como un sistema operativo de Google y que con 5 años ha evolucionado de forma impresionante, mediante las versiones que se sacan periódicamente al mercado.

En el 2003 fue creada una empresa llamada Android Inc. en el estado de California, EE.UU; la cual centraba sus funciones en el desarrollo de software para telefonía móvil. En el 2005 la compañía fue comprada por Google la cual desarrolló una plataforma basada en el kernel de Linux para dispositivos móviles que fue anunciada a los fabricantes de dichos dispositivos con el compromiso de dar un sistema actualizable y flexible.

En el 2007, Google crea un consorcio llamado Open Handset Alliance (OHA)[3], con varias compañías dedicadas a la telefonía móvil, software, comercialización, semiconductores y fabricantes electrónicos para estrenar el Software Development Kit (SDK)[4] de android, una plataforma para dispositivos móviles construida bajo la versión 2.6 del kernel de Linux. Y finalmente en el 2008 sale al mercado el primer dispositivo que ejecutó Android OS.

Una de las mayores ventajas que posee Android es su sistema de código abierto, por lo tanto cualquier persona puede aportar sus conocimientos y continuar desarrollando el software; razón por la cual se ha convertido en el sistema operativo más utilizado por los usuarios de smartphones a nivel mundial.

1.1.2. El futuro de Android.

Android, es considerado como el Sistema Operativo del futuro y se desarrolló para los terminales móviles hasta convertirse en el más popular, principalmente para aplicaciones comerciales como: servicios de información, anuncios, servicios basados en contenido, en localización o aplicaciones de pago sobre móvil.

[3] http://www.openhandsetalliance.com/
[4] http://developer.android.com/sdk/index.html

En los últimos años Android ganó popularidad conjuntamente con los dispositivos móviles de pantalla táctil hasta convertirse en el favorito por los usuarios y en el más vendido en el 2012.

Android nació como una alternativa ante los líderes del mercado de los dispositivos inalámbricos como windows phone, symbian, palm, etc y no es más que un sistema operativo de código abierto dotado para ejecutarse en smartphones y con la facilidad de recibir actualizaciones periódicas.

Ofrece un ámbito de cobertura para sus aplicaciones muy amplio en el que se encuentran muchas de las área de interés socio-económico, tales como el sector financiero a través de la banca electrónica o por medios sociales como los clientes de microblogging y difusión de la información. Y debido a su portabilidad y facilidad de uso su demanda es cada vez mayor por lo que su diseño, funcionalidades y servicios que ofrece son cada vez actualizados a fin de garantizar un correcto funcionamiento ahorrando recursos como memoria y batería.

Según un estudio publicado por la IDC en diciembre del 2013 acerca del envío de tabletas por todo el mundo, sugiere un bajo crecimiento de 1 solo dígito para el 2017.

Se espera que para el 2014 un total de 270.5 millones de unidades de tabletas sean vendidas, mientras que para el 2017 se espera un alcance de 386.3 millones de unidades vendidas, la cual está por debajo de lo pronosticado acerca de 407 millones de unidades vendidas. (San Mateo, 2013)

En la Tabla 1 se puede apreciar una comparativa entre los años 2012, 2013 y lo que se espera para el 2017 según la IDC.

Tabla 1: Venta de tabletas, 2012 - 2017

Tablet OS	2012 Market Share	2013 Market Share	2017 Market Share
Android	52.0%	60.8%	58.8%
iOS	45.6%	35.0%	30.6%
Windows	0.9%	3.4%	10.2%
Other	1.4%	0.8%	0.4%
Grand Total	**100.0%**	**100.0%**	**100.0%**

Fuente: IDC Worldwide Quarterly Tablet Tracker, December 2013

1.2. Redes móviles Ad hoc

Una red “ad hoc”, consiste en un grupo de ordenadores que se comunican cada uno directamente con los otros a través de las señales de radio sin usar un punto de acceso.

Solamente los ordenadores dentro de un rango de transmisión definido pueden comunicarse entre ellos. En fin, en la tecnología “ad hoc”, cada terminal de comunicación se comunica con sus compañeros para hacer una red “punto a punto.” (Chanye, Mercado, Figueroa, 2011)

“Una red inalámbrica se refiere a una red en la cual las piezas especializadas del equipo que forman la espina dorsal de una red física simplemente no están presentes. En su lugar, los dispositivos clientes actúan como nodos clientes y como nodo backbone, el cual determina que en una red estándar los dispositivos clientes se conectan a la red.” (Y. Liang B. Brownlee.)

De acuerdo al Departamento de Ciencias de la Computación e Ingeniería de la Universidad Atlántica de Florida, USA (LOU, Wei and WU, Jie, march 2003) define que hay esencialmente dos componentes principales de una red de comunicaciones ad hoc:

Backbone: El backbone de la red consiste de varios switches (puentes), routers (ruteadores) y vínculos virtuales y/o físicos entre esos dispositivos. La única función del backbone es la de facilitar el transporte de información de inicio a fin de la comunicación. El backbone típicamente no suele generar ningún tráfico de sí mismo.

Clientes: Los clientes consisten de smartphones, computadoras, sensores y cualquier otro dispositivo responsable de la generación o la recuperación de información.

Una red móvil ad hoc es idéntica a una red inalámbrica ad hoc con la diferencia de que los dispositivos están en constante movimiento, esto ha sido considerado como el efecto de cambio constante de topología en la red, destruyendo con ello la estabilidad de la topología de la red e incluyendo en la capacidad de éstos dispositivos alcanzar la información de ruta entre ellos; dichos dispositivos móviles irán dentro y fuera del rango del dispositivo más cercano. Esto requiere la creación de un sistema que permita a los dispositivos abandonar o unirse a la red y de alguna manera propagar los cambios de la topología a todos los dispositivos.

En efecto el problema de tener que mantener el enrutamiento y el direccionamiento coherente ante un constante cambio de topología de red es el problema fundamental asociado con la investigación Mobile Adhoc Network (MANET)[5].

[5] http://es.wikipedia.org/wiki/Mobile_ad_hoc_network

Con lo anterior mencionado, sumado a ello el requisito de ser capaz de interoperar con la internet en su conjunto, son las principales limitaciones que un diseñador de protocolo debe tomar en consideración cuando se crean nuevos protocolos MANET.

1.2.1. Aplicaciones de las redes ad hoc.

Los avances actuales en localización y tecnologías orientadas a un entorno social tienen el potencial para desarrollar muchas aplicaciones necesarias para salvar vidas en respuesta a situaciones de emergencia y misiones de rescate (SONG, Wen; WANG, Fei; DAI, Jianbo, oct. 2010). Muchas de las aplicaciones existentes hoy en día disponen de un factor en común el cual consiste en la constante comunicación y difusión de la información a través de la internet, lo cual imposibilita su utilización en redes ad hoc.

1.2.1.1 Aplicaciones de video.

El monitoreo en video e imagen que capturan las aplicaciones han sido mencionadas en comunicaciones de emergencia en minas. En efecto las aplicaciones de video son una de las tres principales prioridades de los servicios de Ethernet (SONG, Wen; WANG, Fei; DAI, Jianbo, oct. 2010). El costo a nivel de recursos de hardware y software para la trasmisión de video en tiempo real es relativamente alto con respecto a aquellas aplicaciones que únicamente trasmiten texto en tiempos cortos.

1.2.1.2 Wifi mesh.

Las redes de malla tuvieron su origen en aplicaciones militares, con el fin de permitir a los soldados tener comunicación confiable de banda ancha en cualquier lugar. De esta forma, la confiabilidad y robustez requerida en los entornos militares se hereda a los ambientes civiles, donde las redes en malla están encontrando un nicho importante. Uno de los requisitos principales era contar con comunicaciones de banda ancha sin tener que instalar grandes torres o antenas. Así, el equipo de radio de cada soldado contribuía a la formación de una malla de unidades de radio que automáticamente aumentaba su cobertura y robustez conforme se unían nuevos usuarios a la misma (GARCIA, Ja; MACÍAS Evelio, 2006).

Una red wifi mesh en un Sistema de Comunicación de Emergencia basado en una Red Inalámbrica Mesh (ECS-WMN) (SONG, Wen; WANG, Fei; DAI, Jianbo, oct. 2010.) consiste de una colección de ruteadores inalámbricos los cuales trabajan juntos para proveer acceso inalámbrico a los clientes. Los ruteadores mesh enrutan los paquetes desde un router a otro en orden para alcanzar el destino. Uno de los mayores factores que ayuda a clasificar los routers disponibles es el algoritmo usado para el ruteo.

En la tabla 2 se puede apreciar los posibles escenarios y/o servicios en los que se puede utilizar redes ad hoc.

Tabla 2. Aplicaciones sobre redes móviles ad hoc

Aplicaciones	Posibles escenarios/servicios
Redes tácticas	1. Comunicaciones y operaciones militares
Servicios de emergencia	1. Búsqueda y operaciones de rescate 2. Recuperación de desastres 3. Sustitución de infraestructura fija en caso de desastres ambientales 4. Vigilancia y extinción de incendios 5. Apoyo a los médicos y enfermeras en los hospitales
Entornos comerciales y civiles	1. Comercio electrónico: Pagos electrónicos en cualquier lugar y hora 2. Negocios: Acceso dinámico a base de datos, oficinas móviles 3. Servicios vehiculares: Transmisión de las condiciones del clima, la red de taxis y las redes entre vehículos 4. Estadios deportivos, ferias y centros comerciales 5. Redes de visitantes en aeropuertos
Hogar y redes empresariales	1. Redes inalámbricas de oficinas 2. Conferencias y salas de reuniones 3. Personal del área de redes (PAN), Personal de redes (PN) 4. Redes en sitios de construcción.
Educación	1. Campus universitario 2. Salas de clases virtuales 3. Comunicaciones ad hoc durante reuniones o conferencias.
Entretenimiento	1. Juegos multiusuario 2. Redes inalámbricas P2P 3. Mascotas robóticas
Cobertura / extensión	1. Ampliar el acceso celular 2. Vinculación con la internet, intranets, etc.

Fuente: JEROEN HOEBEKE, Bart Dhoedt; MOERMAN, Ingrid and DEMEESTER, Piet.. An overview of mobile ad hoc networks:applications and challenges. In An Overview of Mobile Ad Hoc Net-works:Applications and Challenges .

1.2.2. Características de una red ad hoc.

Las redes móviles ad hoc se caracterizan por una topología de red multi-salto que puede cambiar con frecuencia debido a la movilidad.

Eficientes protocolos de enrutamiento son necesarios para establecer vías de comunicación entre los nodos, sin causar excesiva sobrecarga de control de tráfico o carga computacional en los dispositivos (SONG, Wen; WANG, Fei; DAI, Jianbo, january. 1999). Un gran número de soluciones ya han sido propuestas, siendo algunas de ellas objeto de estandarización dentro del IETF (Internet Engineering Task Force)[6]. "Un número de soluciones proactivas intenta tener una ruta actualizada para todos los otros nodos en todo momento. Con este fin estos protocolos de intercambio rutean el control de información periódicamente y cuando existen cambios topológicos. Los protocolos de enrutamiento reactivos solo establecen rutas a nodos con los que se comunican y esas rutas se mantienen en memoria el mayor tiempo que sea necesario (JEROEN HOEBEKE, Bart Dhoedt; MOERMAN, Ingrid and DEMEESTER, Piet)."

1.2.3. Geolocalización mediante redes wifi en Android.

Todos los dispositivos móviles con sistema operativo Android instalado tienen diferentes maneras de conocer su ubicación en cualquier momento, entre ellas:

A través de triangulación de señales de telefonía móvil. Usando el dispositivo GPS.
Basándose en las redes inalámbricas a nuestro alrededor.

Estas opciones son utilizadas por muchos de los servicios preinstalados en los dispositivos y que necesitan conocer la localización del mismo. Sin embargo, por razones de privacidad los usuarios deben previamente dar su expreso consentimiento para utilizar estas opciones, pudiendo habilitar y deshabilitarlas siempre que consideren oportuno a través de los ajustes de su dispositivo presentados en la Figura 2 por lo que no todas las opciones están disponibles en cualquier momento.

En el presente proyecto se utiliza la antena receptora de GPS para determinar la ubicación de un usuario debido que brinda una mejor posición georeferencial.

La opción de geolocalización mediante redes wifi hace uso del sistema creado por Google para tal fin. De esta manera, cuando la opción de conocer la ubicación del dispositivo está habilitada, y la tarjeta inalámbrica activada, se escanean las redes inalámbricas alrededor del dispositivo para determinar la posición geográfica del mismo. La precisión obtenida puede variar desde decenas hasta cientos de metros (PÉREZ BLANCO, Carlos, july 2010).

[6] http://www.ietf.org/

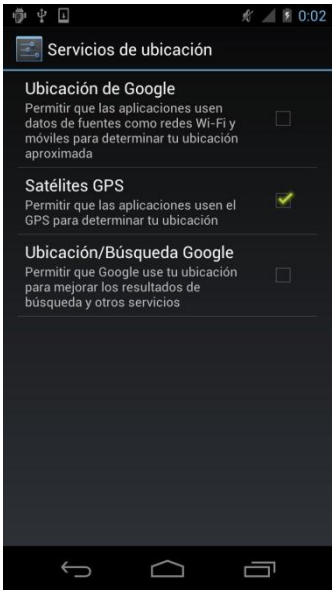

Figura 2. Configuración de ubicación de un dispositivo Android

1.3. Aplicaciones orientadas a situaciones de emergencia

Los desastres pueden ocurrir en un instante, ya sea que estén relacionados con el clima, por el hombre o por alguna otra causa, los desastres ocurren a menudo con poca o ninguna advertencia. Basado en aquello surge la necesidad de la creación de una aplicación que permita una evacuación de víctimas en el menor tiempo posible.

El objetivo principal de las aplicaciones orientadas a situaciones de emergencia consiste en mantener informados a los usuarios que pertenecen a un grupo sobre una situación en la cual es necesaria algún tipo de ayuda.

El éxito de toda aplicación orientada a un ámbito de emergencia es la comunicación clara. Los dispositivos móviles como smartphones y tabletas pueden ayudar a sus usuarios a establecer una comunicación entre sí y con las autoridades u organismos de socorro, la difusión de información es esencialmente importante, especialmente en situaciones de crisis. Ayudar a mantener las líneas de comunicación abiertas es el objetivo principal de decenas de aplicaciones móviles diseñadas específicamente para la preparación para emergencias.

La mayoría de las aplicaciones existentes en la actualidad utilizan una conexión de datos a fin de hacer uso de servicios web existentes los cuales se vuelven inoperables en entornos cerrados o fuera del alcance de una conexión de datos. La aplicación a desarrollar en el presente proyecto no utiliza una conexión a la internet, lo cual resuelve el problema de la operatividad en entornos cerrados o que estén fuera del alcance de una conexión fuera de una intranet.

1.3.1. iRescate / iRescue.[7]

iRescue ayuda a los profesionales en emergencias a obtener información para el rescate de personas en accidentes de tráfico.

Características:

- Está disponible en español e inglés.
- Permite visualizar la posición del dispositivo en el mapa.
- Enviar una foto por correo electrónico, adjuntando o no, la posición del GPS.

De esta forma, los centros de coordinación o cualquier autoridad pueden ser informados en el acto.

1.3.2. Emergency Alert.[8]

Emergemcy Alert es una herramienta útil para alertar de eventos urgentes vía SMS, incluso cuando el dispositivo está en silencio. Puede crear sus propias reglas las cuales se pueden ejecutar con mensajes vía SMS, o que provienen de un número específico.

Características:

- Cualquier texto enviado después del estado de alerta se mostrará en la ventana emergente en la pantalla.
- Puede alertarle de tres maneras:
    - Green Alert (Alerta Verde).- Esta alerta solo muestra un mensaje emergente del remitente.
    - Yellow Alert (Alerta Amarilla).- Alerta que vibrará hasta que lo retire, permitiéndole ser cuidadosamente notificado de un asunto urgente.
    - Red Alert (Alerta Roja).- Esta alerta notificará con una alerta sonora y vibraciones hasta que lo retire, incluso si el dispositivo está en silencio con un volumen medio.

Es una herramienta útil cuando se necesita notificar de manera urgente a un amigo o familiares sobre algún percance, así también se lo puede utilizar en situaciones de desastres, terremotos, alertas de inundaciones o amenazas de seguridad para su debida notificación.

Para activar las reglas, el remitente o el sistema de alerta automatizado necesita conocer el mensaje SMS: redalert, yellowalert, o greenalert.

[7] http://www.rescue-app.com.
[8] https://play.google.com/store/apps/details?id=com.xankle.mobile.sos&hl=es_419.

Si tiene un auricular conectado, usará el volumen actual, sin auriculares, la alerta se reproducirá a volumen completo a través de los parlantes, asegurándose de que no se pierda una notificación.

1.3.3. FireAlert.[9]

FireAlert es una aplicación ideal para el personal de cuerpo de bomberos y otras profesiones en las cuales es importante una respuesta inmediata en el lugar exacto. Puede ser usado por:

- Bomberos.
- Servicio médico de emergencia. Guardacostas.
- Servicio de rescate de montaña. Cruz Roja.
- Despacho asistido por computadora.

FireAlert también puede ser utilizado como una herramienta para el monitoreo de la temperatura en entornos cerrados tales como sala de servidores u otros dispositivos de constante vigilancia.

Características:

- Soporte para el envío de mensajes de texto
- Soporte para el envío de mensajes multimedia Muestra el mensaje en una ventana emergente
- Reproduce sonido incluso cuando el móvil está en modo silencioso
- Posee diferentes patrones de vibración.
- FireAlert es libre y sin ningún tipo de publicidad.

A continuación en la Tabla 3 se presenta un cuadro comparativo de las principales características con las que cuenta cada aplicación:

[9] http://www.b4exit.de.

Tabla 3. Cuadro comparativo de las aplicaciones de emergencia

	iRescue	Emergency Alert	FireAlert
Conexión a internet	Si	Si	Si
Envío de msg de texto	No	Si	Si
Msg multimedia	No	Si	Si
Licencia Libre	No	No	Si
Geolocalizacion	Si	Si	Si
Idioma	Inglés/Español	Ingles	Inglés
Notificación con audio	Si	Si	Si
Notificación por email	Si	No	No

Todas las aplicaciones que se han revisado en este apartado tiene un requerimiento a tener en cuenta el cual consiste en la necesidad de tener una conexión a internet constante debido a que sus servicios están orientados a compartir información mediante la web, ya sea para enviar o recibir dicha información; por el contrario la solución propuesta en el siguiente capítulo de este proyecto plantea una alternativa diferente la cual básicamente no depende de una conexión a internet y su funcionamiento se proyecta a entornos debidamente definidos en áreas con cobertura wifi.

CAPÍTULO 2
SOLUCIÓN PROPUESTA

En las soluciones previamente analizadas, ninguna aborda la solución para diseñar una aplicación móvil independiente de la conectividad de datos.
Muchas de las soluciones disponibles hasta la actualidad hacen uso del consumo de datos basados en servicios web o APIs (Interfaz de programación de aplicaciones) consumidas remotamente, limitando su utilización únicamente en entornos donde existe una red de datos disponible, esto a su vez no se adapta a entornos de emergencia o desastres que ocasionan la pérdida en su totalidad de los enlaces de comunicación, por lo tanto el ámbito de su aplicación es limitada.

El presente trabajo se enfoca en el desarrollo de una aplicación que se adapte a entornos en los cuales no exista una red con acceso a internet, únicamente es necesario tener acceso a una red wifi local, logrando así proveer a los usuarios que se encuentren en situaciones de emergencia visualizar los puntos de evacuación más cercanos y su respectiva ruta de salida desde la ubicación actual hasta la zona de evacuación más cercana, evitando los obstáculos que hayan sido reportados por otros usuarios, los cuales impiden un normal tránsito peatonal o vehicular.

En la Figura 3 se presenta el diagrama de la solución propuesta, la cual contiene tres elementos principales tales como: usuarios, la aplicación y los datos a visualizar por parte del usuario dentro de la aplicación.

Cada uno de estos tres elementos que se plantean en la solución propuesta está estrechamente relacionado debido a que cada uno envía o recibe información de los demás componentes. La interacción entre los componentes se realiza dentro del área de cobertura, en una zona con acceso a una red wifi.

Usuarios.- Son todos los usuarios que estén dentro del área de cobertura y que a su vez ejecuten la aplicación. Tampoco es necesario que el usuario tenga conexión a internet pero a su vez si es necesario que tenga permiso para usar el GPS del dispositivo, finalmente es necesario tener el wifi activado y encontrase dentro de la red.

Aplicación.- Es la solución a desarrollar que se instalará en los dispositivos móviles con android, la misma que deberá funcionar en una red wifi local sin acceso a internet. La aplicación es la encargada de recibir los datos del cliente, ya sean estos para reportar la existencia de un obstáculo o para reportar la existencia de una nueva zona segura; una vez que los datos hayan sido recibidos correctamente la aplicación realiza

las operaciones que el usuario ha solicitado y finalmente los resultados son visualizados en el dispositivo móvil.

Visualización de resultados.- Representa a todos los datos que son visualizados por el usuario en el dispositivos móvil.

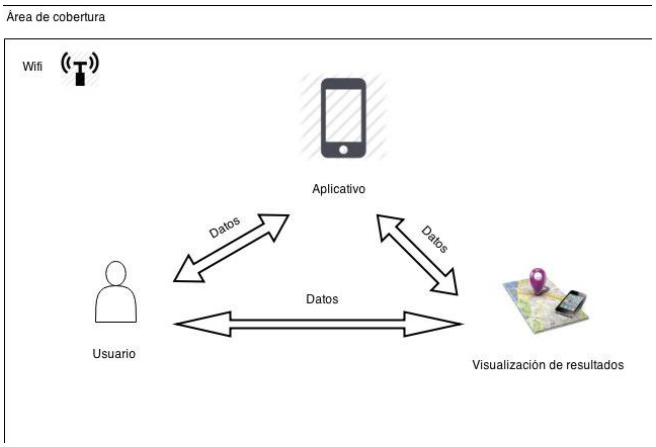

Figura 3. Diagrama de solución propuesto

La información sobre los posibles obstáculos existentes se deberá encontrar en cualquier nodo de la red y éste deberá contener toda la información existente hasta el momento, la cual a su vez estará disponible para que todos los usuarios puedan consultar, así mismo cada usuario puede reportar la existencia de un obstáculo o zona segura.

Con la información que la aplicación obtiene desde el nodo más cercano se puede graficar la mejor ruta de salida desde la ubicación actual, cabe mencionar que cada vez que exista un cambio en la información sobre los obstáculos, se redefine el algoritmo para la búsqueda de una ruta de salida.

Cada usuario en cualquier momento puede consultar si existe nueva información ya sea de los obstáculos, usuarios existentes, mapa o ruta de salida.

La sincronización de la información se la puede hacer de manera manual o automática, siendo, en la primera, necesaria la intervención del usuario para que realice dicha sincronización con el nodo que tenga la información necesaria a fin de obtener los datos correctos y completos.

Por otro lado, la aplicación en modo automático no necesita que sea actualizada por el usuario, los datos son actualizados cada cierto periodo de tiempo dependiendo de las preferencias de usuario configuradas.

2.1. Característica de la solución propuesta

En la solución propuesta se ha definido ciertas características que debe cumplir a fin de mejorar su prestación de servicios, los cuales se detalla a continuación:

- Funciona independiente del estado de la conectividad a la internet, la aplicación únicamente utiliza la red wifi local disponible.
- Uno o más dispositivos móviles serán los encargados de almacenar la información de todos los obstáculos, datos del mapa, zonas de evacuación y usuarios existentes en la red y enviárselos a los demás usuarios siempre y cuando éstos lo soliciten.
- Los datos sobre el reporte de un obstáculo es alimentado por cada de uno de los usuarios existentes.
- Cada usuario obtiene su propia ruta de salida hacia la zona de evacuación, la cual previamente debe ser definida.
- El alcance de la aplicación se basa tanto en la disponibilidad de la red wifi local como de la existencia en el nodo que contenga toda la información acerca de las rutas de evacuación y sus obstáculos.
- Funciona únicamente dentro de un área predefinida, pudiendo ésta ser cambiada por otra.
- Su área de cobertura puede ser expandida, esto requiere que la matriz base sea modificada.

CAPÍTULO 3

ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

Se analiza en detalle el modelado de la solución propuesta, ésta información sirve para determinar los requerimientos funcionales que debe cumplir la aplicación. El objetivo en la ERS (Especificación de Requerimientos de Software) es definir de una manera clara y formal todas las funcionalidades y restricciones de la aplicación que se desea construir.

3.1. Requerimientos de arquitectura

- La aplicación debe correr sobre terminales con android 2.2 o superior.
- La aplicación debe difundir los datos de geolocalización de todos los obstáculos existentes a cualquier usuario nuevo.
- La aplicación debe trabajar sobre una red wifi.

3.2. Requerimientos funcionales

En esta sección se mencionan los requisitos funcionales que debe cumplir la aplicación para el sistema operativo Android.

3.2.1. Requisitos de Usuario.

- El usuario podrá visualizar a todos los usuarios existentes en su red.
- El usuario podrá navegar a través del mapa visualizando la ubicación de cada usuario.
- El usuario podrá reportar la existencia de un obstáculo desde la ubicación en la que se encuentre.
- El usuario podrá visualizar en cualquier momento las posibles rutas para llegar a la zona de evacuación previamente definida.
- El usuario podrá solicitar a la red todos los obstáculos existentes y éstos deben visualizarse en el mapa.
- El usuario podrá visualizar todos los obstáculos existentes en el mapa, los cuales deben estar debidamente enumerados.

3.2.2. Requisitos del Sistema.

- La aplicación debe iniciarse independientemente de la existencia de una conexión a la internet.
- La aplicación debe desplegar un mapa con todos los obstáculos y usuarios existentes.

- La aplicación deberá presentar menús de opciones para el usuario.
- La aplicación debe cargar el mapa que previamente se haya descargado de la web e instalado en la tarjeta del dispositivo.
- La aplicación deberá tener acceso a una red wifi existente localmente.
- La aplicación será capaz de detectar si tiene cobertura e informar al usuario mediante un mensaje.
- La aplicación deberá tener acceso a los servicios de geolocalización (GPS) y detectar si están o no activos; se informará al usuario mediante un mensaje.

3.3. Restricciones

- La aplicación debe iniciarse en cualquier android 2.2 o superior, independientemente de las redes móviles existentes.
- No tiene un número de nodos predefinido, esto dependerá del tamaño del buffer de mensajes almacenados en cada dispositivo.

3.4. Análisis y Diseño de Alto Nivel

3.4.1. Diseño de la arquitectura.

Para el desarrollo de este proyecto se ha optado por la utilización del IDE (Entorno de Desarrollo Integrado) “Eclipse v3.7” y la instalación de la SDK de Android. Además, para hacer posible el desarrollo en Eclipse, se decide instalar el complemento de las herramientas de desarrollo de Android, ADT (Android Development Tools)[10], de tal manera que se pueda disponer de las herramientas de depuración y librerías necesarias para el desarrollo de la aplicación.

3.4.2. Diseño de la aplicación.

En la Figura 4 se muestra un diseño general de la estructura modular de la aplicación. A continuación se detalla cada uno de los componentes que conforman la aplicación:

Map.- Es el módulo encargado de obtener los datos necesarios para renderizar el mapa al usuario, es decir recibe los datos del módulo controlador y los dibuja en la pantalla del usuario.

[10] http://developer.android.com/sdk/index.html

User.- Módulo encargado de recibir todas las entradas de los usuarios, éstas entradas son todas las acciones que dicho usuario realiza en el dispositivo móvil siempre que se encuentre en ejecución la aplicación.

Controller.- Este quizá sea el módulo más crítico en la aplicación del cual dependen todos los demás, debido a que en el presente módulo se gestiona todo el comportamiento y la lógica de negocio que poseerá la aplicación, los datos que ingresan o salen del dispositivo pasan necesariamente por este módulo, el cual determina su destino.

Services.- En este módulo se envían y reciben todos los datos provenientes de un servicio ya sea interno como una comunicación vía TCP entre dos o más dispositivos o a su vez servicios externos tales como un servicio web.

Providers.- Módulo responsable de proporcionar un mecanismo para permitir compartir información entre aplicaciones. Se encarga de mantener la disponibilidad de la información de manera controlada para el resto de módulos de la aplicación, este módulo a su vez proporcionará un *content provider* a través del cual se pueda realizar el acceso a dicha información.

Intents.- Módulo encargado de invocar componentes, en android se entiende por componentes las *activities*, la cuales son componentes de interfaz de usuario, servicios, código ejecutándose en segundo plano, *broadcast receivers*. Cada una de las invocaciones que se realizan a un componente se utiliza para compartir información con otros servicios.

Activities.- Este módulo contiene la interfaz de usuario de nuestra aplicación. En la aplicación se crean *activities* de acuerdo a la necesidad de cada funcionalidad. El objetivo principal de este módulo es interactuar con el usuario desde el momento que una pantalla se renderiza hasta el momento en que ésta se oculta. Este módulo no recibe ningún dato del usuario únicamente se encarga de mantener los componentes del usuario en forma visible.

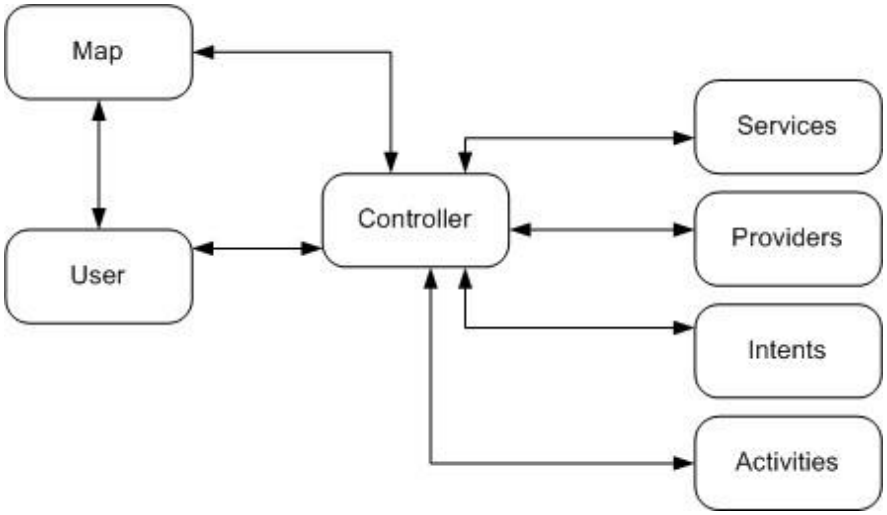

Figura 4. Estructura Modular de la Aplicación

3.5. Modelo de casos de uso

Se incluyen en el diseño los diagramas de casos de uso que contienen acciones principales que un usuario o actor de la aplicación podrá llevar a cabo con la misma, tales acciones se describen en el modelo de casos de uso de la Figura 5.

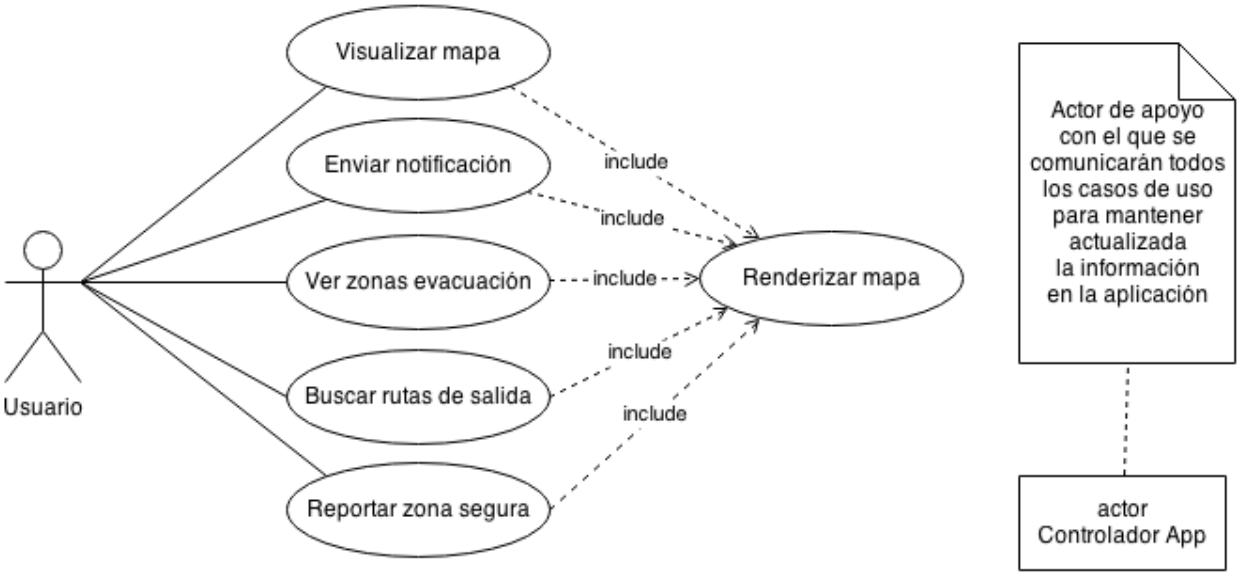

Figura 5. Modelo de casos de uso

3.5.1. Especificación de casos de uso.

El modelo de casos de uso nos proporciona una visión general del funcionamiento, ya que describe en lenguaje natural la funcionalidad completa de un sistema a desarrollar y su empleo en el proceso de especificación de requerimientos del sistema. A continuación se describen cada uno de los casos de uso los cuales detallan el flujo que debe seguir un determinado proceso.

Visualización del Mapa

La Tabla 4 describe el proceso que sigue la aplicación para que el usuario pueda visualizar el mapa en su pantalla, permitiéndole desplazarse por el mismo. Cabe recalcar que la renderización del mapa es independiente del estado de conectividad del dispositivo, todos los datos necesarios ya se encuentran almacenados en el dispositivo, haciendo que la carga del mapa tenga un tiempo mínimo.

Tabla 4. Caso de Uso UC1: Visualizar Mapa

Especificación del caso de uso: Visualizar Mapa	
Código	1
Nombre	Visualizar Mapa
Descripción	Este caso de uso permite al usuario visualizar de manera gráfica el mapa de openstreetmap dentro del dispositivo móvil en modo offline.
Actores	Usuario
Precondición	1. Es necesario pasar por el menú principal de la aplicación para acceder al mapa. 2. Establecer en el Android Manifest el uso de internet y de GPS, para tener los permisos para su uso.
Postcondición	Al ingresar al mapa se debe activar el GPS, para empezar a captura la posición del usuario para que sea representado en el mapa con su posición actual.
Flujo normal	1. El usuario ingresa a la aplicación. 2. La aplicación presenta los menús disponibles. 3. El usuario selecciona "Ver Mapa". 4. La aplicación renderiza el mapa con todos los obstáculos y usuarios existentes hasta el momento.
Excepciones	Si no se puede capturar ninguna posición válida solo cargar el mapa.
Anotaciones	El periodo de actualización del GPS será cada 30 segundos si permanece estático y cada 10 metros si está en movimiento.

Envío de Notificaciones

En la Tabla 5 describe el funcionamiento del envío de una notificación de un usuario hacia otro, en dicha notificación se envían los datos de geo-localización del usuario emisor, los datos enviados corresponden a la existencia de un obstáculo en medio de una calle que se encuentra dentro del área de cobertura de la aplicación.

Tabla 5. Caso de Uso UC2: Enviar Notificación

Especificación del caso de uso: Enviar Notificación	
Código	2
Nombre	Enviar Notificación
Descripción	Permite el envío de notificaciones a los usuarios de la red cuando existe un nuevo obstáculo.
Actores	Usuario
Precondición	1. El usuario debe visualizar que estén correctamente establecidos los datos de geolocalización. 2. Debe estar activado el GPS para obtener la localización.
Postcondición	Los datos de la ubicación del obstáculo reportado son renderizados en el mapa.
Flujo normal	1. El usuario ingresa a la aplicación. 2. La aplicación presenta la latitud y longitud de la ubicación actual 3. El usuario da click en aceptar para enviar la información.
Excepciones	Si los datos no han podido ser enviados se presenta un mensaje indicando el error.
Anotaciones	Una vez que una notificación ha sido enviada, ésta queda almacenada en la memoria del dispositivo receptor con el fin de leer dichos datos y tomarlos en cuenta para trazar la ruta de salida desde el lugar en el que se encuentra un usuario cualquiera hacia el lugar de evacuación más cercano existente en la localidad.

Ver Zonas de Evacuación

En la Tabla 6 se describe el flujo a seguir para la visualización de las zonas de evacuación existentes dentro del área de cobertura de la aplicación, dichas zonas de evacuación normalmente se encuentran fuera del área céntrica de la ciudad, lugar en el cual la seguridad de una persona está resguardada.

Tabla 6. Caso de Uso UC3: Ver zonas de evacuación

Especificación del caso de uso: Ver Zonas de Evacuación	
Código	3
Nombre	Ver Zonas de Evacuación
Descripción	Permite visualizar las zonas de evacuación existentes alrededor de la ubicación actual del usuario.
Actores	Usuario
Precondición	El usuario debe iniciar la aplicación y seleccionar la opción de ver zonas de evacuación.
Postcondición	Todas las zonas de evacuación deben ser visibles claramente en cualquier momento.
Flujo normal	1. El usuario ingresa a la aplicación. 2. El usuario selecciona la opción de ver zonas de evacuación. 3. La aplicación recolecta la información necesaria y renderiza todas las zonas de evacuación existentes.
Excepciones	Si no existen zonas de evacuación la aplicación presenta un mensaje indicando que no existe ninguna zona disponible.
Anotaciones	Ninguna

Búsqueda de Rutas de Salida

Dentro del área de cobertura puede existir una o más zonas de evacuación, depende del número de zonas seguras que hayan sido reportadas por los usuarios como se detalla en el caso de uso 5 (UC5).

En la Tabla 7 se puede observar el flujo a seguir para completar la salida desde el lugar de ubicación actual hasta una zona de evacuación más cercana dentro del área de cobertura, en caso de no existir ninguna zona de evacuación, la aplicación presentará un mensaje informando al usuario de tal situación.

Tabla 7. Caso de Uso UC4: Buscar rutas de salida

Especificación del caso de uso: Buscar Rutas de Salida	
Código	4
Nombre	Buscar Rutas de Salida
Descripción	Permite buscar las posibles rutas de salida desde el punto de ubicación actual hacia el punto de evacuación.
Actores	Usuario
Precondición	1. El usuario debe iniciar la aplicación y visualizar el mapa correctamente. 2. Debe estar activado el GPS para obtener la localización.
Postcondición	La aplicación debe presentarle todas las rutas existentes desde el punto de ubicación actual hasta una zona de evacuación más cercana
Flujo normal	1. El usuario ingresa a la aplicación. 2. El usuario selecciona la opción de rutas de salida. 3. La aplicación empieza a buscar las posibles zonas de evacuación existentes. 4. La aplicación recolecta la información necesaria y renderiza todas las rutas que el usuario tiene desde su ubicación actual.
Excepciones	Si no existe ninguna ruta de salida hacia una zona de evacuación, la aplicación presenta un mensaje de información al usuario.
Anotaciones	Ninguna

Reportar Zona Segura o Zona de Evacuación

Finalmente en el caso de uso 5 presentado en la Tabla 8 se describe el flujo para la notificación de un usuario sobre la existencia de una zona segura en el lugar donde un usuario se encuentre en un determinado momento, cabe recalcar que si dos o más usuarios reportan la existencia de una zona segura, estando éstas a menos de 20 m de distancia entre sí, la aplicación solo tomará como zona segura a la que primero haya sido reportada.

Tabla 8. Caso de Uso UC5: Reportar Zona Segura o Zona de Evacuación.

Especificación del caso de uso: Reportar Zona de Evacuación	
Código	5
Nombre	Reportar Zona de Evacuación
Descripción	Permite reportar la existencia de una zona segura desde el lugar de ubicación actual.
Actores	Usuario
Precondición	1. El usuario debe iniciar la aplicación y seleccionar la opción de reportar zona segura. 2. Debe estar activado el GPS para obtener la localización.
Postcondición	La aplicación debe notificarle al usuario sobre el estado del reporte de la nueva zona segura.
Flujo Normal	1. El usuario ingresa a la aplicación. 2. El usuario selecciona la opción de reportar zona segura. 3. La aplicación verifica que las coordenadas donde se intenta reportar la nueva zona segura no se encuentran ya reportadas por otro usuario. 4. En caso de que las coordenadas de la nueva zona segura no existan, la aplicación envía dichas coordenadas. 5. La aplicación debe informar al usuario que el envío de sus datos han sido exitoso o fallido.
Excepciones	Si el envío de los datos de la zona segura han sido fallidos, la aplicación debe permitirle reenviar dichos datos.
Anotaciones	Ninguna

CAPÍTULO 4
DISEÑO DE LA SOLUCIÓN

En este capítulo se describe en detalle el desarrollo completo de la aplicación enfocado a una situación de emergencia utilizando una red wifi a la cual tendrán acceso todos los usuarios que se encuentren dentro del área previamente definida y en la cual cada dispositivo móvil almacenará toda la información necesaria acerca de los usuarios, rutas de salida, zonas seguras y obstáculos existentes.

Se ha estructurado la aplicación de manera que sea fácil e intuitiva para el usuario, haciendo que los componentes a visualizar sean lo más descriptivos posibles. Asimismo, el acceso a las distintas secciones de la aplicación se integra con las diferentes posibilidades que ofrecen los terminales con sistema operativo Android.

4.1. Arquitectura de la aplicación

El principal objetivo de una arquitectura es diseñar un modelo de aplicación que cumpla con los requerimientos funcionales y que sean fáciles de entender y utilizar. Para poder lograrlo, debe alinear las metas del negocio con los objetivos de los requerimientos.

La arquitectura propuesta está basada en tres niveles: una capa de presentación, una capa de lógica de negocio y una capa de acceso a datos.

Capa de presentación.- Es la interfaz gráfica con la cual el usuario puede visualizar la información proporcionada al usuario, en esta capa también se ingresan los datos proporcionados por el usuario.

Capa de lógica de negocio.- Es la capa que define el intercambio de información entre el modelo (capa de acceso a datos) y la capa de presentación, en este nivel se definen las reglas lógicas que contiene la aplicación.

Capa de acceso a datos.- Es la encargada del acceso a datos, la información puede provenir desde una base de datos, desde un servicio web externo, api, xml, texto, etc.

Una vez se tiene una idea general del funcionamiento y características generales de la aplicación, la figura 6 muestra un diagrama que contiene la arquitectura de todo el sistema completo.
La señal wifi propuesta es externa es decir puede ser accesible por cualquier otro equipo de cómputo existe.

El servidor que en se propone en la arquitectura de la Figura 6 es un dispositivo móvil común el cual será el encargado de almacenar toda la información que será enviada a los demás nodos de la red, la información que este servidor móvil tendrá consta de los datos de los usuarios existentes en la red, los obstáculos reportados y las zonas reportadas como seguras.

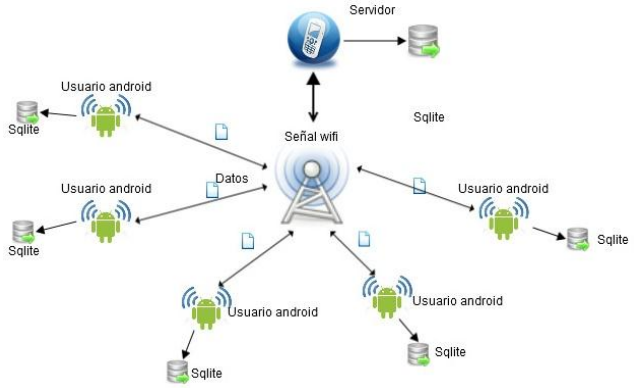

Figura 6. Arquitectura de la aplicación

4.2. Descripción de la arquitectura propuesta

En la arquitectura propuesta (Figura 6) se define los componentes presentados en la Tabla 9:

Tabla 9. Simbología de la arquitectura propuesta

Símbolo	Descripción
	Punto de acceso de la red wifi a la cual se conectan los usuarios.
	Usuario con un dispositivo android.
	Dispositivo móvil android que contiene la información del mapa, los obstáculos y los usuarios existentes en la red.
	Base de datos sqlite en la cual se almacena la información receptada desde el dispositivo móvil (servidor), ésta base de datos es por cada usuario
	Es la información que se envía desde un cliente hacia el nodo que actúa como servidor y viceversa.

Para el funcionamiento de la aplicación es necesario que el dispositivo móvil que contenga la información a sincronizar esté en funcionamiento en la red wifi, a su vez los

usuarios acceder a dicho dispositivo para descargar los datos del mapa, los obstáculos existentes y los usuarios existentes en la red.

4.3. Sincronización de información

En esta sección se define que información se obtendrá desde el dispositivo que actúa como servidor hacia el cliente o viceversa, los datos que viajan a través de la red necesariamente tienen que ser serializados para su transporte en la red. La información a sincronizar se divide en:

Mapa.- Esta información define los parámetros necesarios que tiene el mapa conjuntamente con la matriz base la cual es la encargada de definir el área de cobertura de la aplicación.

Obstáculos.- Información correspondiente a los obstáculos existentes dentro del área de cobertura, dichos obstáculos se toman en cuenta posteriormente para trazar la ruta de salida hacia el punto de evacuación.

Usuarios.- Información que contiene los datos de geolocalización de todos los usuarios existentes dentro del área de cobertura, este tipo de información también es enviada desde los usuarios hacia el nodo servidor, en los datos enviados consta la ruta que un usuario realizó desde que activó el envió de su ubicación de forma automática en cada dispositivo cliente.

Puntos de evacuación.- Corresponde a todos los puntos de evacuación existentes dentro del área de cobertura, esta información inicialmente se encuentra almacenada en el dispositivo al cual el usuario se enlazó previamente, los clientes puede acceder ella mediante la sincronización de los puntos de evacuación.

En la Figura 7 se puede apreciar la sincronización entre dos dispositivos móviles. El modelo de sincronización es para todos los casos en los cuales es necesario obtener y enviar información a un nodo destino.

Figura 7. Sincronización de la información

Toda la información que se envía y se recibe desde o hacia un dispositivo móvil no utiliza ningún método de encriptación, debido a que no es recomendable incrementar

el tamaño de los bytes a trasmitir en la red lo cual puede retardar el envío-recepción de los datos.

4.3.1. Tipos de mensajes de sincronización.

Cada vez que se necesita sincronizar cierta información con otro nodo, el cliente envía un mensaje que contiene el tipo de sincronización que solicita, los tipos de mensajes son:

Obstáculo.- Utilizado para obtener todos los obstáculos existentes dentro del área de cobertura que han sido reportados por los usuarios.

Usuario.- Utilizado para obtener la ubicación de todos los usuarios existentes en la red wifi que se encuentran dentro del área de cobertura de la aplicación.

Zona.- Utilizado para obtener todas las zonas de evacuación existentes dentro del área de cobertura a las cuales es posible trazar una ruta de salida desde una cierta ubicación del usuario en cualquier momento.

Ubicación.- Utilizado para enviar la ubicación de un usuario al nodo destino de forma automática.

Mapa.- Mensaje utilizado para obtener los datos necesarios para que el mapa sea renderizado correctamente en los dispositivos clientes.

4.4. Prototipo

En la siguiente sección se presentan los prototipos que se han definido con el fin de obtener un diseño previo de la interfaz de usuario conjuntamente con sus funcionalidades a fin de cumplir con los requerimientos del usuario.

4.4.1. Pantalla de inicio y opciones disponibles.

El sistema muestra los datos a modo de menús de opciones, listas de resultados y mapas. Cuando un usuario instala y arranca la aplicación por primera vez, se encuentra con una pantalla inicial como se aprecia en la Figura 8 donde se le presenta al usuario las diferentes opciones a elegir.

Figura 8. Pantalla inicial de la aplicación

4.4.2. Mapa de obstáculos.

En el menú “Principal” de la Figura 8 se puede apreciar las opciones a visualizar como se presenta en la Figura 9; la primera opción existente es la de Obstáculos, la cual nos permitirá ver todos los obstáculos existentes dentro del área de cobertura.

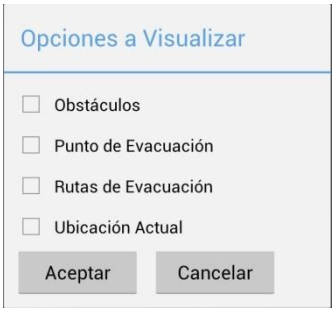

Figura 9. Opciones a visualizar en el mapa

Una vez que seleccionemos la opción de Obstáculos se podrá ver todos los obstáculos como se aprecia en la Figura 10.

Figura 10. Mapa con los obstáculos existentes

Todos los obstáculos se presentan al usuario del dispositivo móvil independientemente de su ubicación, así mismo cabe recalcar que un usuario puede eliminar un obstáculo que aún se mantenga en el mapa, siempre que dicho usuario se encuentre cerca del lugar del obstáculo a eliminar, caso contrario no se puede eliminar debido a que la ubicación de la persona que quiere eliminar dicho obstáculo puede encontrarse muy distante.

.

Todos los obstáculos tendrán una numeración adecuada que será gestionada internamente por la aplicación para tener un mejor control sobre la ubicación de cada uno de ellos y evitar que se envíen datos innecesarios, por otro lado los usuarios acorde a su ubicación y consideración reportarán dichos obstáculos a toda la red de usuarios contribuyendo a que los datos sean almacenados por cada usuario de forma autoritaria, es decir la aplicación automáticamente almacenará en cada nodo todos los datos necesarios del mapa en su propio dispositivo, quedando opcional si se elimina o no dicha información una vez que se haya terminado una sesión activa.

4.4.3. Puntos de evacuación.

Los puntos de evacuación deben encontrase dentro del área de cobertura para que sea posible su visualización, en las opciones de la Figura 9 se puede apreciar la segunda opción la cual permitirá ver todos los puntos de evacuación existentes como en la Figura 11.

Figura 11. Mapa con los puntos de evacuación existentes

4.4.4. Rutas de evacuación.

Las rutas de evacuación se marcan con una línea de color rojo como se presenta en la Figura 12, la cual realiza una trazado de salida desde la ubicación actual del usuario hacia el punto de evacuación más cercano.

En la ruta calculada no se toma en cuenta la dirección de las calles debido a que la aplicación está diseñada para la utilización por personas que se desplazan caminando.

El algoritmo utilizado es el de Dijkstra[11] el cual se implementa en la sección 5.6.1 del capítulo 5 del presente proyecto.

Figura 12. Rutas de evacuación

[11] http://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra

El punto de partida o inicial es el círculo de color azul, el cual representa la ubicación actual del usuario, a partir de ahí se trazará una ruta de salida hacia la ruta de evacuación más cercana el cual está representada por una estrella, lugar que se ha definido como una zona de evacuación segura.

En el mapa se aprecia claramente la zona de evacuación hacia donde los usuarios deben llegar en algún momento, para lo cual la aplicación generará las rutas para cada uno de ellos dependiendo del lugar donde se encuentren actualmente, dando al usuario una trayectoria clara y de fácil uso.

En el caso de que exista algún obstáculo como el que se puede apreciar en la Figura 12 representado por el círculo rojo, la aplicación debe re-calcular las rutas de salida con el fin de obtener una nueva y así llegar hasta el lugar que queremos, cabe recalcar que estos obstáculos son ubicados dependiendo del reporte que un usuario pueda emitir o no.

4.4.5. Aspectos a considerar para la renderización de las rutas de evacuación.

Para que las rutas sean renderizadas correctamente es necesario que previamente se tomen en cuenta lo siguiente:

- Todos los puntos de acceso wifi deben tener comunicación fluida y estar en condiciones operativas.
- El trazado de las rutas tiene una variación con respecto a la visualización en un dispositivo móvil simulado o real debido al redondeo de los datos.
- El cálculo de la distancia hasta el punto de llegada se maneja en metros (m).
- No se considera el tiempo que tome el traslado desde el lugar del usuario hasta la zona de evacuación.
- No se considera la dirección de las vías debido a que únicamente es para personas.
- Cuando se genera un nuevo mapa de otra ciudad es necesario que conjuntamente se genere la matriz de rutas y subirlas al dispositivo.
- El archivo gpx generado por Mobac no debe ser modificado.

4.5. Definición del área de cobertura

La información que puede ser visualizada en el mapa consta de:

a) Puntos de la matriz base.
b) Área de cobertura.

Cada uno de los ítems expuestos son opcionales su visualización, siempre y cuando consten dentro del área de cobertura.

El objetivo principal de ésta sección es la definición del área de cobertura dentro del cuál va a funcionar la aplicación, si el reporte de un obstáculo ocurre fuera del rango de cobertura previamente definido no es posible visualizarlo en el mapa y todos los datos enviados se perderán.

La delimitación del área de cobertura queda a criterio del administrador del nodo que actúa como servidor, dicha área se delimita al momento de crear la matriz de rutas; en la parte central de la Figura 13 se encuentra señalado el entorno sobre el cual la aplicación debe funcionar, dentro de dicha área se deben ubicar todos los puntos que conforman la matriz como se observa en la Figura 14.

El límite del área de cobertura queda a criterio del administrador pudiendo éste ampliarla o no de acuerdo a su criterio.

4.5.1. Funciones principales.

Dentro de las funciones que debe cumplir la definición del área de cobertura constan las siguientes:

- Definir al área de cobertura de la aplicación.
- Visualización de los puntos que forman la matriz del área de cobertura. Interfaz de integración entre de Openstreetmap y Mobac.

4.5.2. Funcionamiento.

Para la generación del mapa se ha utilizado la herramienta Mobac la cual está disponible de forma libre y gratuita pasa su uso.

Inicialmente Mobac ofrece la posibilidad de definir un área de cobertura sobre la cual posteriormente se generará el mapa a utilizar. El área que se ha seleccionado se puede apreciar en la Figura 13 y es sobre la cual se van a ubicar los puntos que conformarán la matriz base descrita posteriormente.

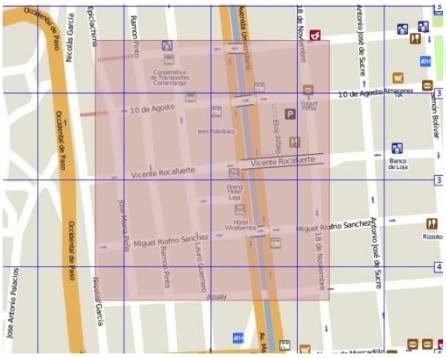

Figura 13. Definición del área de cobertura

El área que el usuario selecciona para su generación se puede ajustar con parámetros como el nivel del zoom el cual refleja los pies de altura aproximada.

4.6. Matriz de Rutas

Para buscar las rutas por las cuales un usuario puede encontrar la salida desde el lugar actual se ha utilizado la herramienta Mobac para generar una matriz con los puntos principales que existirán dentro de un área específica y servirán como base para calcular las rutas.

La matriz de rutas la puede generar el administrador del servidor con la ayuda de la herramienta Mobac, la cual permite exportar un archivo con extensión .gpx el cuál posteriormente deberá ser ubicado en el dispositivo móvil que actúa como servidor el cual a su vez será utilizado por la aplicación para devolver los datos del nuevo mapa a los demás nodos de la red.

4.6.1. Funciones principales de la matriz de rutas.

- Renderización del mapa en modo offline.
- Opciones de zoom en el mapa.
- Trazado de las rutas de la zona de evacuación para cada usuario.
- Re-calcular las rutas en caso de existir algún obstáculo reportado recientemente.
- Cuando el usuario no esté dentro del área de cobertura trazado por la matriz de rutas, la aplicación informará de tal circunstancia.
- Las funciones de trazados de rutas únicamente funciona dentro del área de cobertura como se indica en la sección 4.5.

Es importante mencionar que para calcular las rutas de salida previamente se debe cargar una matriz generada con la herramienta Mobac para realizar la búsqueda y trazado de las rutas de salida en el mapa; el área donde se colocan los puntos de la matriz depende del usuario que genera dicha matriz.

Los requisitos para la búsqueda de las rutas son:

1. **Matriz de puntos generada correctamente**.- Esta matriz debe contener todos los puntos ubicados en las secciones de las calles dentro de una área definida.
2. **Mapa Source**.- Es el proveedor del mapa sobre el cual se trazaran las rutas, en este caso el proveedor seleccionado es OpenStreetMap CloudMade Default Style, éste proveedor de mapa tiene un nivel de detalle más específico del área de cobertura sobre la cual se trabaja.
3. **Exportación a formato GPX**.- Una vez que ya se tiene marcado todos los puntos importantes se debe exportar la matriz a un formato gpx que posteriormente será utilizado por la aplicación.
4. **Zona de evacuación**.- Esta zona segura de evacuación debe estar claramente definida dentro del mapa, así mismo se debe tener en cuenta que la matriz de las rutas debe estar relacionada directamente con la ubicación de dicha zona.

La aplicación buscará las rutas únicamente dentro de la matriz de puntos que se hayan definido claramente

En la Figura 14 se puede apreciar como es la generación de la matriz de rutas para que sea utilizada por la aplicación y encontrar las rutas de salida en cualquier momento hacia la zona 1 que sería la zona de evacuación.

Conjuntamente al mapa generado se debe descargar el archivo gpx con la matriz de puntos establecidos para el manejo de las rutas, el formato de dicho archivo se describe a continuación.

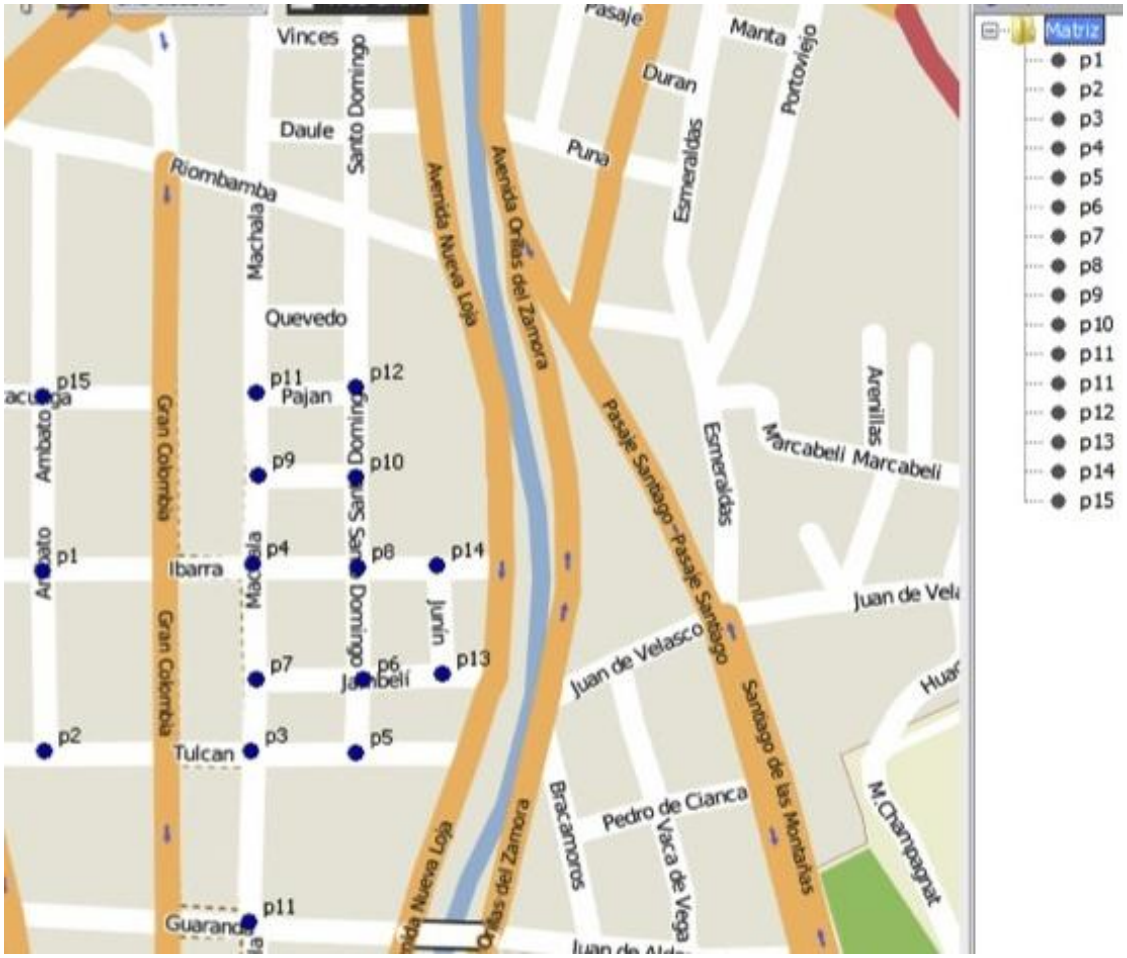

Figura 14. Matriz de Puntos

4.6.2. Formato Gpx en Mobac.

El formato en el que Mobac genera el archivo gpx con la matriz se describe a continuación:

```
// Formato de de archivo .gpx generado por Mobac.
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<gpx xmlns="http://www.topografix.com/GPX/1/1" creator="Mobile Atlas
Creator (MOBAC)"
<metadata>
<name>Matriz</name>
</metadata>
<wpt lon="-79.205138683319091796875" lat="-
3.985077686957426479352761816699057 81">
<name>p1</name>
</wpt>
<wpt lon="-79.202810525894165039062 5" lat="-
3.985677048898699226242570148315280 6">
```

```
<name>p13</name>
</wpt>
</gpx>
```

Donde los metadatos se describen a continuación:

xmlns: Es el namespace que Mobac le asigna a nuestro archivo.

versión: Es la versión del archivo generado.

metadata: Es el nombre que se le asignado al conjunto de puntos que están presentes en la matriz.

wpt: (***waypoints***) Etiqueta para contener un punto de geolocalización conformado por la longitud y latitud cada uno con sus respectivos valores.

lon: Es el valor relacionado a la longitud de la ubicación expresado como un tipo de dato double.

lat: Es el valor relacionado a la latitud de la ubicación expresado como un tipo de dato double.

4.6.3. Matriz Base.

Es la matriz predeterminada sobre el mapa en el cual se trabajará para la ubicación de los diferentes objetos según estén disponibles. Dicha matriz base es necesaria debido a que posteriormente se la utilizará para trazar las rutas y ubicar de forma exacta a un objeto. Cabe mencionar que ésta matriz solo estará disponible en tiempo de desarrollo para poder realizar los cálculos de distancias al momento de buscar la mejor ruta.

Una vez que se tiene definido el área de cobertura posteriormente se ubican los puntos que conformarán la matriz base como se observa en la Figura 15

Figura 15. Puntos de la matriz base

Dichos puntos ubicados en el mapa posteriormente serán exportados con formato gpx para que sean reconocidos por la aplicación y se obtenga una matriz de navegación dentro del área de cobertura.

Dentro de los aspectos que se pueden destacar tenemos los siguientes:

- Generación de los mapas independiente del dispositivo en el que se vaya a utilizar.
- Se ha logrado ubicar los puntos que conformaran la matriz base sobre la cual trabajará la aplicación.
- A mayor número de puntos mayor es la precisión con que los objetos serán renderizados.
- Cada punto se encuentra en línea recta con respecto a su vecino inmediato.

4.6.4. Cálculo de distancia entre dos puntos.

En la aplicación del presente proyecto es importante conocer a detalle la distancia que existe entre dos coordenadas geográficas, debido a que es necesario al momento de trazar

una ruta de salida, por lo cual cada distancia debe ser cuidadosamente calculada teniendo en cuenta el margen de error que puede derivarse, dicho margen de error puede variar hasta un máximo de 15 metros, su variación depende de las características del dispositivo que la emplea.
Calcular la distancia entre dos puntos sobre un plano podría llegar a ser relativamente sencillo, sin embargo, cuando éstos dos puntos los ubicamos sobre la esfera terrestre, es decir, lo que pretendemos es calcular la distancia lineal entre dos posiciones dadas (latitud + longitud), la tarea conlleva cierto grado de complejidad.

Básicamente se complica porque en el cálculo de la distancia entre ambas posiciones debemos contemplar la curvatura terrestre. Es aquí donde entra en escena la Fórmula del Haversine,[12] la cual se detalla a continuación:

$$hav(h) = hav(lat2 - lat1) + \cos(lat1) \cdot \cos(lat2) \cdot hav(lon2 - lon1) \quad (1)$$

donde *h* representa al ángulo central de un circunferencia

$$h = \frac{d}{r} \quad (2)$$

Despejando *d* de (1)

$$d = r\, hav^{-1}(h) \quad (3)$$

La fórmula del haversine está definida por:

$$hav\,(lat) = sen^2\left(\frac{lat}{2}\right) \quad (4)$$

Reemplazando (4) en (3)

$$d = 2r.arcsen\left(\sqrt{h}\right) \quad (5)$$

Reemplazando (1) en (5) se tiene:

$$d = 2r.arcsen\left(\sqrt{hav(lat2 - lat1) + \cos(lat1) \cdot \cos(lat2) \cdot hav(lon2 - lon1)}\right) \quad (6)$$

Reemplazando (4) en (6) y resolviendo *d*:

$$d = 2r.arcsen\left(\sqrt{sen^2\left(\frac{lat2-lat1}{2}\right)} + \cos(lat1) \cdot \cos(lat2) \cdot sen^2\left(\frac{lon2-lon1}{2}\right)\right) \quad (7)$$

d = Es la distancia entre los dos puntos tomando como base el círculo mayor de una esfera,

r = Radio de la tierra,

lat1 = La latitud del punto 1,

lat2 = La latitud del punto 2,

lon1 = longitud del punto 1

lon2 = longitud del punto 2

[12] http://es.wikipedia.org/wiki/Formula_del_Haversine.

Para utilizar la Fórmula del Haversine es necesario definir *r* cuyo valor es relativo a la latitud, pues al no ser la Tierra perfectamente redonda, el valor del radio ecuatorial es de 6378 km mientras que el polar es de 6357 km.

En la Figura 16 se tienen los punto *B* y *A*, la distancia entre éstos dos puntos está determinada por ángulo entre el punto de origen y destino, cada punto tiene una longitud y latitud asociada, siendo *B* el punto de inicio y *A* el punto de destino.
Para calcular la distancia entre A y B se reemplazan los valores de los puntos de origen y destino en (7).

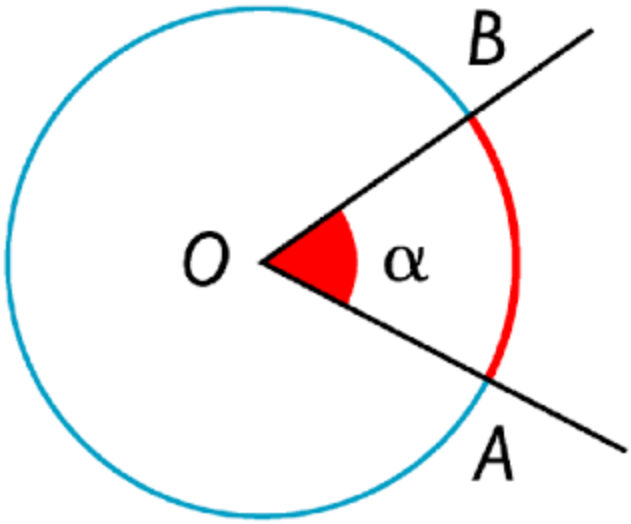

Figura 16. Ángulo central en un círculo
Fuente: http://chickilinas.blogspot.com/

El ángulo central α es el ángulo que tiene su vértice en el centro de la circunferencia y los lados *BO* y *AO* son radios de ella.
La medida del arco *BA* está determinado por el ángulo central formado por *BOA*.

CAPÍTULO 5
DESARROLLO DE LA SOLUCIÓN

En este capítulo se implementa la solución previamente planteada en el capítulo anterior, de tal forma que cumpla con las especificaciones de software descritos en el capítulo 4.

5.1. Descripción de la solución

Para el desarrollo de la solución a la problemática planteada de comunicación entre dispositivos móviles es necesario hacer uso de la tecnología wifi, tomando en cuenta que el ámbito de funcionamiento de la aplicación será en un área con dicha tecnología.

5.2. Herramientas utilizadas en el presente proyecto

Para el desarrollo de la aplicación se utilizaron las siguientes herramientas.

5.2.1 Eclipse ID 3.7

5.2.2 SDK tools Android

5.2.3 Android S.O. v4.1

5.2.4 Mobac (Mobile Atlas Creator) Osmdroid-android

5.2.5 Antena receptora de GPS Archivos Gpx

5.2.1. Eclipse ID 3.7 Indigo.

Es un entorno de desarrollo integrado (IDE por sus siglas en inglés) de código abierto multiplataforma para el desarrollo de aplicaciones empresariales bajo distintos lenguajes de programación tales como Java, Php, Python, C, C++, Ruby, Javascript, etc. Así mismo el IDE eclipse contiene una comunidad de usuarios desarrolladores que constantemente están entendiendo las funcionalidades a fin de mejorar sus servicios como entorno de desarrollo. Inicialmente eclipse fue desarrollado por IBM como el sucesor de VisualAge.[13] En la actualidad es desarrollado y mejorado por la fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios.

5.2.2. SDK tools Android.

El SDK Android provee bibliotecas API (Application Programming Interface)[14] y herramientas de desarrollo necesarias para construir, testear y depurar aplicaciones para dispositivos Android.

[13] http://http://www-01.ibm.com/software/awdtools/visgen/.
[14] http://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones

5.2.3. Android OS v4.1.

"Un sistema operativo basado en Linux, diseñado principalmente para móviles con pantalla táctil como dispositivos inteligentes o tabletas. La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de bibliotecas de Java en una maquina virtual Dalvik con compilación en tiempo real."[15]

El nuevo sistema operativo de Google, lanzado el 21 de Octubre de 2008, es un sistema operativo open source y gratuito, y sobre todo las características traen consigo.

5.2.4. Mobac (Mobile Atlas Creator).

Es una aplicación multiplataforma de código abierto (GPL)[16] que crea mapas para dispositivos GPS y aplicaciones de dispositivos móviles como TrekBuddy[17], AndNav[18] y otras aplicaciones de Android y WindowsCE[19], para utilizarlos sin necesidad de conexión de datos. Dispone de una gran cantidad de fuentes para generar los mapas, aunque a partir de la versión 1.9 beta 2 se han eliminado bastantes fuentes a causa del reclamo hacia los programadores por parte de los propietarios de los mapas.

5.2.5. Osmdroid-android.

Es una librería desarrollada en java que permite la utilización de la api de OpenStreetMap para renderizar el mapa en nuestras aplicaciones ya sean de escritorio o vía web, actualmente ésta librería está desarrollada específicamente para dispositivos con el sistema operativo android en cualquiera de sus versiones.

5.2.6. Antena receptora de GPS.

Este componente es uno de los que indispensablemente debe contener el dispositivo móvil sobre el cual va a funcionar la aplicación, se utiliza para obtener la ubicación actual y enviárselo a los demás usuarios cuando se reporta la existencia de un obstáculo en un momento dado.

"El sistema GPS está constituido por 3 segmentos que son: El espacial, el de control y el del usuario"[20] Los distintos tipos de receptores disponibles en el mercado para uso civil constituyen la parte esencial de este último.

[15] http://es.wikipedia.org/wiki/Android.
[16] http://es.wikipedia.org/wiki/GNU_General_Public_License
[17] www.trekbuddy.net
[18] www.andnav.org
[19] http://es.wikipedia.org/wiki/Windows_CE
[20] http://www.reparacionestopograficas.com/~manuales/Manual3-GPS.pdf.

Los receptores GPS cubren las más diversas posibilidades de aplicación de este sistema. Ordenándolos de menor a mayor prestación, se los puede clasificar en:

- Receptores de navegación.
- Receptores de posicionamiento mono frecuencia.
- Receptores de posicionamiento de doble frecuencia.

En los dispositivos que se utilizaron en el presente proyecto se hace uso de los receptores de navegación debido a que son los más económicos y usualmente son pequeños y portátiles lo cual facilita su inclusión en dispositivos móviles.
La precisión de éstos equipos no supera los +- 10 m.

5.2.7. Archivos Gpx.

"GPX, o GPS eXchange Format (Formato de Intercambio GPS) es un esquema XML para transferir datos GPS entre aplicaciones. Se puede usar para describir puntos (waypoints), recorridos (tracks), y rutas (routes)."[21] Los archivos gpx se encuentran en modo texto y basados en XML, lo que en resumidas cuentas facilita enormemente la conversión de la información a cualquier otro formato. Es un formato estándar, cuya especificación es pública y de uso libre.

5.3. Configuración inicial de una aplicación wifi

Toda aplicación wifi de android debe contener al menos los siguientes permisos para establecer una comunicación con otro dispositivo:

```
<uses-sdk android:minSdkVersion="14"/>
<uses-permission
android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission
android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission  android:name="android.permission.INTERNET"/>
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
```

[21] http://es.wikipedia.org/wiki/GPX.

5.3.1. Cálculo de la distancia entre dos coordenadas.

Para el cálculo de la distancia entre dos coordenadas es necesario el punto de inicio y el punto de llegada, ambos puntos deben tener sus coordenadas de geolocalización previamente definidas. Unos de los inconvenientes existentes con los puntos a calcular la distancia entre si es que las latitudes y longitudes se exponen en grados / minutos / segundos y por tanto es necesario convertirlas a radianes como se presenta en el algoritmo 5.1.

Algoritmo 5.1: Algoritmo de cálculo de distancia entre dos puntos.

```
/**
* Calcula la distancia en metros que existe  entre dos coordenadas
@param  lon1. Longitud del punto de origen
@param lat1. Latitud del punto de origen
@param lon2. Longitud del punto de destino
@param lat2. Latitud del punto de destino
@return. Distancia en metros  entre los dos puntos, origen y destino.
 */
private static double CalcularDistanciaHaversine(double lon1, double lat1,
double lon2, double lat2) {
double radio = 6378; // km Radio de la tierra

lat1 = Math.toRadians(lat1);
lon1 = Math.toRadians(lon1);
lat2 = Math.toRadians(lat2);
lon2 = Math.toRadians(lon2);

double dlon = (lon2 – lon1);
double dlat = (lat2 – lat1);

double sinlat = Math.sin(dlat / 2);
double sinlon = Math.sin(dlon / 2);

double a = (sinlat * sinlat) + Math.cos(lat1)*Math.cos(lat2)*(sinlon*sinlon);
double c = 2 * Math.asin (Math.min(1.0, Math.sqrt(a)));

double distancia = radio * c * 1000;

return distancia;
}
```

5.4. Renderización de objetos en el mapa

En esta sección se define el proceso a seguir para renderizar el mapa y los usuarios sobre el mismo, para que dichos usuarios se presenten en el dispositivo primero estos deben enviar una notificación con los datos de su ubicación actual.

Funciones principales:

1. Renderización del mapa en modo offline
2. Visualización de todos los usuarios activos.
3. Opciones de zoom del mapa
4. Navegabilidad dentro del área del mapa definido al momento de la descarga.
5. Actualización del mapa en cualquier momento con el fin de visualizar la última ubicación de los usuarios.
6. Solo un nodo será el encargado de brindar al último usuario que ingresa la información de geolocalización de todos los usuarios existentes.

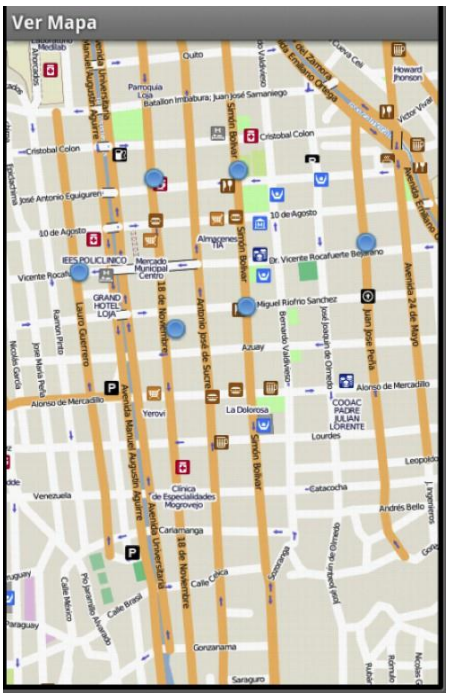

Figura 17. Usuarios existentes

En la Figura 17 se visualizan los usuarios existentes en el mapa dentro de un área definida, cada usuario que se visualiza puede ir cambiando su posición, lo cual conlleva a que su ubicación sea actualizada constantemente o cuando el usuario lo requiera.

A continuación en la Tabla 10 se presenta la simbología que representa a un usuario en el mapa.

Tabla 10. Simbología de usuario

Símbolo	Significado
	Usuario existente dentro del área definida.

Cabe recalcar que no existirá la opción de comunicación directa entre los usuarios de la red, debido a que Android SO no permite una conexión entrante sin el consentimiento del usuario.

5.4.1. Sincronización de la información.

Algoritmo 5.2: Algoritmo de sincronización de información

```
Mensaje_data  mensaje = new  Mensaje_data();
// Se define el tipo de mensaje que sera  enviado
entre dos  dispositivos
mensaje.accion = TIPO_SINCRONIZACION;
oos.writeObject(mensaje);
saveDataFromServer((ArrayList<Matriz>)
ois.readObject());
```

En el algoritmo 5.2 se presenta la sincronización de un cliente con los datos de un nodo, siempre y cuando este último se encuentre disponible y operable.
En la constante *TIPO_SINCRONIZACION* se establece el tipo de sincronización que se llevará a cabo, posteriormente a través del método *writeObject* se escribe el mensaje a enviar al nodo destino; y finalmente a través de *ois.readObject()* se leen los datos devueltos por dicho nodo destino, los cuales posteriormente utilizando el método *saveDataFromServer* se almacenan en la base de datos local de cada dispositivo android.
Cabe recalcar que el paso de mensajes entre el nodo destino que a su vez actúa como servidor y los nodos restantes de la red se realiza únicamente por medio de la serialización de objetos, haciendo que éstos sean transportables en la red.

5.4.2. Apertura del socket de comunicación.

Para establecer un canal de comunicación entre dos dispositivos android se realiza mediante sockets como se describe en el algoritmo 5.3 el cual crea una instancia de la clase Socket, la cual a su vez recibe dos argumentos de entrada que son la dirección ip del nodo destino definida en *IP_SERVIDOR* y el puerto definido en *PUERTO_SERVIDOR*.
Si la conexión ha sido exitosa el objeto *sk* contiene una instancia activa del enlace hacia el nodo destino y sobre el cual se puede escribir los tipos de mensaje a enviar.

Algoritmo 5.3: Apertura de un socket de comunicación en un cliente

```
Socket sk = new Socket(IP_SERVIDOR, PUERTO_SERVIDOR);
```

Por otro lado, en el nodo que actúa como servidor es necesario crear un nuevo objeto de tipo Socket a la espera de la conexión de un nodo, como se puede apreciar en el algoritmo 5.4, el objeto *client* contiene una instancia de enlace hacia dicho nodo.

Algoritmo 5.4: Apertura de un socket de comunicación entre un nodo origen y otro destino

```
Socket  client = serverSocket.accept();
client.getInputStream();
```

Una vez que se ha establecido conexión entres dos dispositivos, éstos pueden enviar o recibir información mediante el método *oos.writeObject(objectSend)*. Para que los datos se envíen desde y/o hacia el nodo destino es necesario que tanto el cliente como el nodo destino implementen la misma jerarquía de paquetes, es decir el contexto de la aplicación del nodo destino debe ser la misma que en el cliente.

5.5. Localización

Se trata del escenario principal de localización. Se compone de tres partes: obtención de las intensidades de los puntos de acceso de la red, recuperación de la longitud-latitud y representación en el mapa. En la primera parte el programa se encuentra a la espera del primer sondeo de los nodos existentes, a partir de ese momento cada barrido se realiza bajo petición del usuario. Una vez hecho el barrido se obtiene la información para la localización, es decir, los datos de geolocalización de cada nodo detectado. Una vez obtenidos y almacenados los valores se pasa a la siguiente fase la difusión de la información a los demás usuarios conectados a la red; en esta segunda fase, como se ha descrito en el párrafo anterior, se procede a compartir la información

con los demás usuarios de tal manera que cada nodo conozca lo que existe por las inmediaciones a él. Finalmente, cuando se envía la información a los demás usuarios, ésta es almacenada tanto en el dispositivo emisor como en el receptor. En la Figura 18 se presenta el flujograma para renderizar los usuarios en el mapa. Estas tres fases se realizan periódicamente hasta que el usuario decide finalizar la aplicación o el dispositivo se encuentra tan comprometido de memoria como para cerrar la aplicación.

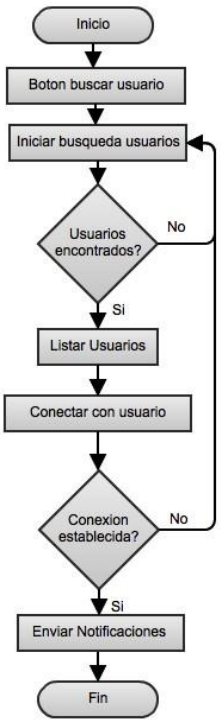

Figura 18. Enviar notificación a un usuario

5.6. Matriz de rutas de salida

Para buscar las rutas por las cuales un usuario puede encontrar la salida desde el lugar actual, se ha utilizado la herramienta Mobac para generar una matriz con los puntos principales que existirán dentro de un área específica y servirán como base para calcular las rutas.

Es importante mencionar que para calcular las rutas de salida previamente se debe cargar una matriz generada con la herramienta Mobac para realizar la búsqueda y trazado de las rutas de salida en el mapa; el área donde se colocan los puntos depende del usuario que genera dicha matriz.

El requisito para la búsqueda de las rutas es la matriz de puntos generada correctamente, esta matriz debe contener todos los puntos ubicados en las intersecciones de las calles dentro de un área definida.

5.6.1. Algoritmo de Dijkstra

Según (Alonso J, 2008), el algoritmo de Dijkstra[22] está diseñado para encontrar las rutas más cortas entre el nodo origen y cada uno de los nodos de la red. Este tipo de algoritmo es de tipo "greedy"[23] porque en cada iteración elige la mejor opción de las posibles con la esperanza de encontrar así la mejor solución global.

Este algoritmo trabaja por etapas, y toma en cada etapa la mejor solución sin considerar consecuencias futuras; si posteriormente se encuentra una mejor ruta a la existente, entonces el camino obtenido hasta una determinada etapa es cambiado por la nueva ruta más corta.

En el Algoritmo 5.5 se presenta la implementación del algoritmo de Dijkstra el cual trabaja de forma recursiva y es el encargado de buscar y trazar la ruta de evacuación más cercana desde el punto de origen hacia la zona de evacuación más cercana.

5.6.1.1. Funcionamiento.

Para iniciar la búsqueda del camino más corto, inicialmente se marcan todos los vértices como no utilizados. El algoritmo parte de un vértice origen que será el punto de inicio, a partir de ese vértice se evalúa sus adyacentes más próximos, como el algoritmo de Dijkstra usa una técnica greedy.

La técnica greedy utiliza el principio de que para que un camino sea óptimo, todos los caminos que contiene también deben ser óptimos, entre todos los vértices adyacentes se busca el que esté más cerca de nuestro punto origen, posteriormente se toma como punto intermedio y se comprueba si es posible llegar más rápido a través de este vértice a los demás. Posteriormente se escoge al siguiente más cercano (con las distancias ya actualizadas) y se repite el proceso. Esto se lo repite tantas veces sea necesario hasta que el vértice no utilizado más cercano sea el nodo destino.

Al proceso de actualizar las distancias tomando como punto intermedio al nuevo vértice se le conoce como relajación.

Este algoritmo se lo aplicó específicamente para encontrar la mejor ruta de evacuación de un usuario desde el punto de localización hasta la zona de evacuación más cercana.

[22] http://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra
[23] http://apuntesinformaticauade.blogspot.com/2013/02/capitulo-16-algoritmos-greedy-parte-i.html.

Algoritmo 5.5: Algoritmo de Dijkstra

```
/**
* Encuentra la ruta mas corta  desde un nodo origen a
un nodo  destino
                  utilizando el algoritmo de
Dijkstra
@param  inicio. Nodo  inicial  para  calcular  la
ruta mas corta. @param fin. Nodo  destino en el cual
termina la ruta
*/
publicString encontrarRutaMinimaDijkstra(String
inicio, String fin)
{
// calcula la ruta mas corta del inicio a los
demas
encontrarRutaMinimaDijkstra(inicio);
// recupera el nodo final de la lista de terminados
Nodo tmp = new Nodo(fin);
if(!listos.contains(tmp))
{
System.out.println("Error, nodo no alcanzable");
return "Bye";
}
   tmp  = listos.get(listos.indexOf(tmp));
   Double  distancia = tmp.distancia;
   // crea una pila para  almacenar la ruta desde
el nodo final al origen
   Stack<Nodo> pila = new Stack<Nodo>();
   while(tmp != null)
   {
      pila.add(tmp);
      tmp = tmp.procedencia;
   }
   String ruta = "";
```

```
   // recorre la pila para armar la ruta en el
orden  correcto
   while(!pila.isEmpty())
      ruta += (pila.pop().id + ";");
   return ruta;
}
```

5.7. Almacenamiento de información

A esta clase la llamaremos *BDController* y lo primero que debemos hacer es definir una serie de campos estáticos:

Nombre de la base de datos:

```
public static final String DATABASE_NAME =
"androcode_ormlite.db";
```

Versión de la base de datos:

```
Public static final int DATABASE_VERSION = 1;
```

Creación de las tablas: La creación de las tablas se realiza de forma automática dependiendo de la versión de la base de datos, es decir por cada versión nueva se crean todos los objetos en la base de datos. Al momento de ejecutarse la aplicación, si no existe la base de datos, se lanza el evento *onCreate* que es capturado por nuestra clase y genera las tablas necesarias para almacenar la información de los objetos a renderizar en el mapa:

Algoritmo 5.6: Algoritmo para la creación de las tablas

```
@Override
public void onCreate(SQLiteDatabase db,
ConnectionSource connectionSource) {
  try {
TableUtils.createTable(connectionSource,
Usuario.class);
TableUtils.createTable(connectionSource,
Obstaculo.class);
TableUtils.createTable(connectionSource,
ZonaEvacuacion.class);
TableUtils.createTable(connectionSource,
```

```
Matriz.class);
TableUtils.createTable(connectionSource,
Adyacente.class);
TableUtils.createTable(connectionSource,
PersonalLocation.class);

} catch (SQLException e) {
  throw new  RuntimeException(e);
}
}
```

En el constructor de la clase *DBHelper* se pasa el contexto de la clase desde donde está siendo invocado para establecer en el contexto de la aplicación la base de datos con la que se va a trabajar.

```
Public DBHelper(Context  context) {
super(context, ConstantesApp.DATABASE_NAME, null,
ConstantesApp.DATABASE_VERSION);
this.context = context;
}
```

5.8. Almacenamiento de información

En el caso de una actualización de la base de datos, el cual a su vez implica una nueva versión, se lanzará el evento *onUpgrade*, el cual es el encargado de volver a crear la base de datos con nueva información existente.

Algoritmo 5.7: Algoritmo para la actualización de la base de datos.

```
@Override
public void  onUpgrade(SQLiteDatabase db,
ConnectionSource  connectionSource,
 int oldVersion, int newVersion)     {
List<String> allSql = new ArrayList<String>();
onCreate(db, connectionSource);
for (String sql : allSql) {
  db.execSQL(sql);
}
```

}

Cuando existen nuevos cambios en la base es importante que se respalde toda la información debido a que ésta será eliminada y reemplazada por la nueva versión del modelo de datos.

5.8.1. Generación de base de datos.

Se ha utilizado el motor de base de datos sqlite, la cual forma parte de los componentes de android debido a su rapidez y facilidad para acceder a los datos.

Se utiliza HashTables[24] para almacenar los datos de una manera eficiente; y para almacenar la base de datos en el disco se emplea la serialización, que permite guardar un estado persistente y poder recuperar posteriormente desde otro módulo. Se realizan diferentes pruebas para mejorar el control de errores e incrementar el rendimiento del módulo sobre todo en la gestión de archivos y en el uso de memoria. Este módulo servirá durante todo el desarrollo para realizar pruebas de funcionamiento de aplicación utilizando mapas en modo offline.

5.8.2. Integración de módulos.

La aplicación debe ser autónoma, por lo que se deben integrar los módulos para que puedan ser ejecutados desde el dispositivo directamente. Se debe prestar especial atención en los mensajes que cada módulo emite hacia otro para el tratamiento y explotación de los datos. Tras la integración, se realizan las pruebas integradas correspondientes y se genera el paquete que contiene la aplicación. Para realizar las pruebas de usuario y poner en funcionamiento en un ambiente real con dispositivos físicos.

5.8.3. Pruebas de usuario.

Las pruebas de usuario se realizan en un entorno real, probando que la aplicación funciona correctamente y responde a los requisitos solicitados. Se realizan diversos cambios para limitar las acciones disponibles para el usuario y así evitar comportamientos anómalos de la aplicación.

5.8.4. Manejo de errores.

Durante el funcionamiento de la aplicación, si se produjera alguna dificultad interna tanto de la aplicación, como del dispositivo, la aplicación lanzará un mensaje por pantalla indicando que existe un problema y que es necesario cerrar la aplicación.

[24] http://en.wikibooks.org/wiki/Data_Structures/Hash_Tables.

5.9. Visualización de objetos en el mapa

La renderización de objetos en el mapa dentro de un dispositivo móvil de ha realizado de forma offline es decir sin la conexión a la internet y, en su lugar, se han utilizado las funcionalidades que dispone la librearía osmdroid.

Inicialmente las aplicaciones ofrecen la posibilidad de visualizar el mapa con los objetos (obstáculos, ubicación actual del usuario y puntos de evacuación) o simplemente un mapa base sobre el cual se le pueden ir agregando objetos a visualizar.

El despliegue de objetos en un mapa tiene por objetivo principal la visualización de información al usuario que puede ser de mucha utilidad en cualquier momento. Dado que inicialmente se cuenta con datos predefinidos se ha procedido a visualizar toda la información que el usuario cree conveniente.

La renderización del mapa en el dispositivo móvil es independiente de la conectividad existente en el mismo dotando a la aplicación de una funcionalidad en modo offline.
La información que puede ser visualizada en el mapa consta de:

- Obstáculos
- Puntos de evacuación.
- Rutas de evacuación.
- Ubicación actual.
- Matriz base.

La visualización de cada uno de los ítems expuestos es opcional, siempre y cuando consten dentro del área de cobertura.

5.9.1. Funciones principales.

- Visualización de los obstáculos
- Visualización del usuario actual
- Visualización de los puntos de evacuación.
- Trazado de la ruta de salida más corta desde la ubicación actual del usuario hacia los puntos de evacuación.

5.9.2. Aplicación.

Para visualizar el mapa primeramente se deben seleccionar las opciones que se desea renderizar, para ello seleccionamos los objetos como se describe en la Figura 19. Dentro de las opciones a visualizar podemos seleccionar los siguientes objetos:

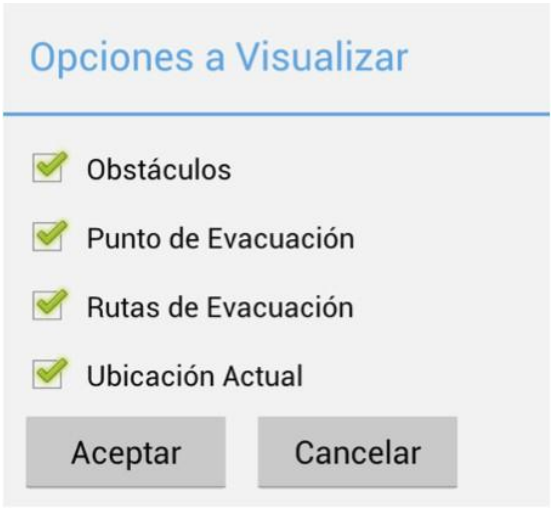

Figura 19. Opciones a visualizar

- **Obstáculos:** Son todos los obstáculos que existen dentro del área de cobertura en el mapa, todos los obstáculos existentes son los mismos que anteriormente han sido reportados por los usuarios.
- **Punto de evacuación:** Son todos los puntos de evacuación existentes dentro de la zona delimitada, y es hacia donde trazará la ruta de salida desde la ubicación actual del usuario.
- **Rutas de evacuación:** Es la ruta que la aplicación renderiza desde la ubicación actual del usuario hacia el punto de evacuación más cercano, utilizando la ruta más corta.
- **Ubicación Actual:** Es la ubicación actual del usuario que va a visualizar el mapa.

  Una vez que el usuario ya ha seleccionado los diferentes objetos a visualizar la aplicación renderiza en el mapa como se muestra en la Figura 20.

Figura 20. Renderización de objetos en el mapa

Todos los objetos que se han utilizado en la visualización de las rutas de salida se pueden apreciar en la Tabla 11, la cual describe a cada objeto de acuerdo a su función.

Tabla 11. Simbología utilizada para renderizar objetos en el mapa

Símbolo	Significado
	Representa la ubicación actual del usuario que esta visualizando el mapa.
	Representa a todos los obstáculos que han sido reportados por los usuarios dentro del área de cobertura.
	Representa el o los puntos evacuación más cercanos a la ubicación actual del usuario.
	Corresponde al trazado de la ruta de salida más corta desde la ubicación actual del usuario hacia el punto de evacuación más cercano.

5.9.3 Funcionamiento

El funcionamiento general de la aplicación para visualizar el mapa con los objetos deseados se puede observar en la Figura 21.

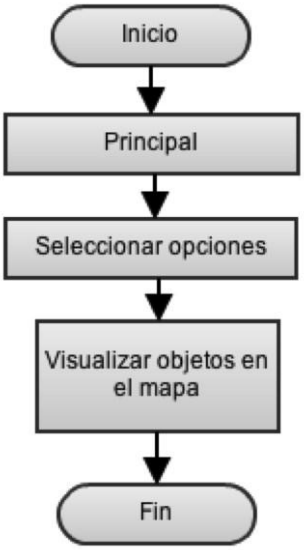

Figura 21. Flujo de funcionamiento de la visualización del mapa.

Se tiene tres fases que son: Principal, Seleccionar Opciones y visualizar objetos en el mapa.

Principal.- Es el menú que inicialmente el usuario selecciona en la aplicación.
Seleccionar opciones.- Son las opciones presentadas en la Figura 19 que tiene el usuario para seleccionar y renderizar en el mapa.

Una vez que se han seleccionado todos los objetos que se quieren visualizar, este proceso renderiza dichos objetos en el mapa como se presenta en la Figura 20, en el cual se han seleccionado todos los objetos.
Como resultado del proceso de renderización de los objetos en el mapa se puede destacar los siguientes aspectos:

1. Selección de objetos a visualizar a gusto del usuario
2. Disponibilidad de la renderización del mapa independientemente de la conexión de datos existente debido a que trabaja en modo offline.
3. Selección de objetos a visualizar a criterio del usuario.
4. Utilización de simbología representativa para indicar los diferentes objetos renderizados.

Finalmente es importante mencionar que todos los datos que se han renderizado en el mapa proviene de información predeterminada debido a que no es posible encontrarnos dentro del área de cobertura de la aplicación.

CAPÍTULO 6
PRUEBAS DE FUNCIONALIDAD

En este capítulo se van a presentar las pruebas realizadas a la aplicación. Se presentará los logs que la aplicación genera al momento de enviar o recibir una notificación como se presenta en la Figura 22.
Las pruebas se realizaron entres tres dispositivos móviles android 4.0.1, ninguno de ellos cuenta con conexión a la internet.

6.1. Entorno de pruebas

Para la realización de las pruebas se ha tomando una área predefinida donde se ha generado un mapa offline con Mobac y se han puesto en funcionamiento 2 dispositivos con el sistema operativo android 4.0.1. En cada uno de éstos dispositivos se ha instalado la aplicación para su funcionamiento y posterior evaluación para lo cual se ha dispuesto de los siguientes requerimientos:

1. Un nodo debe ser maestro, es decir donde se conectarán inicialmente los usuarios existentes de la red.
2. Todos los nodos deben tener acceso a la red wifi disponible.
3. La aplicación debe estar correctamente instalada en todos los dispositivos.
4. Cada dispositivo debe tener permisos de root para que se pueda escribir/leer en el sdcard del dispositivo.
5. El dispositivo debe tener el GPS activado y funcionando.
6. La aplicación en cada dispositivo debe tener descargado el mapa que previamente fue generado con Mobac.

Una vez que se han cumplido con los requerimientos del entorno de producción se continúa con la evaluación de la aplicación tomando en cuenta los parámetros previamente definidos.

6.2. Pruebas de comunicación

En las trasmisiones de datos lo importante es garantizar que los datos de geolocalización sean entregados al último usuario existente en la red para que el mapa con los usuarios sea renderizado sin problemas en el menor tiempo posible, el tamaño de los datos no debe sobre pasar los 2MB para que se pueda enviar y recibir fácilmente.

Así mismo la aplicación internamente maneja un estado de conectividad con la red para poder actualizar los datos enviados y recibidos en cualquier momento y que el retardo de trasmisión sea el menor posible.

La información que será enviada a cada usuario consta de:

- Longitud (obligatorio)
- Latitud (obligatorio)

Los campos que son obligatorios no son modificables por el usuario.

6.2.1. Funciones principales.

- Envío y recepción de datos de geolocalización.
- Renderización de los datos en el mapa.

Los mapas se crean utilizando información geográfica capturada con dispositivos GPS móviles. Con esta cartografía, tanto las imágenes creadas como los datos vectoriales almacenados en su base de datos, se distribuye bajo licencia abierta Open Database Licence(ODbL).[25]

6.2.2. Registro del envío de una notificación.

Una vez que se envían los datos a otro usuario, la aplicación guarda un registro con las acciones realizadas por el sistema operativo tal y como se presenta en la Figura 22.

Figura 22. Log de resultados luego de envío de una notificación

En la trama de resultados obtenidos en la figura 22 como producto del envío de datos a los usuarios de la red se describen los siguientes campos:

- Level.- Define el tipo de mensaje que se va a presentar, éstos pueden ser:
- Log.e: Este tag se utiliza cuando algo ha salido mal. Este tag se utiliza dentro de bloques como try catch.
- Log.w: Se utiliza esta etiqueta cuando se sospecha que algo oculto al programador está sucediendo. Con esta etiqueta no se garantiza el log completo del error, pero puede ayudar a encontrar comportamientos inesperados, Básicamente, se utiliza esta tag para registrar cosas que no esperamos, pero que no necesariamente son un error.

[25] http://es.wikipedia.org/wiki/OpenStreetMap.

- Log.i: Se utiliza este tag para presentar información completa del log. Por ejemplo “La conexión al nodo destino ha sido exitosa”, Básicamente se utiliza para presentar mensajes de éxito de alguna acción.
- Log.d: Este tag se utiliza para propósitos de depuración. Se lo utiliza cuando se desea imprimir una serie de mensajes que le permita registrar el flujo exacto de su programa o mantener un registro de valores de las variables.
- Log.v: Se utiliza esta opción cuando se quiere ir totalmente desordenado con la ejecución de la aplicación. Se utiliza normalmente cuando se quiere ejecutar parte de código del contexto de la aplicación.
- Log.wtf: Se utiliza este tag cuando la ejecución del programa es incierto, es decir no se sabe cómo se va a comportar en un determinado momento. Utilizado para esos bloques catch que capturan excepciones no esperadas.
- Time.- Obtiene la fecha y hora en la cual sucedió el evento relacionado con el level. PID.- Id del proceso relacionado con el evento lanzado.
- Application.- Es el contexto del origen del evento, es decir, desde donde se están lanzado dicho mensaje.
- Tag.- Es un string que permite filtrar los mensajes de acuerdo a un tag que hayamos definido en la aplicación.
- Tex.- Es el mensaje a presentar al usuario dependiendo del tag y level.

6.2.3. Descripción de resultados.

En la Figura 22 se presenta el flujo que sigue la aplicación para el envío de los mensajes de geo-localización a los demás usuarios, en este caso los resultados se basan en un entorno de simulación utilizando dispositivos virtuales android.

Línea 1: En esa línea se especifica al método que llega para iniciar el proceso de envío de datos a los usuarios.

Línea 2: Se realiza la preparación de los datos que van a ser enviados.

Línea 3: Una vez que ya se tienen los datos correctos se inicia el proceso de envío de los datos.

Línea 4: Se hace un escaneo de todos los usuarios existente en la red y se presenta la lista a los que se les enviaran los datos.

Línea 5: Corresponde el mensaje que el usuario a ingresado, adjunto a este mensaje también se envía la longitud y latitud.

Línea 6 y 7: Es el proceso de envío de los datos al servidor, en caso de haber algún error se le informa al usuario dicho error.

Finalmente cuando un reporte es enviado a los usuarios de la red, éstos son notificados del mensaje el cual contiene los datos descritos anteriormente.

6.3. Funcionamiento en un dispositivo real

Para un correcto funcionamiento de la aplicación es de vital importancia que ésta se conecte satisfactoriamente con los servicios a la red wifi que previamente es necesario que se encuentre en funcionamiento. Para probar la integración del sistema se ha procedido a realizar las siguientes pruebas de interfaz.

6.3.1. Menú principal.

Al arrancar la aplicación, el usuario podrá elegir seleccionar entre los diferentes menús disponibles como se presenta en la Figura 23 los cuales se describen a continuación:

Principal.- Objetos que se van a visualizar en el mapa.
Emergencia.- Llamada de emergencia al 911.
Notificaciones.- Notificaciones que son enviadas y/o recibidas.
Seguimiento.- Ruta que ha seguido un usuario y la cual ha sido registrada.
Sincronización.- Opciones de sincronización con otro dispositivo
Preferencias.- Preferencias de usuario para la aplicación.

Figura 23. Menú principal

6.3.1.1. Objetos a visualizar.

Al arrancar la aplicación, el usuario podrá elegir que objetos desea visualizar en el mapa, las opciones existentes son las zonas de evacuación y los obstáculos como se puede apreciar en la Figura 24.

Los demás objetos como los usuarios y la ruta de salida se renderizarán de forma automática.

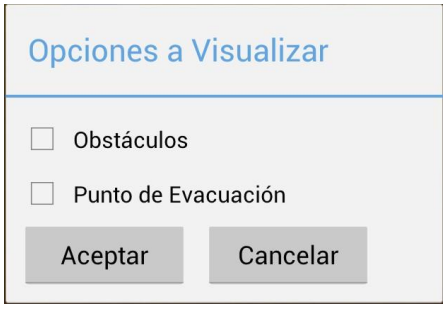

Figura 24. Objetos a visualizar en el mapa

6.3.1.2. Visualización de objetos en el mapa.

En la Figura 25 se pueden apreciar los obstáculos, la ubicación actual del usuario, la zona de evacuación más cercana y la ruta de salida que el usuario debe seguir para llegar a dicha zona de evacuación.

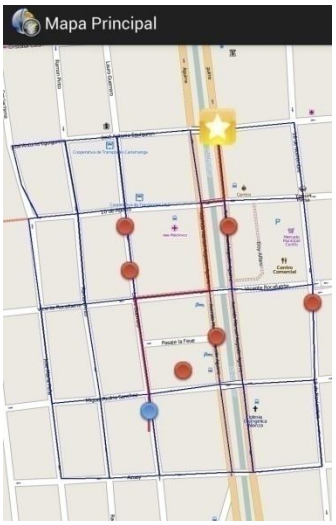

Figura 25. Objetos visualizados

6.3.2. Llamada de emergencia.

Desde el menú principal es posible realizar una llamada de emergencia al 911 como se muestran en la Figura 26, por defecto dicho número se encuentra predefinido para realizar la llamada en el momento que el usuario lo desee.

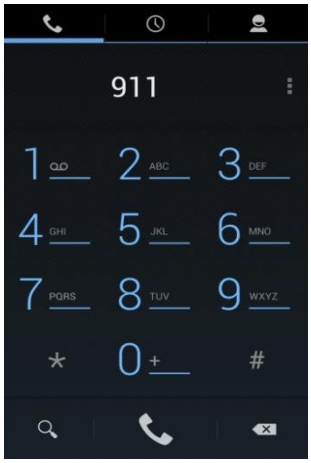

Figura 26. Llamada de emergencia

6.3.3. Notificaciones.

Las notificaciones se refieren al reporte de la existencia de un obstáculo en cualquier lugar dentro del área de cobertura establecida, siempre y cuando uno o varios usuarios hayan reportado la existencia de dicho obstáculo.

En la Figura 27 se puede apreciar todas las notificaciones que han sido enviadas al dispositivo que actúa como servidor por parte de los usuarios, cada notificación consta de un par de coordenadas de geolocalización las cuales se refieren a la localización del obstáculo.

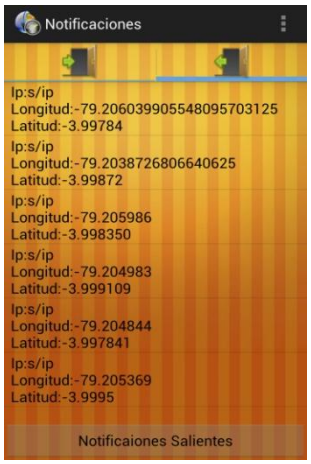

Figura 27. Notificaciones

6.3.4. Seguimiento de la ruta de un usuario.

En el menú de seguimiento se puede iniciar el servicio de trazado de una ruta que un usuario utilizó durante su trayecto en cualquier momento. Este servicio debe ser iniciado de forma manual por el usuario o detenido en cualquier momento, para lo cual cuenta con dos opciones para su funcionamiento como se puede apreciar en la Figura 28. Una vez que el

servicio haya sido iniciado la aplicación empieza a tomar la ubicación del usuario y la envía al nodo servidor cada que exista una nueva localización.

Figura 28. Seguimiento de la ruta de un usuario

6.3.5. Tipos de sincronización de información.

Existen diferentes tipos de sincronización, entre los cuales podemos mencionar a:

Mapa.- Sincroniza los datos del mapa de la aplicación.
Obstáculos.- Todos los obstáculos existentes.
Usuarios.- Todos los usuarios que existen en la red.
P. Evacuación.- sincroniza los puntos de evacuación existentes.

Cada una de las opciones de sincronización como se muestra en la Figura 29 se ejecuta de inmediato conforme el usuario decida hacerlo.

Figura 29. Sincronización de datos

6.3.6. Preferencias de usuario.

En la sección de preferencias de usuario es posible definir parámetros a utilizarse dentro de la aplicación como se puede apreciar en la Figura 30, los parámetros definidos son los siguientes:

Servidor: Es la dirección IP del nodo al cual la aplicación se sincronizará.
Puerto: Es el puerto por el cual se conecta al dispositivo servidor desde un cliente, por defecto es el 8080.
Enviar Ubicación: Es un valor booleano que define si se envía la ubicación o no del usuario al nodo servidor.

Figura 30. Preferencias de usuario

Entorno: En el entorno de funcionamiento de la aplicación es decir define si la aplicación va a funcionar en un entorno real o de pruebas.
Sincronización: Define si la sincronización se va a ejecutar de forma manual o de forma automática.

6.4. Escenarios de funcionamiento

Para la correcta realización, es necesario disponer de ciertos elementos:

1. Un dispositivo móvil Android con la aplicación instalada. Este dispositivo debe contar con una tarjeta de red inalámbrica y GPS activado para poder establecer una conexión con cualquier otro dispositivo. El dispositivo usado para estos experimentos ha sido el Galaxy Nexus One, que cumple con todos los requisitos mencionados.
2. Un entorno con una cierta infraestructura wifi disponible, donde poder descubrir redes inalámbricas sin dificultad.
3. Las pruebas han tenido lugar en dos escenarios distinto de la ciudad de Loja, el primer escenario es un área urbana del centro de la ciudad y el segundo escenario conforma el Campus Universitario donde se encuentran las instalaciones de la Universidad Técnica Particular de Loja.

4. Un dispositivo móvil que actúe como servidor para mantener la información de la red y los usuarios actualizada.

Uno de los principales objetivos en las pruebas es precisamente comprobar el funcionamiento del algoritmo Dijkstra, y la precisión alcanzada a la hora de trazar una ruta de salida desde la ubicación actual hasta el punto de evacuación más cercano.

Gracias al algoritmo de Dijkstra implementado, y al contrario que con otras aplicaciones, las rutas puede ser mucho más personalizadas y adaptadas a cualquier entorno donde exista una señal de GPS activa y cada dispositivo android cuente con dicha funcionalidad activada.

6.4.1. Escenario 1: Área urbana de Loja.

Para el primer escenario se ha seleccionado una área urbana del centro de Loja, donde las calles se encuentra mejor definidas que en una zona marginal o rural, en este escenario es donde mejor prestaciones en cuanto a visibilidad de las rutas se puede tener debido a su ubicación.

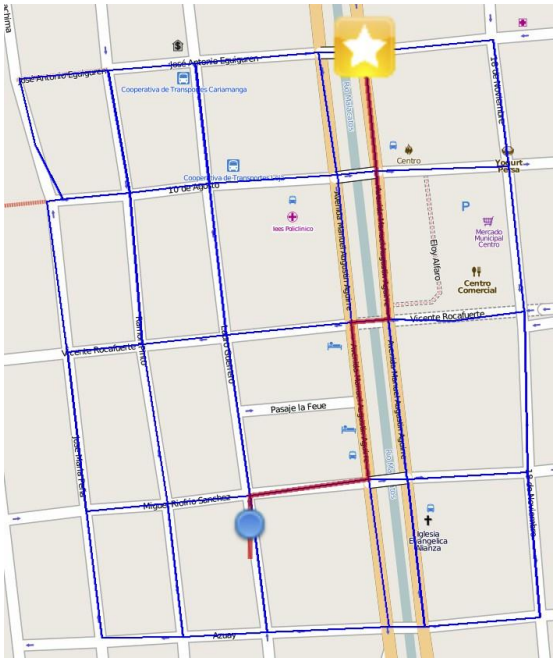

Figura 31. Ruta de evacuación sin obstáculos

En la Figura 31 se puede apreciar el trazado de la mejor ruta de salida en un escenario donde no existen obstáculos; donde, el círculo de color azul representa la ubicación actual, para lo cual el usuario debe guiarse por la línea de color rojo hasta llegar al punto de evacuación disponible.

Por el contrario si existen obstáculos como se aprecian en la Figura 32, el trazado de la ruta cambia en busca de otra ruta que le permita llegar al mismo punto de evacuación.

La ruta de evacuación que se presenta al usuario se puede determinar siempre que el dispositivo móvil se encuentre dentro del área de cobertura, el cual se encuentra definido por el borde azul, si se intenta obtener una ruta de salida estando fuera de dicha área se presenta un mensaje como se muestra en Figura 33, el cual nos indica que nos encontramos fuera del alcance de la aplicación.

En los escenarios donde no existe una área de cobertura se puede ampliar el rango de otra área para que cubra el espacio que se desee, considerando previamente que exista la disponibilidad de una red wifi en correcto funcionamiento.

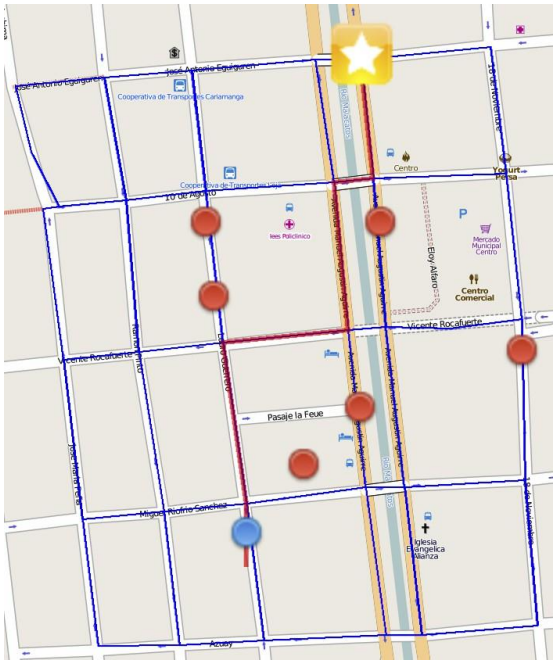

Figura 32. Ruta de evacuación con obstáculos

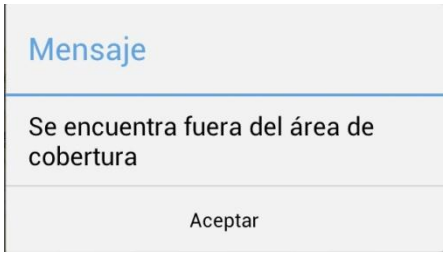

Figura 33. Mensaje fuera del área de cobertura

6.4.2. Escenario 2: Campus universitario de la UTPL.

En la Figura 34 se presenta el área del campus universitario de la UTPL, el cual se encuentra situado en el barrio San Cayetano Alto de la ciudad de Loja. En este escenario de pruebas, el objetivo principal es visualizar el trazado de las rutas de salida en caso de emergencia en entornos donde no existen las calles definidas correctamente, para lo cual se evaluará el comportamiento del algoritmo de búsqueda de rutas a través del número de iteraciones que debe realizar a fin de buscar la mejor ruta posible.

Figura 34. Área del campus de la UTPL

En la Figura 34 se puede apreciar que dentro del campus universitario no se encuentran correctamente definidas las calles, por lo cual se asume que los puntos de evacuación se encuentran dentro del campus.

Dentro del campus se han dispuesto un punto de evacuación en la parte norte de la universidad, mientras que en la parte sur se encuentra un usuario que desea llegar al punto de evacuación; como se muestra en Figura 35, se puede apreciar el trazado de la ruta más corta para llegar al lugar deseado, tomando en cuenta que no existen obstáculos presentes.

Figura 35. Ruta de evacuación dentro del campus utpl

Al igual que en el escenario anterior se debe considerar que el dispositivo móvil se encuentre dentro del área de cobertura, caso contrario se presenta el mensaje de la Figura 33 el cual nos indica que nos encontramos fuera del área de alcance.

La matriz de rutas en la Figura 35 está definida internamente y los puntos de dicha matriz son los vértices que forman al unirse dos segmentos de línea de color azul.

CONCLUSIONES

Luego de realizado el desarrollo de una versión inicial de la aplicación para dispositivos móviles android orientada a reducir los tiempos de evacuación por parte de organismos de socorro. Haciendo uso de una red wifi, dentro del presente proyecto se ha desarrollado la aplicación de activación de emergencia y rescate en la cual se destaca a continuación los siguientes puntos de interés:

1. Utilización de red wifi local para publicar un servicio de sincronización de información entre dispositivos móviles.
2. Desarrollo de una aplicación que puede ser utilizada por los usuarios de dispositivos móviles android para reducir los tiempos de evacuación.
3. Definición de un área de cobertura para el funcionamiento de la aplicación móvil.
4. Desarrollo de una alternativa para reducir los tiempos de respuesta para el personal de los organismos de socorro.
5. Generación de una red wifi independiente de una conexión de datos.
6. Utilización de los elementos de geolocalización de un dispositivo android para determinar la localización de un usuario.
7. Se ha conseguido implementar el desarrollo completo de una aplicación con herramientas Open Source.
8. El emulador que trae el sdk de android es una herramienta muy útil para el desarrollo de aplicaciones y sobre el cual se puede simular su funcionamiento sin necesidad de ningún dispositivo físico, sin embargo para aplicaciones como la que se ha desarrollado en este proyecto, su uso se limita a comprobar el aspecto de la interfaz, pues el uso tanto del GPS como de la tarjeta inalámbrica son complicados de emular.

Este proyecto ha permitido explorar otra característica funcional de los dispositivos móviles con android: su utilización como dispositivos de guía en situaciones de emergencia mediante el empleo de las redes wifi disponibles. Este hecho abre las puertas a multitud de nuevas posibilidades de aplicaciones para el usuario y la industria dispuesta al desarrollo de aplicaciones móviles en entornos pequeños.

RECOMENDACIONES

De acuerdo a las características con las cuales se ha desarrollado el proyecto se sugieren las siguientes recomendaciones:

- Se sugiere no generar una matriz de ruta en Mobac demasiado grande, esto es, que el archivo .gpx generado no exceda los 2mb, de lo contrario la aplicación necesitará más recursos del dispositivo móvil.
- Para generar la matriz de rutas en mobac se sugiere seleccionar como proveedor de fuente del mapa a OpenStreetMap debido a que tiene un mejor detalle de las vías.
- Utilizar estándares de programación para futuras mejoras a la aplicación.
- Firmar las aplicaciones antes de probarlas en un dispositivo físico real, sin esta firma no es posible instalarlas dentro del sistema operativo Android.

TRABAJOS FUTUROS

El presente proyecto sienta las bases para un posible futuro desarrollo, actuando como una posible fuente de ideas para proyectos similares como una primera versión de un sistema mejorado. En esta sección se proponen algunas de esas posibles líneas de trabajo:

1. Ampliar la zona de cobertura para permitir que más usuarios puedan conectarse a la red y utilizar la aplicación.
2. Acelerar la carga del mapa en modo offline y los datos a renderizarse.
3. Mejorar el envío de datos de un nodo hacia otro a través de la compresión de la información con el fin de reducir su tamaño y facilitar su transporte.
4. Mejorar la disponibilidad de la información sobre el estado de una ruta mediante la implementación de una cola de mensajes capaz de almacenar toda información necesaria y disponible para los nuevos usuarios de la red.

BIBLIOGRAFÍA

- ADT. http://developer.android.com/sdk/index.html. Noviembre 2013
- android. Android OS. http://es.wikipedia.org/wiki/Android.
- Dijkstra. http://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra
- emergencyapp. Emergency Alert. https://play.google.com/store/apps/details?id=com.xankle.mobile.sos&hl=es_419
- firealert. Fire Alert. http://www.b4exit.de.
- FUMERO, Antonio; WERTERSKI, Adam; RODRÍGUEZ, Pedro; BLANCO, Paco; CAMARERO, Julio. Metodología de desarrollo ágil para sistemas móviles. Introducción al desarrollo con Android y el iPhone.
- gpx. Antena receptora de GPS- Manual de usuario. http://www.reparacionestopograficas.com/~manuales/Manual3-GPS.pdf.
- gpx. GPS Exchange Format - GPX. http://es.wikipedia.org/wiki/GPX.
- greedy.Técnica Greedy (Algoritmo voraces). http://apuntesinformaticauade.blogspot.com/2013/02/capitulo-16-algoritmos-greedy-parte-i.html.
- Hashtable.Data Structures/Hash Tables. http://en.wikibooks.org/wiki/Data_Structures/Hash_Tables.
- haversine. Formula del Haversine. http://es.wikipedia.org/wiki/Formula_del_Haversine. Noviembre 2013
- idc. International Data Corporation. http://www.idc.com.
- iRescue. iRescue Application for iPhone. http://www.rescue-app.com.
- GARCIA, Ja; MACÍAS Evelio, 2006. Redes Wi-Fi en malla (wi-fi mesh networks).
- JEROEN HOEBEKE, Bart Dhoedt; MOERMAN, Ingrid and DEMEESTER, Piet.. An overview of mobile ad hoc networks:applications and challenges. In An Overview of Mobile Ad Hoc Net-works:Applications and Challenges, .
- LOU, Wei and WU, Jie. A cluster-based backbone infrastructure for broadcasting in manets. In A Cluster-Based Backbone Infrastructure for Broadcasting in MANETs. Department of Computer Science and Engineering Florida Atlantic University, march 2003.
- PÉREZ BLANCO, Carlos. Descubrimiento y geolocalización mediante redes wi-fi. In Universidad Carlos III de Madrid, volume 1, pages 216 -220, july 2010.
- research2guidance. The mobile research spacialist. http://www.research2guidance.com/.

- SONG, Wen; WANG, Fei; DAI, Jianbo. Mobile ad hoc networking (manet). In Routing Protocol Performance Issues and Evaluation, january. 1999.
- SONG, Wen; WANG, Fei; DAI, Jianbo. A emergency communication system based on wmn in underground mine. In Computer Application and System Modeling (ICCASM), 2010 International Conference on, volume 4, pages V4-624 -V4-627, oct. 2010. doi: 10.1109/ICCASM.2010.5620768.
- Source: IDC Worldwide Quarterly Tablet Tracker, December 2013
- VisualAge. VisualAge Generator. http://http://www-01.ibm.com/software/awdtools/visgen/.
- Y. Liang B. Brownlee. Mobile Ad Hoc Networks An Evaluation of Smartphone Technologies.

ANEXOS

Anexo 1: Ejemplos de Código de Componentes Android

De manera que se facilitase la lectura de la memoria, se ha decidido excluir los ejemplos de código de los componentes principales de Android del cuerpo principal de ésta. Por este motivo, se incluye a continuación una recopilación que ayude al lector a comprender el modo de funcionamiento de dichos componentes.

Servicios

Se crean y destruyen mediante un intent:

```
/** Se crea un intent que indica la clase que
implementa el servicio y se inicia mediante el
metodo  startService*/
Intent svc = new Intent(this, MyService.class);
startService(svc);


/** Para detener el servicio, se procede del mismo
modo, solo que invocando el metodo  stopService*/
Intent svc = new Intent(this, MyService.class);
stopService(svc);
```

Un servicio implementa una clase que lo gestiona y que extiende a la clase Service para que pueda ejecutarse correctamente. Habrá una serie de hilos ejecutando en segundo plano:

```
/** Se crea un intent que indica la clase que
implementa el
servicio y se inicia mediante el metodo
startService */

public class MyService  extends Service {
/** Metodos  similares a los de cualquier  Activity
*/
@Override
```

```
public void onCreate() {
super.onCreate();
/** Metodo que realiza toda la actividad */
startService();
}

@Override
public void onDestroy() {
   super.onDestroy();
   shutdownService();
}
private void _startService() {
/** Una practica  habitual es la de crear un
temporizador que lance un hilo que realice una tarea
    */
timer.scheduleAtFixedRate(new TimerTask() {
public void run() {
   Thread cThd = new Thread(new claseTarea());
   cThd.start();
}
}, 0, UPDATE_INTERVAL);

private void _shutdownService() {
   /** Se destruye el temporizador*/
   if (timer != null)
   timer.cancel();
}
}
```

Proveedores de contenidos

Por lo general, el sistema operativo Android fuerza a que los datos de una aplicación estén fuertemente ligados a la misma, es decir, se asegura de que una aplicación no pueda

leer o escribir datos entre paquetes. Sin embargo, existirán aplicaciones que quieran compartir datos y permitir que otras aplicaciones externas puedan leer y escribir datos en su base de datos. Un claro ejemplo son los datos de nuestros contactos. Si Android no proveyese una manera de compartir esta información entre aplicaciones, se forzaría al usuario a mantener una base de datos diferente para cada aplicación específica, y no habría ningún tipo de vínculo entre unos datos y otros. Para permitir esto, existe la API de proveedor de contenidos, a través de la cual se permite a cada aplicación cliente a que solicite la información que necesite al propio sistema operativo, utilizando un mecanismo de identificación uniforme de recursos (de las siglas en inglés URI).

La API de proveedor de contenidos dotará a la aplicación acceso total al contenido, por lo que ésta podrá:

1. Crear nuevos registros
2. Recuperar uno, todos, o un conjunto limitado de registros Actualizar registros
3. Borrar registros, siempre y cuando esto esté permitido

La interfaz común que implementan todos los proveedores de contenidos se emplea a través de objetos ContentResolver. Estos objetos se obtienen invocando al método getContentResolver() desde la implementación de una actividad o cualquier otro componente:

```
ContentResolver cr = getContentResolver();
```

Intents

Para pasar de una actividad a otra, Android dispone de la clase Intent. Existen dos invocaciones distintas para pasar de una actividad a otra. La primera es el paso a la espera de una confirmación/respuesta por parte de la actividad invocada:

```
/* Se crea un Intent  indicando la clase a ejecutar*/
Intent i = new Intent(this, Login.class);
/* Se invoca el metodo que inicia la nueva actividad */
this.startActivityForResult(i, codigoRespuesta);

/* En la actividad destino, una vez finalizada la
tarea y  antes de indicar el regreso a la actividad
origen, se indica el resultado de la operacion */
```

```
setResult(RESULT_OK);
finish();

/* De vuelta la actividad origen, se procede a evaluar
el resultado de la operacion*/
@Override
protected void onActivityResult(int requestCode, int
resultCode, Intent data){
}
```

La segunda, es la invocación a la nueva actividad sin esperar ningún resultado:

```
/* Se crea un Intent  indicando la clase a ejecutar*/
Intent i = new Intent(this, Login.class);
/* Se invoca el metodo que inicia la nueva actividad*/
this.startActivity(i);
finish();
```

Anexo 2: Archivo de configuración AndroidManifest.xml

Como ya se ha mencionado anteriormente, el fichero AndroidManifest es el fichero de control que indica al sistema central lo que debe hacer con todos los componentes que conforman la aplicación. Es especialmente útil para indicarle qué tipo de datos puede manejar o de qué permisos dispone la aplicación en cuanto a ejecución (si tiene acceso a la red, al dispositivo GPS, tarjeta de memoria, etc.). Su inclusión en el directorio raíz del proyecto es obligatoria, ya que sin este fichero, no se podrá compilar ni ejecutar.

A continuación se incluye el contenido del archivo de configuración *AndroidManifest.xml*

Listing 9.1: AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.urban.trackerclient"
    android:versionCode="1"
    android:versionName="1.0" >

     <uses-permission
android:name="android.permission.ACCESS_WIFI_STATE" />
         <uses-permission
android:name="android.permission.CHANGE_WIFI_STATE" />
```

```
          <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />
          <uses-permission android:name="android.permission.INTERNET"
/>
          <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
          <uses-permission
android:name="android.permission.CALL_PHONE" />
          <uses-permission
android:name="android.permission.CALL_PRIVILEGED" />

    <uses-sdk
        android:minSdkVersion="10"
        android:targetSdkVersion="15" />

    <application
        android:icon="@drawable/ic_launcher2"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".HomeActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category
android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MainActivity"
            android:label="Mapa Principal" >
        </activity>
        <activity
            android:name=".SocketActivity"
            android:label="Envio Comunicado" >
        </activity>
        <activity
            android:name=".NotificationsActivity"
            android:label="Notificaciones" >
        </activity>
        <activity

android:name="com.urban.trackerclient.application.PreferencesActivit
y"
            android:label="Configuración" >
        </activity>
        <activity
            android:name=".application.SincronizacionManualServer"
            android:label="Sincronización Manual" >
        </activity>
        <activity

android:name=".application.SincronizacionAutomaticaActivity"
            android:label="Sincronización Automática" >
```

```
        </activity>
        <activity
            android:name=".TrackerActivity"
            android:label="Servicio Ubicación" >
        </activity>
        <activity
            android:name=".NotificacionesEntrantes"
            android:label="Notificaciones Entrantes" >
        </activity>
        <activity
            android:name=".NotificacionesSalientes"
            android:label="Notificaciones Salientes" >
        </activity>
        <service
android:name="com.urban.tracker.service.GpsTrackerLogger"

      android:permission="android.permission.ACCESS_FINE_LOCATION"
                android:label="@string/app_name"
android:exported="true"
                android:enabled="true"/>
        <service
android:name="com.urban.tracker.service.SincronizacionService"
                android:label="@string/app_name"
                android:enabled="true"/>

    </application>

   </manifest>
```

Como se puede apreciar, el manifiesto incluye, en primer lugar, una serie de permisos de los que debe disponer la aplicación para que se ejecute normalmente. Esta lista de permisos, y recursos que requiere la aplicación, es la que se le presenta al usuario final antes de instalar la aplicación para que dé su consentimiento. Toda funcionalidad protegida, a la que se quiera tener acceso desde el sistema desarrollado, deberá incluirse en el manifiesto mediante el tag

<uses-permission>

Tras establecer los permisos, se definen todas las actividades disponibles en el paquete de la aplicación desarrollada. Todas deben estar bajo el tag <application> y se declaran mediante el tag <activity>.

Si una actividad no se ha declarado en este momento, la aplicación no podrá lanzarla y el sistema devolverá un error de acceso a una actividad no existente. Cualquier paquete que se presente como aplicación deberá incluir una actividad que soporte la acción main y la categoría launcher, ya que esta actividad es la que se lanzará al arrancar el sistema:

```
<intent-filter>
<action android:name="android.intent.action.MAIN"/>
<category  android:name="android.intent.category.LAUNCHER"/>
</intent -filter >
```

Anexo 3: DDMS (Dalvik Debug Monitor Service)

Para facilitar la tarea de depuración del desarrollador, Android se distribuye con una herramienta de depuración llamada Dalvik Debug Monitor Service (DDMS). Esta herramienta proporciona, entre otros, los siguientes servicios:

1. Redirección de puertos.
2. Captura de pantalla del dispositivo.
3. Información sobre la pila y los hilos de ejecución del dispositivo Información sobre el estado de los procesos.
4. Simulación de llamadas y envíos de mensajes de texto.
5. Simulación de recepción datos de localización geográfica

Ubicación y modo de uso

DDMS se encuentra en el directorio de herramientas (”tools/”) de SDK. Esta herramienta funcionará tanto con el emulador como con un dispositivo real conectado. Si ambos están conectados y ejecutándose simultáneamente, DDMS cogerá el emulador por defecto. DDMS actúa como intermediario para conectar el entorno de desarrollo a las aplicaciones que se están ejecutando en el dispositivo/emulador. En Android, cada aplicación ejecuta en un proceso propio, cada uno de los cuales hospeda su propia máquina virtual, por lo que cada proceso escucha a un depurador en un puerto distinto. Cuando arranca, DDMS se conecta al adb (de las siglas en inglés, “Android Debug Bridge”) y ejecuta un servicio de monitorización de dispositivos entre ambos, que notificará al DDMS en el momento en que se conecte o desconecte un dispositivo. Al conectar un dispositivo, se crea un servicio de monitorización de máquina virtual entre el adb y el DDMS, que notificará a este último cuando una máquina virtual se arranca o para en el emulador. Una vez que hay una máquina virtual en ejecución, el DDMS obtiene el identificador del proceso de la máquina virtual, vía el adb, y abre una conexión al depurador de dicha máquina virtual. De esta forma queda establecido un canal de comunicación entre el DDMS y la máquina virtual como se muestra en la Figura 36.

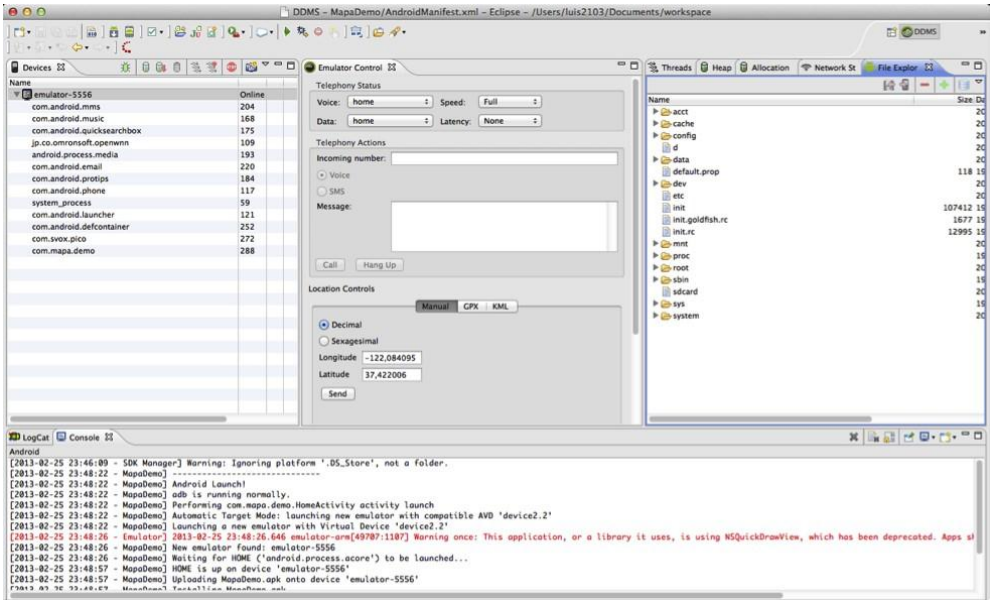

Figura 36. Dalvik Debug Monitor Server

Panel Izquierdo

El panel izquierdo del monitor de depuración muestra cada uno de los emuladores/dispositivos activos, con una lista de todas las máquinas virtudes ejecutándose en cada uno. Éstas se identifican por el nombre del paquete de la aplicación que alberga.

Panel Derecho

En el lado derecho, el monitor de depuración proporciona una serie de pestañas que muestran información general del emulador seleccionado. Esta pestaña permite ver el sistema de ficheros del dispositivo que estamos depurando así como realizar gestiones básicas sobre el mismo, como, por ejemplo, eliminar e insertar ficheros. Asimismo, si se ha montado una imagen de tarjeta SD en el emulador, se podrá acceder a los archivos almacenados en ella desde este mismo panel.

Panel Central

Finalmente, en el centro del monitor se ofrecen al desarrollador una serie de herramientas que permiten emular ciertas funciones:

1. Llamadas
2. Envío de SMS
3. Controles de localización GPS

El que se ha empleado para el desarrollo de este proyecto ha sido el de localización, puesto que permite emular la recepción de coordenadas de posicionamiento GPS en el dispositivo. De

esta forma ha sido posible probar y depurar las funcionalidades sobre el mapa de OpenstreetMap.

Anexo 4: Simulador GPS Android

Utilizando el comando “geo” del simulador de Android se puede interactuar y probar el funcionamiento del GPS.

Para enviar un par de coordenadas a una aplicación en android es necesario hacer uso de aplicaciones que faciliten el envío de datos hacia el emulador.
Para enviar un par de coordenadas de geolocalización se debe seguir el siguiente formato:
geo fix longitud latitud

Tanto la longitud como la latitud se deben enviar en formato decimal.

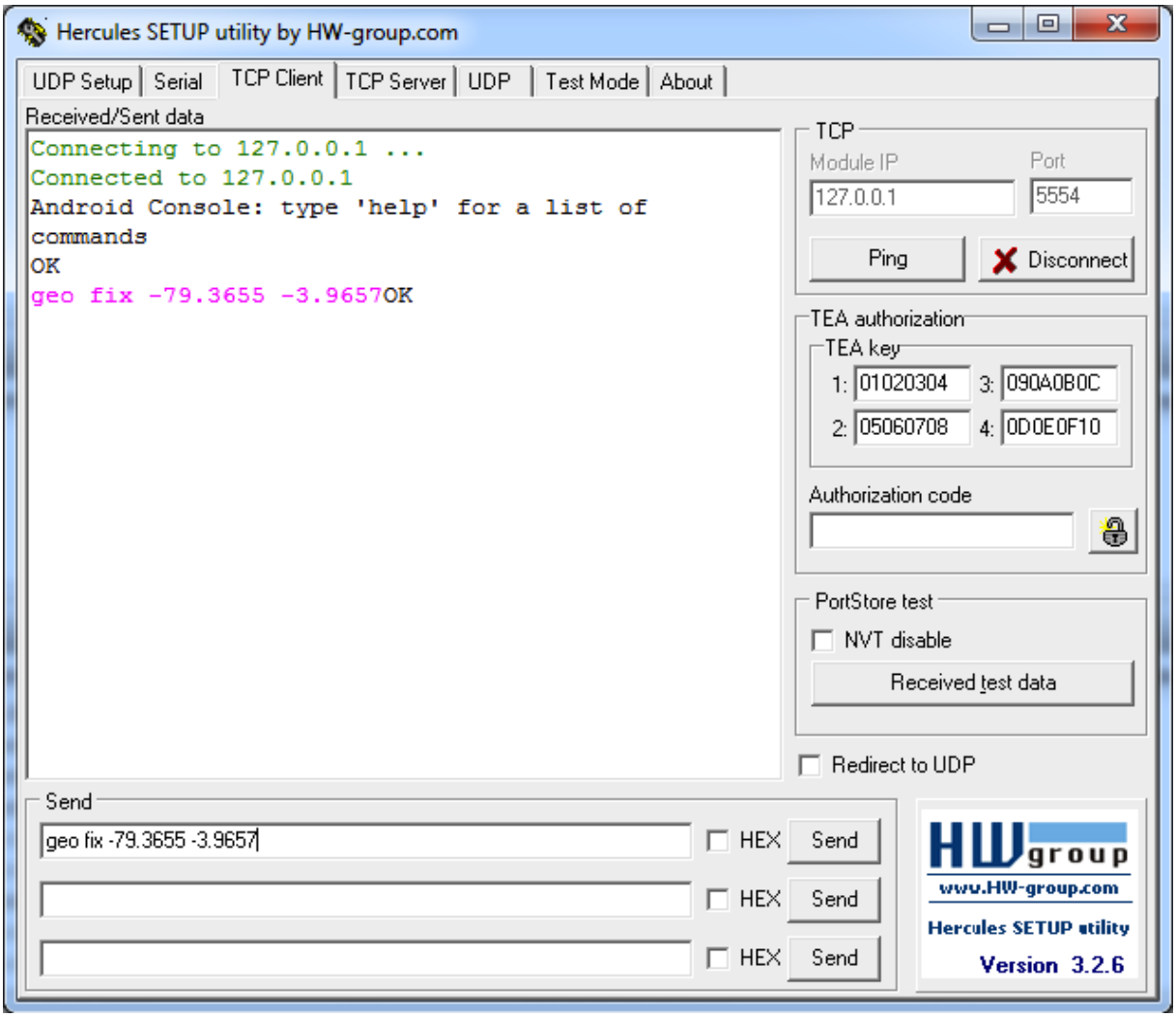

Figura 37. Envío de una posición de geolocalización

Para esta prueba se ha usado el programa Hércules en su versión 3.2.6 como se muestra en Figura 37 el cual sirve para realizar conexiones TCP cliente y servidor, UDP, Serial, etc. Se encuentra disponible en:

http://www.hw-group.com/products/hercules/index_en.html

Para conectarse a un servicio vía TCP es necesario ingresar el Module IP el cual es la dirección ip del equipo al cual se va a conectar, también se debe ingresar el puerto.

En la Figura 37 se ha establecido el puerto 5554 ya que es el puerto por defecto que se crea al momento de crear un nuevo dispositivo, en caso de existir más dispositivos, los puertos varían de acuerdo a la disponibilidad de los mismos.