Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Aplicación Android para apoyar la estrategia de respuesta ante incidentes de peligrosidad en Cuba.

Autor: Elvis Salabarria Aquino

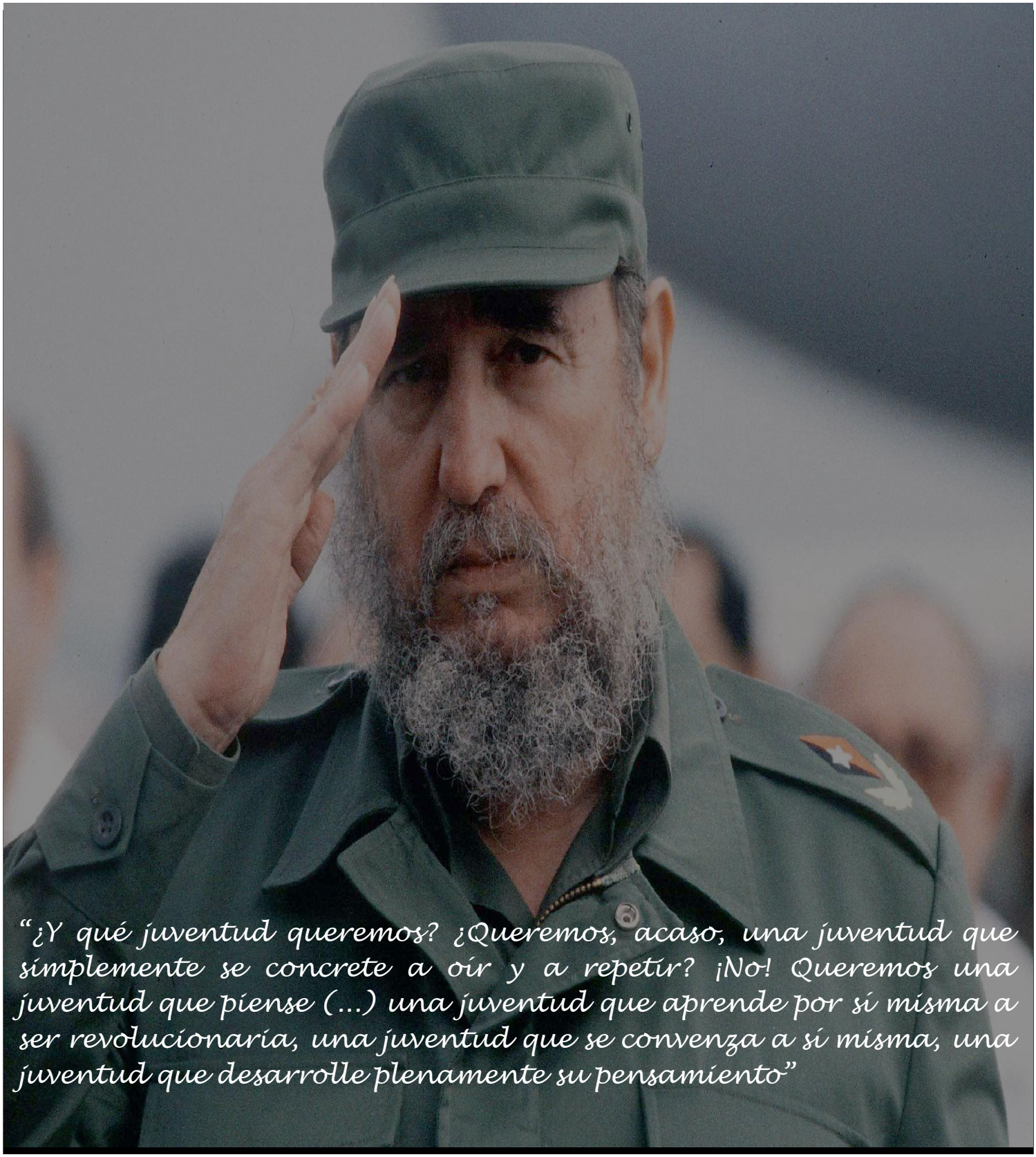
Tutores: Ing. Yaiselis Ramírez Mastrapa

Ing. Leonardo Eloy Saavedra Remón

Ing. Noichel Juan Hernández

La Habana, junio 2017

"Año 59 de la Revolución"


“¿Y qué juventud queremos? ¿Queremos, acaso, una juventud que simplemente se concrete a oír y a repetir? ¡No! Queremos una juventud que piense (…) una juventud que aprende por sí misma a ser revolucionaria, una juventud que se convenza a sí misma, una juventud que desarrolle plenamente su pensamiento”

Declaración de autoría

Declaro por este medio que yo **Elvis Salabarria Aquino**, con carné de identidad **93071111344** soy el autor principal del trabajo titulado **"Aplicación Android para apoyar la estrategia de respuesta ante incidentes de peligrosidad en Cuba"** y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

_____

Firma del autor

Elvis Salabarria Aquino

_____ _____ _____

Firma del tutor
Yaiselis Ramírez Mastrapa

Firma del tutor
Noichel Juan Hernández

Firma del tutor
Leonardo Eloy Saavedra Remón

Datos del autor:

Nombre y apellidos: Elvis Salabarria Aquino

Correo electrónico: esalabarria@estudiantes.uci.cu

Situación laboral: Estudiante.

Institución a la que pertenece: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba, Código Postal 19370.

Datos de los Tutores

Nombre y apellidos: Yaiselis Ramirez Mastrapa

Correo electrónico: yaiselis@uci.cu

Situación laboral: Especialista B

Años de graduado: 3 años.

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Institución a la que pertenece: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba, Código Postal 19370.

Nombre y apellidos: Leonardo Eloy Saavedra Remón

Correo electrónico: lesavedra@uci.cu

Situación laboral: RGA

Años de graduado: 3 años.

Especialidad de graduación: Ingeniero en Ciencias Informáticas.

Institución a la que pertenece: Universidad de las Ciencias Informáticas.

Dirección: Carretera a San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba, Código Postal 19370.

Nombre y apellidos: Noichel Juan Hernández
Correo electrónico: njuan@uci.cu
Situación laboral: Profesor
Años de graduado: 8 años.
Especialidad de graduación: Ingeniero en Ciencias Informáticas.
Institución a la que pertenece: Universidad de las Ciencias Informáticas.
Dirección: Carretera a San Antonio de los Baños, Torrens, Municipio Boyeros, Ciudad de La Habana, Cuba, Código Postal 19370.

Dedicatoria

A mis madre Deysi y a mi padre Feliberto quienes han sido los principales protagonistas en mi vida, a quienes debo todo lo que soy, por su amor, dedicación,confianza y enseñarme a construir mi propio futuro, también quiero dedicarla a mi hermano Erick que ha sido no solo un hermano sino un padre, sus consejos y regaños han hecho en mi una mejor persona profesional.

Agradecimientos

A mi mamá y a mi papá, por ser las persona más especiales de mi vida, por su esmero, comprensión y apoyarme en todas mis decisiones, por tener esas fuerzas para luchar cuando incluso la vida nos pone en difíciles momentos, hoy este logro es gracias a ustedes y me siento orgulloso de tener los mejores padres del mundo, los amo con la vida.

Mi hermano Erick que hubiese sido de mí sin tu apoyo, tus regaños, tus alones de oreja cuando estaba haciendo algo mal, gracias por guiarme por un buen camino, por siempre estar presente cuando necesita la mano de alguien en el que realmente pudiese confiar, has sido y siempre lo serás un ejemplo, no tengo palabras para agradecerte todo el esfuerzo que has hecho para que hoy sea quién soy, más que un hermano puedo sentirme orgullozo de sentirte como mi padre.

A mí abuelita Oria que no pudo verme graduado, donde quiera que estés este momento también es para ti.

Quiero agradecer a Barbarita por haberme acojido como un hijo más en su casa, por todo el apoyo incondicional que me has brindado en los momentos más difíciles de mi carrera, sabía que siempre podía contar contigo.

Oscarito no tengo forma de agradecer todo lo que has hecho por mí, incluso cuando no podías ahí estuviste cuando te necesité y por ser como un hermano para mi en todo momento.

A mi familia de alamar mi tía Bertha, mi tío Ricardo, mis primas Beatriz y Patricia, a mi mulatísima de Ovni como cariñosamente le digo en realidad no tengo como agradecer todo lo que han hecho por mi, los quiero de corazón.

El jefe, Yoa, Elena, Redito, Jennyfer, mi familia de luyanó el dia llegó y no los dejaré pasar por alto, gracias por tanta precupación, gracias por estar siempre ahí cuando más los necesité, por darme tanto cariño y por todo lo que me han ayudado a lo largo de mi carrera.

A mi novia Claudia por esas mala noches que pasamos juntos, por su apoyo,por toda tu paciencia y por demostrarme que siempre se puede, gracias mi princesa.

Daynelis gracias por ser la hermana que no tuve, por tantas cosas lindas que pasamos y por toda la ayuda que me has dado, te quiero mucho.

A mi grupo 1503 gracias por todo el tiempo que convivimos juntos todos los malos y buenos momentos que pasamos, a mi piquete de los 5 fantásticos El Humbe, Javier, Yunier, Yosbel, espero que siempre tengamos presentes los buenos momentos que compartimos e incluso los malos también, Asney a pesar de que cuando entraste al grupo eras la nota discordante te he cogido mucho aprecio y pude comprobar que tenia un mal concepto de tí, gracias por tu apoyo cuando mas lo necesitaba, gracias de corazón hermano.

Mi cajita fuerte no pienses que te quedas fuera tu sabes que eres especial para mí, ya somos ingenieron y aún no se de que municipio eres, espero algún día saberlo con toda sinceridad.

Edel y Lexys gracias por todos sus consejo y por toda esa ayuda que me brindaron en los momentos en que la necesitaba, gracias de todo corazón.

Mi piquete del voly mi gente ustedes saben gracias por todo los momentos buenos que tuvimos y por los pelotazos que me hicieron recapacitar y activar las neuronas jajaja.

Andy gracias por todas las sugerencias que me hiciste, por toda la ayuda que me brindaste sin importar que tan ocupado estuvieras, gracias mi hermano.

Spaek no tengo como agradecer todas las cosas que hiciste por mí, de verdad nunca pensé tener el grado de amistad que logré alcanzar contigo, un millón de gracias mi hermano.

Elementrix la familia que encontré a penas llegue a la UCI, gracias por todos esos momentos buenos que pasamos juntos y espero que continuen.

A mis tutores, gracias por sus regaños, por sus comentarios y por todo la ayuda que me brindaron.

A todos los profes de la facultad que de una forma u otra influyeron en mi preparación profesional.

De una forma u otra a todas aquellas personas que a lo largo de mi vida como estudiantes aportaron un granito de arena para que estuviese aquí hoy. Gracias a todos.

Resumen

Las nuevas tecnologías de la información y las comunicaciones han incidido cada vez más en el desarrollo de aplicaciones para teléfonos inteligentes con el fin de preservar la seguridad ciudadana. De ahí que el objetivo del presente trabajo sea desarrollar una aplicación móvil con Sistema Operativo (SO) Android que sea capaz de aumentar respuesta ante incidentes de peligrosidad en Cuba. El desarrollo de la aplicación se basó en el uso de la Metodología de Desarrollo para Aplicaciones Móviles (MDAM) la misma contiene cinco fases importantes: análisis, diseño, desarrollo, pruebas de funcionamiento y entrega. Finalmente se llevó a cabo la validación de la propuesta a través de los métodos definidos en la investigación.

Palabras clave: aplicación android, emergencias, incidentes, Sistema Operativo Android.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Cada estado tiene la responsabilidad de la seguridad de sus habitantes. “Todo individuo tiene derecho a la vida, a la libertad y a la seguridad de su persona”. “Derecho a la protección por parte del estado, a través de los órganos de seguridad ciudadana regulados por ley, frente a situaciones que constituyan amenazas, vulnerabilidad o riesgos para la integridad física de las personas, sus propiedades, el disfrute de sus derechos y el cumplimiento de sus deberes”(1).

En la última década uno de los temas centrales de inquietud social ha sido la falta de seguridad ciudadana y por lo tanto se ha convertido en una de las principales razones de trabajo de los dirigentes estatales a nivel mundial. En Cuba este tema cobra gran valor y es prioridad del estado, convirtiéndose en uno de los cinco países de América Latina con menos índices de peligrosidad ciudadana, atendiendo a variables como: homicidio, robo, incendios, violaciones, accidentes de Tránsito, derrumbes, estados de pánicos ante problemas de salud, infartos cardíacos y ataques de epilepsia (1).

A pesar de Cuba ser uno de los países que mejor garantiza la seguridad nacional, en el pasado 2016 según la Oficina Nacional de Estadísticas e Información (ONEI) existieron: 430 casos de homicidios, 820 robos con fuerza, 9345 incendios, 682 accidentes de tránsito, 227 derrumbes, 23 626 infartos cardíacos, 29 821 ataques epilépticos trayendo consigo aproximadamente 2034 pérdidas humanas (1). Según Oscar Medero especialista en estadística de la ONEI, el 40 % de estas pérdidas se podían haber evitado con una respuesta rápida de los organismos encargados (1).

Para una mejor protección de la población ante hechos delictivos, situaciones de emergencia y desastres naturales y para dar respuesta eficiente a cualquier irregularidad en Cuba, existe como órgano rector el Ministerio del Interior (MININT) y para garantizar la protección de la población ante problemas de salud se encuentra El Ministerio de Salud Pública (MINSAP) (2).

Dentro del MININT los organismos encargados de minimizar las variables antes expuestas son: Policía Nacional Revolucionaria (PNR), encargada de velar por el orden y la seguridad de los ciudadanos y el Cuerpo de Bomberos de Cuba (CBC) que se encarga de la protección contra incendios y posee una brigada de rescate y salvamento. Dicha brigada entra en acción cuando se producen inundaciones, derrumbes, fuegos de alta magnitud o cualquier otra situación que ponga en peligro la vida humana.

Otra de las instituciones encargadas de dar respuesta ante incidentes de peligrosidad es el MINSAP, la misma cuenta con el servicio de Urgencia Médica que posee 921 ambulancias destinadas para todo el país,

con prioridades que garantizan el traslado gratuito de pacientes hasta los lugares de difícil acceso (3).

En caso de existir alguno de estos incidentes de peligrosidad las personas tienen dos vías para hacer la solicitud del servicio de socorro a los ministerios encargados: la primera es dirigiéndose hacia las instituciones más cercanas en su municipio o zona y la segunda y más factible en estos casos es mediante una llamada telefónica.

El individuo debe llamar a las entidades correspondientes por un número de teléfono determinado, hacer solicitud del servicio y dar a conocer datos específicos del hecho ocurrido, además de una serie de información que le pide la receptora. Existen dos vías para realizar el procedimiento de reporte ante una situación de emergencia, el primero es el proceso de recepción de la información, el cual dura aproximadamente entre 5 y 8 minutos (4) con un tiempo de respuesta que está dado por la magnitud del hecho, por la disponibilidad de recursos, localización, prioridad del hecho y el retraso, la segunda es directamente en cada estación, el individuo es atendido por un oficial de guardia que recoge los datos personales (dígase nombre, apellidos, carné de identidad, dirección particular y descripción del hecho), una vez recogidos estos datos se procede a la investigación por parte de los peritos[1] que son asignados al caso, sin embargo, ante una situación de emergencia se hace muy difícil para la persona situarse geográficamente y encontrar una zona de auxilio médico o de cualquier otro tipo (5).

Sin embargo, en Cuba existe un 80 % de probabilidad, según la Dra. Ángela Olga Hidalgo, que las personas antes de llamar a las entidades correspondientes se auxilien entre distintas comunidades como son: vecinos, amistades, familiares o incluso desconocidos que brindan ayuda. Esto está dado por el nivel de solidaridad que existe entre la población cubana (6).

A pesar de que exista una respuesta casi inmediata ante las distintas dificultades, existen ciudadanos que se encuentran en circunstancias en las que el nivel de respuesta debe ser urgente y se les dificulta la acción de realizar una llamada telefónica o pedir socorro. Un ejemplo de esto es cuando ocurre un robo con fuerza. Para este hecho delictivo es riesgoso pedir auxilio a una estación de PNR o a una comunidad. Denunciar el delito en ese mismo instante podría traer consigo la muerte del individuo. Otra situación a analizar es cuando

[1] Es un profesional dotado de conocimientos especializados y reconocidos, a través de sus estudios superiores, que suministra información u opinión fundada a los tribunales de justicia sobre los puntos litigiosos que son materia de su dictamen.

una persona se encuentra ante los síntomas de un infarto cardíaco o epilepsia, el individuo no sabe a qué hospital puede dirigirse y hablar para pedir ayuda le es muy peligroso y el mismo no tiene una guía para dirigirse.

Mientras el gobierno cubano se enfrasca en aumentar la seguridad ciudadana, las Tecnología de la Informática y las Comunicaciones (TIC)[2] avanzan a toda velocidad en el mundo. Esto trae consigo el uso acelerado de los dispositivos móviles, convirtiéndolos en herramientas primordiales en la vida de un ser humano. Existen actualmente más de 8 000 millones de celulares en todo el planeta y se prevé que para finales del 2017 haya más celulares que personas en el mundo (7), en el trabajo, en la escuela, en la casa, donde quiera que se esté, se hace necesario tener acceso a estos dispositivos. Con el paso del tiempo y de los avances de la ciencia, se ha logrado convertir estos móviles en pequeñas computadoras personales que permiten recordar fechas y citas importantes, escuchar música, controlar el ritmo cardíaco o guiar a través de un mapa hacia una dirección determinada.

La cifra de dispositivos activados diariamente, que usan el Sistema Operativo (SO Android), asciende a más 1.5 millones. Una de las razones que potencian su popularidad son su naturaleza de código abierto (*open source*) y que posee una curvatura de aprendizaje relativamente baja (7). Actualmente en Cuba se fomenta el trabajo con este tipo de dispositivo, debido a que es interés del país fomentar el trabajo con sistemas libres.

La seguridad ciudadana no se ha quedado atrás en la explotación de esta tecnología y ha fomentado el desarrollo de una importante cantidad de aplicaciones que se ejecutan desde dispositivos móviles con Sistema Operativo (SO) Android. Con el fin de elevar el nivel de seguridad de los ciudadanos.

Para dar solución a la problemática antes expuesta se propone como **problema de la investigación:**

¿Cómo apoyar a la estrategia de respuesta ante incidentes de peligrosidad en Cuba, mediante el uso de teléfonos móviles con SO Android?

Teniendo como **objeto de estudio:** Estrategia de respuesta ante Incidentes de peligrosidad en Cuba.

[2] El término tecnologías de la información y la comunicación (TIC) tiene dos concepciones: por un lado, a menudo se usa tecnologías de la información para referirse a cualquier forma de hacer cómputo. Por el otro, como nombre de un programa de licenciatura, se refiere a la preparación que tienen estudiantes para satisfacer las necesidades de tecnologías en cómputo y organización.

Centrándose en el **campo de acción** Aplicaciones móviles para SO Android ante incidentes de peligrosidad.

Para dar solución al problema a resolver se plantea como **objetivo general:** Desarrollar una aplicación móvil para SO Android que apoye la estrategia de respuesta ante incidentes de peligrosidad en Cuba.

Preguntas científicas:

1. ¿Cuáles son los principales referentes teórico–metodológicos que deben sustentar la estrategia de respuesta ante incidentes de peligrosidad en Cuba?
2. ¿Qué características debe tener la solución propuesta para apoyar la estrategia de respuesta ante incidentes de peligrosidad en Cuba?
3. ¿Cómo llevar a cabo el desarrollo de la aplicación móvil para SO Android para apoyar la estrategia de respuesta ante incidentes de peligrosidad en Cuba?
4. ¿Qué estrategia de pruebas aplicar antes las funcionalidades desarrolladas a la aplicación móvil para SO Android para verificar su correcto funcionamiento?

Diseño metodológico de la investigación:

Métodos Teóricos:

- **Histórico-Lógico:** para determinar los antecedentes, tendencias, irregularidades de la estrategia de respuesta ante incidentes de peligrosidad en cuba y de las aplicaciones móviles con SO Android utilizadas para la respuesta ante incidentes de peligrosidad.

- **Analítico-Sintético:** para realizar el estudio teórico de la investigación facilitando el análisis de documentos y la extracción de los elementos más importantes acerca de la estrategia de respuesta ante índices de peligrosidad.

Métodos Empíricos:

- **Observación:** permite la estrategia de respuesta ante índices de peligrosidad para la obtención de conocimiento e información acerca del comportamiento de las aplicaciones existentes para aplicaciones móviles con SO Android utilizadas para respuesta de incidentes de peligrosidad.

- **Entrevista:** la entrevista consiste en un “intento sistemático de recoger información de otra persona”

a través de una comunicación interpersonal que se lleva a cabo por medio de una conversación estructurada donde es muy importante la forma en que se plantea la conversación y la relación que se establece en la entrevista.

El presente trabajo de diploma se ha estructurado de la siguiente manera:

- **Capítulo 1:** Fundamentación Teórica: en este capítulo se exponen los principales conceptos que contribuyen al mejor entendimiento del problema en cuestión. Se especifican detalladamente todos los argumentos que esclarecen el objeto de estudio, a partir del análisis del estado del arte de los sistemas homólogos, enfocados en aumentar la seguridad ciudadana utilizando el SO Android. Además, se realiza la caracterización de la metodología, herramientas y tecnologías que serán utilizadas en el desarrollo de la solución.

- **Capítulo 2:** Análisis y Diseño: contiene un análisis del proceso que lleva la generación de notificaciones a los organismos pertinentes, detallándose como se realizará el diseño del sistema a desarrollar. Se elaborarán los diagramas pertinentes para un mejor entendimiento del sistema.

- **Capítulo 3:** Implementación y Prueba: una vez implementado el sistema se realizarán distintas pruebas, con el fin de comprobar su funcionamiento y robustez.

Capítulo 1. Fundamentación Teórica

Introducción

En el presente capítulo se analizan soluciones similares vinculadas al campo de acción. Se caracteriza la arquitectura del SO Android. Se hace un análisis para definir las tecnologías, metodologías y herramientas a utilizar para la construcción de la solución propuesta haciendo un estudio comparativo de las mismas y se fundamenta la metodología de desarrollo de software utilizada para guiar el proceso de construcción de la solución.

1.1 Conceptos asociados a la investigación

A continuación, se detallan conceptos que se consideran relevantes conocer para un mayor entendimiento de la investigación:

- **Seguridad Ciudadana:** se refiere al mantenimiento y restablecimiento del orden público, el apoyo de la autoridad, la protección de personas, hogares y familias, y al aseguramiento y disfrute de garantías y derechos constitucionales (8).
- **Comunidad:** una comunidad es un grupo de individuos de una o más especies que viven juntos en un lugar determinado. Los individuos de una comunidad están relacionados porque tienen las mismas necesidades (8).
- **Emergencia:** el término emergencia suele ser utilizado por la mayoría de las personas para conjeturar una situación que se salió de control y como consecuencia, provocó un desastre (9).
- **Medicina de emergencia o emergentología:** es la que actúa sobre una urgencia médica, una enfermedad en estado avanzado o algo que amenace a la vida de una persona (9).
- **Catástrofe:** consecuencia destructiva más extendida que afecta a un mayor número de personas y bienes y por tanto supone un ímprobo esfuerzo de coordinación y organización global (9).
- **Incendio:** un incendio es todo aquel fuego grande que se produce en forma no deseada, propagándose y destruyendo lo que no debía quemarse. Puede ser natural o provocado por descuidos humanos o realmente adrede por personas inescrupulosas (9).

- **Asalto:** un intento ilegal, por parte de un hombre, con fuerza o violencia, para infligir una lesión corporal a otro. Un intento o amenaza para vencer a otro (9).
- **Red de datos:** es un conjunto de equipos informáticos y software conectados entre sí por medio de dispositivos físicos que envían y reciben impulsos eléctricos, ondas electromagnéticas o cualquier otro medio para el transporte de datos, con la finalidad de compartir información, recursos y ofrecer servicios (10).
- **Redes móviles:** Son aquellas redes pensadas para que el teléfono o equipo del usuario pueda moverse con libertad en la zona cubierta por dicha red incluso mientras mantiene una conversación o una conexión de datos. Una red móvil debe permitir el movimiento incluso a la velocidad de un coche sin que exista una pérdida de la conexión. Las redes móviles actuales permiten mantener esta conexión incluso a la velocidad de un tren de alta velocidad con velocidades superiores a 300 Km/h (10).
- **Telefonía Móvil:** el teléfono móvil es un dispositivo inalámbrico electrónico basado en la tecnología de ondas de radio, que tiene la misma funcionalidad que cualquier teléfono de línea fija. Su principal característica es su portabilidad, ya que la realización de llamadas no es dependiente de ningún terminal fijo y no requiere ningún tipo de cableado para llevar a cabo la conexión a la red telefónica. Aunque su principal función es la comunicación de voz, como el teléfono convencional, su rápido desarrollo ha incorporado funciones adicionales como mensajería instantánea (SMS[3]), agenda, juegos, cámara fotográfica, acceso a Internet, reproducción de video e incluso GPS[4] y reproductor mp3 (10).
- **Geolocalización:** La geolocalización es la capacidad para obtener la ubicación geográfica real de un objeto, como un radar, un teléfono móvil o un ordenador conectado a Internet. La geolocalización puede referirse a la consulta de la ubicación, o bien para la consulta real de la ubicación.

[3] Mensaje corto de texto que se puede enviar entre teléfonos celulares o móviles.

[4] Sistema de Posicionamiento Global (*Global Positioning System*) es un sistema de navegación basado en satélites

1.2 Análisis de las soluciones existentes

Para garantizar la seguridad ciudadana existen diversas soluciones, entre ellas se destacan aplicaciones para el SO Android, que contribuyen a aumentar las probabilidades de respuesta a los ciudadanos que se enfrentan ante un incidente de peligrosidad.

- ***PF Móvil (México)***

Aplicación de la Policía Federal de México que permite enviar al Centro de Atención del Comisionado (CEAC) mensajes de voz, videos o imágenes para reportar algún delito. El usuario decide si quiere compartir sus datos o hacerlo de manera anónima. En cualquier caso, la información tiene tratamiento confidencial. Dentro de sus funcionalidades se encuentra solicitar información acerca de los diferentes tipos de delitos y es posible solicitar información acerca de diferentes tópicos y consultar las cuentas de Facebook, Instagram y Twitter. En otro aspecto requiere Android 4.0 (11).

- ***Cuadrantes (Colombia)***

Cuadrantes es el nombre de una aplicación móvil desarrollada en Colombia que permite comunicarse directamente con la Policía Nacional. La aplicación identifica el cuadrante o área en la que esté ubicado y lo conecta de manera inmediata con el teléfono de los policías a cargo (12).

- ***Mí policía K8 (México, D.F.)***

Mi policía K8 es una aplicación desarrollada en México que permite hacer reportes y denuncias al Centro de Atención el Secretario (CAS), fácil manejo para los usuarios, así como información de multas y tenencias pendientes. Esta aplicación presenta como principales funcionalidades la geolocalización, realizar llamadas de emergencia y el contacto del policía más cercano a su ubicación. En requerimientos no funcionales requiere Android 4.1 y versiones superiores (13).

- ***Basta ya (Puerto Rico)***

Utilizada para poder reportar una serie de delitos y actos de corrupción indicando fecha, hora, coordenadas, descripción y foto, la aplicación promueve el cambio de actitud de la sociedad civil y el sector publico promoviendo la prevención y el esclarecimiento de los crímenes. Esta aplicación requiere Android 2.2 y versiones superiores (14).

- ***Policía de bolsillo (Venezuela)***

Esta aplicación tiene un botón de pánico que a través de un mensaje de texto y correo electrónico le permite solicitar ayuda a las autoridades informando sobre sus coordenadas, a la vez que podrá avisar a tres personas que haya registrado como contactos de emergencia. Entre sus funcionalidades se encuentran el envío de mensajes predeterminados contactos definidos en casos de urgencias y la geolocalización (15).

Luego de realizar un análisis de las soluciones existentes, se observa que todas están enfocadas a avisar a los organismos encargados de socorrer cuando exista una situación determinada que pueda poner en peligro la vida de una persona, el método o forma que emplea dichas aplicaciones para la comunicación de sus usuarios es mediante el uso de redes móviles, ya sea vía WIFI o red de datos.

Sin embargo, en países como Cuba no se puede hacer uso de los servicios dependientes de redes de datos que brindan estas aplicaciones debido a que para tener acceso a Internet es necesario conectarse a algún punto Wi-Fi o a algún lugar empresarial que cuente con tal servicio.

Cada una de estas aplicaciones tiene características propias que las hace diferentes: como la creación de comunidades, el uso de geolocalización y las llamadas en caso de emergencias. Se puede concluir que es necesario realizar una nueva aplicación para Cuba, teniendo en cuanta las funcionalidades comunes de las aplicaciones mencionadas adaptándolas a los intereses del país.

1.3 Metodología de Desarrollo de Software

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto (16).
El desarrollo de aplicaciones móviles difiere del desarrollo otros software como son sitios web y aplicaciones de escritorio en muchos aspectos, lo que provoca que las metodologías usadas para estos entornos también difieran de las del software clásico. Esto es porque el software móvil tiene que satisfacer unas condicionantes especiales (16) que lo hace más complejo como las que se muestran a continuación:

- ***Canal radio:*** consideraciones tales como la disponibilidad, las desconexiones, la variabilidad del ancho de banda, la heterogeneidad de redes o los riesgos de seguridad han de tenerse especialmente en cuenta en este entorno de comunicaciones móviles.

- ***Movilidad:*** aquí influyen consideraciones como la migración de direcciones, alta latencia debido a cambio de estación base o la gestión de la información dependiente de localización. Sobre esta

última, de hecho, se pueden implementar un sinfín de aplicaciones, pero la información de contexto asociada resulta muchas veces incompleta y varía frecuentemente.

- ***Portabilidad:*** la característica portabilidad de los dispositivos terminales implica una serie de limitaciones físicas directamente relacionadas con el factor de forma de los mismos, como el tamaño de las pantallas (algo que ha variado sustancialmente con la popularización de las pantallas táctiles), o del teclado, limitando también el número de teclas y su disposición.

- ***Fragmentación de la industria:*** la existencia de una considerable variedad de estándares, protocolos y tecnologías de red diferentes añaden complejidad al escenario del desarrollo móvil.

- ***Capacidades limitadas de los terminales:*** aquí incluimos factores como la baja potencia de cálculo o gráfica, los riesgos en la integridad de datos, las interfaces de usuario poco funcionales en muchos aspectos, la baja capacidad de almacenamiento, la duración de las baterías o la dificultad para el uso de periféricos en movilidad. Factores todos que, por otro lado, están evolucionando en la dirección de la convergencia de los ultra portátiles (*netbooks*) con los dispositivos inteligentes (*smartphones*) constituyendo cada vez menos un elemento diferencial.

- ***Diseño:*** desde el punto de vista del desarrollo, el diseño multitarea y la interrupción de tareas es clave para el éxito de las aplicaciones de escritorio; pero la oportunidad y frecuencia de éstas es mucho mayor que en el software tradicional, debido al entorno móvil que manejan, complicándose todavía más debido a la limitación de estos dispositivos.

- ***Usabilidad:*** las necesidades específicas de amplios y variados grupos de usuarios, combinados con la diversidad de plataformas tecnológicas y dispositivos, hacen que el diseño para todos se convierta en un requisito que genera una complejidad creciente difícil de acotar.

A la hora de elegir una metodología a seguir para realizar un desarrollo de una aplicación móvil, se pueden encontrar varias, como por ejemplo Mobile-D, Rapid7 y *Hybrid Methodology Desig*. Entre estas metodologías se encuentra una que se ajusta a las necesidades de las características del desarrollo a seguir en la solución, la misma se llama Metodología para el Desarrollo de Aplicaciones Móviles (MDAM) y se basa en la evaluación del potencial de éxito para servicios de tercera generación denominada 6 M’s, la ingeniería

de software educativo con modelado orientado por objetos (ISE-OO), y principalmente en los valores de las metodologías ágiles (16).

De la ISE-OO se hereda el enfoque de los micro mundos interactivos y la orientación por objetos; los elementos de los micro mundos más utilizados en los servicios móviles interactivos son: Mundo, Escenarios, Personajes y Roles, Argumento e Historia, Variables Compensatorias, Variables de Control, Variables de Resultado, Zonas de Comunicación, Ambientación-Caracterización, Recuperación de Estados Anteriores, Manejo de Información del Usuario, Mecanismos para Análisis de Desempeño, Ampliación de las Posibilidades del Micro mundo, Personalización del Ambiente y Soporte a la Comunicación en Grupo (16).

De las metodologías ágiles se heredan los conceptos inmersos en los cuatro postulados o manifiesto ágil (17).

- Desarrollar software que funciona más que conseguir buena documentación.
- La respuesta ante el cambio es más importante que el seguimiento de un plan.
- Colaboración con el cliente sobre negociación contractual.
- Individuos e interacciones sobre procesos y herramientas.

La metodología se encuentra enmarcada en cinco fases como se muestra en la Figura 1, denominadas: análisis, diseño, desarrollo, pruebas de funcionamiento y entrega. A continuación, se describe cada una de las actividades que intervienen en el desarrollo de la propuesta.

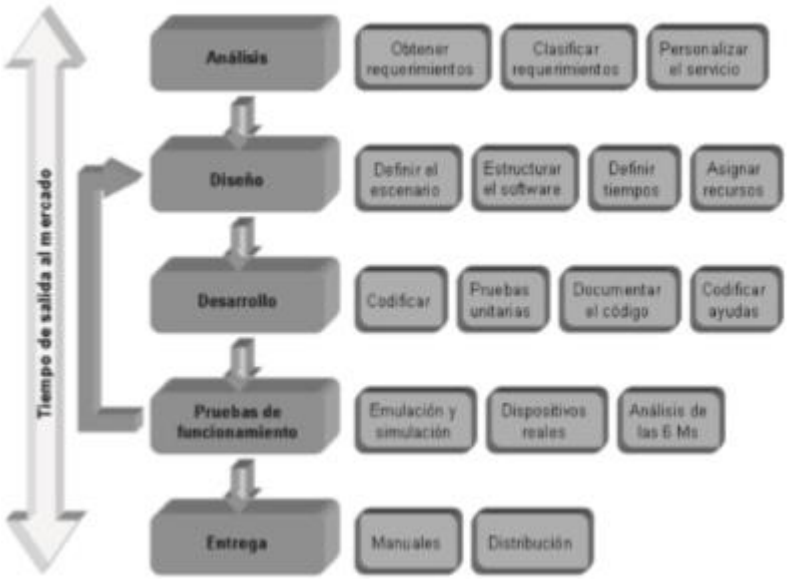


Figura 1 Metodología MADM. Fuente: ver RB[5] (16).

- ***Análisis***

En esta fase se realizan tres tareas: obtener requerimientos, clasificar los requerimientos y personalizar el servicio. Generando como artefactos para la obtención de requisitos la técnica de la entrevista (ver *Anexo 1*), para la clasificación de los requisitos se tuvo en cuenta tres aspectos entorno, funcionales y no funcionales.

- ***Diseño***

El objetivo de esta etapa es plasmar el pensamiento de la solución mediante diagramas o esquemas. Una vez determinado el escenario se generan como artefactos los posibles diagramas que define la metodología. Estos diagramas pueden ser: Diagrama de Caso de Uso del Sistema (CUS), Diseño de Clase (DC) y Diagrama de Despliegue (DD).

- ***Desarrollo***

El objetivo de esta fase es implementar el diseño en un producto de software. En esta etapa se realizan las siguientes actividades:

[5] Referencias Bibliográficas

- **Codificar:** se escribe en el lenguaje de programación seleccionado, cada una de las partes definidas en los diagramas realizados en la etapa de diseño.
- **Pruebas unitarias:** se verifica el funcionamiento de la aplicación.
- **Documentar el código:** a medida que se codifica y se prueba cada elemento, se redacta la pequeña documentación sobre lo desarrollado.
- **Pruebas de funcionamiento**

El objetivo de esta fase es verificar el funcionamiento de la aplicación en diferentes escenarios y condiciones.

- ***Entrega***

Terminada la depuración de la aplicación y atendidos todos los requerimientos de última hora del cliente se da por finalizada la aplicación y se procede a la entrega del ejecutable, el código fuente y la documentación.

1.4 Ambiente de Desarrollo

Una vez definida la metodología que guiará el proceso de desarrollo del software se seleccionaron las herramientas para la construcción del mismo. A continuación, se mencionan dichas herramientas:

1.4.1 Lenguaje de modelado

“El modelado constituye una simplificación de la realidad donde se define lo esencial para la construcción del software con los objetivos de comunicar la estructura de un sistema complejo, especificar el comportamiento deseado del sistema, comprender mejor lo que se está desarrollando y descubrir oportunidades de simplificación y reutilización” (18).

Lenguaje Unificado de Modelado (UML) 2.0

UML es el acrónimo de Lenguaje Unificado de Modelado, es el lenguaje estándar para visualizar, especificar, construir y documentar los artefactos de un sistema, utilizándose para el modelado del negocio y sistemas de software. También ofrece un estándar para describir los modelos, incluyendo aspectos conceptuales como procesos de negocio, funciones del sistema, expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables (19).

1.4.2 Lenguaje de programación

Se selecciona ***Java 8*** como lenguaje de programación porque es el nativo para el desarrollo de aplicaciones para el SO Android, permite optimizar el tiempo y el ciclo de desarrollo (compilación y ejecución). Además, cientos de aplicaciones desarrolladas en Java corriendo satisfactoriamente en la plataforma Android, certifican lo idóneo que resulta este lenguaje de programación.

La principal característica de Java es la de ser un lenguaje compilado e interpretado, a la vez de ser de código abierto. Todo programa en Java ha de compilarse y el código generado es interpretado por una máquina virtual. De este modo se consigue la independencia de la máquina, el código compilado se ejecuta en máquinas virtuales que si son dependientes de la plataforma. Java es un lenguaje orientado a objetos de propósito general. Aunque Java comenzará a ser conocido como un lenguaje de programación de *applets*[6] que se ejecutan en el entorno de un navegador web, se puede utilizar para construir cualquier tipo de proyecto (20).

1.4.3 Herramienta case

Se empleará ***Visual Paradigm*** en su versión 8.0, para modelar los diagramas que se generen en cada una de las etapas del proceso de desarrollo de software.

Visual Paradigm para UML es una herramienta de modelado que soporta el modelado mediante UML y proporciona asistencia tanto a los analistas, ingenieros de software como a los desarrolladores durante todo el ciclo de vida del proyecto (21).

La herramienta de modelado *Visual Paradigm* (en su versión libre) para UML en su versión 8.0 soporta el lenguaje de modelado UML, facilita el trabajo del equipo de desarrollo durante el ciclo de vida del software y permite la estandarización de la documentación que se va generando.

1.4.4 Entorno de desarrollo

Se utiliza **Android Studio** en su versión 2.3 pues es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones Android, basado en IntelliJ IDEA. Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan tu

[6] Es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo, en un navegador web.

productividad durante la compilación de aplicaciones para Android (22), como las siguientes:

- Sistema de compilación flexible basado en Gradle[7].
- Un emulador rápido con varias funciones.
- Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos Android.
- Instant Run, para aplicar cambios mientras la aplicación se ejecuta sin la necesidad de compilar un nuevo APK.
- Integración de plantillas de código y GitHub, para ayudarte a compilar funciones comunes de las aplicaciones e importar ejemplos de código.

1.4.5 Android SDK 25.0.2

El paquete de Desarrollo de Software de Android (SDK, por sus siglas en ingles *Software Development Kit*), no es más que un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para Android. Estos proporcionan bibliotecas de interfaz de programación de aplicaciones creadas con el fin de permitir un mejor uso de las técnicas a desarrollar en este sistema (23).

Para el trabajo con el Android Studio se hace necesario el uso del conjunto de recursos SDK fundamentales que dispone la plataforma Android para el desarrollo de aplicaciones. Posteriormente se debe configurar el IDE para que localice el SDK de Android y disponga de sus recursos en el entorno de desarrollo. Entre los recursos de que dispone el SDK de Android se encuentra un emulador de dispositivos, para probar los desarrollos sin necesidad de realizar instalaciones sobre un dispositivo físico

Los SDK continuamente contienen códigos de ejemplo y notas técnicas de soporte para ayudar a clarificar ciertos puntos del material de referencia. Este marco de trabajo es tan sencillo como las APIs[8], creadas con el objetivo de permitir el uso de cierto lenguaje de programación, que puede incluir también *hardware* sofisticado para comunicarse con este sistema.

[7] Paquete de herramientas de compilación avanzadas, para automatizar y administrar el proceso de compilación.

[8] Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación.

1.4.6 Sistema gestor de base de datos

Se selecciona **SQLite** en su versión 3.8 debido a que es una de las mejores alternativas para la persistencia de los datos sobre texto plano o ficheros en XML ya que es un sistema gestor muy ligero que posibilita la portabilidad local de los datos y no requiere de un gran procesamiento en memoria para su gestión. Android posee herramientas muy útiles que facilitan las tareas de manipulación y consulta sobre este tipo de base de datos. Además, el tiempo de respuesta entre Realm y dicho gestor no sobrepasan los 1ms de respuesta en tiempo de lectura para un volumen de datos menor que las mil tuplas.

SQLite es un sistema de gestión de bases de datos que enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más rápidas y seguras que la comunicación entre procesos (24).

1.4.7 Control de versiones

Se escoge **Git** en su versión 9.2.5 para el control de versiones porque presenta las siguientes ventajas:

- Distribuido, no centralizado: Esta característica es importante porque permite que personas en diferentes estaciones de trabajo puedan trabajar sobre un mismo proyecto.
- Es rápido: trabaja todo localmente gracias a que es un sistema distribuido.
- Comprueba: casi todo es guardado en Git no por nombre, sino por la suma de comprobación de sus contenidos.

El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante (25).

1.5 Conclusiones parciales del capítulo

En este capítulo fueron analizadas distintas herramientas que brindan mecanismos para aumentar la probabilidad de respuesta ante incidentes de peligrosidad en teléfonos con el SO Android indicando que estas herramientas no resuelven las necesidades actuales para Cuba pero si se puede hacer uso de algunas de sus características como el uso de la geolocalización, el envío de SMS y las llamadas telefónicas. Se

seleccionaron JAVA como lenguaje de programación, SQLite en su versión 3.8 como gestor de base de datos y Android Studio en su versión 2.3 como entorno de desarrollo, al ser herramientas de desarrollo más usadas para este tipo de solución todas guiadas por la metodología MDAM.

Capítulo 2. Análisis y Diseño

Introducción

El siguiente capítulo se enfoca en plantear la vía a seguir para la realización de la solución propuesta que resuelve el problema expuesto, siguiendo la metodología MDAM con la finalidad de llegar a obtener un producto con buena calidad, teniendo en cuenta las prácticas y los principios que contribuyan a agilizar el proceso. Se realizará una descripción de la solución propuesta para ofrecer de manera legible la finalidad de este trabajo. Los requisitos serán explicados a detalle basándose en la clasificación que propone la metodología seleccionada, se mostraran distintos diagramas que ayudaran a una mejor comprensión de la solución y se describirán los distintos patrones de diseños utilizados para el cumplimiento de los mismos

2.1 Propuesta de solución

En la presente investigación se propone el desarrollo de una aplicación basada en SO Android con versiones superiores a 4.0 (IceCream Sándwich) con funcionalidades que permitan, desde un dispositivo móvil, hacer uso de las redes de datos para enviar mensajes instantáneos a los contactos. Además, permitirá ubicar geográficamente las estaciones policiales, hospitales y bomberos y emitir una notificación de alerta informando cual es el más cercano.

También contendrá teclas de acceso rápido la cual permitirá hacer una llamada gratuita en caso de una situación extraordinaria ya sea presencia de incendio o robos, por otra parte, también generará un mensaje predeterminado informando a algún contacto que se haya definido para estas situaciones de emergencia. Existen dos vías para mandar mensajes, la primera vía es utilizando los mensajes tradicionales con un valor de 9 centavos y la segunda vía es a través de mensajes multimedia (MMS).Para la comunicación entre usuarios es necesario que cuenten con cuenta de correo nauta ya que el tráfico mensajes instantáneos será usando dicha cuenta, esto posibilitará que el costo de los mensajes sea mínimo. También para todo lo anteriormente descrito es necesario de que el dispositivo contenga cobertura pues la realización de las llamadas será utilizando la línea del ciudadano.

2.2 Fase de análisis.

2.2.1 Obtención de requisitos

Según la metodología seleccionada se utilizó como técnica de obtención de los requisitos la entrevista (ver *Anexo 1*), la misma se le realizo al cliente Teniente Coronel Orlando Perdomo Cruz, para determinar las

necesidades que se pretenden solucionar con las tecnologías móviles, o simplemente, para que señale las características que debe tener la aplicación (26).

Entrevista:

La entrevista es de gran utilidad para obtener información cualitativa como opiniones, o descripciones subjetivas de actividades. Es una técnica muy utilizada, y requiere una mayor preparación y experiencia por parte del analista.

Una vez identificados los requerimientos que debe tener el software, se procede a clasificarlos en Entorno (todo lo que rodea al servicio), Funcionales (los que forman parte del funcionamiento del sistema) y no Funcionales (27).

A continuación, se muestran los requisitos identificados para el desarrollo de la solución (ver Tabla 1), clasificándose de acuerdo a la metodología seleccionada:

Tabla 1Obtención de requisitos

No	Nombre	Descripción	Clasificación
RF 1	Insertar cuenta nauta	El sistema debe permitir insertar una cuenta nauta.	Entorno y Funcional
RF 2	Modificar cuenta nauta	El sistema debe permitir modificar una cuenta nauta.	Entorno y Funcional
RF 3	Eliminar cuenta nauta	El sistema debe permitir eliminar una cuenta nauta.	Entorno y Funcional
RF 4	Insertar contacto	El sistema debe permitir insertar un contacto.	Funcional
RF 5	Eliminar contacto	El sistema debe permitir eliminar un contacto.	Funcional
RF 6	Enviar SMS predeterminado al contacto de salud	El sistema debe permitir enviar el mensaje predeterminado al contacto configurado para salud.	Entorno y Funcional
RF 7	Enviar SMS predeterminado al contacto de delito	El sistema debe permitir enviar el mensaje predeterminado al contacto configurado para delito.	Entorno y Funcional
RF 8	Enviar SMS predeterminado al contacto de bombero	El sistema debe permitir enviar el mensaje predeterminado al contacto configurado para bombero.	Entorno y Funcional

RF 9	Realizar llamada a SIUM	El sistema debe permitir realizar llamada de urgencia al servicio SIUM.	Entorno y Funcional
RF 10	Realizar llamada a Cuerpo de Bomberos	El sistema debe permitir realizar llamada de urgencia al Cuerpo de Bomberos.	Entorno y Funcional
RF 11	Realizar llamada a PNR	El sistema debe permitir realizar llamada de urgencia a la PNR.	Entorno y Funcional
RF 12	Determinar ubicación para bomberos más cercano	El sistema debe permitir ubicar el cuerpo de bombero más cercano.	Funcional
RF 13	Determinar ubicación para hospital más cercano	El sistema debe permitir ubicar el hospital más cercano.	Funcional
RF 14	Determinar ubicación para estación de policía más cercano	EL sistema debe permitir ubicar la estación de policía más cercana.	Entorno y Funcional
RF 15	Actualiza contacto de chat.	El sistema debe sincronizar los contactos con correo.	Entorno y Funcional
RF 16	Insertar contacto de chat.	El sistema debe permitir insertar el contacto seleccionado.	Funcional
RF 17	Eliminar contacto de chat	El sistema debe permitir eliminar el contacto seleccionado.	Funcional
RF 18	Crear mensaje predeterminado	El sistema trae predefinido los mensajes predeterminados para cada caso.	Funcional

2.2.2 Requisitos no funcionales.

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente se aplican a características o servicios individuales del sistema (27).

A continuación, se relacionan los requisitos no funcionales definidos para la propuesta de solución que se presenta.

RNF1. Requerimientos de software

- Para la utilización del sistema se requiere el SO Android en su versión 4.0 (*IceCream Sándwich*) o una versión superior.

RNF2. Requerimientos de hardware

- Capacidad disponible de 6 Mb en la memoria de teléfonos inteligentes para la instalación y ejecución de la aplicación.
- *Smartphone* con soporte de redes GSM.

RNF3. Costo

- El sistema envía mensajes con costo de 1 centavo.

2.3 Fase de diseño

2.3.1 Definición de escenario

Las aplicaciones móviles se pueden diseñar para ejecutarse en diferentes escenarios, dependiendo del sistema de conexión y sincronización con el servidor o aplicación central; el proceso de sincronización se realiza para insertar, modificar o borrar información. Entre los diferentes escenarios que define la metodología, se encuentran los siguientes (16):

- **Desconectado:** los procesos se realizan en el dispositivo móvil desconectado, después de terminar el proceso, si se requiere, puede conectarse con una aplicación central mediante el proceso de sincronización.
- **Semiconectado:** los procesos pueden ejecutarse en el dispositivo móvil desconectado, pero se requiere establecer conexión en algún momento para terminar el proceso, al sincronizar la información con el servidor o aplicación central. En los escenarios desconectado y semiconectado se recomienda utilizar los protocolos y tecnologías que se ajusten al servicio y capacidades tecnológicas del dispositivo. Algunos son: *Media Transfer Protocol (MTP), Near Field Communication (NFC), SlowSync, FastSync y SyncML*.
- **Conectado:** el dispositivo debe estar siempre conectado con la aplicación central o servidor para su correcto funcionamiento, no se almacenan datos o archivos en el móvil, la sincronización se realiza mediante la validación de formularios, usualmente se utiliza el Protocolo de Transferencia de Hipertexto (*Hypertext Transfer Protocol, HTTP*).

En este caso se propone un escenario **semiconectado** pues el dispositivo móvil solo va requerir conexión en pequeños intervalos de tiempo.

2.3.2 Diagrama de casos de uso del sistema

Se utilizó el diagrama de Casos de Uso para detallar la relación que existe entre el actor y el sistema, en este diagrama se ilustran los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo (27). A continuación, (ver Figura 2) se representa el Diagrama de Casos de Uso del Sistema, donde participa el actor y las entidades que fueron definidas para la solución.

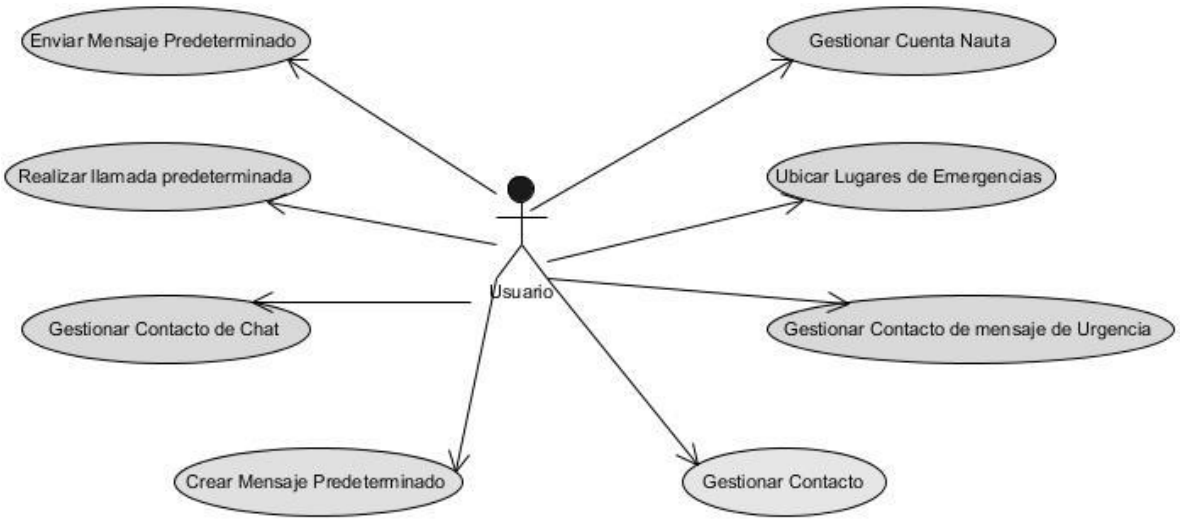


Figura 2 Diagrama de Casos de Uso del Sistema. Fuente: Elaboración propia

Tabla 2 Descripción del actor del sistema

Actor	Objetivo
Usuario	Realizar llamadas gratuitas, realizar geolocalizaciones, visualizar agenda de contactos, realizar la llamada predeterminada, enviar mensaje predeterminado y gestionar contactos de chat.

2.3.2.1 Patrones de CU

Un patrón de casos de uso no describe un uso particular de un sistema, más bien se basa en la captura de técnicas para que el modelado sea mantenible, reusable, y entendible (28). Para el CUS se utilizó el patrón CRUD[9] parcial porque se realiza las acciones insertar, modificar y eliminar.

[9] Es el acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete), que se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.

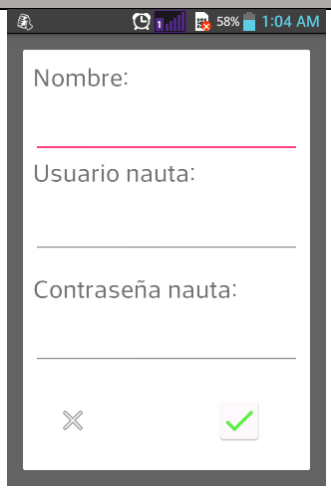
Tabla 3. Descripción del CU Gestionar cuenta

Objetivo	Usuario.	
Actor	El caso de uso se inicia cuando el Usuario decide registrar, modificar o eliminar datos de la cuenta nauta.	
Resumen	Alta	
Complejidad	Crítico	
Prioridad	Usuario ya es cliente de Etecsa	
Precondiciones	Se registra un Usuario, se modifican los datos o se elimina el mismo.	
Postcondiciones	RF1,RF2,RF3	
Referencia	Usuario.	
Flujo de eventos		
Flujo básico.		
	Actor	**Sistema**
1	**a)** Desea registrar, modificar o eliminar los datos de la cuenta nauta.	
2		Muestra un menú con las siguientes opciones: -Registrar cuenta. -Modificar cuenta. -Eliminar cuenta.
3	**b)** Escoge una de las opciones: registrar, modificar o eliminar los datos de la cuenta	

	nauta.	
4		Ejecuta algunas de las siguientes acciones: a) Si decide registrar un usuario, ir a la sección “Insertar cuenta” b) Si decide modificar los datos de la cuenta a la sección “Modificar cuenta”. c) Si decide eliminar la cuenta nauta, ir a la sección “Eliminar cuenta”.
Sección “Insertar cuenta”		
	Actor	**Sistema**
Flujo Básico Insertar cuenta		
1		Muestra una ventana con los siguientes campos a introducir: -Nombre de la cuenta. -Usuario nauta. -Contraseña. -Confirmar contraseña. - Y el botón Registrar cuenta.
2	Introduce los datos (Nombre de la cuenta, Usuario, Contraseña,) y presiona el botón Registrar cuenta.	2. a Verifica que todos los campos estén llenos.
3		2. b Verifica que los datos introducidos estén correctos.
4		2. c Verifica que la cuenta no exista.
5		2. d Almacena los datos de la cuenta y muestra el mensaje “Registro

		satisfactorio". Finalizando así la opción Insertar cuenta.

Prototipo Funcional: "Insertar cuenta"



Sección 1: "Insertar cuenta"

Flujo Alterno de Eventos "Campos vacíos"

Actor		Sistema
Flujo básico Campos vacíos		
1		2. a Muestra el mensaje "Campos vacíos"

Flujo Alterno de Eventos "Datos correctos"

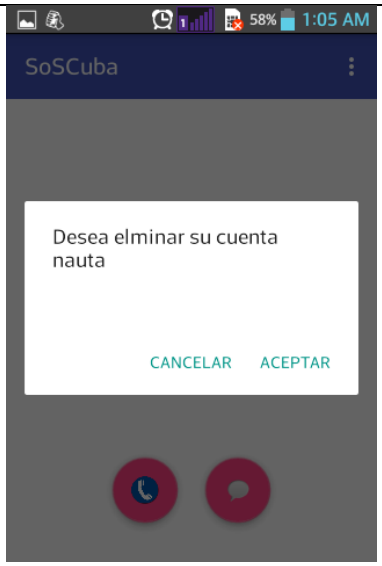
Actor		Sistema
Flujo básico Datos correctos		
1		2. b Muestra el mensaje "Datos incorrectos"

Flujo Alterno de Eventos "Usuario existente"

Actor		Sistema
Flujo básico.		
1		2. c Muestra el mensaje "Cuenta existente"

Sección “Modificar Cuenta”		
Actor		**Sistema**
Flujo básico Modificar Cuenta.		
1		Muestra una ventana con los campos de cuenta y el botón aceptar y cancelar.
2	Selecciona la cuenta presiona el botón aceptar.	
		2.a Muestra una ventana con los datos a modificar: -Nombre de la cuenta. - Usuario nauta. -Contraseña. -Confirmar contraseña. -Y el botón Aceptar o Cancelar
3	Realiza las actualizaciones deseadas en los datos (Nombre de la cuenta, Usuario nauta y Contraseña) y presiona el botón Aceptar.	
		3. a Verifica que todos los campos estén llenos.
		3. b Verifica que los datos introducidos estén correctos.
		Actualiza la información incorporada a la cuenta y se emite un mensaje ¨Modificación satisfactoria¨. Finalizando la opción Modificar cuenta.
Prototipo Funcional: “Modificar cuenta”		

Sección 2: "Modificar cuenta"		
Flujo Alterno de Eventos "Campos vacíos"		
Actor		**Sistema**
Flujo básico Campos vacíos.		
1		3.a Muestra el mensaje "Campos vacíos"
Flujo Alterno de Eventos "Datos incorrectos"		
Actor		**Sistema**
Flujo básico Datos incorrectos.		
1		3.b Muestra el mensaje "Datos incorrectos"
Sección "Eliminar Cuenta"		
Actor		**Sistema**
Flujo básico Eliminar Cuenta.		
1		Muestra una ventana con el usuario nauta y el botón aceptar.
2	Selecciona la cuenta eliminar y presiona el botón Eliminar Cuenta.	
		2. a Muestra el mensaje de confirmación "¿Está seguro que desea eliminar la cuenta

		seleccionado?” y los botones aceptar y cancelar.
3	Presiona el botón aceptar.	
		1 a Elimina la cuenta seleccionada y emite el mensaje “Eliminación satisfactoria”. Finalizando así el caso de uso.
Prototipo Funcional: “Eliminar cuenta”		
 C		
Sección 3: “Eliminar cuenta”		
Flujo Alterno de Eventos “Cancelar eliminación de la cuenta nauta”		
Actor		**Sistema**
Flujo básico. Cancelar eliminación de la cuenta nauta.		
1	2. a Presiona el botón cancelar.	
		Vuelve al paso 2 del flujo básico de eliminar usuario.
Relaciones	**CU Incluidos**	-
	CU Extendidos	-
Requisitos no funcionales	-	

-Relaciones	-

2.4 Modelo de diseño

Un Modelo de Diseño describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con en el entorno de implementación, tiene impacto en el sistema. El modelo de diseño es considerado la entrada fundamental de las actividades de implementación (29).

2.4.1 Diagrama de clase de diseño

Para el desarrollo de la aplicación surge la necesidad de representar a través de las clases, cuales son los atributos que la identifican y la relación que existe entre ellas, ya sea de asociación, herencia o agregación. Por tanto, se modeló un diseño de clases para mostrar lo que el sistema puede hacer. A continuación, (ver Figura 3) se muestra un fragmento del diagrama de clases del sistema.

Las clases que se representan en este diagrama tienen una relación de dependencia entre ellas. A continuación, se describen las clases que componen el diseño del sistema.

- MainActivity: Es la clase principal, donde se encuentran las principales funcionalidades del proceso.
- MainTagget: Es la encargada de gestionar los datos para el envío y recepción de los mensajes usando los datos móviles, también gestiona la inserción y eliminación de los contactos para el chat.
- MapActivity: Se encarga de todo el funcionamiento en cuanto a la geolocalización así como pintar el mapa y ubicar los lugares de urgencias.
- ConexionBD: En esta clase se implementan todos los métodos para hacer uso de la base dato.
- BD: Se crean todas las tablas de la base de datos.

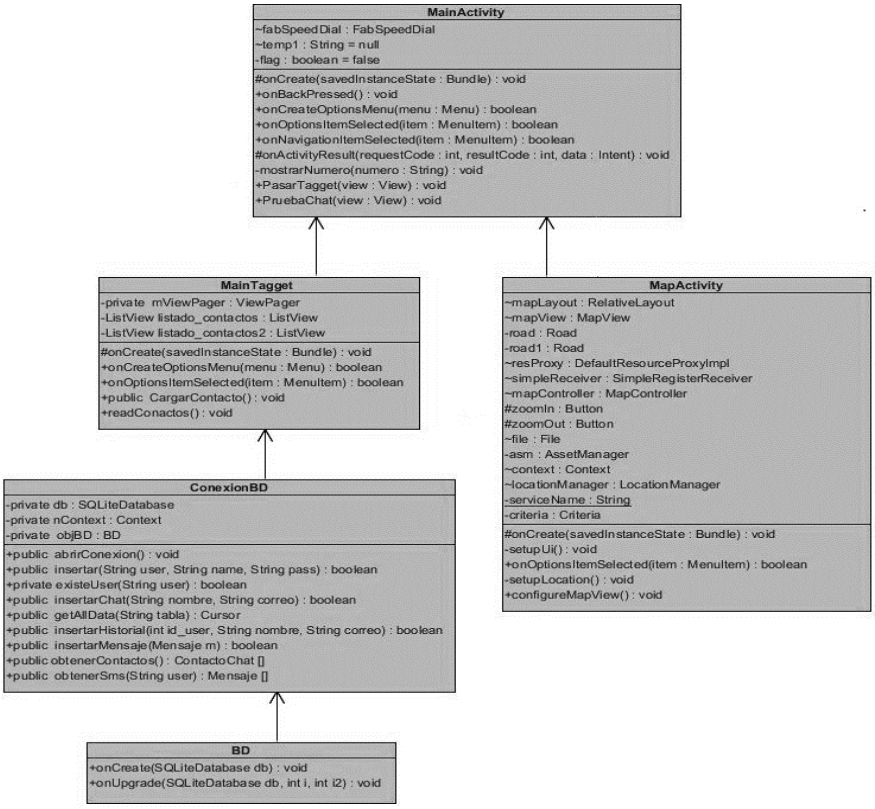


Figura 3 Diagrama de clases del diseño. Fuente: Elaboración Propia.

2.5 Patrones de diseño

Patrones GRASP

Los patrones para Asignar Responsabilidades (GRASP del inglés *Responsability Assignment Patterns*) describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño del objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable (29). En el diseño de la aplicación se emplearon patrones de diseño GRASP tales como:

- **Creador:** Soluciona el problema de ¿quién debería ser responsable de crear una nueva instancia? El patrón creador guía la asignación de responsabilidades relacionadas a la creación de objetos, una tarea muy común en sistemas orientados a objetos. Se aplica en todos los casos donde una clase tiene la responsabilidad de crear una nueva instancia de la otra. Un ejemplo donde se utiliza este patrón es en la clase *ConexionBD* encargada de crear la clase *BD*.

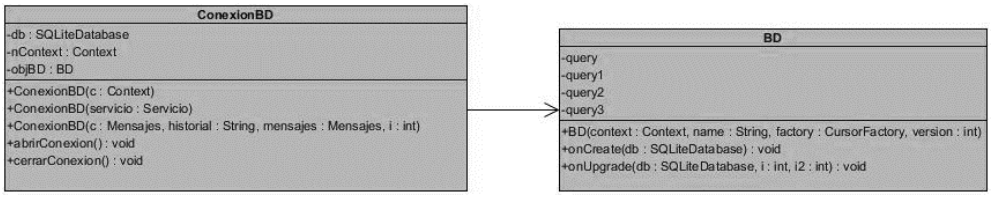


Figura 4 Patrón Creador. Fuente: Elaboración Propia.

- **Experto:** El patrón experto en información soluciona el problema ¿de qué forma se puede saber qué responsabilidad delegar a cada objeto? Es el principio básico de asignación de responsabilidades. Indica que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. Por ejemplo, la clase *ContactoChat* es la encargada de proporcionar toda la información relacionada con los contactos, debido a que es la que representa todos los atributos para la creación de este objeto.

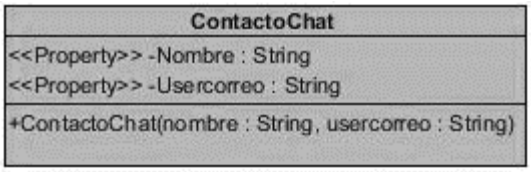


Figura 5 Patrón Experto. Fuente: Elaboración Propia.

- **Controlador:** Resuelve el problema de ¿quién gestiona un evento del sistema? Es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado. La evidencia de este patrón se encuentra en las clases *MainActivity*, *MainTagget*.

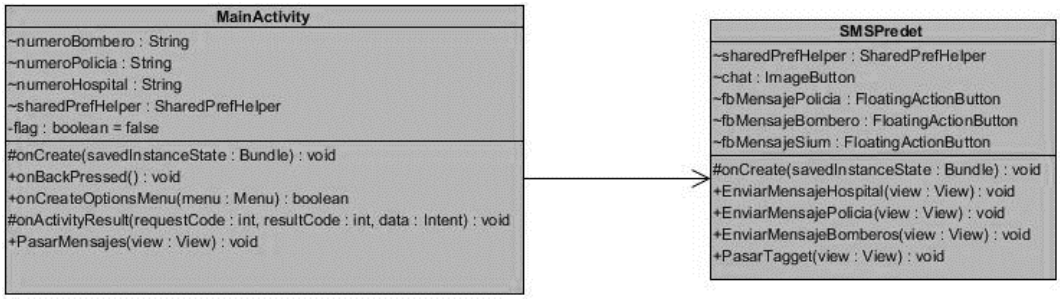


Figura 6 Patrón Controlador. Fuente: Elaboración Propia.

- **Alta Cohesión:** Soluciona el problema de ¿cómo mantener manejable la complejidad?, ya que asigna responsabilidades de manera que la información que almacena una clase sea coherente y esté relacionada con la clase. Este patrón se evidencia en todas las clases ya que ellas almacenan la información necesaria para trabajar, sin involucrar otras clases.
- **Bajo Acoplamiento:** Soluciona el problema de ¿cómo dar soporte a las bajas dependencias y al incremento de la reutilización? Plantea tener las clases lo menos ligadas entre sí, de tal forma que, en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las mismas (33). Este patrón se evidencia en todas las clases porque existe la mínima dependencia entre ellas para realizar modificaciones sin alterar su comportamiento.

Patrones GOF

- **Singlenton:** Su objetivo es restringir la creación de objetos pertenecientes a una clase, de modo que solo se tenga una única instancia de la clase para toda la aplicación, garantizando así un punto de acceso global al objeto creado. Este patrón se evidencia en la clase ConexiónBD porque a través de esta clase es donde se realizan las operaciones de insertar y actualizar los datos (29).

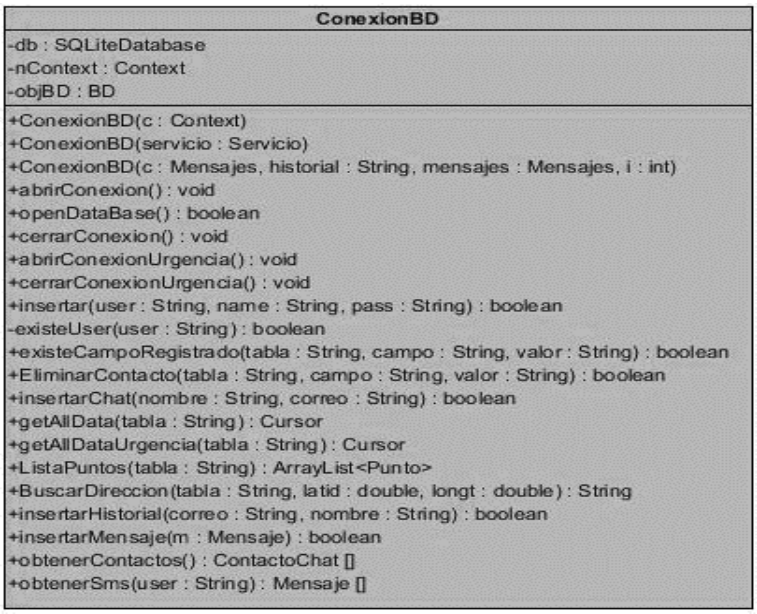


Figura 7 Patrón Singlenton. Fuente: Elaboración Propia.

Para el CU definido se generó el siguientes diagrama de clase y su descripción (ver Figura 8).

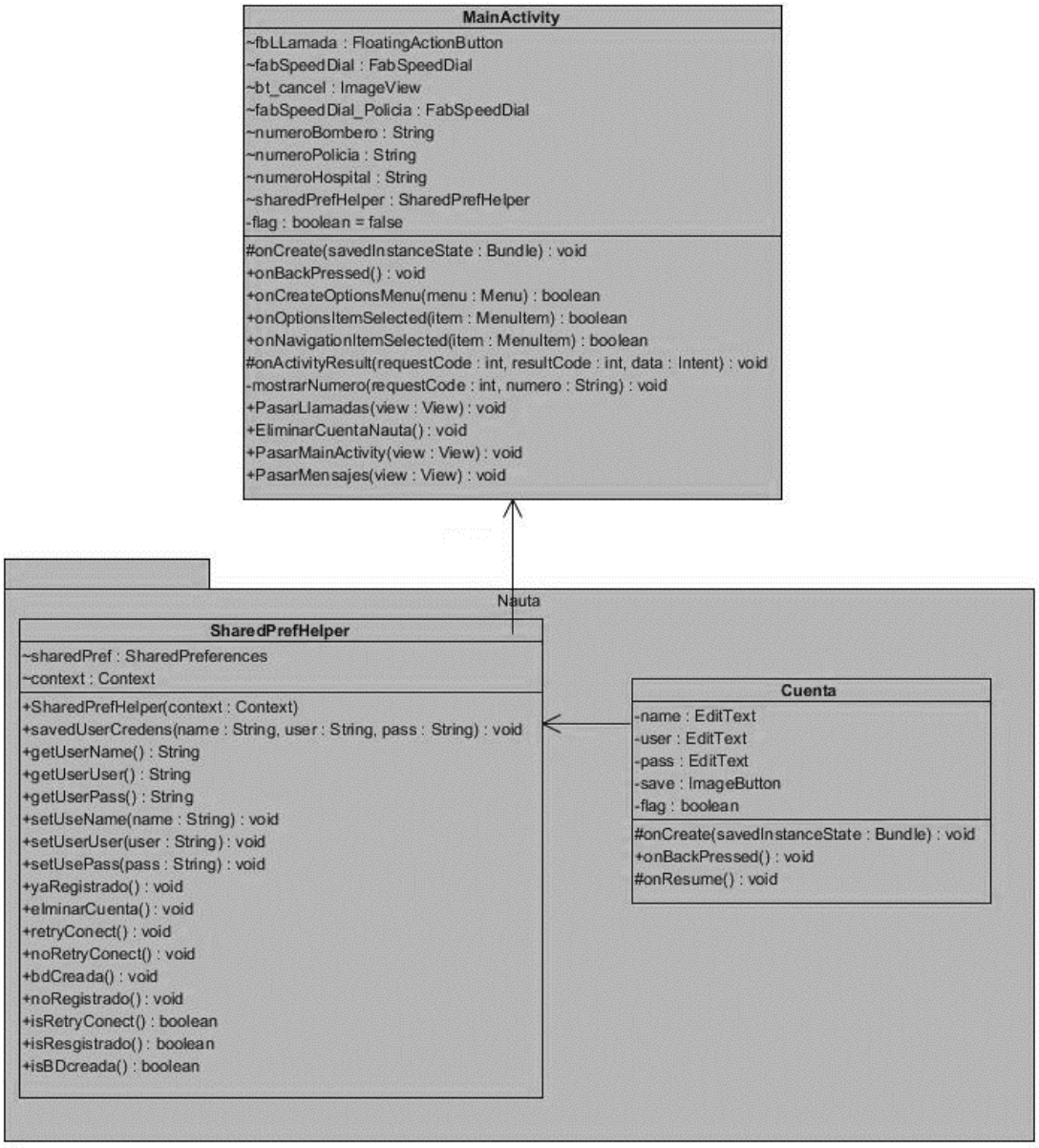


Figura 8 Diseño de clases para el CU Gestionar Cuenta. Fuente: Elaboración Propia

2.5.1 Diagrama de paquete

Una forma de representan el empaquetado de la aplicación, fue agrupando elementos relacionados semánticamente y a su vez mostrando las dependencias entre esas agrupaciones (29). A continuación, (ver

Figura 9*)* se muestra el diagrama de paquetes correspondiente al sistema.

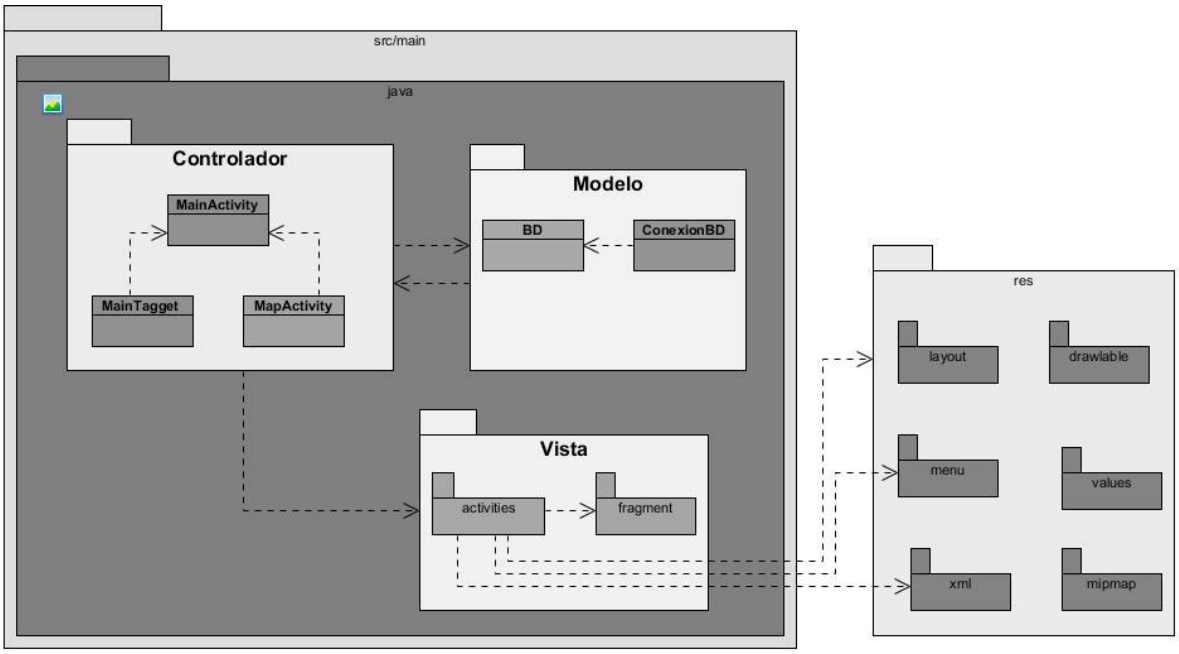


Figura 9 Diagrama de Paquetes. Fuente: Elaboración propia

Los proyectos Android eventualmente van construyendo en los archivos .apk todo el código fuente, recursos y elementos necesarios para el funcionamiento del software. Estos archivos .apk son los ejecutables de instalación de las aplicaciones. Algunos de estos ficheros .apk son generados por el desarrollador por defecto, mientras que otros solo en caso de ser necesario. La estructura de paquetes y archivos de un proyecto de este tipo es la siguiente:

- **src/:** Contiene los archivos de clases y actividades. Estos son almacenados en la dirección: src/your/package/namespace/ActivityName.Java.
- **bin/:** Directorio de salida de la construcción del archivo .apk y otros recursos compilados.
- **gen/:** Contiene los archivos de Java generados por el ADT, como el archivo R.Java para el control de los identificadores de todos los elementos implicados en la herramienta.
- **res/:** Contiene los recursos de la aplicación, como archivos *drawable*, *layouts* y los valores *string* (archivo XML que contiene los textos visualizados en la herramienta).
- **drawable/:** Para archivos de imágenes.png, .jpeg o .gif; archivos XML que describen las formas

drawable u objetos *drawable* que contengan múltiples estados.

- **layout/:** Archivos XML que son compilados en los *layouts* de pantalla.
- **values/:** Para archivos XML que son compilados en diversos tipos de recursos. A diferencia de otros recursos del directorio res/, los recursos que son escritos a archivos XML en esta carpeta no son referenciados por su nombre, sino que el tipo de elemento XML contenido es controlado a través de la clase R.
- **libs/:** Contiene las librerías privadas que son utilizadas internamente en el proyecto.

2.6 Patrón arquitectónico

Un patrón de arquitectura de software es un esquema genérico probado para solucionar un problema particular, el cual es recurrente dentro de un cierto contexto. Este esquema se especifica describiendo los componentes, con sus responsabilidades y relaciones. Para el desarrollo de la solución se utilizará el patrón arquitectónico Modelo-Vista–Controlador (ver Figura 10) (30).

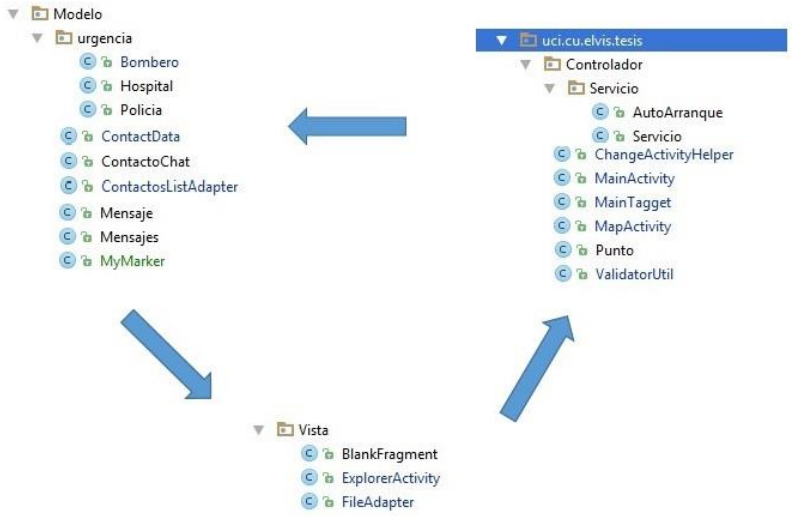


Figura 10 Patrón Arquitectónico MVC. Fuente: Elaboración Propia

Modelo-Vista–Controlador (MVC): el patrón proporciona grandes ventajas como la organización de código, reutilización y flexibilidad. La programación es organizada pues separa los datos de la aplicación, la interfaz de usuario y la lógica de los datos en tres componentes diferentes, en el cual cada capa se especializa en una función específica.

- **Modelo:** representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos. En esta capa se encuentran las clases *BD* y *ConexionBD.*
- **Vista:** muestra el modelo en un formato adecuado para interactuar. Maneja la presentación visual de los datos representados por el modelo. En esa capa se encuentran Mensaje y Mensajes.
- **Controlador:** responde a eventos e invoca cambios en el modelo y probablemente en la vista. Dentro la capa Controlador se encuentran clases como *MainActivity*, *FileAdapter* y *ExplorerActivity.*

2.7 Conclusiones parciales

Mediante el método de entrevista se lograron extraer 25 requisitos clasificándolos según la metodología en entorno, funcional y no funcional logrando así identificar las características con las que contará la aplicación, las cuales darán respuesta a las necesidades del problema. Se selecciona el escenario semidesconectado al ser el más idóneo en soluciones como estas cuando el dispositivo móvil solo requiere conexión en pequeños intervalos. Los patrones de diseño empleados garantizaron aspectos como la facilidad de entendimiento del código para su posterior mantenimiento. Y la elaboración de distintos diagramas permitió plasmar el pensamiento de la solución mediante artefactos.

Capítulo 3. Implementación y Prueba

Introducción

En el presente capítulo se definen los modelos de implementación para desarrollar la aplicación que permite aumentar las probabilidades de respuesta ante incidentes de peligrosidad. Se definen los estándares de codificación utilizados para el desarrollo del sistema y se ejecutan las pruebas de software para verificar el correcto funcionamiento de la aplicación.

El objetivo de este capítulo luego de haber realizado el diseño es desarrollar la solución y luego someter la herramienta a pruebas internas pues podría presentar fallas, mediante casos de pruebas donde se validarán las soluciones y finalmente poder realizar el despliegue en los sistemas de producción.

3.1 Modelo de implementación

La implementación constituye una de las fases más importantes del desarrollo de software. En ella se toman como punto de partida los resultados obtenidos en el diseño, implementándose el sistema en términos de componentes como ficheros de código binario, código fuente, *scripts* y ejecutables. Su importancia reside en que se obtiene como consecuencia un sistema ejecutable, siendo esto uno de los principales objetivos en el desarrollo de software (29).

3.1.1 Diagrama de componente.

Un componente es una parte física de un sistema (módulo, base de datos, programa ejecutable). Se puede decir que un componente es la materialización de una o más clases, porque una abstracción con atributos y métodos pueden ser implementados en los componentes (29).

Un diagrama de componentes representa la separación de un sistema de *software* en componentes físicos (por ejemplo: archivos, módulos, paquetes, base de datos, código fuente, binario o ejecutable) y muestra las dependencias y organización existente entre estos componentes.

A continuación, se representan el diagrama de componente correspondiente al CU Gestionar Cuenta (ver Figura 11).

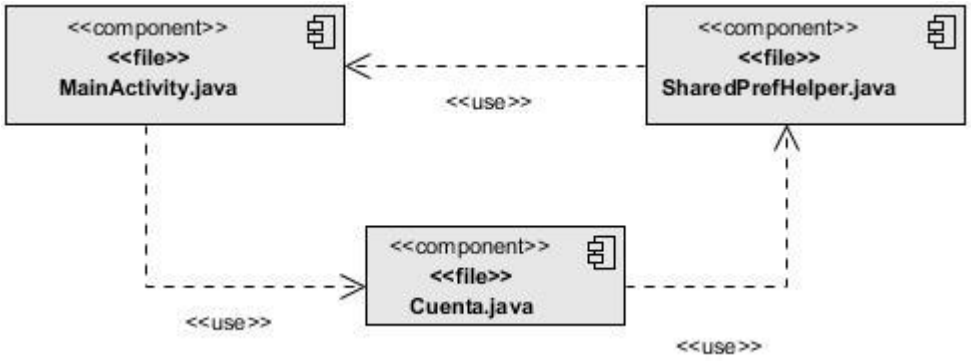


Figura 11 Diagrama de componentes para el CU Gestionar cuenta. Fuente: Elaboración Propia

MainActivity: es la clase principal encargada de realizar todas las acciones dentro de la aplicación.

Cuenta: se encarga de recoger todos los parámetros para establecer la conexión con el servidor nauta.

SharedPrefHelper: esta clase se encarga de guardar los valores de registros de la cuenta nauta.

3.1.2 Diagrama de Despliegue

El diagrama de despliegue modela la topología del *hardware* sobre el que se ejecuta un sistema, el cual muestra la configuración de los nodos que participan en la ejecución de los componentes que residen en ellos. Además, representa el despliegue físico de un componente. La misma será desplegada en un Smartphone con SO Android. En la Figura 12 se muestra el diagrama de despliegue para el sistema (29).

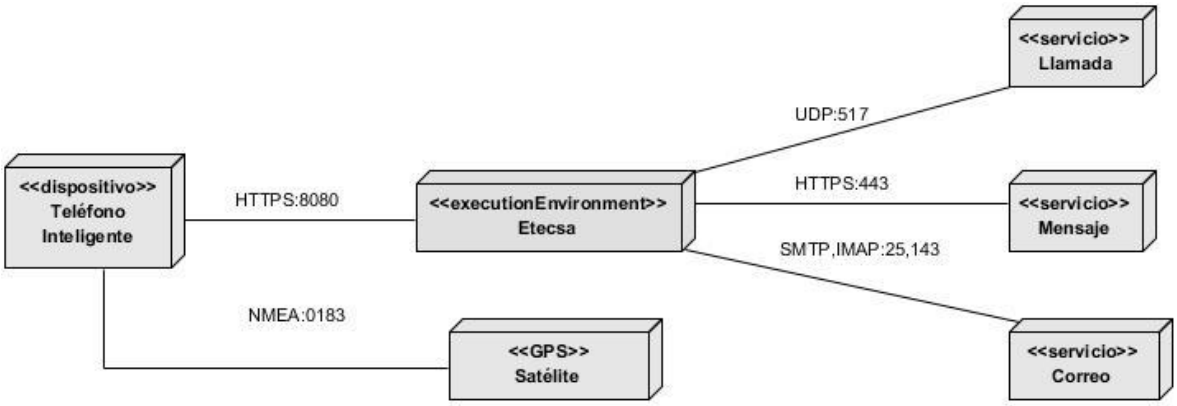


Figura 12 Diagrama de Despliegue. Fuente: Elaboración Propia

El nodo Teléfono Inteligente se conectará utilizando el protocolo de conexión segura HTTPS por el puerto 8080 con el nodo Etecsa el cual establecerá la conexión con los nodos Llamada, Mensaje y Correo en

dependencia del servicio que solicite el cliente, esto se realizará utilizando los puertos y protocolos definidos para cada servicio, además dicho teléfono podrá establecer comunicación utilizando el protocolo NMEA por el puerto 0183 para establecer la conexión con el servicio de GPS.

3.2 Estilos de programación

Un estilo de programación es un término que describe convenciones para escribir código fuente en ciertos lenguajes de programación, además de brindarle una mayor legibilidad al código. En la implementación de la aplicación se utilizaron diferentes estilos que se describen de la manera siguiente:

- **Número de declaraciones por línea**: se debe declarar cada variable en una línea distinta, de esta forma cada variable se puede comentar por separado (ver Figura 13).

  Ejemplo:

```
private static final int PICK_CONTACT = 0;
private ArrayList<ContactData> chat;
private ArrayList<ContactData> activos;
private ArrayList<PlaceholderFragment> fragments;
```

Figura 13 Declaración por línea. Fuente: Elaboración propia

- **Espacio en blanco:** se debe usar una línea en blanco entre: métodos, variables locales de un método y la primera sentencia, entre diferentes secciones lógicas dentro de un fichero (ver Figura 14).

  Ejemplo:

```java
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();

    if (id == R.id.nav_bomberos) {
        String rutaMap = sharedPrefHelper.getRutaMap();

        if (rutaMap.isEmpty()) {
            Toast.makeText(this, "Debe especificar ubicación del mapa", Toast.LENGTH_SHORT).show();
            Intent inten = new Intent(this, MainActivity.class);
            startActivity(inten);

        } else {
            Intent pasar = new Intent(this, MapActivity.class);
            startActivity(pasar);
        }
```

Figura 14 Espacio en blanco. Fuente: Elaboración propia

- **Asignación de nombres:** cada tipo de elemento debe nombrarse con una serie de reglas determinadas (ver Figura 15*)*.

  Clases e interfaces: la inicial en mayúscula ya sea simple o compuesta su nombre.

  Ejemplo

```java
public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {
```

Figura 15 Asignación de nombres (clases e interfaces). Fuente: Elaboración propia

Métodos: la primera letra de la primera palabra en minúsculas, el resto de las palabras empiezan por mayúsculas (ver Figura 16).

Ejemplo:

```java
private void mostrarNumero(final int requestCode, final String numero) {
    AlertDialog.Builder dialogo = new AlertDialog.Builder(MainActivity.this);
    dialogo.setTitle("Desea guardar este contacto para URGENCIAS");
    dialogo.setMessage(numero);
```

Figura 16 Asignación de nombres (Métodos). Fuente: Elaboración propia

Variables: Deben comenzar por minúscula. No se utilizará en ningún caso el carácter "_" (ver Figura 17).

Ejemplo:

```
FloatingActionButton fbLLamada;
FabSpeedDial fabSpeedDial;
ImageView bt_cancel;
```

Figura 17 Asignación de nombres (Variables). Fuente: Elaboración propia

3.3 Interfaces del Sistema

A continuación, se muestran algunas de las interfaces del sistema.

La Interfaz Principal (ver Figura 18), muestra los servicios de Llamadas y Mensajes que pueden ser utilizados en la aplicación.

La interfaz de Logueo (ver Figura 19), muestra un formulario con los campos que el usuario debe llenar. Una vez autenticado el usuario puede hacer servicio de los mensajes instantáneos utilizando los datos móviles.

Para los servicios de Llamadas y Mensajes (ver Figura 20) el usuario debe definir contactos para casos de emergencia a cada uno de estos servicios.

La aplicación es capaz de mostrar y ubicar en un mapa los puntos para casos de emergencia más cercanos, dígase Bomberos, Policías y Hospitales (ver Figura 21).

Figura 18 Interfaz Principal. Fuente: Elaboración Propia

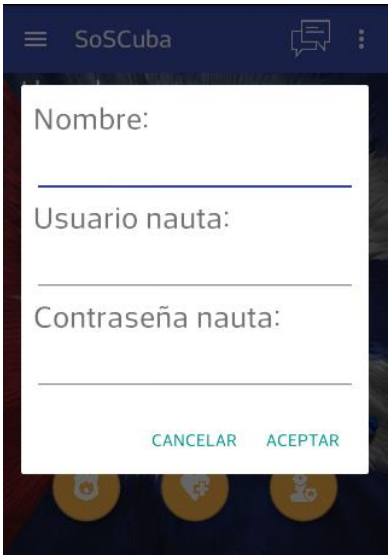


Figura 19 Interfaz de Logueo. Fuente: Elaboración Propia

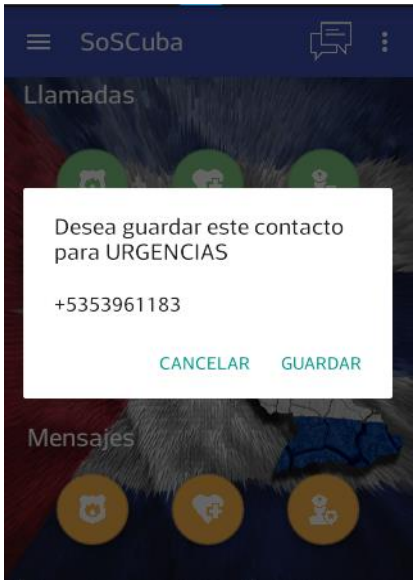


Figura 20 Interfaz para contacto. Fuente: Elaboración Propia

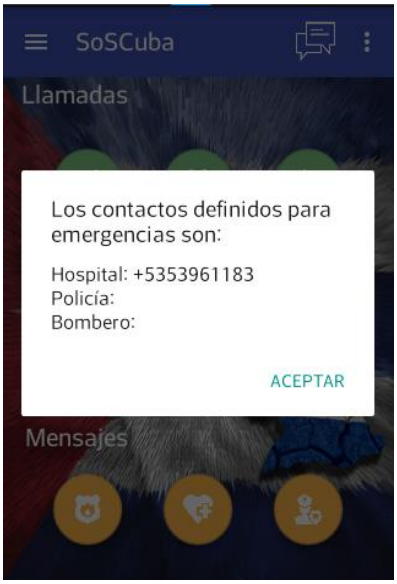


Figura 21 Interfaz para contacto de urgencias. Fuente: Elaboración Propia

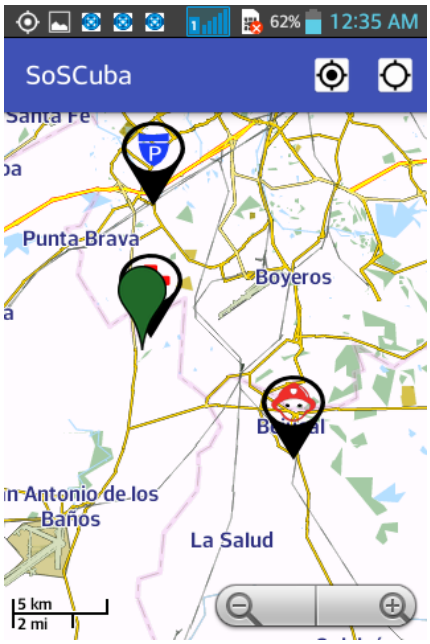


Figura 22 Interfaz del Mapa

3.4 Pruebas del software

Las pruebas de *software* son un elemento crítico para la garantía de la calidad del *software* y representan una revisión final de las especificaciones, del diseño y de la codificación. El objetivo fundamental de las pruebas es descubrir diferentes clases de errores con la menor cantidad de tiempo y de esfuerzo. Aunque las pruebas no pueden asegurar la ausencia de defectos; sí pueden demostrar que existen defectos en el *software* (31).

Estas actividades se planean con anticipación y se realizan de manera sistemática. Cuando se aplican pruebas a un *software* es necesario tener en cuenta el objetivo que se persigue, debido a que las pruebas son agrupadas por niveles que se encuentran en distintas etapas del proceso de desarrollo.

La metodología que guía el proceso de desarrollo de la presente investigación propone realizar pruebas unitarias, la prueba de las 6 M's y de aceptación.

3.4.1 Estrategias de prueba

Una estrategia de prueba del *software* integra los métodos de diseño de casos de prueba en una serie bien planteada de pasos que va a converger en la eficaz construcción del *software*. Dicha estrategia debe ser lo suficientemente flexible como para promover un enfoque personalizado, y al mismo tiempo lo adecuadamente rígido como para originar una planeación razonable y un seguimiento administrativo del avance del producto (32).

Se plantean las siguientes estrategias de pruebas:

- Pruebas unitarias
- Prueba de las 6 M's
- Pruebas de aceptación

Pruebas unitarias

Se puede definir el concepto de prueba de unidad o unitaria como la actividad de probar el funcionamiento de un módulo *software* (código) con el objetivo de lograr su correcto funcionamiento (33).

Para comprobar que cada sentencia de código se ejecuta al menos una vez, se realizaron pruebas al código de las funcionalidades más complejas desde el punto de vista de la programación en cada uno de los paquetes. Para ello se empleó el método de caja blanca con la técnica del Camino Básico; a partir de la obtención de la medida de la complejidad de un procedimiento o algoritmo y la obtención de un conjunto

básico de caminos de ejecución de este, los cuales son utilizados para obtener los casos de prueba.
Para obtener los casos de prueba a partir de la técnica seleccionada se debe construir el grafo de flujo correspondiente al código de la funcionalidad. Luego se determina la complejidad ciclomática V (G) del grafo resultante, la cual es un indicador del número de caminos independientes que existen en un grafo, es decir, es cualquier camino dentro del código que introduce por lo menos un nuevo conjunto de sentencias de proceso o una nueva condición.

La prueba de unidad que se realiza hace uso del método de caja blanca y de la técnica del camino básico. El método del camino básico permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño procedural y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

Tabla 4 Técnica camino básico funcionalidad onOptionsItemSelected. Fuente: Elaboración Propia.

(1)	public boolean onOptionsItemSelected(MenuItem item)
(2)	int id = item.getItemId();
(3)	Log.e("id", "" + id);
(4)	if (id == R.id.*ruta_mapa*) {
(5)	Intent pasar = new Intent(this, ExplorerActivity.class);
(6)	startActivity(pasar); }
(7)	if (id == R.id.*action_chat*) {
(8)	Intent intent = new Intent(this, MainTagget.class);
(9)	startActivity(intent);}
(10)	return super.onOptionsItemSelected(item);

A continuación, se muestra un grafo que representa el flujo del código para el método ***onOptionsItemSelected ()*** (ver Figura 23).

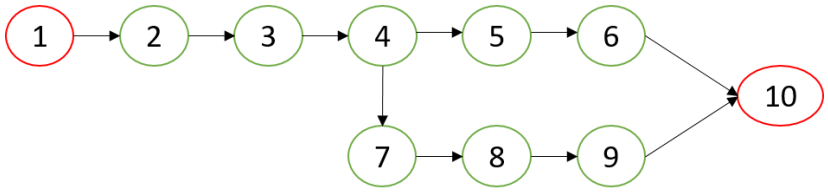


Figura 23 Grafo de flujo del código del método onOptionsItemSelected (). Fuente: Elaboración Propia

La complejidad ciclomática puede ser calculada de 3 formas:

- V (G) = a - n + 2, siendo a el número de arcos o aristas del grafo y n el número de nodos.
- V (G) = r, siendo r el número de regiones cerradas del grafo.
- V (G) = c + 1, siendo c el número de nodos de condición.
- V (G)=r=2

Por lo que el conjunto de caminos básico sería:

- Camino Básico *1:*1-2-3-4-5-6-10
- Camino Básico *2:*1-2-3-4-7-8-9-10

Luego se definen los casos de prueba para cada uno de los caminos básicos obtenidos. A continuación, se muestra el resultado de las pruebas aplicadas a los caminos básicos 1 y 2.

Tabla 5 CP para el Camino Básico 1. Fuente: Elaboración Propia

Caso de Prueba de Unidad	
No. ruta: 1	**Ruta:** 1-2-3-4-5-6-10
Nombre de la persona que realiza la prueba: Yaiselis Ramírez Mastrapa	
Descripción de la prueba: El usuario escoge la opción Cargar Mapa.	
Entrada: Se envía como parámetro el id del ítem para seleccionar la opción.	
Resultado esperado: El sistema carga el mapa y envía una notificación con la ruta.	
Evaluación de la Prueba: Satisfactoria. Se obtuvo la notificación esperada.	

Tabla 6 CP para el Camino Básico 2. Fuente: Elaboración Propia

Caso de Prueba de Unidad	
No. ruta: 2	**Ruta:** 1-2-3-4-5-6-10
Nombre de la persona que realiza la prueba: Tte.Cor Orlando Perdomo Cruz	
Descripción de la prueba: El usuario escoge la opción mensajes.	
Entrada: Se envía como parámetro el id del ítem para seleccionar la opción.	
Resultado esperado: El sistema muestra la interfaz de la sala de chat.	
Evaluación de la Prueba: Satisfactoria. El sistema muestra la interfaz correspondiente.	

Se realizaron tres iteraciones de la prueba unitaria. En la primera iteración se detectaron 4 no conformidades; en la segunda, 1 no conformidad la cual fue resuelta para la tercera iteración. Las no conformidades detectadas estaban asociadas a errores de validación.

Prueba de las 6'M

De las 6 M's se extrae la concepción de que las aplicaciones móviles deben garantizar el cumplimiento de las necesidades de los usuarios y al mismo tiempo generen ingresos. Las 6 M's debe su nombre a los seis atributos que se miden para evaluar el éxito del servicio propuesto: *Movement* (Movimiento), *Moment* (Momento), *Me* (Yo), *Multi-user* (Multiusuario), *Money* (Dinero) *y Machines* (Máquinas).
Una vez aplicada esta prueba, se obtuvieron los siguientes resultados:

Tabla 7. Prueba de software 6'M. Fuente: Elaboración Propia

Variables	Calificación	Resultado
Movement (Movimiento)	4	El usuario tiene acceso al servicio en cualquier lugar del país, no siendo así donde no hay cobertura. (Atributo: ubicación)
Moment (Momento)	5	El usuario puede hacer uso de la aplicación en el momento de la ocurrencia del problema posibilitando una disminución del tiempo de respuesta. Y puede hacer uso de la geolocalización en cualquier momento. (Atributo: tiempo)
Me (yo)	5	Es una aplicación personalizada ya que el usuario maneja sus contactos y hace

		uso de la misma por una cuenta personal. (Atributo: servicio personalizado)
Multi-user (multi-usuario)	5	Es una aplicación multi-usuario, cada usuario puede interactuar con otros usuarios, no existe administrador, ni tipos de usuario. (Atributo: social)
Money (dinero)	3	Dicha aplicación no es libre de costo para el usuario pues hace uso de los servicios de Etecsa, lo que comparado con mandar SMS el costo disminuye 8 centavos. Además, puede enviar el SMS tradicional. (Atributo: Ingresos)
Machines (máquina)	4	La interfaz gráfica es amigable y entendible por los usuarios por lo que puede ser fácil su uso. Lo que el uso de la APK depende de que el usuario tenga comprada una cuenta nauta. (Atributo: tecnología)

Las 6 M's afirman que cualquier servicio que brinde un gran valor en cualquiera de las 6 m tiene un buen potencial para el éxito como servicio móvil. Analizando los atributos se puede decir que una vez instalada la aplicación se puede hacer uso de ella en cualquier lugar que exista cobertura, permitiendo el ahorro de tiempo una vez activado la opción de ubicar en caso de emergencia, además de disminuir el costo de los mensajes con la utilización de los datos móviles, también posibilita que el usuario sea quien gestione sus propios contactos y muestra una interfaz sencilla lo que contribuye a que su uso sea fácil. Por lo que atendiendo a los resultados de la tabla antes expuestos y teniendo en cuenta que las calificaciones fueron superiores a 3 se puede concluir que la aplicación tiene resultados satisfactorios.

Prueba de Aceptación

Es la prueba planificada y organizada formalmente para determinar si se cumplen los requisitos de aceptación marcados por el cliente. Sus características principales son las siguientes (34):

- Participación del usuario.
- Está enfocada hacia la prueba de los requisitos de usuarios especificados.

- Está considerada como la fase final del proceso para crear una confianza en que el producto es el apropiado para su uso en explotación.

Para representar las pruebas de aceptación se definieron los siguientes elementos:

- **Código:** Representa al caso de prueba, incluye el número del CU, de la prueba y si posee diferentes escenarios.
- **CU:** Nombre del CU al cual pertenece.
- **Nombre:** Junto al código conforma el identificador del caso de prueba.
- **Descripción:** Acción que debe realizar el sistema.
- **Condiciones de ejecución:** Describe las características y elementos que debe contener el sistema para realizar el caso de prueba.
- **Entrada/Pasos de Ejecución:** Incluye las entradas necesarias para realizar el sistema, además de los pasos para realizar el caso de prueba.
- **Resultados Esperados:** Descripción de la respuesta del sistema ante el caso de prueba.
- **Evaluación de la prueba:** Clasificación de la prueba en satisfactoria o insatisfactoria.

La siguiente tabla muestra las pruebas de aceptación del CU1 perteneciente a la primera iteración de sistema. La cual fue escogida por ser una de los procesos más relevantes en el desarrollo del sistema.

Tabla 8. Caso de prueba Gestionar cuenta. Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Código Caso Prueba **SoSCuba-1-1**	**CU: Gestionar cuenta**
Nombre: Insertar cuenta	
Nombre de la persona que realiza la prueba: Yaiselis Ramirez Mastrapa	
Descripción de la Prueba: Comprobar que se inserte una nueva cuenta nauta verificando que no existe, en caso contrario mostrar un mensaje de alerta “Cuenta existente”.	

Condiciones de Ejecución: El usuario debe comprobar que se inserte la cuenta utilizando caracteres válidos como son letras y números para el nombre y el usuario, y para la contraseña una combinación de números, letras y otros caracteres. El usuario y la contraseña de la cuenta nauta no debe exceder de 32 caracteres, ni puede comenzar con comillas (',") y and per se and (&). Las cuenta nauta debe tener como dirección fija *ejemplo**@nauta.com.cu.***
Entradas/ Pasos de Ejecución: - Seleccionar en el menú la casilla que dice "Nombre de la cuenta" y especificar el nombre de la cuenta que el usuario desea para identificarse. - Seleccionar en el menú la casilla que dice Cuenta Nauta y crear la cuenta nauta. - Seleccionar en el menú la casilla que dice Contraseña y establecer la contraseña.
Resultado esperado: El usuario creó la cuenta nauta y se notifica el éxito de la creación de la cuenta nauta.
Evaluación de la prueba: Prueba satisfactoria

Tabla 9. Caso de prueba Enviar mensaje predeterminado. Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Código Caso Prueba **SoSCuba-2-2**	**CU: Enviar mensaje predeterminado**
Nombre: Enviar mensaje predeterminado.	
Nombre de la persona que realiza la prueba: Yaiselis Ramirez Mastrapa	
Descripción: El usuario debe comprobar que el contacto predefinido existe.	
Condiciones de Ejecución: El usuario debe comprobar que el mensaje ha sido enviado.	
Entradas/ Pasos de Ejecución: - Seleccionar de la pantalla principal el icono de mensaje. - Luego seleccionar de la vista posterior mensaje predeterminado.	

- Se envía el mensaje predeterminado al contacto que ha sido predefinido en la configuración de la aplicación.
Resultado esperado: La prueba se ha cumplido satisfactoriamente.
Evaluación de la prueba: Prueba satisfactoria

Tabla 10. Caso de prueba Realizar llamada predeterminada. Fuente: Elaboración Propia

Caso de Prueba Aceptación	
Código Caso Prueba SoSCuba-3-3	**CU: Realizar llamada predeterminada**
Nombre: Realizar llamada predeterminada	
Nombre de la persona que realiza la prueba: Tte.Cor. Orlando Perdomo Cruz	
Descripción: El usuario debe comprobar que para realizar las llamadas de emergencia debe estar en un lugar con cobertura.	
Condiciones de Ejecución: El usuario debe comprobar que se ha realizado la llamada.	
Entradas/ Pasos de Ejecución: - Seleccionar de la vista principal el icono llamada. - Seleccionar de la vista siguiente que el usuario quiere llamar a los bomberos, SIUM o PNR.	
Resultado esperado: La prueba se ha cumplido satisfactoriamente.	
Evaluación de la prueba: Prueba satisfactoria	

Diseño de Casos de Prueba

Pruebas del sistema	CU	Iteración	NC	Cerrada	No Procede
		1ra	5	5	0

	7	2da	2	2	0
		3ra	1	1	0

Una vez realizados los casos de Pruebas para un total de 7 Casos de Uso, con tres iteraciones se eliminaron las 7 no conformidades (ver Figura 24).

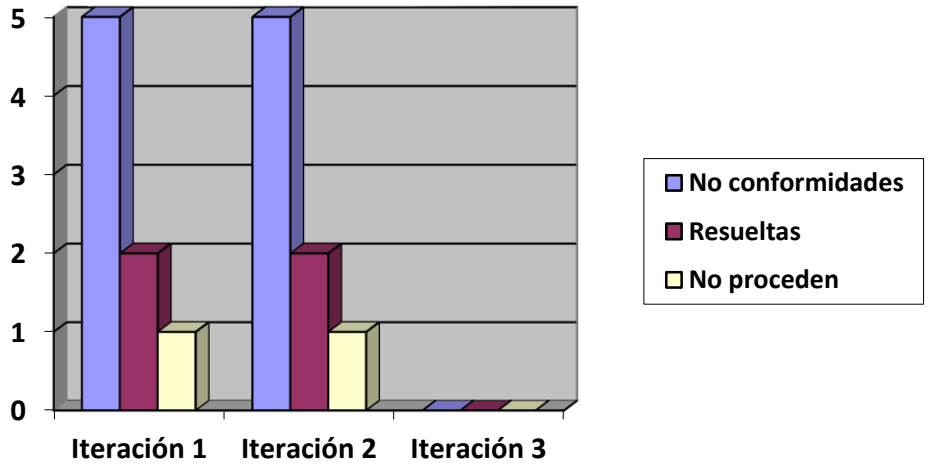


Figura 24 Pruebas del Sistema. Fuente: Elaboración propia

Como resultado se detectó que los errores encontrados pueden clasificarse según su tipo:

- Para una cantidad de cinco No Conformidades se revelan errores de Presentación porque no se visualiza correctamente toda la información que se desea mostrar.
- Para las siguientes dos No Conformidades los errores fueron de Redacción porque se localizaron varios errores ortográficos siendo estos generalmente de acentuación.
- Para la restante No conformidad el error detectado fue de Ejecución, durante el despliegue del sistema fueron detectadas diferentes fallas, siendo estas ocasionadas porque el sistema no realizaba todos los procedimientos definidos.

3.5 Conclusiones Parciales

Tras definir las características de implementación y pruebas de la propuesta de solución, se concluye que se realizaron los diagramas de componentes y despliegue para el modelado de la estructura general del sistema y de una topología de *hardware* donde se ejecuta el mismo. Además, se diseñaron los casos de prueba para rectificar a tiempo defectos que pudiera presentar el sistema y se validó la solución propuesta a través de las pruebas de sistema y aceptación las que permitieron evaluar el funcionamiento del sistema.

Conclusiones generales

Con la realización de este trabajo se desarrolló una Aplicación Android para apoyar la estrategia de respuesta ante incidentes de peligrosidad en Cuba. De esta forma se da cumplimiento al objetivo propuesto al inicio de la investigación, además se comprobó que:

- Las relaciones existentes entre los principales conceptos asociados al dominio de la presente investigación, permitieron una mayor comprensión de la propuesta de solución.

- El análisis de las soluciones existentes permitió determinar las características que constituyen la base para el diseño de las funcionalidades que se definen en la propuesta de solución.

- La elaboración de los artefactos propuestos por la metodología de desarrollo y la especificación de requisitos permitieron un mejor entendimiento de la aplicación que se propone desarrollar, así como las características de la misma.

- La realización de las pruebas al sistema, permitió apreciar que todas en su gran medida fueron satisfactorias, lográndose de esta manera el cumplimiento de los requisitos exigidos inicialmente y la validación de la solución propuesta mediante la completa satisfacción del cliente.

Recomendaciones

Se recomienda:

- Implementar la funcionalidad trazar ruta para resaltar en el mapa cual es el recorrido más corto para llegar a los bomberos, policías y hospitales.
- Agregarle al mensaje predeterminado la dirección del ciudadano.
- Internacionalizar la aplicación utilizado otros lenguajes para que los ciudadanos de otras naciones puedan hacer uso de la misma en Cuba.

Referencias Bibliográficas

1. **Visual Paradigm. [En línea] [Citado el: 11 de noviembre de 2014.] http://es.scribd.com/doc/65619842/Visual-Paradigm.**
2. Laman, Craig. *UML y Patrones. Introducción a1 análisis y diseño orientado a objetos.* México : PRENTICE HALL, 1999.
3. SQLite. [En línea] [Citado el: 23 de 2 de 2017.] http://www.sqlite.org/.
4. PRESSMAN, Roger. *Software Engineering: A practitioner´s Approach. .* New York : Seventh, 2010.
5. Hidalgo, Ángela Olga. *Servicio de Urgencia.* Cuba : s.n., 2012.
6. Secretaria de Seguridad Publica. *Secretaria de Seguridad Publica.* [En línea] http://www.ssp.df.gob.mx/mi-policia.html.
7. Secretaria de Gobernación.Comisión Nacional de Seguridad. *Secretaria de Gobernación.Comisión Nacional de Seguridad.* [En línea] 2013. http://www.cns.gob.mx/portalWebApp/wlp.c?__c=18e76.
8. *Redes y Comunicación.* Cajamarca, Perú : s.n., 2017, Academia.
9. Real Academia Española. *Real Academia Española.* [En línea] 2017. http://dle.rae.es/?id=GjqhajH.
10. Simón, Jesús García. *Prioridades para el país.* Cuba : s.n., 2016.
11. Portal web de la Unión de Empresas Constructoras Caribe, UNECA S.A. [En línea] [Citado el: 10 de diciembre de 2014.] http://www.uneca.com.cu.
12. Patrones de Casos de Uso. *Patrones de Casos de Uso.* [En línea] [Citado el: 16 de 3 de 2017.] https://sg.com.mx/revista/6/patrones-casos-uso.
13. P. Letelier, J.H.Canós y C.Penadés. Metodologías Ágiles en el Desarrollo de Software. *Metodologías Ágiles en el Desarrollo de Software.* [En línea] 2013.
14. Mantill, Maira Cecilia Gasca. *Methodology for mobile application development.* Colombia : s.n., 2015.
15. Java. *Java.* [En línea] Oracle. https://www.java.com/es/download/faq/java8.xml.
16. SOMMERVILLE, Ian. *Ingeniería del software.* s.l. : 7ma, 2005. ISBN 84-7829-074-5.
17. PRESSMAN, Roger. *Ingeniería de software: Un Enfoque Práctico.* New York.Estados Unidos : Sexta, 2005.
18. Pressman, Roger S. *Ingeniería de software.Un enfoque práctico.*
19. http://www.sld.cu/. *Centro Nacional de Información de Ciencias Médicas.* [En línea] Infomed, 1999-2016. [Citado el: 6 de Diciembre de 2016.] http://web.archive.org/web/20090114065904/http://www.dne.sld.cu/minsap/.
20. Cristiá Álvares, Aldo, y otros. Guía para la personalización de PostgreSQL 8.4. [En línea] [Citado el: 12 de noviembre de 2014.] http://postgresql.uci.cu/?page_id=21.
21. Git. *Git.* [En línea] 2017. https://git-scm.com/book/es/v1/Empezando-Acerca-del-control-de-versiones.
22. *Estructura de las Aplicaciones Orientadas a Objetos.El patrón Modelo-Vista-Controlador (MVC).* Fernández Romero, Yenisleidy. Habana : s.n., Revista Digital de las Tecnologías de la Información y las Comunicaciones.

23. Jacobson, Ivar, Booch, Grady y Rumbaugh, James. ***El Proceso Unificado de Desarrollo de Software.*** **España : Pearson Educación, 2000.**
24. El Mundo. [En línea] 2014. [Citado el: 8 de Diciembre de 2016.] http://www.elmundo.es/tecnologia/2016/03/03/56d85088268e3ea0338b4670.html.
25. El Impulso.com. ***El Impulso.com.*** **[En línea] 2017. http://www.elimpulso.com/noticias/nacionales/policia-de-bolsillo-aplicacion-venezolana-contra-la-inseguridad.**
26. EcuRed. [En línea] [Citado el: 21 de 2 de 2017.] https://www.ecured.cu/Comunidad.
27. Cubadebate. ***Cubadebate.*** **[En línea] 2014. [Citado el: 8 de diciembre de 2016.] http://www.cubadebate.cu/noticias/2012/11/18/servicios-de-urgencia-medica-una-prioridad-cubana/.**
28. Rodriguez, Elena Valdés. ***Comportamiento y primeros auxilios.*** **Cuba : s.n., 2014.**
29. Colombia.com. ***Colombia.com.*** **[En línea] 2017. http://www.colombia.com/tecnologia/noticias/sdi/33785/cuadrantes-la-app-movil-de-seguridad-en-territorio-colombiano.**
30. BastaYaPR. ***BastaYaPR.*** **[En línea] 2017. http://www.bastayapr.org/.**
31. Información, Oficina Nacional de Estadística e Información. ***Anuario Estadístico de Cuba 2015.*** **Cuba : s.n., 2015.**
32. ANDROIDPIT. [En línea] [Citado el: 22 de 2 de 2017.] http://www.androidpit.es/sdk-android.
33. Android Studio. ***Android Studio.*** **[En línea] https://developer.android.com/studio/intro/index.html?hl=es-419.**
34. Jacobson, Ivar, Grady, Booch y Jame, Rumaugh. ***El lenguaje Unificado de Modelado. Manual de referencia.***

Anexo 1 Entrevista al cliente

¿Cuál es el procedimiento establecido por el MININT para dar respuesta, ante incidentes de peligrosidad?

¿Cómo cree que un dispositivo móvil puede apoyar la estrategia establecida por el MININT para socorrer personas ante incidentes de peligrosidad?

¿Cree que una aplicación móvil puede agilizar el tiempo de auxilio ante un incidente de peligrosidad? En caso de responder si, explique ¿cómo?

¿Cuáles son los incidentes de peligrosidad que más afectan a la población?

¿Cuáles son los incidentes de peligrosidad que les gustaría tratar en una solución móvil? ¿Qué características usted cree que no pueden faltar en esa aplicación móvil?

¿Usted cree que una herramienta que permita ubicarse y ubicar los lugares que brindan servicios de urgencia ayude al ciudadano a socorrerse con mayor facilidad? En caso de responder si, diga ¿cómo?

Anexo 2 Casos de Prueba

Tabla 11 CP Gestionar Cuenta Nauta-Modificar. Fuente: Elaboración Propia.

Caso de Prueba Aceptación	
Código Caso Prueba **SoSCuba-1-2**	**CU: Gestionar Cuenta Nauta**
Nombre: Modificar Nombre Cuenta	
Nombre de la persona que realiza la prueba: Yaiselis Ramirez Mastrapa	
Descripción: Comprobar que se modifique el nombre que identifica la cuenta nauta.	
Condiciones de Ejecución: El usuario debe comprobar que se modifique el nombre que identifica la cuenta nauta utilizando solamente letras y números, El nombre de la cuenta nauta no debe exceder de 32 caracteres, ni puede comenzar con comillas (',") y and per se and (&).	
Entradas/ Pasos de Ejecución: - Seleccionar en el menú la casilla que dice "Nombre Nuevo" y modificar el nuevo nombre que identifica la cuenta nauta.	
Resultado esperado: El usuario modificó el nombre.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 12 CP Gestionar Cuenta Nauta-Eliminar. Fuente: Elaboración Propia.

Caso de Prueba Aceptación	
Código Caso Prueba **SoSCuba-1-3**	**CU: Gestionar Cuenta Nauta**
Nombre: Eliminar cuenta nauta	
Nombre de la persona que realiza la prueba: Yaiselis Ramirez Mastrapa	
Descripción: Comprobar que se elimine la cuenta nauta que el usuario no desea utilizar, se muestra un cartel: ¿" Está seguro que desea eliminar su cuenta nauta"?, en caso afirmativo, se muestra el siguiente	

cartel: “Cuenta eliminada satisfactoriamente”.
Condiciones de Ejecución: El usuario debe comprobar que se elimina la cuenta nauta.
Entradas/ Pasos de Ejecución: - Seleccionar del menú lateral la opción Eliminar Cuenta Nauta. - Se muestra un mensaje de información: ¿” Está seguro que desea eliminar su cuenta nauta”? - Se muestran dos opciones Aceptar y Cancelar. - En caso de Aceptar se muestra el siguiente cartel: “Cuenta eliminada satisfactoriamente”. - En caso de Cancelar no se elimina la cuenta.
Resultado esperado: El usuario eliminó la cuenta nauta.
Evaluación de la prueba: Prueba satisfactoria

Tabla 13 CP Gestionar Contacto-Insertar. Fuente: Elaboración Propia.

Caso de Prueba Aceptación	
Código Caso Prueba SoSCuba-4-1	**CU: Gestionar Contacto**
Nombre: Insertar contacto	
Nombre de la persona que realiza la prueba: Tte.Cor. Orlando Perdomo Cruz	
Descripción: El usuario debe comprobar que el contacto que ha seleccionado tiene correo nauta.	
Condiciones de Ejecución: El usuario debe comprobar que el o los contacto(s) ha sido agregado.	
Entradas/ Pasos de Ejecución:	

- Seleccionar de la pantalla principal el icono mensaje. - Seleccionar de la segunda pantalla el icono de robo. - Seleccionar el campo Contacto. - Presionar el botón Adicionar Contacto. - Seleccionar el contacto.
Resultado esperado: La prueba se realizó satisfactoriamente.
Evaluación de la prueba: Prueba satisfactoria

Tabla 14 CP Gestionar Contacto-Eliminar. Fuente: Elaboración Propia.

Caso de Prueba Aceptación	
Código Caso Prueba **SoSCuba-4-2**	**CU: Gestionar Contacto**
Nombre: Eliminar contacto	
Nombre de la persona que realiza la prueba: Tte.Cor. Orlando Perdomo Cruz	
Descripción: Una vez seleccionado el contacto se muestra el siguiente cartel: ¿" Está seguro que desea eliminar el contacto"? En caso afirmativo se elimina el contacto satisfactoriamente, en caso contrario se mantienen los contactos.	
Condiciones de Ejecución: El usuario debe comprobar si se ha eliminado o no el contacto.	
Entradas/ Pasos de Ejecución: - Dejar presionado contacto a eliminar. - Se muestra un cartel de confirmación. - Seleccionar Eliminar.	

Resultado esperado: La prueba se ha cumplido satisfactoriamente.
Evaluación de la prueba: Prueba satisfactoria

Anexo 3 Descripción de los casos de uso del Sistema

<table>
<tr><td colspan="2">Objetivo</td><td colspan="3">Permitir realizar una llamada predeterminada a los Bomberos, Hospital o la PNR.</td></tr>
<tr><td colspan="2">Actores</td><td colspan="3">Usuario</td></tr>
<tr><td colspan="2">Resumen</td><td colspan="3">El caso de uso se inicia cuando el usuario decide realizar una llamada en caso de emergencia a los Bomberos, Hospital o PNR.</td></tr>
<tr><td colspan="2">Complejidad</td><td colspan="3">Alta</td></tr>
<tr><td colspan="2">Prioridad</td><td colspan="3">Crítico</td></tr>
<tr><td colspan="2">Precondiciones</td><td colspan="3">Usuario tiene cobertura.</td></tr>
<tr><td colspan="2">Postcondiciones</td><td colspan="3">Se realiza la llamada de emergencia.</td></tr>
<tr><td colspan="2">Objetivo</td><td colspan="3">Permitir realizar una llamada predeterminada a los Bomberos, Hospital o la PNR.</td></tr>
<tr><td colspan="5">Flujo de eventos</td></tr>
<tr><td colspan="5">Flujo básico.</td></tr>
<tr><td></td><td colspan="2">Actor</td><td colspan="2">Sistema</td></tr>
<tr><td></td><td colspan="2">1. El usuario desea llamar en caso de emergencia a los contactos que tiene definidos para ese caso.</td><td colspan="2">1.1 - Seleccionar de la vista principal el icono llamada.
- Seleccionar de la vista siguiente el contacto (Bomberos, Hospital o PNR), al que el usuario desea hacerle la llamada en caso de emergencia. Finalizando así el caso de uso.</td></tr>
<tr><td colspan="2" rowspan="2">Relaciones</td><td>CU Incluidos</td><td colspan="2"></td></tr>
<tr><td>CU Extendidos</td><td colspan="2"></td></tr>
<tr><td colspan="2">Requisitos no funcionales</td><td colspan="3"></td></tr>
</table>

Figura 25 Descripción del CU Realizar llamada predeterminado. Fuente: Elaboración Propia

Anexo 4 Diagrama de clase para el CU Realizar llamada predeterminada

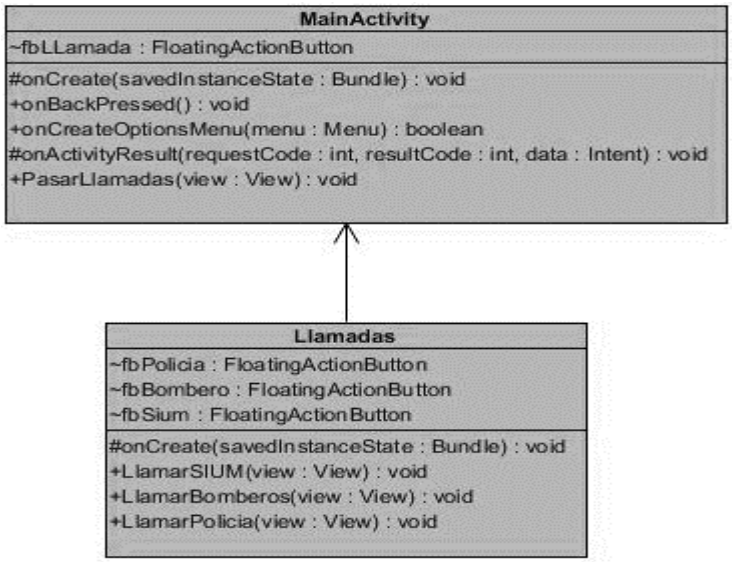


Figura 26 Diseño de clases para el CU Realizar Llamada Predeterminada. Fuente: Elaboración Propia