UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN
FACULTAD DE INGENIERÍA

UCSC

MANTENCIÓN DE UNA FAMILIA DE PRODUCTOS EN EL DOMINIO DE ADMINISTRACIÓN DE EMERGENCIAS

Por

ANA MARÍA MONTERO CÁCERES

MEMORIA PRESENTADA A LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD CATÓLICA DE LA SANTÍSIMA CONCEPCIÓN, PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INFORMÁTICO.

PROFESOR GUÍA: PEDRO ROSSEL CID.

CONCEPCIÓN, CHILE
AGOSTO 2017

Dedicado a Dios, mis padres, mi esposo y mi hija Isabella. También a mi abuelita, mis hermanos, cuñadas, sobrinas y en especial a mi puntito en cielo, quienes han sido una guía en mi camino y sin ellos nada de esto sería posible.

Tabla de contenido

Índice de Tablas

Índice de Ilustraciones

Resumen

En el dominio de la administración de emergencias del Cuerpo de Bomberos, se necesita de mucha coordinación, comunicación y fluidez, entre otros, para la distribución y asignación de recursos ante una emergencia. Es por ello que se ha ido desarrollando en diversos proyectos una aplicación móvil, llamada *MapaMóvil*, que les permita a bomberos reducir el uso de radio para comunicarse y evitar así su saturación.

Con el paso del tiempo, se ha hecho necesario realizar una serie de cambios al proyecto original para que se ajuste cada vez más a las necesidades actuales de bomberos, tales como el uso de una nueva interfaz para obtener una mejor usabilidad; un cambio de nombre a *CollabMap* y la ampliación de la familia de productos existente, fomentando así la reutilización de software. Es esto último, que genera la idea central de este proyecto de título, el cual busca entregar una herramienta integral al cuerpo de bomberos, de tal forma que su uso sea imprescindible ante una emergencia, que para este proyecto se enfocará en los incendios.

Para aumentar la familia de productos de *CollabMap*, se seleccionaron nuevos componentes que fuesen reutilizables y que generen nuevas aplicaciones móviles dependiendo del cargo que desempeñe el bombero, complementando sus funciones adecuadamente ante un incendio. Es así como se continuó con la construcción de estos nuevos componentes, siendo desarrollados e incorporados en base a la aplicación móvil original *CollabMap*.

Las nuevas funcionalidades integradas, sin duda aportan en el crecimiento de la aplicación móvil. Sin embargo, aún quedan varias áreas que explorar como aumentar la cantidad de dispositivos al que está orientada la aplicación según el sistema operativo y en base a las nuevas tecnologías que se siguen desarrollando continuamente y que serán mencionadas en el presente informe.

Summary

In the field of emergency management of the Fire Department, much coordination, communication and fluidity are needed, among others, for the distribution and allocation of resources in the event of an emergency. This is why a mobile application, called Mobile Map, has been developed in various projects, allowing firefighters to reduce the use of radio to communicate and thus avoid saturation.

With the passage of time, it has become necessary to make a series of changes to the original project to be increasingly adjusted to the current needs of firefighters, such as the use of a new interface for better usability; A change of name to *CollabMap* and the extension of the existing product family, thus encouraging the reuse of software. It is the latter, which generates the beginning of this title project, which seeks to provide an integral tool to the fire department, so that its use is essential in an emergency, which for this project will focus on fires.

To increase *CollabMap*'s family of products, new components were chosen that were reusable and generate new mobile applications depending on the position of the firefighter, complementing their functions properly in the event of a fire. This is how we continued with the construction of these new components, being developed and incorporated based on the original mobile application *CollabMap*.

The new integrated functionalities certainly contribute to the growth of the mobile application. However, there are still several areas to explore such as increasing the number of devices to which the application is oriented according to the operating system and based on the new technologies that continue to be developed continuously and that will be mentioned in the present report.

1. Introducción

1.1. Presentación del proyecto

En el dominio de administración de emergencias es imprescindible que las acciones a ejecutar sean en el tiempo menor posible, además de establecer procedimientos predefinidos dependiendo del tipo de emergencia que se desarrolla. Es por esto que nace CollabMap, una aplicación móvil diseñada para colaborar conjuntamente con el Cuerpo de Bomberos ante una emergencia, específicamente los incendios, a modo de facilitar el trabajo desde que es recibida una emergencia en la Central de Bomberos hasta que ésta ha finalizado.

CollabMap es un proyecto que se ha ido desarrollando y modificando de acuerdo a las exigencias actuales de Bomberos y a las actualizaciones en los APIs de Android. En base al estudio de la aplicación original CollabMap, se desarrollaron nuevos componentes de software para ser integrados en la aplicación original y que permitan complementar de mejor forma el trabajo de bomberos ante un incendio. La coexistencia entre los componentes originales y los nuevos fomenta la reutilización de éstos, lo que permite crear nuevas aplicaciones móviles orientadas a los distintos tipos de funciones que posee bomberos. Además, este reuso también permite seguir añadiendo componentes y ajustarse cada vez más a las necesidades de bomberos.

1.2. Objetivos del proyecto

En esta sección se mostrarán el objetivo general que se busca cumplir al finalizar el proyecto y los objetivos específicos que son necesarios para llevar a cabo esto.

1.2.1. Objetivo general

Expandir la familia de productos existentes en el dominio de administración de emergencias, por medio del desarrollo de componentes reutilizables a partir de la aplicación móvil CollabMap.

1.2.2. Objetivos específicos

1) Estudiar los componentes de software desarrollados en el proyecto de título anterior.

2) Establecer nuevos productos en la familia de productos pre-establecida.

3) Desarrollar nuevos componentes de software de tal forma que puedan ser reutilizados posteriormente.

4) Construir una nueva aplicación en base a los nuevos componentes desarrollados.

Debido a que este proyecto de título es una mantención a uno previamente desarrollado, es que estos objetivos fueron logrados en base a un estudio exhaustivo del proyecto de título de Erika Ormeño [15], permitiendo conocer tanto el contexto del problema al que apunta el

proyecto como a la aplicación móvil que será actualizada de acuerdo a los componentes escogidos en base a la línea de productos de software que sigue el presente trabajo de título y que deberán ser integrados a la aplicación móvil original. Para el desarrollo de los componentes mencionados en el párrafo anterior y que llevan también a cumplir el objetivo general, fue necesario un estudio a fondo del lenguaje y el entorno en el que se desarrollaron los componentes, así como también una investigación sobre las temáticas asociadas a este proyecto de título de tal forma de lograr un desarrollo integral.

1.3. Justificación del proyecto

Dentro del Cuerpo de bomberos, existen distintos cargos que se deben ejercer ante una emergencia, por lo que cada uno requiere desempeñar distintas funciones dentro del equipo. Es por esta razón que se precisa ampliar la familia de productos junto al desarrollo de nuevos componentes de software, permitiendo incrementar el número de aplicaciones móviles y abarcar así un mayor número de tareas a resolver dependiendo de lo que el usuario necesite para una mejor ejecución de su cargo en una emergencia.

Por otro lado, *CollabMap* busca facilitar el manejo de las emergencias con estas nuevas aplicaciones, fomentando así la reutilización de código logrando disminuir el tiempo de desarrollo de éstas.

Además, cabe mencionar que el presente proyecto es parte de una mejora continua, en base a la investigación realizada desde hace algunos años, a través de distintos proyectos que se ensamblan con el fin de mantener la aplicación *CollabMap*, para luego ser utilizada por el Cuerpo de Bomberos ante un incendio, de tal forma que su usabilidad les brinde grandes beneficios y simplifique la comunicación entre ellos, además de entregarles información relevante de la emergencia, tales como ubicación de los recursos, recorridos para llegar al lugar de la emergencia visualizados en un mapa, carros bomba disponibles, pedir refuerzos si es necesario, entre otros.

1.4. Delimitación del proyecto

El presente proyecto está orientado a la creación de aplicaciones en el dominio de la administración de emergencias, específicamente en los incendios atendidos por el Cuerpo de Bomberos, a través del desarrollo de nuevos componentes de software.

Utilizando los estudios previos a este proyecto, se busca reutilizar los componentes ya desarrollados para la creación de nuevos, a modo de integrarlos entre sí. Para la generación de componentes, se utilizó el entorno de desarrollo Eclipse junto al plugin ADT y deberán ser operativas en dispositivos móviles Android 2.3.4 o superior.

1.5. Metodología aplicada

La metodología que se aplicó durante el proyecto puede dividirse en 3 etapas que se detallarán a continuación:

1) Etapa de Aprendizaje:

En esta primera etapa, fue necesario un estudio de todo aquello que permitiría llevar a cabo el desarrollo del proyecto:

- Estudio del proyecto de título de Erika Ormeño, lo que permitió entender a *CollabMap* de manera más detallada, de tal forma de integrar las nuevas funcionalidades a lo desarrollado anteriormente.

- Estudio de la base de datos creada por Erika a través de PostgreSQL, lo que permitió entender el funcionamiento de las tablas que ahí se incluyen, cómo se relacionan con las funcionalidades de *CollabMap* para, finalmente, añadir nuevos datos para ser utilizados en los nuevos componentes a desarrollar.

- Estudio del lenguaje de programación Java para Android y el sistema operativo ya mencionado y todos sus requerimientos, con el fin de conocer el ambiente de desarrollo mejorando así las habilidades que permitieron el desarrollo de los nuevos componentes.

- Instalación de las aplicaciones utilizadas en el proyecto de título anterior para desarrollar aplicaciones Android, tales como el kit de desarrollo para Java (JDK), PostgreSQL que fue utilizado originalmente para la base de datos y el complemento ADT para Eclipse, a modo de realizar la instalación de CollabMap para su mantención.

2) Etapa de Identificación:

Esta etapa permitió identificar los nuevos componentes que se debieron desarrollar, en base a las necesidades de bomberos ante un incendio. Estos componentes son:

- Envío y recepción de audio.
- Visualización de ruta hacia la emergencia.

3) Etapa de Desarrollo:

a) Método de desarrollo: El método de desarrollo a aplicar en este proyecto fue el modelo Incremental, que es un modelo que permite obtener versiones o incrementos del sistema en base a las nuevas funcionalidades a desarrollar, así "cada versión añade funcionalidad a la versión anterior" [21], esto es, se trabajó con cada una de las funcionalidades mencionadas en la etapa anterior de identificación por separado, obteniendo un incremento o versión del sistema de cada función para tener luego, como resultado, la versión final del sistema.

Las etapas serán las siguientes para cada versión del sistema: Análisis y definición de requerimientos, Diseño del sistema, Implementación y Pruebas.

Debido a que este proyecto es una mantención a un desarrollo previo, los requerimientos debiesen comprenderse claramente, sin generar cambios relevantes al problema original.

b) Mantenibilidad: Debido que el presente proyecto es una mantención de un proyecto anterior, es importante contar con la documentación necesaria de los componentes que se desarrollaron, con el fin de generar nuevas aplicaciones móviles con funcionalidades diferentes en base a lo que bomberos necesiten.

1.6. Capítulos presentes en el proyecto de título

El presente trabajo de título está organizado según los siguientes capítulos mencionados a continuación:

- **Capítulo 2. Marco Teórico:** Este capítulo trata sobre los tópicos que permitirán comprender el problema y las temáticas asociadas a él, esto es, en base a una aplicación móvil creada con anterioridad junto a su entorno de desarrollo, de tal forma de entregar la información necesaria para dar inicio al desarrollo del proyecto.

- **Capítulo 3. Estado del Arte:** Este capítulo trata sobre la situación actual de la aplicación CollabMap y a su vez, de aquellas aplicaciones similares a ella, con el fin de obtener retroalimentación que permita cumplir con los objetivos de este proyecto.

- **Capítulo 4. Análisis y definición de requerimientos:** Este capítulo trata sobre la definición del problema para posteriormente continuar con su desarrollo. Además, se

dará a conocer las funcionalidades que ya posee la aplicación para luego detallar las nuevas funcionalidades que se implementaron a lo largo del proyecto.

- **Capítulo 5. Diseño del sistema:** En este capítulo se muestra el diagrama de la base de datos creada por Erika Ormeño para la aplicación original. También se muestran los diagramas de flujo de datos que corresponden a las funcionalidades desarrolladas para este proyecto, además de proporcionar las capturas de pantalla de dichas funcionalidades.

- **Capítulo 6. Implementación y Pruebas:** En este capítulo se muestran lo cambios que se realizaron al proyecto original, con el fin de llevar a cabo una actualización del sistema en base a las nuevas funcionalidades a desarrollar. Además, se muestran las pruebas que se hicieron a las nuevas funcionalidades para analizar el funcionamiento de la aplicación en distintos dispositivos móviles.

- **Capítulo 7. Conclusiones:** Este capítulo muestra el análisis del funcionamiento del sistema versus el cumplimiento de los objetivos propuestos para el presente proyecto. Además, se muestran las dificultades obtenidas durante el desarrollo y los trabajos futuros.

2. Marco Teórico

2.1. Reutilización de software

Hoy en día la reutilización, definida como "el proceso de crear sistemas de software a partir de software existente" [18], juega un rol importante al momento del desarrollo de software, esto gracias a que esta técnica proporciona grandes beneficios utilizando activos que fueron creados y probados con anterioridad para la construcción de nuevos sistemas. Dentro de estos beneficios se encuentran:

- Menores costos en el desarrollo y mantención de sistemas.
- Disminución de los riesgos a lo largo del desarrollo.
- Utilización de un número menor de personas en el equipo de trabajo.
- Reducción en el tamaño de la documentación, pues solo se deberá documentar los nuevos desarrollos.
- Entrega del producto o la puesta en venta al mercado es más rápida.

Se destacan dos categorías para poner en práctica la reutilización: Reutilización con desarrollo y Reutilización sin desarrollo [6], las que se detallan a continuación:

- Reutilización con desarrollo: Se refiere a la reutilización durante el desarrollo de un software y se encuentran, entre otros, el enfoque Sistemático, que implica una inversión inicial para obtener beneficios a futuro, creando componentes reutilizables y genéricos para ser utilizados fácilmente, es decir, hay una planificación, y el enfoque Oportunista, en el que se utilizan partes que se ajustan al problema en cuestión, obtenidos de un repositorio en el que fueron almacenados estos componentes que esperan ser usados.

- Reutilización sin desarrollo: Se refiere a la reutilización de un producto, lo que quiere decir que hay reutilización de un sistema completo. En otras palabras, "en vez de construir un sistema específico para un sólo cliente, se construye un sistema más general que puede ser utilizado por muchos clientes", un ejemplo claro de esto es *Windows,* pues es un solo sistema operativo que puede usarse en múltiples configuraciones de hardware.

Algunos ejemplos en los que se utiliza la reutilización, ya sea con o sin desarrollo, son sistemas operativos, compiladores, patrones de diseño, generadores de aplicaciones, línea de productos de software, entre otros.

2.2. Línea de productos de software

Una línea de productos de software (SPL), también conocida como familia de productos, es un método que permite disminuir el tiempo y costos de sistema al momento de desarrollar un software, basándose principalmente en la reutilización. Una línea de productos es "un conjunto de aplicaciones construidas en forma planificada a partir de un conjunto de activos comunes y que satisfacen las necesidades de un segmento del mercado" [2]. Estos activos del software son aquellos componentes que fueron elaborados durante el desarrollo del sistema y que son potencialmente reutilizables. Cabe mencionar que el SPL posee un costo inicial que disminuye al ir construyendo activos que permitan el desarrollo de múltiples aplicaciones [17].

Dentro de un SPL se pueden identificar dos fases, denominadas ingeniería del dominio e ingeniería de la aplicación:

- Ingeniería del dominio: Se refiere al desarrollo y mantención de los activos comunes reutilizables.

- Ingeniería de la aplicación: Se refiere a aquellos productos que son construidos mediante la combinación de los activos ya desarrollados [17].

Algunos de los beneficios de utilizar SPL son que aquellos sistemas grandes podrán ser desarrollados en menor tiempo y esfuerzo que si se realiza de la forma tradicional, aumentando de esta forma la productividad para el desarrollo del software. Además, al basarse en la reutilización, se disminuyen costos de personal pues los activos se desarrollarán una única vez dependiendo de lo que los clientes requieran, es decir, se reutilizarán activos creados anteriormente y solo se desarrollarán aquellas partes nuevas que se requieran para completar el nuevo software. Otra ventaja es que se posee un mejor control de lo desarrollado, pues los activos debiesen ser probados con anterioridad, disminuyendo posibles errores en el nuevo sistema y contribuyendo así a un mejor funcionamiento de éste.

2.3. Patrones de diseño

El patrón de diseño es "una descripción del problema y la esencia de su solución, de forma que la solución se pueda reutilizar en diferentes situaciones. El patrón no es una especificación detallada, antes bien, puede pensarse en él como una descripción del conocimiento y experiencia acumulados. Es una solución adecuada a un problema común" [21].

Existen cuatro elementos esenciales en los patrones de diseño:

1) Un nombre característico del patrón de tal forma de obtener una referencia adecuada de él.
2) Una descripción del entorno detallando a que problemas puede aplicarse.
3) Una descripción de las partes que tienen la solución del diseño, sus relaciones y responsabilidades.

4) Una declaración acerca de aplicar el patrón, pues se debe estar consciente de las consecuencias de utilizarlo en un determinado problema.

Según lo anterior, claramente los patrones de diseño son una manera de reutilizar software, adquiriendo los beneficios que esto otorga como mejorar la productividad y la eficiencia en el desarrollo. Por otro lado, según el catálogo de patrones de diseño, se encuentran disponibles 23 patrones que se distribuyen dentro de los tipos de patrones de creación, estructurales y de comportamiento, de ellos solo se definirán ocho, que son los que conciernen al presente proyecto: Abstract Factory, Factory Method, Singleton, Adapter, Decorator, Bridge, Observer y Template Method [8]:

- *Abstract Factory o Fábrica Abstracta*: Proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas.

- *Factory Method o Método de Fabricación*: Define una interfaz para crear un objeto, pero deja que sean las subclases quienes decidan que clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.

- *Singleton o Único:* Pertenece al tipo creacional al igual que el anterior y busca asegurar que una clase posea una instancia de tal forma de entregar un punto de acceso global a la misma.

- *Adapter o Adaptador*: Convierte la interfaz de una clase en otra interfaz que es la que esperan los clientes. Permite que cooperen clases que de otra forma no podrían por tener interfaces incompatibles.

- *Decorator o Decorador:* Es un tipo de patrón estructural y lo que hace es añadir dinámicamente nuevas responsabilidades a un objeto, de tal forma de proporcionar una alternativa flexible a la herencia ampliando su funcionalidad.

- *Bridge o Puente:* Se encuentra dentro del tipo de patrón estructural y permite desacoplar una abstracción de su implementación, de manera que puedan variar de forma independiente.

- *Observer u Observador*: Define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambie de estado se modifiquen y actualicen automáticamente todos los objetos que dependen de él.

- *Template Method o Método Plantilla*: Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.

2.4. Administración de emergencias

Una emergencia se define como una situación crítica, tales como incidentes naturales o aquellos provocados por el hombre, que requieren medidas que deben ser llevadas a cabo de inmediato, reduciendo las consecuencias adversas y proteger la vida y propiedad de las personas [5]. A causa de estas emergencias y dependiendo del área al cual pertenecen, existen en Chile entidades a cargo, que principalmente son:

- Unidad de Salud.
- Bomberos.
- Carabineros.

En primer lugar, gestiona la forma en que se llevará a cabo la atención y así controlar la situación a través de los protocolos presentes. En Chile, el protocolo a seguir ante una emergencia está dado por las siguientes fases [3]:

1) Fase de alarma: Esta fase es primordial al momento de iniciar el control de la emergencia, pudiéndose incluso salvar vidas si se hace con eficiencia. Es por esto que la alarma debe ser compartida por las tres instituciones nombradas anteriormente, de

las cuales una será la encargada de dirigir la emergencia y las otras restantes servirán de apoyo si son requeridas según la situación.

2) Fase de desplazamiento, que se refiere a todo aquello referente al trayecto de los vehículos de emergencia hacia el lugar del incidente, desde el uso de balizas y sirenas hasta el acceso del evento.

3) Fase de organización en el sitio de la emergencia, que tiene que ver con la comunicación entre los jefes de cada institución, establecer perímetros de seguridad haciendo que se respeten para la seguridad de la población, atención de víctimas o lesionados si hubieren, además de la entrega de información del incidente ya sea a familiares, medios de prensa y autoridades.

Como la delimitación de este proyecto solo involucra a los bomberos, se habla de emergencia o alerta al evento de despachar carros bomba para atender la solicitud de una o más personas que llamaron a bomberos [20]. Dependiendo de la emergencia, éstas pueden clasificarse según el área al que pertenecen, ya sea accidente de tránsito, incendios, rescates, entre otros. Ante la llamada de alerta, se requerirá cierta información como el número de teléfono de quien llama, ubicación de la emergencia y otros de acuerdo al tipo de emergencia como el tipo de accidente, que es lo que se quema, entre otros [3].

2.5. Componentes de software

Un componente es un tipo de activo reutilizable para el desarrollo de nuevos sistemas, cuyo uso permite disminuir tiempos tanto de construcción como de entrega del software. Un componente está definido como “una unidad de software independiente que puede estar compuesta por otros componentes y que se utiliza para crear un sistema software” [21]. Según esto, la principal característica del componente es la reusabilidad que posee, pero además debe responder a las siguientes características [13]:

1) Es identificable: Para un repositorio de componentes, cada uno de ellos se debe tener una buena identificación para facilitar la búsqueda y uso del componente.

2) Accesible solo a través de su interfaz: El componente debe ser observable por el usuario según las operaciones que realiza en una interfaz y no sobre su implementación; de esta forma puede reutilizarse para otras aplicaciones con distintas interfaces.

3) Servicios invariantes: Las operaciones de un componente no deben variar en la interfaz, esto es, puede ser modificado el componente pero no deben haber cambios visibles para el usuario en la interfaz.

4) Documentado: Un componente debe poseer una buena documentación acorde a las operaciones que posee, de tal forma de facilitar su posterior uso en nuevos desarrollos.

Además, se desea que un componente pueda ser utilizado en múltiples sistemas de acuerdo a lo requerido por el cliente, que exista independencia, es decir, que en lo posible no dependa de otros componentes para su utilización, que pueda ser mantenido ya sea para optimizar, corregir o mejorar las operaciones del componente y que pueda ser utilizado en cualquier plataforma independiente del lenguaje y del entorno en el cual fue desarrollado [13].

Por otro lado, el reusar componentes, en vez de volver a desarrollarlos nuevamente, genera beneficios en cuanto a productividad y eficiencia al integrar componentes para la producción de nuevas aplicaciones y mientras más genérico sea, dependiendo del dominio al que está enfocado y los servicios que entrega, más puede reutilizarse.

2.6. Android

Android es un sistema operativo enfocado a los dispositivos móviles que está basado en Linux, cuya ventaja es ser un sistema operativo libre, gratuito y multiplataforma. Este sistema operativo consta de una arquitectura de cuatro capas basadas en software que permiten una visualización de las aplicaciones, las cuáles se muestran en la Figura 1 y se detallan a continuación [9].

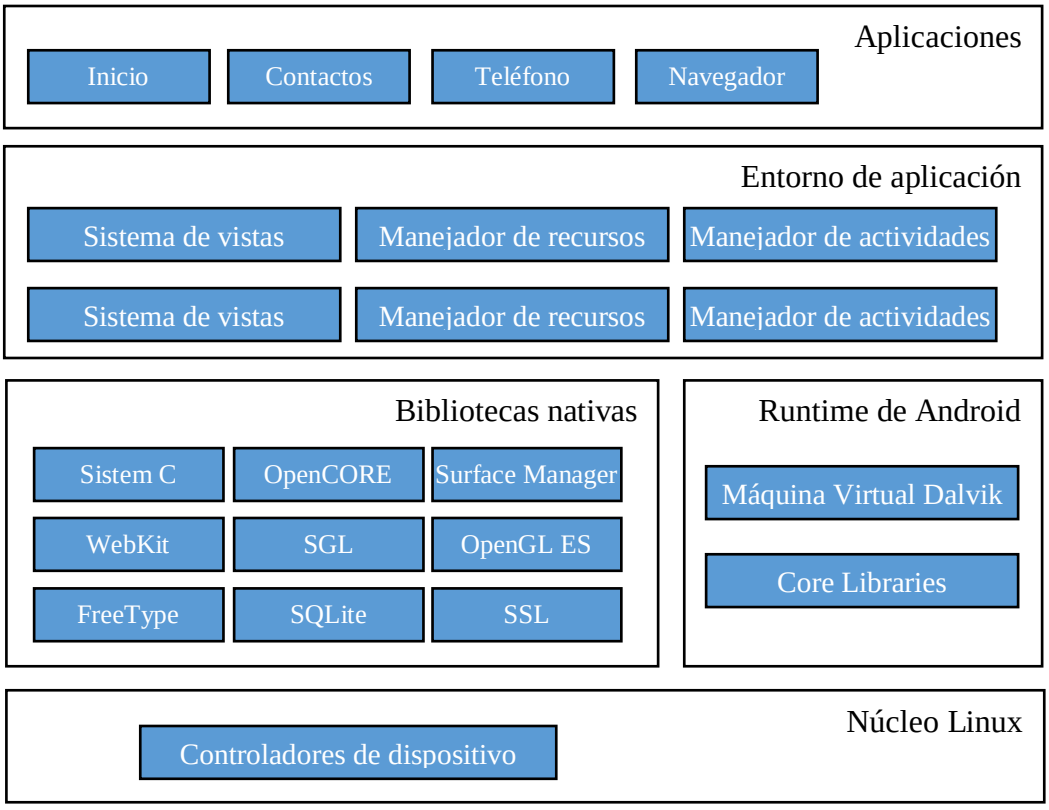


Figura 1. Arquitectura de Android por capas.

1) Núcleo Linux: El núcleo de Android está basado en por el sistema operativo Linux. Esta capa depende del hardware y provee diversos servicios tales como:

   a) Seguridad
   b) Manejo de la memoria

c) Multiprocesos
d) Soporte para dispositivos

2) Runtime de Android: Esta capa se basa en el concepto de máquina virtual en Java y como ésta dependería de las limitaciones del dispositivo para funcionar, Google creó una máquina virtual llamada Dalvik, que respondiera mejor incluso ante un procesador limitado y con poca memoria. Así, cada aplicación corre en su propio proceso con ayuda de esta máquina virtual.

3) Bibliotecas nativas: Esta capa contiene diversas librerías en C/C++, pues están compiladas en el código nativo del procesador y de ahí proviene su nombre. Estas bibliotecas son usadas en diversos componentes de Android y gran parte de ellas son de código abierto.

4) Entorno de aplicación: Esta capa entrega una plataforma de desarrollo libre de aplicaciones y permite el reuso de componentes utilizando el lenguaje de programación Java. Dentro de los servicios más importantes que se incluyen en esta capa se encuentra la parte visual de los componentes, manejo del ciclo de vida de las aplicaciones, manejo de alertas personalizadas en la barra de notificaciones y acceso a datos de otras aplicaciones como los contactos, entre otros.

Finalmente, para desarrollar aplicaciones en Android pueden utilizarse diferentes IDEs disponibles, como por ejemplo Android Studio o Eclipse, agregándole a este último el plugin de Android ADT, siendo Android Studio el IDE oficial de Android en la actualidad. Cabe mencionar además, que se deben entender aquellos conceptos de java como la programación orientada a objetos y sus métodos.

2.7. Java para Android

Java es un lenguaje de programación y una plataforma informática comercializada por Sun Microsystems. Está presente en computadores, centros de datos, consolas para videojuegos, teléfonos móviles, Internet, entre muchos otros usos y puede ser descargado gratuitamente desde su página oficial java.com. Este lenguaje de programación, puede ser utilizado desde distintas plataformas como son Windows, Mac y Linux a través de múltiples entornos de desarrollo. En la Figura 2 se muestra un pequeño extracto del desarrollo de un componente en el presente proyecto, que corresponde al envío y recepción de audio entre los usuarios, dentro de la aplicación *CollabMap*.

```
public class ConcreteAudioMakerA implements AudioMaker {

    public void make(String savePath, UIInstance UIInstance){

        if(savePath != null && UIInstance != null){

            if(UIInstance instanceof Activity){

                Uri audioUriPath = Uri.fromFile(new File(savePath));

                Intent takeAudioIntent = new Intent();
                takeAudioIntent.setAction(android.provider.MediaStore.Audio.Media.RECORD_SOUND_ACTION);

                takeAudioIntent.putExtra(MediaStore.EXTRA_OUTPUT, audioUriPath);
                ((Activity) UIInstance).startActivityForResult(takeAudioIntent, 1);

            }

        }
```

Figura 2. Extracto de código en lenguaje de programación Java.

3. Estado del arte

A continuación se listarán aplicaciones que actualmente están en funcionamiento y que son similares a *CollabMap*.

3.1. Aplicaciones similares

3.1.1. VIPer

VIPer es una aplicación móvil, disponible de forma gratuita para las plataformas Android e iOS. Esta aplicación fue desarrollada por la empresa chilena *Exefire*, la cual permite enviar notificaciones entre teléfonos móviles que pertenezcan a un grupo seleccionado de usuarios, que puede ser desde un grupo de amigos, familiares, bomberos, universidades y clínicas, de tal forma de mantenerlos informados según se requiera. Dentro de las funcionalidades que posee *VIPer* se encuentra la integración de mapas, imágenes, rutas utilizando *Waze*, instrucciones y cualquier otra información para el grupo en cuestión a través de notificaciones. El cuerpo de bomberos de Ñuñoa tiene 11 compañías y el cuerpo de bomberos de Santiago tiene 22 compañías, las que en 2014 decidieron firmar un convenio con la empresa desarrolladora de *VIPer*, para obtener una licencia gratuita para cada uno de los voluntarios de las compañías respectivas y utilizar la aplicación con el fin de optimizar los tiempos de respuesta ante una emergencia, mantener a los voluntarios informados en tiempo real, enviando información relevante a los dispositivos móviles como por ejemplo, la ubicación exacta de la emergencia y gestionar de forma más sencilla la comunicación entre bomberos. *VIPer* resulta ser una vía de comunicación muy buena entre voluntarios, pues se enfoca principalmente en difundir información relevante y actualizada de la emergencia y no solo mensajería sino que de imágenes, mapas y rutas de tal forma de evitar saturaciones en la comunicación por radio, permitiendo así utilizarla en casos específicos e importantes, que es lo que busca también entregar *CollabMap* a bomberos.

Al ser una aplicación que mantiene el contacto entre un grupo de personas cualquiera y no específicamente voluntarios, se pierden ciertas funcionalidades que merecen ser

incorporadas en beneficio de bomberos para lograr disminuir el tiempo de respuesta ante una emergencia, como por ejemplo, la ubicación de sitios de interés como grifos u otros carros bomba, característica que sí es incorporada por *CollabMap*. Por otro lado, se tiene que CollabMap, por lo pronto, funcionaría solo para alerta de incendios, que a diferencia de *VIPer* es utilizado para cualquier tipo de emergencia, reuniones y otros informativos.

Según usuarios de *VIPer*, la interfaz es una característica importante y que debiese mejorarse para lograr un manejo más fácil de la aplicación. Es por esto que los desarrolladores han implementado una nueva interfaz, además de agregar configuraciones a las notificaciones y un punto importante, que es el soporte en inglés. En la Figura 3, se muestra la nueva interfaz desarrollada, de la que se observa un listado con todas las emergencias disponibles durante la consulta y que incluye información detallada como fecha y ubicación de la emergencia, además de hacer uso de los códigos utilizados por Bomberos de Chile. También se observa una captura de pantalla correspondiente a la ubicación de la emergencia seleccionada mediante un marcador fijo posicionado en un mapa.

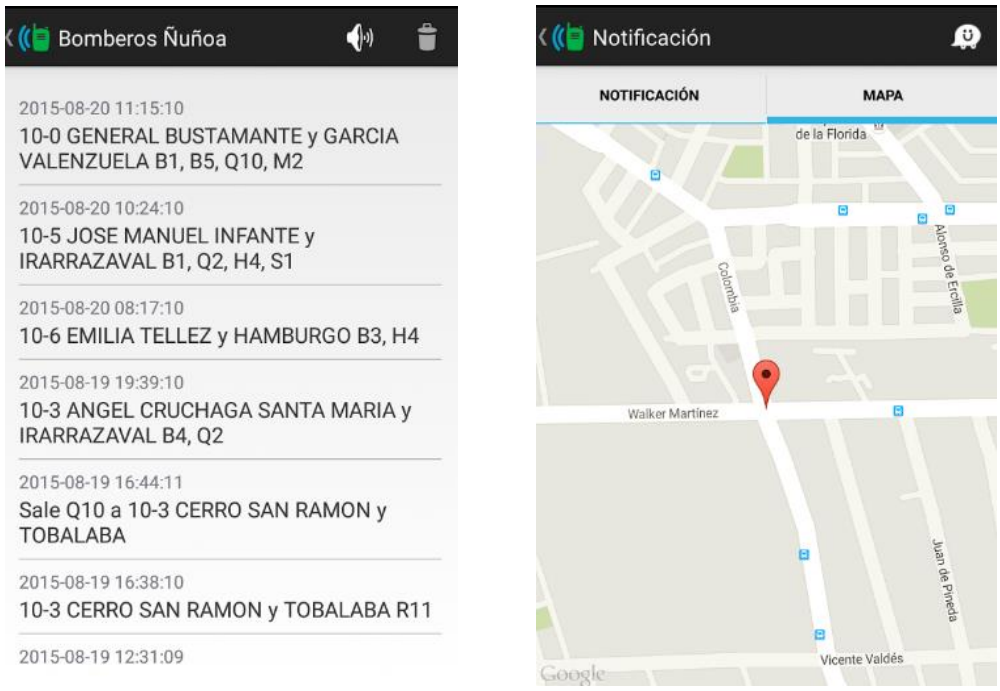


Figura 3. Interfaz de la aplicación VIPer.

3.1.2. Bomberos CR

Bomberos CR es una aplicación móvil creada por la empresa *aPlicativa*, desarrollada para ser utilizada principalmente por los bomberos de Costa Rica y por la población costarricense. Ésta informa una emergencia en tiempo real a través de notificaciones, logrando una coordinación entre la central de bomberos y el personal voluntario, la unidad de policía, la prensa nacional y la población en general que tenga en su móvil la aplicación. Se encuentra disponible para descargarla tanto para sistemas Android como para iOS, permitiendo a cualquier usuario acceder a ella sin costo alguno.

Bomberos CR ofrece a sus usuarios el acceso a la información completa e inmediata de una emergencia, como por ejemplo su ubicación, el tipo de emergencia, la unidad de bomberos a cargo y el mapa del lugar, lo que permite además personalizar qué tipos de emergencia desean recibir, el número de notificaciones diarias de una estación de bomberos en particular o las emergencias correspondientes a un lugar específico.

Algunas de las funciones de la aplicación son notificar con sonidos especiales según el interés del usuario, seleccionar en pantalla la cantidad de emergencias, estaciones de bomberos y una mayor variedad de emergencias que se pueden filtrar como notificaciones o emergencias, consultar datos históricos o más frecuentes de las emergencias gracias a gráficos estadísticos, pudiéndose ver por datos generales o por zona geográfica, que también incluye filtros de consulta para una mejor información [1].

Observando las características y funciones que entrega *Bomberos CR*, se puede decir que es una aplicación muy completa, que logra cubrir una gran cantidad de necesidades relacionadas con el procedimiento de una emergencia, y que no solo sirve para ser utilizada por bomberos sino por toda la ciudadanía en general, manteniéndola al día de aquellos sucesos que surgen en la población costarricense, evitando que utilicen y saturen las líneas de emergencia constantemente preguntando por información. Además, que sea personalizable juega un rol importante en la aplicación, debido a que el usuario solo recibirá emergencias que le parezcan significativas, ya sea por la ubicación, por la compañía de bomberos, u otros.

Que el sistema mantenga una conexión entre bomberos y la policía es una gran ventaja que facilita el apoyo entre instituciones ante una eventual emergencia y que *CollabMap* podría incorporar a sus funcionalidades y por qué no, añadir la colaboración de unidades de salud, cumpliendo así con el protocolo según el manual de operaciones ante una emergencia de Chile [3].

En la Figura 4(a), se puede observar la interfaz de la aplicación *Bomberos CR*, que muestra el menú de opciones que consta de un listado de emergencias, ubicación en mapa de las estaciones de bomberos, mapa de las emergencias, información sobre la aplicación BomberosCR, gráficos estadísticos sobre las emergencias y configuración para que el usuario reciba notificaciones y visualice las emergencias o estaciones de bomberos más significativas para él. Además, en la Figura 4(b) la aplicación permite al usuario configurar el tipo de emergencia que prefiere visualizar en el listado de emergencias, por ejemplo, que la aplicación le muestre o no incendios, emergencias por agua, aire, eléctricas, entre otras. Cabe mencionar que actualmente han incorporado gráficos que permiten ver estadísticas sobre las emergencias, además de nuevos filtros de búsqueda.

a) *b)*

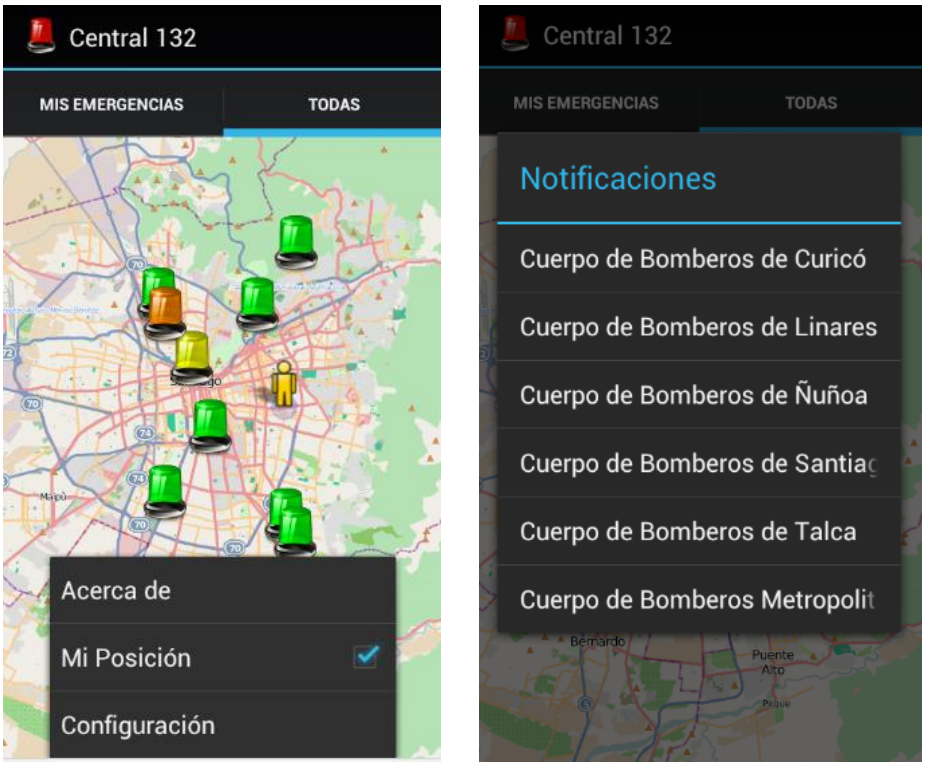
Figura 4. Interfaz de la aplicación BomberosCR.

3.1.3. Central 132

Central 132 es una aplicación móvil desarrollada en 2014 por la empresa chilena Huilliche Ltda. para sistemas Android e iOS. Este software está orientado a mejorar la comunicación y el trabajo que realiza bomberos ante un llamado de alerta. Se busca entregar información en tiempo real y de manera oportuna de las emergencias hacia los voluntarios del cuerpo de bomberos. Actualmente, *Central 132* está en funcionamiento en diversas ciudades del país y en más de 50 compañías de bomberos a lo largo de Chile, con un servicio que incluye la ubicación satelital (ver Figura 5(a)) y de desplazamiento de las unidades, botón de emergencia en tablero del vehículo para indicar a la Central la ocurrencia de un evento mayor, la información de ubicación de grifos e incluye comunicación vía celular con la central, entre otros.

Esta aplicación permite recibir información de emergencias de acuerdo a la configuración que cada usuario le dé, pudiendo elegir por comuna, tipo de la emergencia o de la compañía de bomberos de interés. Esta característica resulta ser muy importante al momento de recibir alertas filtrando aquella información que es relevante para el usuario. *Central 132* comenzó como una aplicación simple que enviaba notificaciones de emergencias y que ha logrado incrementar sus funcionalidades en base a las necesidades propias de bomberos. *CollabMap* tiene mucho que mejorar si pretende entrar al mercado de la administración de emergencias en Chile, incorporando funcionalidades que los usuarios valoran, tales como recibir alertas personalizadas de una compañía, emergencia o comuna, además de entregar información a toda la comunidad y no solo a la entidad en cuestión. Por el contrario, *CollabMap* posee funciones que esta aplicación no tiene como el añadir recursos digitales que pueden visualizarse en el mapa, que puede ser útil ante el actuar de bomberos.

En la Figura 5(a) se muestra dentro de un mapa, la ubicación de todas las emergencias disponibles. Además permite al usuario conocer más acerca de la aplicación, visualizar su ubicación dentro del mapa en el que se muestran las emergencias y configurar la aplicación con respecto a la recepción de notificaciones. En tanto, en la Figura 5(b) se muestra al usuario la configuración de notificaciones, permitiéndole escoger de cuál estación de bomberos desea recibir notificaciones, a modo de filtrar las emergencias según los intereses actuales de cada bombero.



a) *b)*

Figura 5. Interfaz de la aplicación Central 132.

3.1.4. Bomberos Alerta

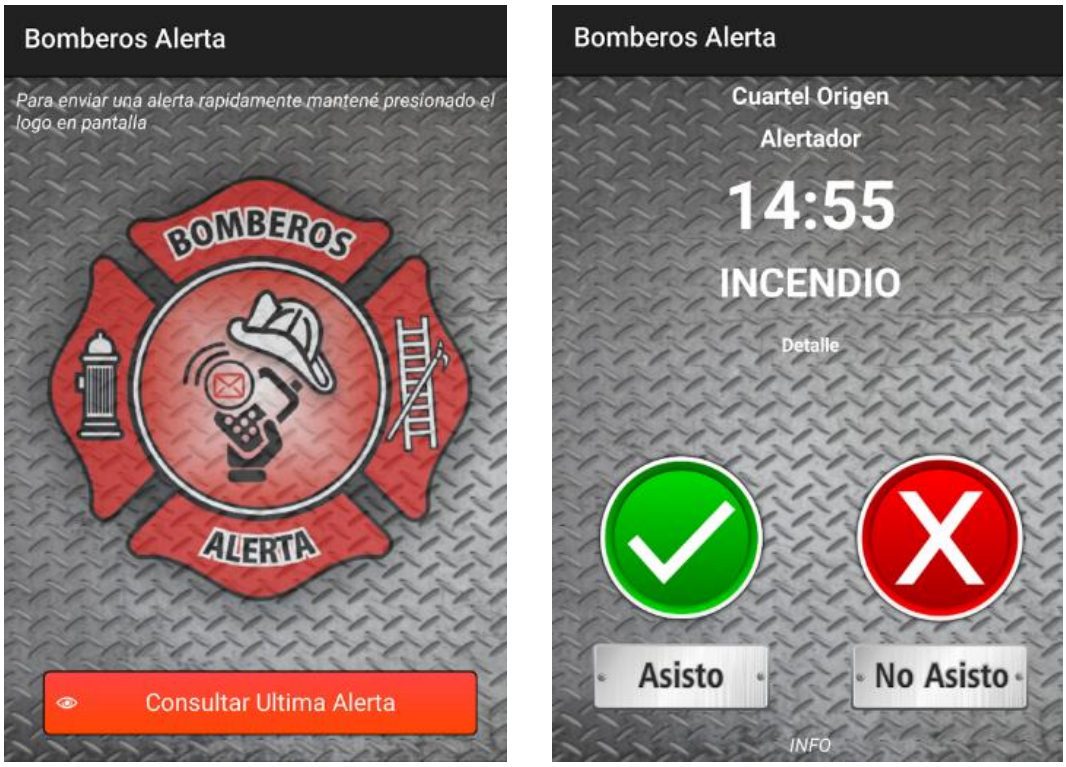
Bomberos Alerta es un sistema de alerta desarrollado de forma independiente por profesionales argentinos, con el objetivo de enviar y recibir alertas a bomberos, quienes pueden responder al llamado de alerta dependiendo de su disponibilidad. Esto permite tener un control de la cantidad de voluntarios que se dispondrá ante una emergencia. Las notificaciones son enviadas desde el cuartel al que pertenece el bombero hasta sus dispositivos móviles. De esta forma, lo que se busca con esta aplicación es mejorar el tiempo de respuesta ante una emergencia entregando un mejor servicio hacia la comunidad. Otra funcionalidad que posee *Bomberos Alerta* es que se pueden recibir alertas correspondientes a otras emergencias que se estén desarrollando en otra parte del país.

Si un cuartel de bomberos desea utilizar esta aplicación, debe ingresar a la página oficial y registrar el cuartel de bomberos al que pertenece, lo cual permitirá generar un PIN que se asociará a cada bombero, quienes deberán descargar la aplicación, ingresar el PIN correspondiente y ya podrá utilizar la aplicación obteniendo los beneficios de éste.

Al igual que en *CollabMap*, es una aplicación de uso exclusivo de bomberos y se puede llevar un control de los voluntarios que asistirán a una emergencia, esto es, cada bombero puede decidir si asistirá o no a la emergencia (ver Figura 6 (b)).

Como la mayoría de las aplicaciones móviles dirigidas a la administración de emergencias de bomberos, ésta permite mejorar la comunicación en el cuartel, alertando a los voluntarios de las emergencias que se originen a través de notificaciones. Una de las desventajas que se puede observar en *Bomberos Alerta* es en cuanto a la plataforma de uso de la aplicación, que es solamente Android. Según un estudio, Android es el sistema operativo más utilizado. Sin embargo le siguen otros que cada vez toman más fuerza como son iOS y Windows Mobile [10], por lo que en la actualidad se hace necesario desarrollar aplicaciones móviles multiplataforma con el fin de obtener diversidad, de la cuál *Bomberos Alerta* no posee y se vuelve también un punto a considerar para *CollabMap* y llegar así a un número más grande de usuarios.

Finalmente, en la Figura 6(a), se permite al usuario acceder a la última emergencia disponible. Además, al presionar el logo que aparece en pantalla, el usuario puede enviar un mensaje de alerta indicando si es un incendio, auxilio u otro.



a) *b)*

Figura 6. Interfaz de la aplicación Bomberos Alerta.

4. Análisis y Especificación de Requisitos

Según lo mencionado anteriormente en el método de desarrollo utilizado (ver sección 1.5. Metodología aplicada) en el presente proyecto, se tiene esta primera etapa de análisis, que conlleva la determinación de los objetivos y las limitantes del sistema, con el fin de alcanzar dichos objetivos y dar solución al problema.

4.1. Descripción del problema

CollabMap es una herramienta colaborativa desarrollada para ser usada en dispositivos móviles por bomberos ante una emergencia, específicamente en incendios.

Debido a que constantemente se generan cambios, ya sea en el sistema operativo, el lanzamiento de nuevos *smartphones* o *tablets* con nuevas funciones incorporadas y por supuesto, a las necesidades de bomberos, entre otros, se vuelve necesario realizar una mantención del sistema, que para este proyecto significa añadir nuevas funcionalidades e integrarlas a la aplicación junto a aquellas ya construidas. Estas funcionalidades ya elaboradas, se detallan a continuación [15]:

- Uso de datos en línea mediante la sincronización entre un repositorio de datos local y una base de datos externa.
- Compartir la ubicación del usuario con todos aquellos que se encuentren conectados.
- Compartir el status del usuario, relacionado a la emergencia atendida por éste, con todos aquellos que se encuentren asistiéndola o solo visualizándola.
- Crear un recurso digital para una determinada emergencia.
- Compartir un recurso digital con aquellos que estén asistiendo la misma emergencia que el usuario.

4.2. Funcionalidades desarrolladas

En base a lo estipulado anteriormente, se tienen las siguientes funcionalidades que fueron desarrolladas para este proyecto de título, que considera la mantención de la aplicación *CollabMap*:

- *Creación, envío y recepción del recurso digital de audio:* Esta funcionalidad permite a un usuario grabar un audio asociado a una emergencia en específico y guardarlo tanto el repositorio de datos local como en la base de datos externa, para luego dar la opción al usuario de compartir dicho recurso con usuarios que estén conectado en la misma emergencia.

  Cabe mencionar que parte de esta funcionalidad fue una modificación a la ya implementada [15], de tal forma de añadir la generación de audio a los recursos digitales existentes.

  Para llevar a cabo esta funcionalidad, se crearon/modificaron los siguientes componentes:

  - ConcreteAudioMakerA: Este componente es el encargado de crear el audio.

  - AudioMaker: Este componente corresponde a la interfaz implementada para la clase ConcreteAudioMakerA.

  - DigitalResourcesMaker: Este componente permite al usuario seleccionar qué recurso digital (fotografía, video o audio) desea crear, siendo cada uno de ellos implementado por su componente respectivo.

  - ConcreteOptionalComponentsA: Este componente está encargado de administrar los componentes opcionales presentes en la aplicación.

- DefaultOptionalComponents: Este componente permite gestionar las instancias de aquellos componentes opcionales e independientes del sistema operativo Android.

- OptionalComponents: Este componente corresponde a la interfaz implementada para la clase “ConcreteOptionalComponentsA”.

- ConcreteAudioOpenerA: Este componente permite gestionar la apertura de un audio.

- AudioOpener: Este componente corresponde a la interfaz implementada para la clase “ConcreteAudioOpenerA”.

- ConcreteDigitalResourcesOpener: Este componente permite gestionar la apertura de los tres recursos digitales presentes en el proyecto, esto es, fotografía, videos y audio.

- DigitalResourceTypes: Este componente permite definir los formatos y los tipos de recursos digitales presentes formatos y los tipos de recursos digitales y que son gestionados por el componente “ConcreteDigitalResourceTypes”.

Los componentes modificados de la lista anterior para llevar a cabo esta funcionalidad fueron:

- “DigitalResourcesMaker”
- “ConcreteOptionalComponentsA”
- “DefaultOptionalComponents”
- “OptionalComponents”
- “ConcreteDigitalResourcesOpener”

- "DigitalResourceTypes"

El resto fueron clases creadas para el desarrollo del presente proyecto.

- *Visualización de ruta hacia la emergencia*: Esta funcionalidad permite a un usuario, que es partícipe de una emergencia, visualizar la ruta desde el punto en el que está ubicado hasta el lugar de la emergencia, pensado para aquellos bomberos que desconocen la ubicación exacta de la emergencia y minimizar así los tiempos de llegada.

  Cabe mencionar que parte de esta funcionalidad fue una modificación a la ya implementada [15], de tal forma de añadir la visualización de ruta hacia la emergencia.

  Para llevar a cabo esta funcionalidad, se crearon/modificaron los siguientes componentes:

  - EmergencyMenu: Esta actividad está conformada por un menú de opciones existentes sobre una emergencia.

  - CommonLabels: Es la encargada de administrar los rótulos que son usados por varios componentes

  - RouteToEmergencyMap: Este componente es el encargado de trazar la ruta desde el origen, que es la ubicación del usuario, hasta el lugar de la emergencia.

  - DirectionsJSONParser: Este componente es el encargado de conectar con el web service de Google Maps y recibir el resultado en formato JSON, que retorna un listado que contiene latitudes y longitudes para luego interpretar el resultado.

Los componentes modificados de la lista anterior para llevar a cabo esta funcionalidad fueron:

- "EmergencyMenu"
- "CommonLabels"

El resto fueron clases creadas para el desarrollo del presente proyecto.

La especificación de requisitos de cada funcionalidad mencionada se detalla a continuación:

Id Requisito	**R1**
Nombre Función	Crear recurso digital de audio
Descripción	Esta función permite a un usuario, que asiste a una determinada emergencia, grabar audio con la opción de compartirlo con otros usuarios que también estén asistiendo la emergencia.
Entrada	digital_resource_name, format, file, description, date_sent, related_emergency, sender, latitude, longituded.
Proceso	**R1.1:** digital_resource_name es el nombre del recurso digital que puede ser Fotografía, Video o Audio y que será almacenado. **R1.1.1:** digital_resource_name es una cadena de tamaño máximo 50 y acepta caracteres especiales.

	R1.1.2: digital_resource_name comienza con el prefijo 'EFILE_' seguido de un número correlativo que inicia en 1. **R1.1.3:** No pueden existir digital_resource_name idénticos. **R1.2:** format es el formato del recurso digital y es una cadena de tamaño máximo 10 y acepta caracteres especiales. **R1.2.1:** format para el recurso digital de Fotografía puede contener los formatos 'gif', 'jpg', 'jpeg', 'bmp', 'png' y 'tiff'. **R1.2.2:** format para el recurso digital de Video puede contener los formatos '3gp', 'mpg', 'mpeg', 'avi', 'mp4' y 'mkv'. **R1.2.3:** format para el recurso digital de Audio puede contener los formatos 'mp3', 'aac', 'wav', 'rec', 'wma', 'amr', 'm4a'. **R1.2.2:** No pueden existir format idénticos. **R1.3:** file es la imagen, video o audio que se almacena como tipo bytea. **R1.4:** description es la descripción que el usuario da al recurso creado y es una cadena de tamaño máximo 100 y acepta caracteres especiales. **R1.5:** date_sent es los datos de fecha y hora correspondiente al envío del recurso digital, que

	sigue el formato AAAA-MM-DD para la fecha y HH:MM:SS para la hora. **R1.5.1:** AAAA se define para el año. **R1.5.2:** MM se define para el mes y puede tomar valores entre 01 y 12. **R1.5.3:** DD para el día y puede tomar valores entre 01 y 31. **R1.5.4:** HH se define para la hora y puede tomar valores entre 00 y 23. **R1.5.5:** MM se define para los minutos y puede tomar valores entre 00 y 59. **R1.5.6:** SS para los segundos y puede tomar valores entre 00 y 59. **R1.6:** related_emergency es un número entero con el cual se identifica a una emergencia en la cual se ha creado el recurso digital. **R1.6.1:** related_emergency es un número correlativo que comienza en 1. **R1.7:** sender corresponde al usuario que envía el recurso digital y es una cadena de tamaño máximo 50 y acepta caracteres especiales. **R1.8:** latitude y longitude corresponden a la ubicación desde donde se genera el recurso digital.

	R1.8.1: latitude y longitude son del tipo double precision y puede ser tanto negativo como positivo o una combinación de ellos. **R1.9:** Todas las variables mencionadas en los requisitos anteriores no deben ser nulos, a excepción de 'description'. **R1.10:** Para que un usuario cree un nuevo recurso digital, éste deberá estar asistiendo a la emergencia correspondiente. **R1.10.1:** Se desplegará un menú de opciones en el cual el usuario podrá escoger qué recurso digital crear, que puede ser imagen, video o audio. **R1.10.2:** Luego que el usuario escoja el recurso digital, se abrirá el grabador de audio o la cámara, para foto o video, dependiendo de la elección. **R1.11:** Si la operación se efectúa correctamente se desplegará por pantalla el mensaje "El archivo ha sido almacenado!" y el archivo se habrá almacenado en el dispositivo. **R1.11.1:** Luego de que el sistema guarde el archivo, se preguntará al usuario si desea compartir el archivo. **R1.11.2:** Si el usuario acepta compartir el archivo, deberá seleccionar si envía el recurso a todos los destinatarios o solo a algunos. Y

	además, opcionalmente, podrá escribir una descripción. **R1.11.3:** Si el usuario presiona el botón enviar, se enviará el archivo a los destinatarios seleccionados y se desplegará el mensaje: “Recurso digital enviado!” **R1.11.4:** Si el usuario presiona el botón enviar pero no se efectúa el envío se desplegará el mensaje: "El recurso digital no pudo ser enviado..." y el archivo no se enviará. **R1.12:** Si no se efectúa correctamente la operación, se desplegará por pantalla el mensaje “El archivo no pudo ser almacenado…” y se volverá al menú de opciones para escoger crear un nuevo recurso digital, sin haber guardado el archivo.
Salida	**Salida1:** digital_resource_name, format, file, description, date_sent, related_emergency, sender, latitude, longituded. **Salida2:** Despliegue de mensajes según los requisitos **R1.11, R1.11.3, R1.11.4 y R1.12**

Id Requisito	**R2**
Nombre Función	Visualizar ruta hacia la emergencia
Descripción	Esta función permite a un usuario que asiste a una determinada emergencia, visualizar el trayecto desde su punto de ubicación hasta el lugar de la emergencia.
Entrada	id_emergency, address, latitude, longitude
Proceso	**R2.1:** id_emergency es un número entero con el cual se identifica a una emergencia. **R2.1.1:** id_emergency es un número correlativo que comienza en 1. **R2.2:** address es la dirección de la emergencia y es una cadena de tamaño máximo 50 y acepta caracteres especiales. **R2.3:** latitude y longitude corresponden a la ubicación desde donde se genera el recurso digital. **R2.3.1:** latitude y longitude son del tipo double precision y puede ser tanto negativo como positivo o una combinación de ellos. **R2.4:** Todas las variables mencionadas en los requisitos anteriores no deben ser nulos. **R2.5:** No es necesario que el usuario esté asistiendo a la emergencia para visualizar la ruta. **R2.6:** Para que el usuario visualice la ruta desde su ubicación hasta la emergencia, deberá seleccionar en el menú principal de la emergencia escogida

	"Mostrar ruta hacia la emergencia" y se visualizará la ruta. **R2.6.1:** La ubicación actual del usuario será obtenida utilizando el GPS del dispositivo a través de Google Maps, por lo que éste deberá permanecer activo. **R2.6.2:** El lugar de la emergencia está dado por la latitud y longitud asignada a la emergencia (id_emergency) a través de la dirección (address).
Salida	**Salida1:** Visualización en el dispositivo móvil de la ruta, desde la ubicación del usuario hasta el lugar de la emergencia.

4.3. Restricciones del sistema

- El uso de la aplicación está dirigida al tipo de usuario bombero e incluye: "Bombero", "Comandante Incidente", "Bombero F/S", "Conductor Camión" y "Líder Equipo".

- El sistema deberá utilizarse en dispositivos móviles: *Smartphones* y *Tablets*.

- El sistema solo estará disponible para dispositivos móviles con sistema operativo Android con la versión mínima 2.3.4.

- El dispositivo móvil deberá contar con los servicios de *Google Play Services* v13 o superior para una función correcta de la aplicación.

- El dispositivo móvil debe contar con acceso a Internet y activar el GPS en cada uso.

- El dispositivo móvil deberá otorgar los permisos correspondientes a Almacenamiento, Cámara, Micrófono, Teléfono y Ubicación.

- El dispositivo móvil deberá contar con un espacio mínimo de almacenamiento de 30 MB para su instalación.

- El sistema se desarrollará en lenguaje Java para Android utilizando el entorno de desarrollo Eclipse, más el complemento ADT.

- Se utilizará el servidor local *Wamp Server*, debido a la construcción de la base de datos original.

- Se utilizará *PostgreSQL* para la base de datos del sistema y el lenguaje de programación SQL para las modificaciones respectivas.

5. Diseño del sistema

Siguiendo con el diseño del sistema, de acuerdo a las especificaciones detalladas en la sección anterior, se mostrará a continuación tanto el diseño de los datos de la aplicación original como el diseño de la interfaz de usuario de la aplicación móvil correspondiente a las nuevas funcionalidades desarrolladas para el presente proyecto.

5.1. Diseño de datos

Debido a que el presente proyecto es una mantención al proyecto original desarrollado por Erika Ormeño [15], se mostrará en la Figura 7 el diagrama correspondiente al modelo de entidad-relación asociado a la base de datos original, incluyendo la modificación al tipo de archivo que se podrá crear con la nueva funcionalidad de audio, con el fin de dar a conocer cómo está conformada para el almacenamiento de datos.

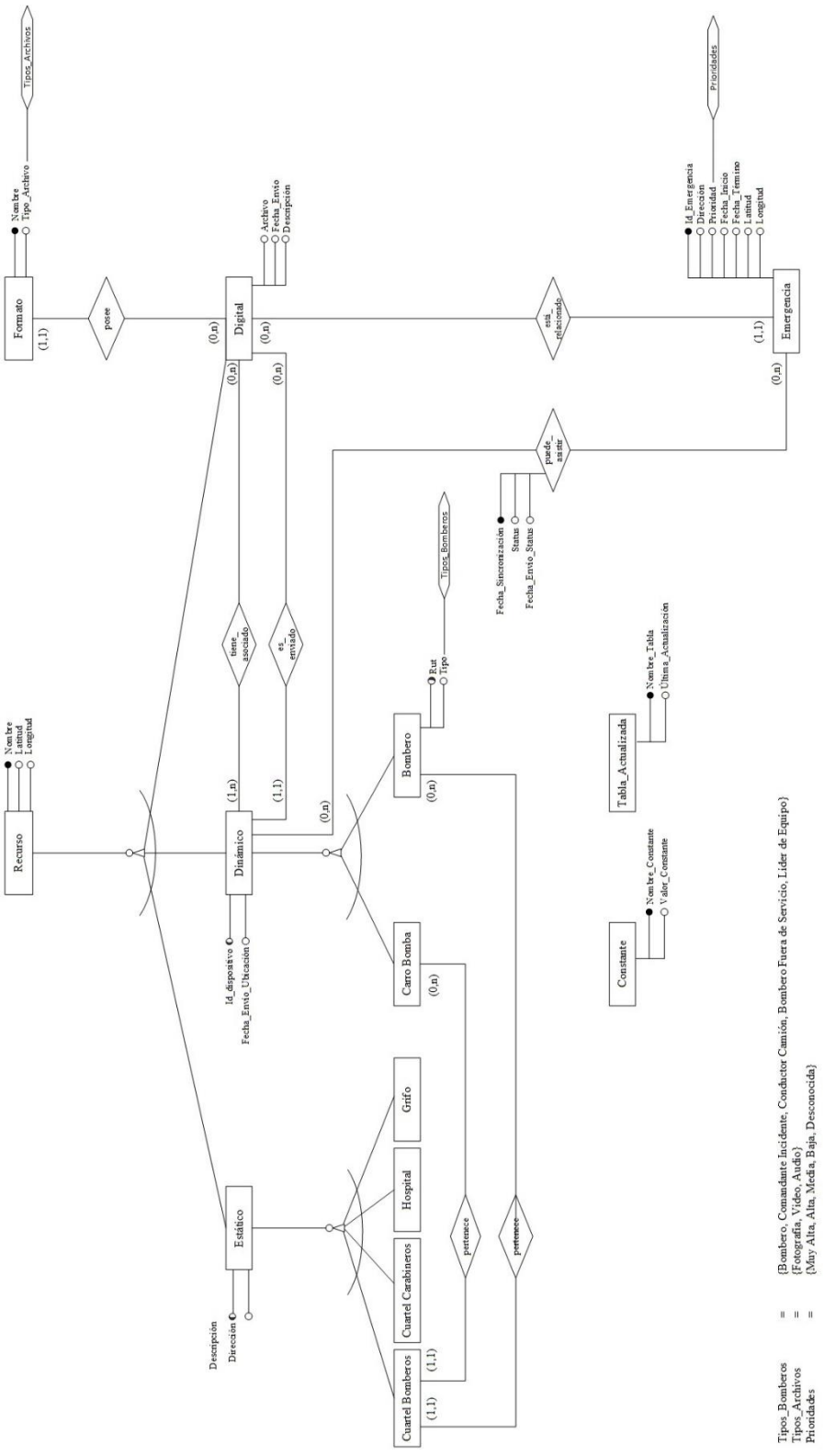


Figura 7. Modelo Entidad-Relación de la base de datos actual.

5.1.1. Nuevos datos en la Base de Datos

Para el desarrollo de la funcionalidad correspondiente a la "*creación, envío y recepción del recurso digital de audio"* mencionada en la sección 4.2 (Funcionalidades Desarrolladas), se establece que no hubo modificaciones a la base de datos original, sino que se debieron añadir datos a las tablas ya existentes.

Como se puede ver en el modelo entidad-relación de la Figura 7, existen 3 tipos de recursos:

- *Estático:* Este tipo de recurso corresponde a aquellos puntos de interés fijos que estarán disponibles visualmente en un mapa, con el fin de encontrar el más cercano a la emergencia consultada en caso de ser requeridos. Además, cuenta con datos tales como la dirección y una descripción del recurso.
  Éste recurso puede tomar los siguientes valores:

  - Cuartel Bomberos
  - Cuartel Carabineros
  - Hospital
  - Grifo

- *Dinámico:* Este tipo de recurso corresponde a los tipos de usuarios que la aplicación móvil posee, tomando los siguientes valores:

  - Carro Bomba
  - Bombero

  Bombero puede ser del tipo: Bombero o Comandante incidente, de acuerdo a la actividad que desempeñe el usuario.

- *Digital:* Este tipo de recurso está asociado a un recurso dinámico, esto es, puede ser creado por un recurso dinámico para luego ser almacenado como contenido digital asociado a una emergencia en específico. Este tipo de recurso puede tener diferentes formatos dependiendo del tipo de archivo (Tipo_Archivo) escogido, esto es, "Fotografía" o "Video".

  Para "Fotografía" y "Video" se encontraban los formatos definidos (Nombre_Formato), y que son aceptados en la creación del recurso, gestionada por el usuario en la aplicación. Estos formatos se muestran en la Tabla 1.

  Por otro lado, para añadir el nuevo recurso digital "Audio", fue necesario añadir este valor al dominio "Tipo_Archivo" que se encuentra en la base de datos y además agregar los siguientes formatos (Nombre_Formato), mediante comandos SQL.

  Los nuevos formatos a añadir a la tabla y que serán aceptados para el recurso digital de Audio son: 'mp3', 'aac', 'wav', 'rec', 'wma', 'amr', 'm4a'.

  Así, este nuevo atributo "Audio", queda establecido junto a los ya existentes y se muestra en la Tabla 1, a continuación:

Fotografía	gif jpg jpeg bmp png tiff

Video	3gp mpg mpeg avi mp4 mkv
Audio	mp3 aac wav rec wma amr m4a

Tabla 1. Formatos aceptados para la creación de un recurso digital.

5.1.2. Diagrama de flujo de datos

A continuación, se modelará el diagrama de flujo de datos que corresponde a las funcionalidades desarrolladas y mencionadas en el Capítulo 4 (Análisis y Especificación de Requisitos) del presente proyecto de título, de acuerdo a lo que allí se especificó para llevar a cabo el desarrollo del proyecto.

Para la primera funcionalidad se tiene:

- *Creación, envío y recepción del recurso digital de audio:* Tal como se observa en la Figura 8, el usuario (firefighter_name) previamente ingresado al sistema, ya sea bombero o carro bomba, procederá a escoger una de las emergencias listadas en la pantalla principal de la aplicación móvil y obtener así el identificador de la emergencia (id_emergency). Posteriormente, deberá aceptar asistir a la emergencia si desea crear algún recurso digital sino el sistema no lo permitirá. Si el usuario asiste a la emergencia podrá elegir crear uno de los 3 recursos disponibles, "Fotografía", "Video" o "Audio", siendo este último el que se muestra en el diagrama por ser la nueva funcionalidad disponible. Finalmente, después de ser creado el audio, se guardará el recurso en la base de datos local del sistema (dispositivo móvil) y además se dará la opción para ser compartido con otros usuarios que también asistan a la misma emergencia.

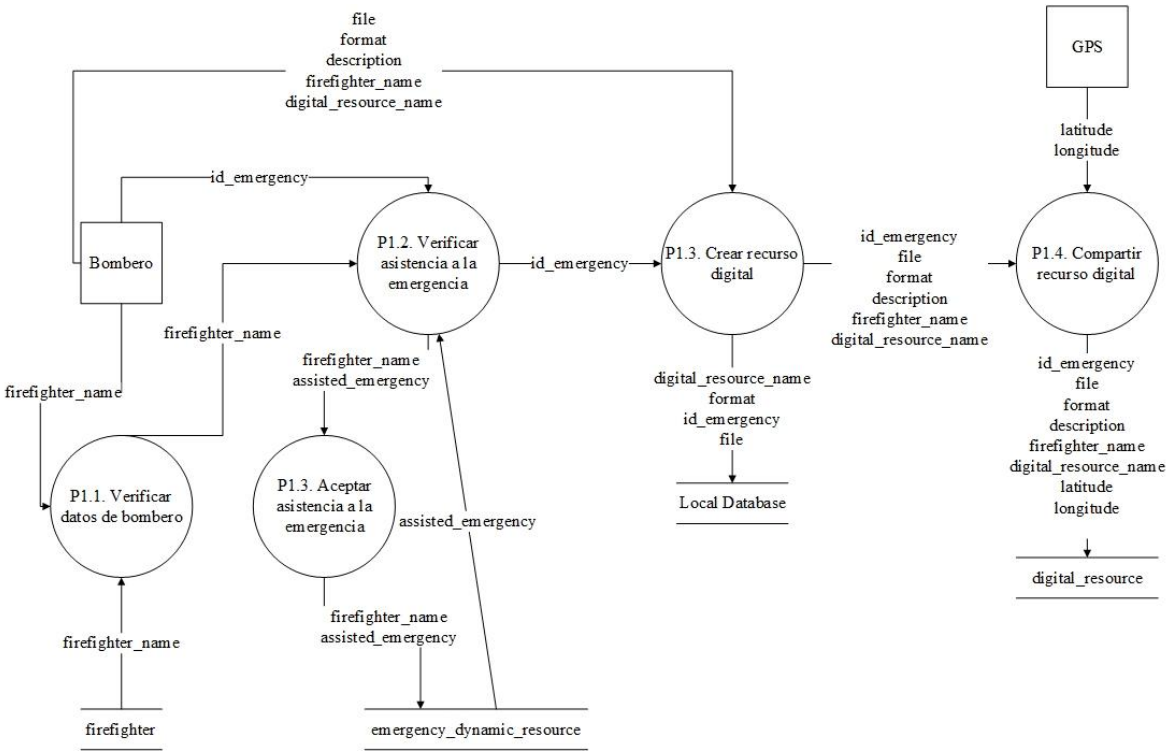


Figura 8. Diagrama de flujo de datos para crear y compartir audio

- *Visualización de ruta hacia la emergencia*: Esta funcionalidad, tal como lo muestra la Figura 9, consiste en que el usuario, luego de seleccionar una emergencia específica dentro del listado que aparece en el menú principal de la aplicación, escoja dentro del menú de la emergencia la opción para mostrar la ruta hacia la emergencia, esto recibiendo datos del repositorio tanto del usuario como de la emergencia para dibujar en el mapa la ruta que lo lleve hacia la emergencia.

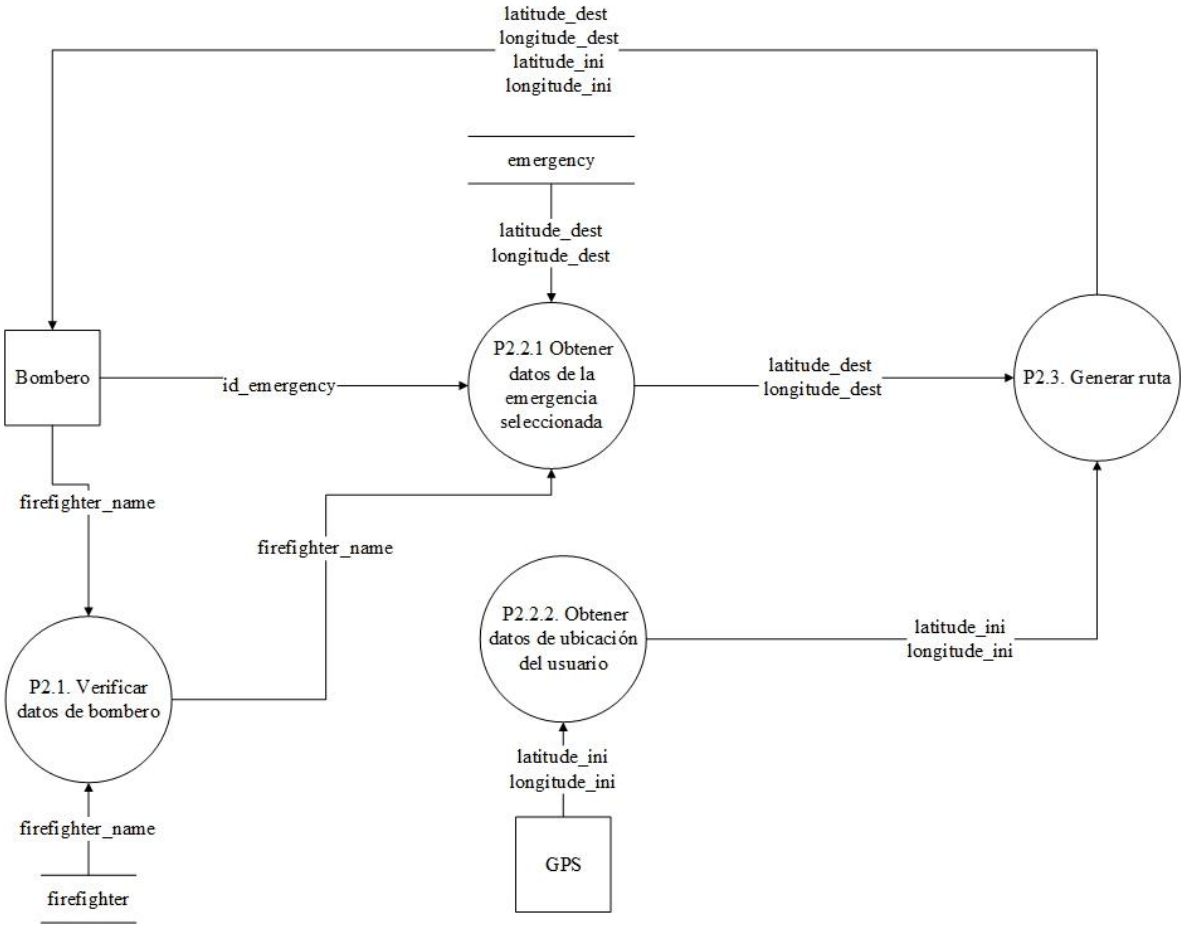


Figura 9. Diagrama de flujo de datos para visualizar ruta hacia la emergencia.

5.2. Diseño de interfaz de usuario

Como ya se mostraron los diagramas relacionados con el diseño de datos, se darán a conocer los aspectos que tienen que ver con la interfaz de usuario de la aplicación, en base a las funcionalidades desarrolladas y a su vez comentar su funcionamiento.

Para el inicio de la aplicación se observa la pantalla principal que consta de un listado con todas las emergencias ingresadas a la base de datos, mostrando para cada una el nombre de la emergencia, la distancia aproximada con respecto a la ubicación actual del usuario, el tiempo de inicio del incendio y la cantidad de carros bomba que asisten. Al escoger una de las emergencias, se despliega un menú de opciones que permitirá al usuario seleccionar la funcionalidad a la que desea ir dentro de la aplicación móvil, esto es, podrá seleccionar si asistirá a la emergencia, mostrar su ubicación actual en un mapa, mostrar en un mapa la ruta trazada hacia la ubicación de la emergencia, mostrar qué recursos están disponibles para la emergencia: Bomberos, Carros Bomba y Comandantes de Incidente y finalmente, crear y visualizar recursos digitales: Fotografía, Video y Audio (ver Figura 10).

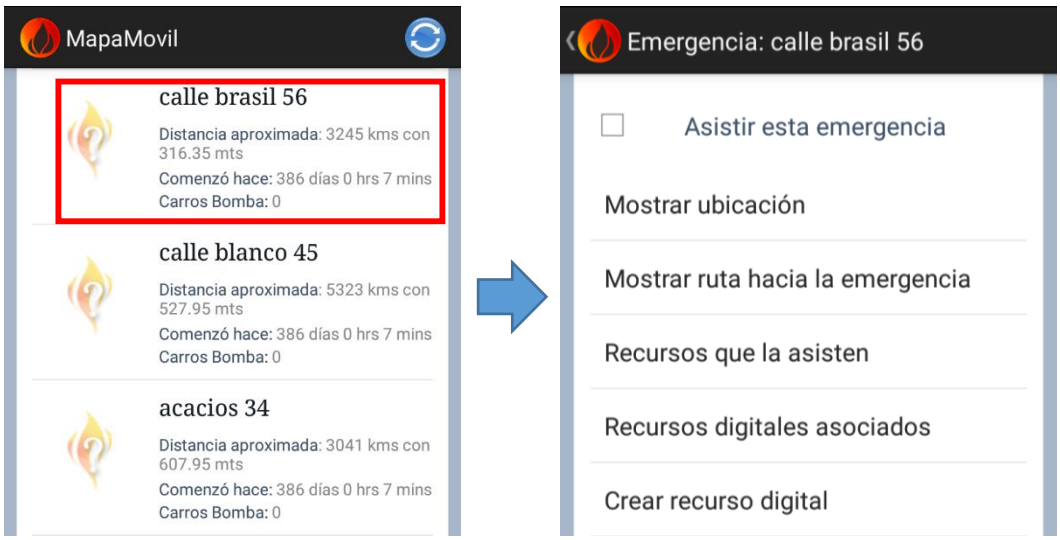


Figura 10. Menú de opciones asociado a una emergencia seleccionada del listado principal.

5.2.1. Interfaz componente de audio

En la Figura 11 se observa el despliegue de menú asociado a una emergencia para crear un recurso digital. Para grabar un audio el usuario deberá, en primer lugar, seleccionar la opción marcada en rojo “Crear recurso digital”, de la cual se desplegará un menú para crear tres tipos de recursos digitales disponibles: Fotografía, Video y Audio. En este caso, solo se verá la creación de audio, por lo que el usuario deberá seleccionar la opción “Grabar audio” y se desplegará el componente correspondiente para iniciar la grabación utilizando el grabador por defecto de Android. Esta grabación tiene una duración máxima que dependerá de la disponibilidad de almacenamiento que tenga el usuario en su dispositivo móvil. Luego de terminar la grabación, el usuario deberá decidir si almacenarla en la base de datos local o descartarla.

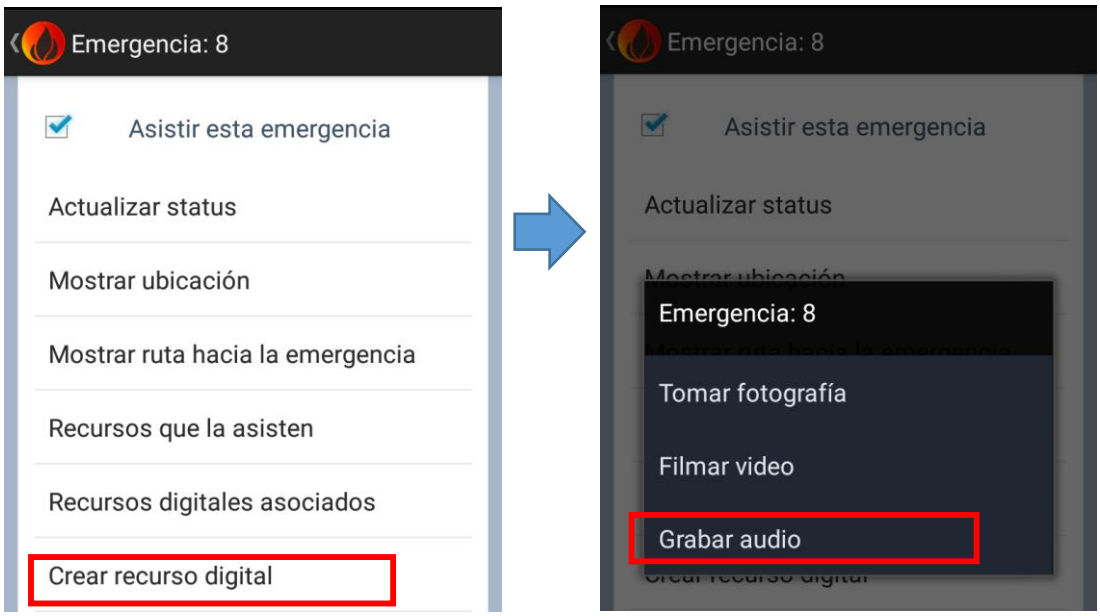


Figura 11. Interfaz del menú de una emergencia para la creación de audio.

En la Figura 12 se observa el proceso para compartir un audio, esto es, posterior a la grabación del audio el sistema desplegará un mensaje donde preguntará al usuario si desea compartir el archivo recién grabado (ver Figura 12(a)). Si el usuario no acepta compartir, simplemente volverá al menú de opciones para la creación de uno de los tres tipos de recurso digital (ver Figura 11). Por el contrario, si el usuario acepta compartir el recurso digital se desplegará un formulario, mostrado en la Figura 12(b) en el cuál el usuario puede seleccionar a qué destinatarios enviar el audio recién grabado y una descripción que es opcional. Luego de enviar el archivo, éste se compartirá mostrando al usuario una barra de progreso para luego desplegar un mensaje de éxito (ver Figura 12 (c) y (d), respectivamente).

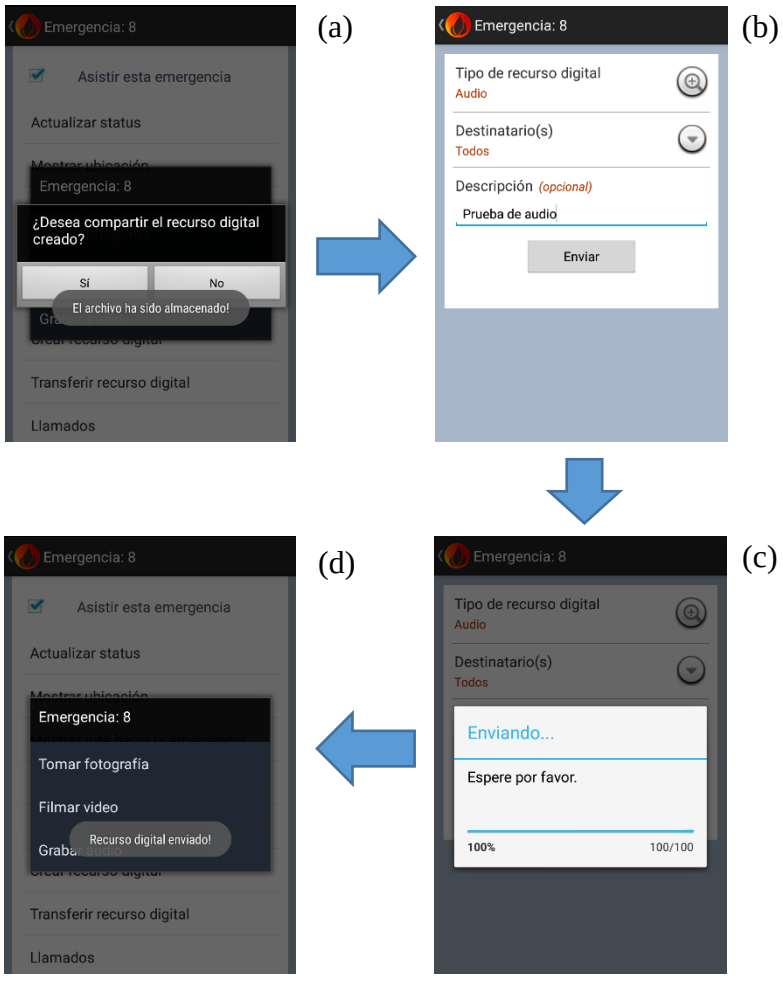


Figura 12. Proceso para compartir un audio grabado.

Todos los archivos de audio compartidos se verán reflejados en un listado al cuál se accede desde el menú de la emergencia y que está marcado en rojo en la Figura 13(a). Este listado de recursos digitales se divide dos secciones, una para los archivos “Nuevos” y otra para los archivos “Descargados”, cuya categorización dependerá de si es un recurso digital que ha sido descargado por el usuario o no, esto es, si el usuario selecciona un recurso digital de la sección “Nuevo” se descargará automáticamente el recurso digital y formará parte de la sección “Descargados” (ver Figura 13 (b), (c) y (d), respectivamente). Además de descargar el archivo, el sistema permite 2 opciones: Mostrar la ubicación donde está almacenado o abrir el recurso digital para su reproducción (ver Figura 13 (e) y (f), respectivamente).

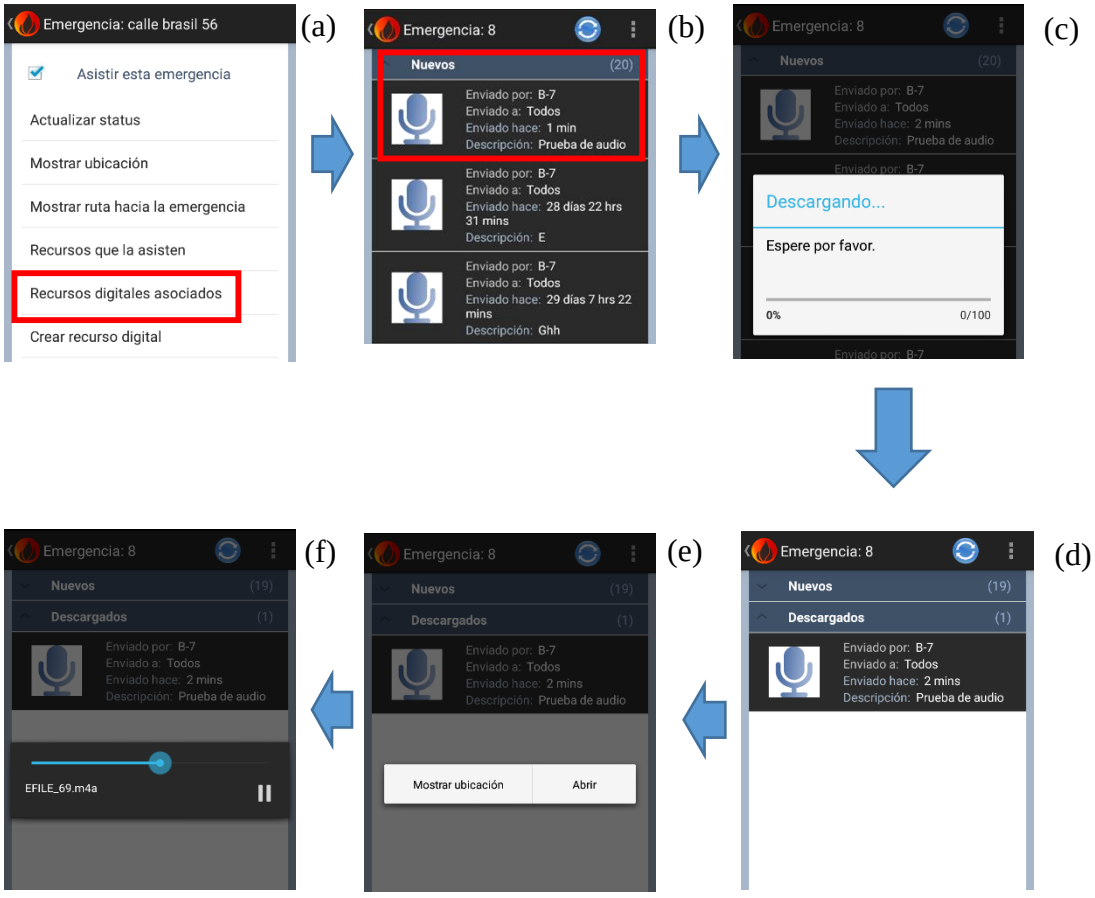


Figura 13. Proceso de mostrar al usuario los recursos digitales de una emergencia.

Finalmente, para integrar definitivamente el componente de audio, se establece el ícono de la Figura 14(c) que corresponde a aquel utilizado para definir a simple vista que el archivo compartido es un audio (Ver Figura 13(b)) y por otro lado, los otros dos íconos pueden visualizarse en el mapa correspondiente a cada emergencia para mostrar los recursos digitales asociados, ya sea uno nuevo, Figura 14(a), o uno descargado Figura 14(b). Estos íconos deben añadirse al *workspace* de Eclipse correspondiente al proyecto según la ruta "res/drawable-mdpi/" con el nombre de "ic_poi_audio", "ic_poi_downloaded_audio", "ic_list_audio" según la Figura 15 (a), (b) y (c) respectivamente.

(a) (b) (c)

Figura 14. Íconos ic_poi_audio, ic_poi_downloaded_audio e ic_list_audio.

5.2.2. Interfaz componente ruta hacia la emergencia

En la Figura 15 se observa el despliegue de menú asociado al trazado de la ruta hacia una emergencia. Para que el usuario pueda visualizar la ruta que debe seguir desde su ubicación actual hasta la ubicación de la emergencia seleccionada, este deberá seleccionar la opción marcada en rojo "Mostrar ruta hacia la emergencia", de la cual se desplegará un mapa con la información requerida.

Cabe mencionar que para esta funcionalidad, no es obligatorio estar asistiendo a la emergencia previamente seleccionada para visualizar la ruta.

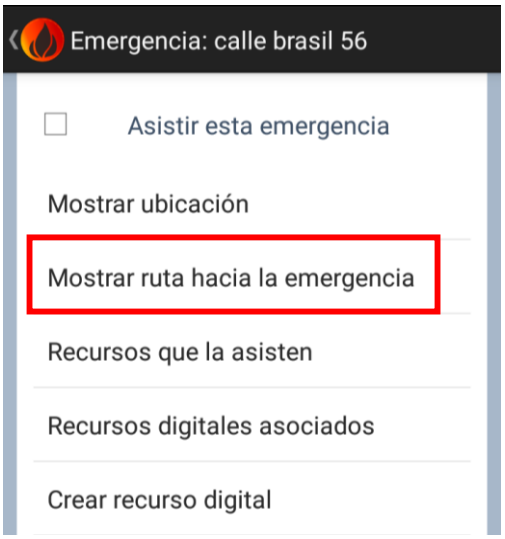


Figura 15. Selección para mostrar ruta en un mapa.

En la Figura 16, se observa que al seleccionar la opción para mostrar la ruta se desencadena un despliegue de la ruta en un mapa para la emergencia seleccionada, mostrando en primera instancia el lugar de ubicación del usuario marcado en rojo (ver Figura 16(a)), mientras que el trazado de la ruta se muestra como una línea de color azul dibujada por las calles.

Al desplazarse por el mapa, también se puede observar el destino de la ruta, que en este caso es donde se ubica la emergencia y se muestra el marcador correspondiente a la prioridad de la emergencia y que se encuentra encerrado en rojo en la Figura 16(b).

De la misma Figura es posible observar la distancia que existe entre la ubicación del usuario y la emergencia, además de un botón de sincronización que permite actualizar la ruta a medida que el usuario avanza. Además, se tiene que la ruta trazada en el mapa es procesada utilizando la ruta mas rápida para llegar al destino.

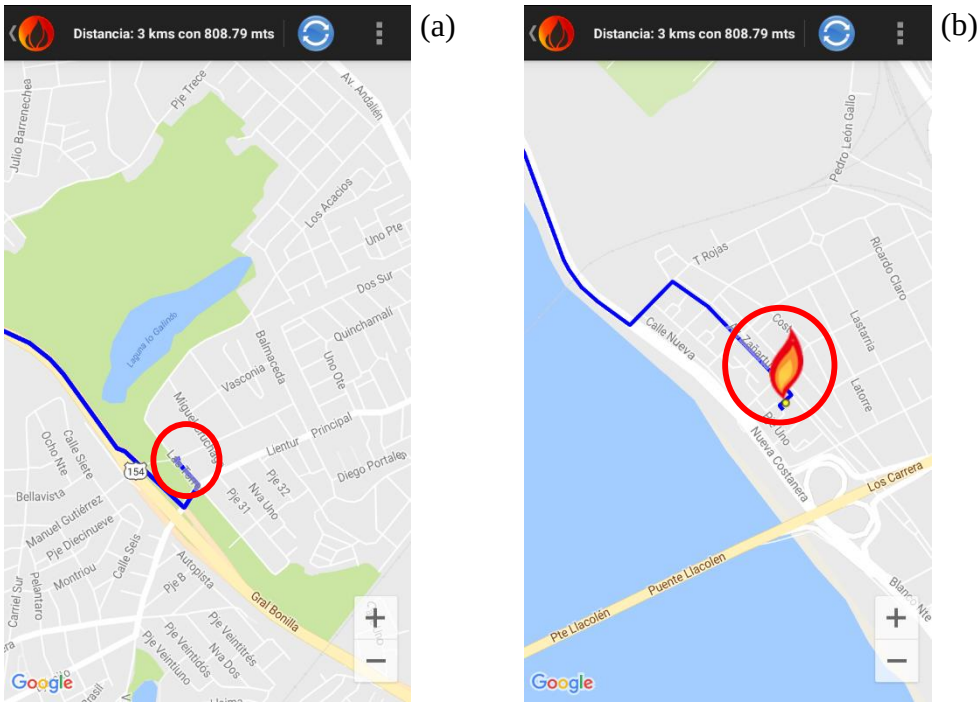


Figura 16. Visualización de ruta hacia la emergencia.

Finalmente, en la Figura 17, se puede observar un ejemplo de la ruta trazada en el mapa, desde una vista mas lejana, de tal forma de visualizar la ruta completa para una emergencia en específico.

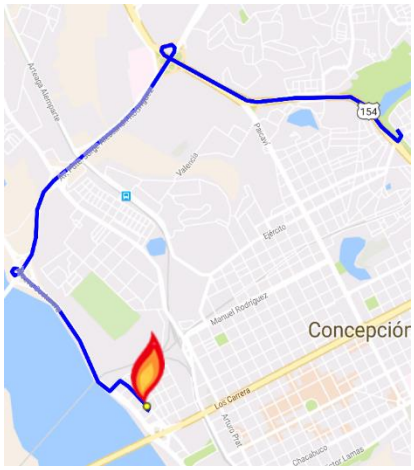


Figura 17. Visualización de ruta completa hacia la emergencia.

5.3. Diccionario de Datos

Debido a que en el presente proyecto no hubo modificaciones a la base de datos original, sino solo hubo adición de datos en la tabla "format" correspondiente a los formatos de los recursos digitales aceptados por el sistema, (ver sección 5.1.1. Nuevos datos en la Base de Datos), es que el diccionario de datos se construyó de acuerdo a la base de datos creada en el proyecto de título anterior [15].

A continuación se muestra el diccionario de datos correspondiente solo de la tabla a la cual se le agregaron datos:

format					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
format_name file_type	PK	VarChar(10)	Not Null Not Null	 file_types	

Tabla 2. Tabla de la Base de Datos para el formato de los archivos digitales.

Donde file_types puede ser del tipo:

- Fotografía
- Video
- Audio

Para revisar la tabla de datos completa del proyecto, dirigirse al Anexo A.

6. Implementación y Pruebas

En este capítulo se mostrarán las modificaciones que se hicieron al código Java de la aplicación *CollabMap,* desarrollada por Erika Ormeño, utilizando para ello el IDE Eclipse. Además, se mostrarán pruebas que permitirán ver el funcionamiento de la aplicación a través de distintos dispositivos móviles con el fin de observar cada funcionalidad desarrollada.

6.1. Desarrollo del proyecto

Para la instalación de la aplicación se escogió una de las tres aplicaciones disponibles elaboradas por Erika, para su posterior actualización. Estas 3 aplicaciones previamente desarrolladas son:

- [Application][Base] NewMobileMap.rar
- [Application][Only Basic] NewMobileMap.rar
- [Application][Original] NewMobileMap.rar

De ellas fue escogida la aplicación "[Base] NewMobileMap.rar" para realizar cambios al código fuente, porque de las tres mencionadas, es la más completa y representa el código fuente de la arquitectura base, además de contener todos los componentes desarrollados anteriormente.

6.1.1. Actualización en la Base de Datos

Antes de añadir los nuevos componentes a la aplicación, se debieron actualizar algunos datos en la Base de Datos original, con el fin de permitir al componente de audio, aceptar ciertos tipos de formatos predefinidos y que fueron mencionados en la sección 5.1.1 (Nuevos datos en la Base de Datos). Para ello, se deben seguir los siguientes pasos dependiendo si la Base de Datos ya está creada en el sistema o no.

1. Abrir pgAdmin III y luego seleccionar el editor SQL que allí se encuentra.

2. Abrir el archivo [Tables][Functions][Views][Triggers][Rules][Key Records]EDB_Emergencies_Response ubicado en \Erika\Server Files\[DB Files] Emergencies_Response_External_DB

Si no se ha creado la base de datos externa:

3. Modificar:

```
CREATE TYPE file_types AS ENUM

('Fotografía','Video');
```

Por:

```
CREATE TYPE file_types AS ENUM

('Fotografía','Video','Audio');
```

4. Después de la línea:

```
insert into format (format_name,file_type) values ('mkv','Video');
```

Añadir:

```
insert into format (format_name,file_type) values ('mp3','Audio');

insert into format (format_name,file_type) values ('aac','Audio ');
```

```
insert into format (format_name,file_type) values ('wav','Audio');
insert into format (format_name,file_type) values ('rec','Audio');
insert into format (format_name,file_type) values ('wma','Audio');
insert into format (format_name,file_type) values ('amr','Audio');
insert into format (format_name,file_type) values ('m4a','Audio');
```

5. Guardar cambios en el documento y seguir los pasos del manual de instalación del servidor y la base de datos externa según proyecto de título anterior [15].

Si ya se ha creado la base de datos externa:

3. Después de:

```
CREATE TYPE file_types AS ENUM
('Fotografía','Video');
```

Añadir la línea:

```
ALTER TYPE file_types ADD VALUE 'Audio' AFTER 'Video';
```

4. Seleccionar la línea añadida en el paso anterior y presionar F5 para agregar el tipo “Audio” a la base de datos ya instalada.

5. Después de la línea:

```
insert into format (format_name,file_type) values ('mkv','Video');
```

Añadir:

```
insert into format (format_name,file_type) values ('mp3','Audio');
insert into format (format_name,file_type) values ('aac','Audio ');
insert into format (format_name,file_type) values ('wav','Audio');
insert into format (format_name,file_type) values ('rec','Audio');
insert into format (format_name,file_type) values ('wma','Audio');
insert into format (format_name,file_type) values ('amr','Audio');
insert into format (format_name,file_type) values ('m4a','Audio');
```

6. Seleccionar las líneas añadidas en el paso anterior, de una en una, y presionar F5 para agregar los formatos a la base de datos ya instalada.

6.1.2. Desarrollo del componente de audio

Los archivos creados/modificados, ordenados según los paquetes listados por defecto en la aplicación, fueron:

Paquete	Nombre clase
cl.ucsc.projectoftitle.newmobilemap.application.ui	▪ DigitalResourcesMaker
cl.ucsc.projectoftitle.newmobilemap.components.compmanagers.optional	▪ ConcreteOptionalComponentsA ▪ DefaultOptionalComponents

	▪ OptionalComponents
cl.ucsc.projectoftitle.newmobilemap.components. digitalmakers	▪ ConcreteAudioMakerA ▪ AudioMaker
cl.ucsc.projectoftitle.newmobilemap.components. digitalopeners	▪ ConcreteAudioOpenerA ▪ AudioOpener ▪ ConcreteDigitalResourcesOpener
cl.ucsc.projectoftitle.newmobilemap.components.entities. types	▪ DigitalResourceTypes

Tabla 3. Archivos creados/modificados para el componente de audio.

Principalmente, para implementar el componente de audio fue necesario crear la clase "ConcreteAudioMakerA", que permite crear el audio y cuya interfaz está implementada por la clase creada "AudioMaker".

El Audio será capturado mediante el grabador que provee Android, lo que es ventajoso pues la aplicación a utilizar será conocida por el usuario. Para esto se invocará la aplicación de grabación recuperando posteriormente el audio grabado, tal como se observa en la Figura 18, en el cuál al presionar el botón de grabar audio, por medio del Intent *takeAudioIntent*, se activa el grabador de Android por defecto y se procede a la grabación.

Posteriormente, se llama al método *startActivityForResult* para recuperar dicha grabación después de finalizada, a través del *ActivityResult* que se encuentra ya definido en la "DigitalResourcesMaker.java", donde el archivo es guardado en la base de datos local, esto es, en el almacenamiento del dispositivo móvil. Además, se da la opción de compartirlo con otros usuarios que también asisten a la misma emergencia (ver anexo B para mayor detalle del código).

```
if(savePath != null && UIInstance != null){

    if(UIInstance instanceof Activity){

        Uri audioUriPath = Uri.fromFile(new File(savePath));

        Intent takeAudioIntent = new Intent();
        takeAudioIntent.setAction(android.provider.MediaStore.Audio.Media.RECORD_SOUND_ACTION);

        takeAudioIntent.putExtra(MediaStore.EXTRA_OUTPUT, audioUriPath);
        ((Activity) UIInstance).startActivityForResult(takeAudioIntent, 1);
```

Figura 18. Clase ConcreteAudioMakerA.java que permite crear de audio.

Luego de efectuar la grabación y compartirla, es posible acceder a su reproducción. Para ello, se debió implementar la clase "ConcreteAudioOpenerA" la cual es la encargada de la apertura del recurso digital de Audio a través de su interfaz creada por la clase "AudioOpener". En la Figura 19 se puede observar el código correspondiente a la apertura del audio, a través de un Intent se llama a la acción ACTION_VIEW que permite mostrar al usuario la información, en este caso el audio, por medio del *startActivity()*.

```
if(file != null && UIInstance != null){

    if(UIInstance instanceof Activity){

        Uri uriPath = Uri.fromFile(file);

        Intent openIntent = new Intent();
        openIntent.setAction(Intent.ACTION_VIEW);
        openIntent.setDataAndType(uriPath,"audio/*");

        ((Activity) UIInstance).startActivity(openIntent);

    }
```

Figura 19. Clase ConcreteAudioMakerA.java que permite reproducir un audio.

Por otro lado, para incorporar la mayor cantidad de dispositivos móviles, se debieron agregar los formatos de audio mencionados en la sección 5.1.1 (Nuevos datos en la Base de Datos) y que fueron también agregados a la base de datos externa, además de los

íconos mostrados en la Figura 14 (sección 5.2.1 Interfaz componente de audio). Estos formatos e íconos fueron añadidos a la clase "DigitalResourceTypes", tal como se muestra en la Figura 20, otorgando a cada uno una etiqueta que hace referencia a sus nombres.

Es importante mencionar que el valor de TYPE_NAME debe ser idéntico a como fue establecido en el dominio "file_types" de la base de datos externa, que para este componente toman el valor "Audio", tal como lo muestra la Figura 20. Así mismo, el valor de las constantes FORMAT_ debe ser idéntico al establecido en la tabla "format" de la base de datos externa.

```
public interface Audio{
    public static final String TYPE_NAME = "Audio";
    public static final String TYPE_NAME_IN_PLURAL_FOR_UI = "Audios";
    public static final String TYPE_POI_ICON_NAME = "ic_poi_audio";
    public static final String TYPE_DOWNLOAD_POI_ICON_NAME = "ic_poi_downloaded_audio";
    public static final String TYPE_LIST_ICON_NAME = "ic_list_audio";

    public static final String FORMAT_MP3 = "mp3";
    public static final String FORMAT_AAC = "aac";
    public static final String FORMAT_WAV = "wav";
    public static final String FORMAT_REC = "rec";
    public static final String FORMAT_WMA = "wma";
    public static final String FORMAT_AMR = "amr";
    public static final String FORMAT_M4A = "m4a";
}
```

Figura 20. Valor de las constantes correspondientes al formato de audio.

6.1.3. Desarrollo del componente de ruta hacia la emergencia

Los archivos creados/modificados, ordenados según los paquetes listados por defecto en la aplicación, fueron:

Paquete	Nombre clase
cl.ucsc.projectoftitle.newmobilemap.application.ui	▪ EmergencyMenu ▪ RouteToEmergencyMap
cl.ucsc.projectoftitle.newmobilemap.application.utilities	▪ CommonLabels

cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.json	▪ DirectionsJSONParser

Tabla 4. Archivos creados/modificados para el componente de ruta.

Para incorporar este nuevo componente, lo primero que se hizo fue mostrar en el menú principal de cada emergencia la opción "Mostrar ruta hacia la emergencia". De esta forma el usuario podrá acceder a la ruta de una emergencia previamente seleccionada. En la Figura 21, se muestra lo anterior y que fue implementado, en primer lugar, en la clase "CommonLabels" donde se declaró la etiqueta LABEL_SHOW_ROUTE_TO_EMERGENCY_MAP, mostrada en rojo, y que toma el valor "Mostrar ruta hacia la emergencia" para luego hacerse visible para el usuario en el menú de opciones.

```
public static final String LABEL_SHOW_USER_STATUS_FORM = "Actualizar status";
public static final String LABEL_SHOW_EMERGENCY_MAP = "Mostrar ubicación";
public static final String LABEL_SHOW_DYNAMIC_RESOURCES_LIST = "Recursos que la asisten";
public static final String LABEL_SHOW_DIGITAL_RESOURCES_LIST = "Recursos digitales asociados";
public static final String LABEL_SHOW_DIGITAL_RESOURCES_MAKER = "Crear recurso digital";
public static final String LABEL_SHOW_ROUTE_TO_EMERGENCY_MAP = "Mostrar ruta hacia la emergencia";
```

Figura 21. Declaración de etiqueta para agregar al menú de opciones.

En tanto en la clase "EmergencyMenu", se hace uso de la etiqueta declarada anteriormente, añadiéndola al listado de opciones que surge después de haber seleccionado una emergencia en específico. Si se observa la Figura 22, solo se añadió la etiqueta utilizando el método add, a diferencia de LABEL_SHOW_DIGITAL_RESOURCES_LIST o LABEL_SHOW_DIGITAL_RESOURCES_MAKER, donde es necesario comprobar que el usuario está asistiendo a la emergencia antes de ejecutar una acción. En otras palabras, el usuario puede ver la ruta hacia una determinada emergencia, asista o no a ella.

```
tmp.add(CommonLabels.LABEL_SHOW_EMERGENCY_MAP);
tmp.add(CommonLabels.LABEL_SHOW_ROUTE_TO_EMERGENCY_MAP);
tmp.add(CommonLabels.LABEL_SHOW_DYNAMIC_RESOURCES_LIST);

if(reflectionUtilities.checkExistence(DigitalResourcesList.class))
    tmp.add(CommonLabels.LABEL_SHOW_DIGITAL_RESOURCES_LIST);

if(reflectionUtilities.checkExistence(DigitalResourcesMaker.class))
    tmp.add(CommonLabels.LABEL_SHOW_DIGITAL_RESOURCES_MAKER);
```

Figura 22. Código para visualizar la etiqueta al menú de opciones.

Para trazar la ruta, es necesario obtener la latitud y longitud tanto del origen como del destino de ella, donde el origen está dado por la ubicación actual del bombero y el destino será la ubicación de la emergencia. En la Figura 23, se muestra la forma de obtener esas latitudes y longitudes de origen y destino. Para el origen, la latitud y longitud se obtiene desde la API de *Google Maps* utilizando la variable "startPoint" marcada en rojo, para luego establecer una marca en el mapa correspondiente. El origen puede tomar valores variables que dependerán de la ubicación del usuario. En tanto, el destino tomará un valor fijo que se guardará en la constante "destinyPoint", marcada en verde, y se obtendrá desde la base de datos externa del sistema debido a que la latitud y longitud del destino se determinan al momento de ingresar una emergencia en la aplicación. Al igual que en el origen, luego de obtener el destino se fijará una marca en el mapa mostrando la ubicación de la emergencia.

```
    LatLng startPoint = new LatLng(mLatitude, mLongitude);

    mMarkerPoints.add(startPoint);
    MarkerOptions optionStart = new MarkerOptions();

    // Configurando la posición del marcador de origen
    optionStart.position(startPoint);
    optionStart.icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_poi_available_user));
    mGoogleMap.addMarker(optionStart);

}
LatLng destinyPoint = new LatLng(emergency.getLatitude(), emergency.getLongitude());
mMarkerPoints.add(destinyPoint);

MarkerOptions optionsDestiny = new MarkerOptions();

// Configurando la posición del marcador de destino
optionsDestiny.position(destinyPoint);
optionsDestiny.icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_poi_emergency));
mGoogleMap.addMarker(optionsDestiny);
```

Figura 23. Código para la obtención de las latitudes y longitudes.

Luego de obtener las latitudes y longitudes del origen y destino, se procede a llamar al método "getDirectionsUrl" de la Figura 24, en el cuál se construye la URL que posteriormente se enviará al *Web Service* de *Google Maps*, con el fin de obtener una serie de puntos con la ruta a trazar, lo que devolverá con formato JSON. Estos puntos de la ruta luego serán enviados a una tarea de descarga en segundo plano llamada "DownloadTask" (Ver clase en anexo C), la cual permitirá descargar datos desde Google Directions URL.

```
private String getDirectionsUrl(LatLng origin,LatLng dest){

    String str_origin = "origin="+origin.latitude+","+origin.longitude;

    String str_dest = "destination="+dest.latitude+","+dest.longitude;

    String sensor = "sensor=false";

    String parameters = str_origin+"&"+str_dest+"&"+sensor;

    String output = "json";

    String url = "https://maps.googleapis.com/maps/api/directions/"+output+"?"+parameters;

    return url;
}
```

Figura 24. Método que permite obtener los puntos para trazar una ruta.

Posteriormente, en la Figura 25 se muestra un extracto código de la clase "ParserTask", la cual interpreta el resultado obtenido del *Web Service* de *Google Maps* y crea así lo puntos que conforman la ruta desde el origen hasta la ubicación de la emergencia. Luego de interpretar y obtener la ruta hacia la emergencia, se otorgan características tales como color y grosor de la línea a trazar (marcado en rojo), para finalmente dibujarlas en el mapa formando una ruta (marcado en verde) desde la ubicación del bombero hasta la ubicación de la emergencia seleccionada.

Para este componente se estableció el color azul para la línea trazada y un grosor de tamaño 8.

```
        // Obteniendo la i-ésima ruta
        List<HashMap<String, String>> path = result.get(i);

        // Obteniendo todos los puntos en la i-ésima ruta
        for(int j=0;j<path.size();j++){
            HashMap<String,String> point = path.get(j);

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

            points.add(position);
        }

        // Añadiendo todos los puntos a la ruta con LineOptions
        lineOptions.addAll(points);
        lineOptions.width(8); //Grosor linea de la ruta
        lineOptions.color(Color.BLUE);
    }

    // Dibujando el polyline en el Mapa
    mGoogleMap.addPolyline(lineOptions);
```

Figura 25. Extracto de código de la clase "ParserTask".

6.2. Pruebas del proyecto

En este capítulo, se mostrarán las pruebas que se realizaron para observar el funcionamiento de la nueva aplicación generada. Cabe mencionar que se utilizaron los siguientes *smartphones* para llevar a cabo las pruebas:

Dispositivo Móvil	Sistema Operativo
Moto G3	Android 6.0
Huawei	Android 6.0.1
Lenovo A6020	Android 5.1.1
Own S4035	Android 5.1
LG G3 Stylus	Android 5.0.2

Tabla 5. Dispositivos en los que se ejecutó la aplicación móvil para ser testeada.

6.2.1. Pruebas al componente de Audio

A continuación se dará a conocer la secuencia de acciones, con respecto al funcionamiento de la aplicación y de aquello que se espera que el componente de audio haga durante la ejecución del sistema:

1. Abrir la aplicación CollabMap para acceder al servidor y a la base de datos.
2. Seleccionar una emergencia del listado.
3. Seleccionar “Asistir esta emergencia”.
4. Seleccionar del menú “Crear Recurso Digital”.
5. Seleccionar el recurso digital “Grabar audio”.
6. Grabar mensaje de audio y aceptar.
7. Seleccionar “Sí” en la ventana de diálogo ¿Desea compartir el recurso digital creado?
8. Llenar el formulario con mensaje opcional, seleccionar destinatarios y presionar Enviar.
9. Seleccionar “Sí” en la ventana de diálogo “¿Desea enviar el recurso digital?”
10. Esperar mensaje exitoso “Recurso digital enviado!”.

La secuencia anterior, se ejecutó en el mismo orden en que están listadas las acciones y se cumple lo siguiente para cada uno de los 5 dispositivos móviles de la Tabla 5.

N° acción	Motorola	Huawei	Lenovo	Own	LG
1	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓

5	✓	✓	✓	✓	✓
6	×	✓	✓	✓	✓
7	×	✓	✓	✓	✓
8	×	✓	✓	✓	✓
9	×	✓	✓	✓	✓
10	×	✓	✓	✓	✓

Tabla 6. Análisis sobre qué acciones fueron exitosas durante la prueba de audio.

Se observa de la Tabla 6 que los dispositivos móviles Huawei, Lenovo, Own y LG, funcionan sin problemas tanto en la creación del audio como en el envío de éste. Sin embargo, el *smartphone* Moto G3, no cumple con iniciar el grabador, debido a que el dispositivo no tiene incorporado el grabador de audio que provee Android. Éste error se produjo luego de una actualización el sistema del Smartphone mencionado.

Por otro lado, si el usuario en el punto 7 de la secuencia selecciona la opción contraria, es decir "No", se observó que la grabación se guardó satisfactoriamente en cada base de datos local de su respectivo dispositivo móvil, esto es, en el almacenamiento predeterminado del *smartphone*. Así mismo, si el usuario selecciona "No" en la opción 9 de la secuencia de opciones, se cancela el envío del recurso digital, guardándose solo en la base de datos local de los dispositivos.

6.2.2. Pruebas al componente de ruta hacia la emergencia

A continuación se dará a conocer la secuencia de acciones, con respecto al funcionamiento de la aplicación y de aquello que se espera que el componente de ruta hacia la emergencia haga durante la ejecución del sistema:

1. Abrir la aplicación CollabMap para acceder al servidor y a la base de datos.
2. Seleccionar una emergencia del listado.
3. Seleccionar del menú “Mostrar Ruta hacia la emergencia”.
4. Visualizar ruta desde la ubicación actual del usuario hasta la ubicación de la emergencia.

La secuencia anterior se ejecutó en el mismo orden en que están listadas las acciones y se cumple lo siguiente para cada uno de los 5 dispositivos móviles de la Tabla 7.

N° acción	Motorola	Huawei	Lenovo	Own	LG
1	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓

Tabla 7. Análisis sobre qué acciones fueron exitosas durante la prueba de ruta.

Las pruebas que se realizaron para este componente resultaron satisfactorias con respecto a la secuencia de acciones listada anteriormente. Mencionar además, que al momento de desviarse de la ruta, la aplicación se actualiza calculando una nueva ruta respecto de la nueva ubicación del usuario. Este nuevo cálculo de ruta sucede al momento de la sincronización de la aplicación, tanto automática como manual, esto último, presionando el botón de actualización ubicado en la esquina superior derecha.

Finalmente, si el usuario visualiza otra sección o lugar en el mapa y pierde la visualización de la ruta, es posible volver a la ubicación actual del usuario seleccionando la opción “Ver mi ubicación”. Esto se pudo observar en cada uno de los dispositivos móviles en los que se ejecutó la aplicación.

7. Conclusiones

A continuación, se analizará el cumplimiento de los objetivos del proyecto y si efectivamente se llevaron a cabo según lo requerido. Además, se mencionarán las dificultades encontradas y trabajos futuros asociados al presente trabajo de título.

7.1. Cumplimiento de objetivos

El cumplimiento de los objetivos específicos del proyecto y que, por tanto, llevan a cumplir el objetivo general son analizados a continuación:

- El *estudio de los componentes de software desarrollados en el proyecto de título anterior* fue necesario debido que el presente proyecto es una mantención de él. Este estudio se llevó a cabo satisfactoriamente, utilizando el informe de Erika Ormeño [15] como guía para comprender el funcionamiento de los componentes y de la aplicación en general, con el propósito de integrar los nuevos componentes de forma correcta. Además, también hubo un estudio con respecto a la base de datos utilizada por estos componentes y que son fundamentales para el funcionamiento de los nuevos componentes.

- Se *establecieron nuevos productos en la familia de productos pre-establecida*, los cuales fueron definidos en el Análisis del proyecto (ver sección 4.2, Funcionalidades desarrolladas) y que implicó ampliar la familia de productos con respecto a la ya existente, conforme a las necesidades actuales de los usuarios (bomberos).

  El resultado final que persigue este objetivo es crear una mayor cantidad de aplicaciones móviles, utilizando la reutilización de código, para que pueda ser utilizado en las distintas actividades que realiza un bombero dependiendo de su área; por lo tanto, agregar nuevos productos a la familia es un punto importante dentro de

la mantención de la aplicación. Sin embargo, la cantidad de componentes creados en este proyecto no necesariamente es la adecuada para obtener una aplicación móvil integral y que se adecúe a las necesidades de bomberos en cada emergencia.

- En base al punto anterior acerca de ampliar la familia de productos y de los componentes elegidos para ello, se *desarrollaron nuevos componentes de software de tal forma que puedan ser reutilizados posteriormente*; esto se llevó a cabo en la implementación del proyecto según el modelo de desarrollo escogido (ver sección 6.1.), donde se observa a grandes rasgos la forma de integrar cada componente. Se destaca que al agregar estas nuevas funcionalidades no se modificó en lo absoluto el funcionamiento de la aplicación original.

  Por otro lado, mencionar que los nuevos componentes fueron desarrollados y probados individualmente e independiente de la aplicación móvil original con el fin de adaptar el componente a lo que realmente se requiere y facilitar la corrección de errores para posteriormente integrarlo a la aplicación.

- La c*onstrucción de una nueva aplicación en base a los nuevos componentes desarrollados* se llevó a cabo integrando las nuevas funcionalidades a la aplicación original. Tal como se mencionó anteriormente, esta adición no influye en nada en el funcionamiento de los componentes ya existentes. La nueva aplicación obtenida corresponde al *CollabMap* original más los componentes de envío y recepción de audio y el componente de trazar una ruta hacia la ubicación de la emergencia.

Con lo anterior, se cumple el objetivo general ya descrito en los objetivos del proyecto (ver sección 1.2) y permiten ampliar la familia de productos ya existentes incorporando los nuevos componentes creados. Además, estos componentes pueden reutilizarse en un futuro para construcción de nuevas características y aplicaciones que complementen el trabajo de bomberos según su necesidad.

Expandir la familia de productos existentes en el dominio de administración de emergencias, por medio del desarrollo de componentes reutilizables a partir de la aplicación móvil *CollabMap*.

7.2. Dificultades encontradas

Algunas de las dificultades encontradas fueron:

- En primera instancia hubo dificultad en lo que respecta a la instalación de la aplicación móvil en el sistema, pues el IDE arrojaba errores que tenían que ver con los paquetes "*Google Play Services*" y "*Android Support Library*", esto debido a la versión que se utilizó al intentar instalar la aplicación. Lo anterior se solucionó utilizando los paquetes de la misma versión que se utilizó en el proyecto de título anterior [15] y que dejó disponibles para ser utilizadas para los desarrollos posteriores a los de ella.
- Luego de ser instalada correctamente la aplicación en el computador personal con sistema operativo Windows y luego en el dispositivo móvil con sistema operativo Android 5.1, el *smartphone* comenzó a tener problemas con el envío y recepción de fotografías y videos, pues al intentar crear un recurso digital, la aplicación fallaba y se cerraba. Este inconveniente fue resuelto desactivando las dependencias en las propiedades del proyecto "- Base2 – NewMobileMap" tal como se muestra en la Figura 26. Desactivar estas dependencias permite un funcionamiento correcto de la aplicación original y continuar así con el desarrollo del presente trabajo de título.

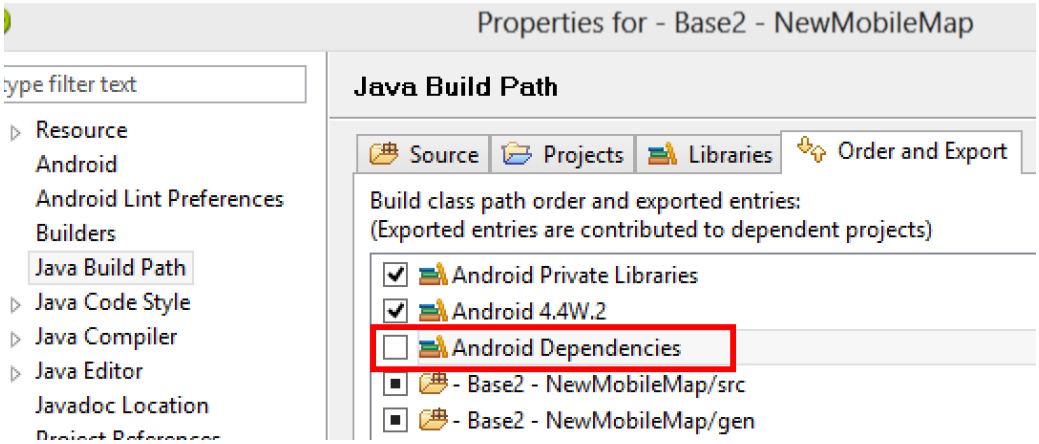


Figura 26. Propiedades del proyecto en el IDE Eclipse.

- Durante el desarrollo de los componentes surgió un problema integración de los componentes nuevos a los ya desarrollados, pues inicialmente se creaba el componente independiente de la aplicación original. Finalmente, esto se solucionó haciendo un estudio más exhaustivo de *CollabMap*, para saber en qué lugar debieron crearse las funcionalidades nuevas.

- Debido a las actualizaciones de uno de los dispositivos móviles que sucedió después de unos meses desde el comienzo del proyecto, el componente de audio dejó de funcionar en el *smartphone* Moto G3, siendo que inicialmente funcionaba correctamente. El sistema operativo Android del *smartphone* Moto G3 se actualizó de la versión 5.1 a la versión 6.0, lo que no permitió acceder a un grabador de audio por defecto, pues no existía.

7.3. Trabajos Futuros

Debido a que este trabajo de título es una mantención a un sistema ya creado, la idea de extenderlo agregando nuevos productos a la familia de productos que ya existe se hace aún

más necesaria, debido a que siempre se generarán nuevas necesidades para bomberos como también cambios en los dispositivos móviles.

Cada vez se ve como surgen nuevos dispositivos móviles con más funcionalidades, sensores incorporados, actualizaciones de sistema, entre otros, y que tarde o temprano llevarán a actualizarla de acuerdo a los nuevos requerimientos, así como también ampliarse al sistema iOS. Además de los cambios en la tecnología, también puede extenderse la aplicación con respecto al tipo de emergencia, pues actualmente solo se aplica a los incendios.

Finalmente, existen nuevos componentes que pueden agregarse y que estuvieron inicialmente contemplados en el proyecto, los que son:

- Uso de sensores que permitan verificar los signos vitales de bomberos.
- Envío y recepción de recursos digitales que ya se encuentren almacenados en el dispositivo móvil.
- Recepción de notificaciones ante nuevos eventos tales como una nueva emergencia, nuevo recurso digital recibido proveniente de otro bombero.

Glosario

- **Android:** Sistema operativo desarrollado por Google, destinado principalmente a su uso en dispositivos móviles con pantalla táctil tales como *smartphones* o tablets. Se programa principalmente en Java y su núcleo está basado en Linux [4].

- **ADT (Android Development Tools)**: Herramienta de desarrollo que instala una serie de complementos en Eclipse, de forma que el entorno de desarrollo se adapte al desarrollo de aplicaciones para Android [9].

- **Eclipse:** Entorno de desarrollo integrado y de código abierto basado en Java. Esta plataforma entrega las herramientas para la programación, desarrollo y compilación de elementos como páginas web, aplicaciones Java o programas en C++ [7].

- **Java:** Lenguaje de programación comercializada por Sun Microsystems y que está presente en computadores, centros de datos, consolas para videojuegos, teléfonos móviles, Internet, entre muchos otros usos [14].

- **API (Application Programming Interface):** Conjunto de funciones y procedimientos que cumplen una o muchas funciones con el fin de ser utilizadas por otro software [16].

- **Dalvik:** Máquina virtual basada en registros creada por Google, que facilitan la optimización de recursos, ya sea por poseer un procesador limitado o de poca memoria en un dispositivo [9].

- **PostgreSQL:** Es un sistema multiplataforma de administración de bases de datos relacionales orientadas a objetos. PostgreSQL usa un modelo cliente/servidor y es un

sistema basado en Open Source, esto quiere decir que posee licencia libre, por lo que cualquier persona puede usar, modificar y distribuir PostgreSQL de manera libre y gratuita para cualquier propósito, sea privado, comercial o académico [12].

- **JDK (Java Development Kit):** Es un software que provee herramientas de desarrollo para la creación de programas en java. Puede instalarse en una computadora local o en una unidad de red [11].

Referencias Bibliográficas

[1] APLICATIVA. (s.f.). *Google Play*. Recuperado el 18 de 08 de 2015, de https://play.google.com/store/apps/details?id=net.aplicativa.apps.bomberoscr&hl=es_419

[2] CLEMENTS, P., & NORTHROP, L. (2002). *Software Product Lines: Practices and Patterns* (Tercera ed.). Addison Wesley.

[3] CONASET. (2001). Manual de Operaciones Multi-institucional ante Emergencias. Chile.

[4] DEPTO. DE CIENCIA DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL. (2014). Libro Electrónico. *Desarrollo de aplicaciones para Android*. Alicante, España.

[5] ENGELBRECHT, A., BORGES, M., & VIVACQUA, A. (2011). Digital Tabletops for Situational Awareness in Emergency Situations. *15th International Conference on Computer Supported Cooperative Work in Design.* Rio de Janeiro, Brasil.

[6] FIGUEROA, C. (2010). Proyecto de Título. *Reutilización del Software, Metodologías y Aplicaciones más Frecuentes*. Guatemala.

[7] GALLARDO, D. (2012). *IBM developersWorks.* Obtenido de https://www.ibm.com/developerworks/ssa/library/os-ecov/

[8] GAMMA, E., HELM, R., JOHNSON, R., & VLISSIDES, J. (2003). *Patrones de Diseño: Elementos de Software Orientado a Objetos Reutilizables* (Primera ed.). Madrid: Pearson Educación.

[9] GIRONÉS, J. T. (2013). *El gran libro de Android.* México: Alfaomega Grupo Editor.

[10] INTERNATIONAL DATA CORPORATION. [s.a]. *Worldwide Quarterly Mobile Phone Tracker*. Obtenido de http://www.idc.com/promo/smartphone-market-share/vendor

[11] LATORRE, G. (22 de Marzo de 2010). *Programación II.* Recuperado el Abril de 2017, de http://gl-epn-programacion-ii.blogspot.cl/2010/03/jvm-jdk-jre-conceptos-fundamentales-de.html

[12] LÓPEZ ACEVEDO, L. (21 de Julio de 2017). Tutorial de PostgreSQL. *Publicación 9.1.0.* Recuperado el Abril de 2017

[13] MONTILVA, J., ARAPÉ, N., & COLMENARES, J. (2003). Artículo. *Desarrollo de sofware basado en componentes*.

[14] ORACLE CORPORATION. ([s.a.]). *Java.com*. Recuperado el Abril de 2017, de Conozca más sobre la tecnología Java: https://www.java.com/es/about/

[15] ORMEÑO SANHUEZA, E. (2015). *Construcción de componentes reutilizables para la elaboración de aplicaciones móviles que apoyen el trabajo colaborativo en emergencias.* Proyecto de Título, Universidad Católica de la Santísima Concepción, Concepción.

[16] PÉREZ PORTO, J., & GARDEY, A. (2017). *Definicion.de*. Recuperado el Abril de 2017, de Definición de API: http://definicion.de/api/.

[17] ROSSEL CID, P., & HERSKOVIC, V. (2013). Building a Domain Model for Mobile Collaborative Systems: Towards a Software Product Line. *19th International Conference on Collaboration and Technology*, 290-305.

[18] SAMETINGER, J. (1997). *Software Engineering with Reusable Components* (Primera ed.). Berlin: Springer.

[19] SCHACH, S., & TOMER, A. (2000). Development/Maintenance/Reuse: Software Evolution in Product Lines. *Software Product Lines; Experiences and Research Directions.* Denver, Colorado, USA: Springer.

[20] SIU, T. (2012). Trabajo de Título. *Interfaces Humano-Computador en Dispositivos Móviles para Situaciones de Emergencia*. Santiago, Chile.

[21] SOMMERVILLE, I. (2011). *Libro Ingeniería del Software* (Novena ed.). Addison-Wesley.

Anexos

A. Diccionario de datos

Debido a que solo hubo cambios en los datos de las tablas y no a la base de datos original, esto es, solo se agregó un nuevo tipo de recurso digital además de nuevos datos a la tabla de formato para el nuevo recurso digital de audio, se muestra a continuación el diccionario de datos correspondiente a la aplicación original.

constant					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
constant_name	PK	VarChar(50)	Not Null		
constant_value		VarChar(50)	Not Null		

digital_resource					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
digital_resource_name	PK	VarChar(50)	Not Null		
format	FK	VarChar(10)	Not Null		format
file		bytea	Not Null		
description		VarChar(100)	Null		
date_sent		timestamp	Not Null		
related_emergency	FK	integer	Not Null		emergency
sender	FK	VarChar(50)	Not Null		dynamic_resource
latitude		double precision	Not Null		
longitude		double precision	Not Null		

Tabla 8. Tablas constant y digital_resource, correspondiente a la base de datos de la aplicación.

digital_resource_addressee					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
digital_resource_name	PK/FK	VarChar(50)	Not Null		digital_resource
addressee	PK/FK	VarChar(50)	Not Null		dynamic_resource

dynamic_resource					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
dynamic_resource_name	PK	VarChar(50)	Not Null		
id_device	Unique	VarChar(16)	Not Null		
latitude		double precision			
longitude		double precision			
date_sent_of_location		timestamp			

Emergency					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
id_emergency	PK	serial	Not Null		
address		VarChar(50)	Not Null		
priority			Not Null	priorities	
latitude		double precision	Not Null		
longitude		double precision	Not Null		
start_date		timestamp	Not Null		
end_date		timestamp			

Tabla 9. Tablas digital_resource_addressee, dynamic_resource y emergency, correspondiente a la base de datos de la aplicación.

emergency_dynamic_resource					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
assisted_emergency	PK/FK	integer	Not Null		Emergency
dynamic_resource_name	PK/FK	VarChar(50)	Not Null		dynamic_resource
synchronization_date	PK	timestamp	Not Null		
status		VarChar(50)			
date_sent_of_status		timestamp			

fire_hydrant					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
fire_hydrant_name	PK/FK	VarChar(50)	Not Null		static_resource

fire_station					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
fire_station_name	PK/FK	VarChar(50)	Not Null		static_resource

fire_truck					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
fire_truck_name	PK/FK	VarChar(50)	Not Null		dynamic_resource
fire_station_name	FK	VarChar(50)	Not Null		fire_station

Tabla 10. Tablas emergency_dynamic_resource, fire_hydrant, fire_station y fire_truck, correspondiente a la base de datos de la aplicación.

Firefighter					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
firefighter_name	PK/FK	VarChar(50)	Not Null		dynamic_resource
firefighter_rut,	UNIQUE	numeric(8)	Not Null		
fire_station_name	FK	VarChar(50)	Not Null		fire_station
firefighter_type			Not Null	firefighter_types	

Format					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
format_name	PK	VarChar(10)	Not Null		
file_type			Not Null	file_types	

hospital					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
hospital_name	PK/FK	VarChar(50)	Not Null		static_resource

police_station					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
police_station_name	PK/FK	VarChar(50)	Not Null		static_resource

Tabla 11. Tablas firefighter, format, hospital y police_station, correspondiente a la base de datos de la aplicación.

static_resource					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
static_resource_name	PK	VarChar(50)	Not Null		
address	UNIQUE	VarChar(50)	Not Null		
description		VarChar(100)			
latitude		double precision	Not Null		
longitude		double precision	Not Null		

updated_table					
Atributo	Clave	Tipo_dato	Valor_nulo	Dominio	Referencia
table_name	PK	VarChar(50)	Not Null		
last_update		timestamp	Not Null		

Tabla 12.Tablas static_resource y updated_table, correspondiente a la base de datos de la aplicación.

B. Código fuente de los nuevos componentes.

A continuación, se mostrará el código relacionado a cada uno de los nuevos componentes. Cabe mencionar que algunos de los archivos fueron modificados y otros fueron creados con el fin de integrar el componente a la aplicación original.

Aquellos archivos que fueron modificados se mostrarán las líneas de código agregadas encerradas en rojo.

B1. Componente de audio

Los archivos modificados/creados para el componente de audio fueron:

1. DigitalResourceMaker.java (modificado)
2. ConcreteOptionalComponentsA.java (modificado)
3. DefaultOptionalComponents.java (modificado)
4. OptionalComponents.java (modificado)
5. AudioMaker.java (creado)
6. ConcreteAudioMakerA.java (creado)
7. AudioOpener.java (creado)
8. ConcreteAudioOpenerA.java (creado)
9. ConcreteDigitalResourcesOpener.java (modificado)
10. DigitalResourceTypes.java (modificado)

Los cuales se muestran a continuación:

1. *DigitalResourceMaker.java*

```
package cl.ucsc.projectoftitle.newmobilemap.application.ui;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Calendar;

import cl.ucsc.projectoftitle.newmobilemap.R;
import
cl.ucsc.projectoftitle.newmobilemap.application.utilities.CommonLabels
;
import
cl.ucsc.projectoftitle.newmobilemap.application.utilities.StringsOfExt
ras;
import
cl.ucsc.projectoftitle.newmobilemap.components.compmanagers.Components
Manager;
import
cl.ucsc.projectoftitle.newmobilemap.components.compmanagers.ConcreteCo
mponentsManagerA;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.PictureMa
ker;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.VideoMake
r;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.AudioMake
r;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmanager.DigitalR
esourceManager;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitals
ender.DigitalResourceSender;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.lrsynchr
onizer.LocalRepositorySynchronizer;
import
cl.ucsc.projectoftitle.newmobilemap.components.entities.Emergency;
import
cl.ucsc.projectoftitle.newmobilemap.components.entities.types.DigitalR
esourceTypes;
import
cl.ucsc.projectoftitle.newmobilemap.components.localrepository.reader.
LocalRepositoryReader;
```

```
import
cl.ucsc.projectoftitle.newmobilemap.components.user.data.UserData;
import
cl.ucsc.projectoftitle.newmobilemap.components.user.location.UserLocat
ionReader;
import
cl.ucsc.projectoftitle.newmobilemap.components.utilities.UIInstance;
import
cl.ucsc.projectoftitle.newmobilemap.components.utilities.reflection.Re
flectionUtilities;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.annotation.SuppressLint;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.res.Configuration;
import android.database.Cursor;
import android.util.Log;
import android.view.Gravity;
import android.view.MenuItem;
import android.view.View;
import android.view.WindowManager;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.AdapterView.OnItemClickListener;

public class DigitalResourcesMaker extends Activity implements
OnItemClickListener, UIInstance {

      private ComponentsManager compManager;

      private LocalRepositoryReader lrReader;
      private Emergency emergency;

      private LocalRepositorySynchronizer localRSync;

      private UserLocationReader userLocation;
      private UserData userData;

      private ListView drrCreatorList;

      private DigitalResourceManager drrManager;
      private PictureMaker pictureMaker;
      private VideoMaker videoMaker;
      private AudioMaker audioMaker;

      private ReflectionUtilities reflectionUtilities;
```

```
	private String pictureSavePath;
	private String videoSavePath;
	private String audioSavePath;

	private boolean isFileInDigitalResourcesFolder = true;

	private AlertDialog.Builder sendMessageDialog;

	private Toast tmpMessage;


	private static final String LABEL_TAKE_PICTURE = "Tomar
fotografía";
	private static final String LABEL_FILM_VIDEO = "Filmar video";
	private static final String LABEL_RECORD_AUDIO = "Grabar audio";

	private static final String LABEL_THE_FILE_HAS_BEEN_SAVED = "El
archivo ha sido almacenado!";
	private static final String LABEL_THE_FILE_COULD_NOT_BE_SAVED =
"El archivo no pudo ser almacenado...";
	private static final String
LABEL_DO_YOU_WANT_SHARE_THIS_DIGITAL_RESOURCE = "¿Desea compartir el
recurso digital creado?";


	@SuppressLint("ShowToast")
	@Override
	protected void onCreate(Bundle savedInstanceState) {

		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_digital_resource_maker);


	getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN
_ON);

		compManager =
ConcreteComponentsManagerA.getInstance(getApplicationContext());

		lrReader =
compManager.getBasicComponents().getLocalRepositoryReader();

	      Intent intent = getIntent();
	      emergency =
lrReader.getParticularEmergency(intent.getIntExtra(StringsOfExtras.EXT
RA_ID_EMERGENCY,0));

		userData = compManager.getBasicComponents().getUserData();
```

```
          if(emergency != null){

                  if(userData.getAssistedEmergency() ==
emergency.getIdEmergency()){

                        setupActivityTitle();

                        localRSync =
compManager.getBasicComponents().getLocalRepositorySynchronizer();

                        userLocation =
compManager.getBasicComponents().getUserLocationReader();

                        drrCreatorList = (ListView)
findViewById(R.id.digital_resource_creator_list);

                        drrManager =
compManager.getBasicComponents().getDigitalResourceManager();
                        pictureMaker =
compManager.getOptionalComponents().getPictureMaker();
                        videoMaker =
compManager.getOptionalComponents().getVideoMaker();
                        audioMaker =
compManager.getOptionalComponents().getAudioMaker();

                        reflectionUtilities =
compManager.getBasicComponents().getReflectionUtilities();

                      tmpMessage =
Toast.makeText(getApplicationContext(), "", Toast.LENGTH_SHORT);


                        setMenu();

                  }
                  else
                        backToThePreviousActivity();

          }

      }

      private void setupActivityTitle() {

            String emergencyAddress = emergency.getAddress();
            setTitle(Emergency.LABEL_EMERGENCY + ": " +
emergencyAddress);

      }
```

```
	@Override
	protected void onResume(){

		super.onResume();

		EmergenciesList.commonBehavoirInOnResumeMethod(localRSync,
userLocation, this);

	}

	private void setMenu(){

		String[] listItems = setUpListItems();

		if(listItems != null){

			drrCreatorList.setAdapter(new
ArrayAdapter<String>(this, R.layout.digital_resource_maker_childs,
listItems));
			drrCreatorList.setOnItemClickListener(this);

		}

	}

	private String[] setUpListItems(){

		String[] result = null;

		ArrayList<String> tmp = new ArrayList<String>();

		if(pictureMaker != null &&
DigitalResourceTypes.Picture.TYPE_NAME != null &&
userData.getAssistedEmergency() == emergency.getIdEmergency())

			tmp.add(LABEL_TAKE_PICTURE);

		if(videoMaker != null &&
DigitalResourceTypes.Video.TYPE_NAME != null &&
userData.getAssistedEmergency() == emergency.getIdEmergency())

			tmp.add(LABEL_FILM_VIDEO);

		if(audioMaker != null &&
DigitalResourceTypes.Audio.TYPE_NAME != null &&
userData.getAssistedEmergency() == emergency.getIdEmergency())

			tmp.add(LABEL_RECORD_AUDIO);

		if(tmp.size() > 0)
			result = (String[]) tmp.toArray(new
String[tmp.size()]);
```

```
            return result;

      }

      @Override
      public void onItemClick(AdapterView<?> adapterView, View view,
int position, long id) {

            TextView textView = (TextView)
view.findViewById(android.R.id.text1);
            String currentRow = textView.getText().toString();

            if(currentRow == LABEL_TAKE_PICTURE &&
DigitalResourceTypes.Picture.TYPE_NAME != null){


                  pictureSavePath =
drrManager.getSavePathForUserDigitalResource(DigitalResourceTypes.Pict
ure.FORMAT_PNG);
                  pictureMaker.make(pictureSavePath, this);

            }

            if(currentRow == LABEL_FILM_VIDEO &&
DigitalResourceTypes.Video.TYPE_NAME != null){


                  videoSavePath =
drrManager.getSavePathForUserDigitalResource(DigitalResourceTypes.Vide
o.FORMAT_MP4);
               videoMaker.make(videoSavePath, true, 60,
DigitalResourceSender.MAX_FILE_SIZE_IN_BYTES, this);

            }

            if(currentRow == LABEL_RECORD_AUDIO &&
DigitalResourceTypes.Audio.TYPE_NAME != null){


                  audioSavePath =
drrManager.getSavePathForUserDigitalResource(DigitalResourceTypes.Audi
o.FORMAT_MP3);
                  audioMaker.make(audioSavePath, this);

            }

    }
```

```
      @Override
      protected void onActivityResult(int requestCode, int resultCode,
Intent data){

            String tmpData = null;

            if(requestCode == 1 && resultCode == Activity.RESULT_OK){

                if(data != null){

                    if(data.getData() != null)
                        tmpData = data.getData().toString();

            }

            if(pictureSavePath != null){

                  tmpData = pictureSavePath;
                  pictureSavePath = null;

            }

            if(tmpData != null){

                  File fileToSend = getfileToSend(tmpData);

                  if(fileToSend.exists()){


      tmpMessage.setText(LABEL_THE_FILE_HAS_BEEN_SAVED);

      tmpMessage.setDuration(Toast.LENGTH_SHORT);
                              tmpMessage.setGravity(Gravity.NO_GRAVITY,
Gravity.CENTER_HORIZONTAL, Gravity.CLIP_VERTICAL);
                              tmpMessage.show();


      if(reflectionUtilities.checkExistence(DigitalResourcesForm.class
)){

                                    double senderLatitude =
userLocation.getLatitude();
                                    double senderLongitude =
userLocation.getLongitude();
                                    String dateOfLastLocation =
userLocation.getDateOfLastLocation();

                              Calendar c = Calendar.getInstance();
                              c.add(Calendar.MINUTE, -1); // La
ubicación obtenida más arriba debe estar dentro del intervalo indicado
( [now() - 1min, now()] )
                              Timestamp waitingTime = new
Timestamp(c.getTimeInMillis());
```

```
                                if(senderLatitude != 0.0 &&
senderLongitude != 0.0 &&
Timestamp.valueOf(dateOfLastLocation).after(waitingTime))

      showSendMessage(fileToSend.getAbsolutePath(), senderLatitude,
senderLongitude, true);

                                else

      showSendMessage(fileToSend.getAbsolutePath(), senderLatitude,
senderLongitude, false);

                           }

                     }
                           else{


      tmpMessage.setText(LABEL_THE_FILE_COULD_NOT_BE_SAVED);

      tmpMessage.setDuration(Toast.LENGTH_LONG);
                                tmpMessage.setGravity(Gravity.NO_GRAVITY,
Gravity.CENTER_HORIZONTAL, Gravity.CLIP_VERTICAL);
                                tmpMessage.show();

                           }

               }

               }

      }

      private File getfileToSend(String dataFromOnActivityResult){ //
Función encargada de obtener el archivo recién creado, además de
informar si este se encuentra o no en la carpeta de recursos
digitales, esto con el fin de saber si es necesario moverlo a la ruta
donde originalmente debía crearse.

               File fileToSend = null;

               File recentMediaFile =
getTheMostRecentFileOfMediaFolderIfExists();

               if(recentMediaFile == null && dataFromOnActivityResult !=
null){                                    //No es un archivo reciente
de la carpeta multimedia de Android

                     if(dataFromOnActivityResult.contains("file://"))
                           fileToSend = new
File(dataFromOnActivityResult.substring(7));
```

```
else
                        fileToSend = new
File(dataFromOnActivityResult);

            }

            else if(recentMediaFile != null &&
!dataFromOnActivityResult.contains("content:/")){    //Es un archivo
reciente y replicado de la carpeta multimedia de Android

                  recentMediaFile.delete();
                  fileToSend = new File(dataFromOnActivityResult);

                  isFileInDigitalResourcesFolder = false;

            }

            else{

      //Es un archivo reciente de la carpeta multimedia de Android

                  fileToSend = recentMediaFile;
                  isFileInDigitalResourcesFolder = false;

            }

            return fileToSend;

      }

      private File getTheMostRecentFileOfMediaFolderIfExists(){
//Función encargada de obtener el último archivo multimedia creado en
el dispositivo

            File recentMediaFile = null;

            File tmpPicture =
getTheMostRecentPictureOfMediaFolderIfExists();
            File tmpVideo =
getTheMostRecentVideoOfMediaFolderIfExists();
            File tmpAudio =
getTheMostRecentAudioOfMediaFolderIfExists();

            if(tmpPicture != null && tmpVideo != null && tmpAudio !=
null){

                  if((new
Timestamp(tmpPicture.lastModified())).after(new
Timestamp(tmpVideo.lastModified())) && (new
Timestamp(tmpPicture.lastModified())).after(new
Timestamp(tmpAudio.lastModified())))
                        recentMediaFile = tmpPicture;
```

```
else if((new Timestamp(tmpAudio.lastModified())).after(new
Timestamp(tmpPicture.lastModified())) && (new
Timestamp(tmpAudio.lastModified())).after(new
Timestamp(tmpVideo.lastModified())))
                        recentMediaFile = tmpAudio;

                  else
                        recentMediaFile = tmpVideo;

            }

            if(tmpPicture != null && tmpVideo == null && tmpAudio ==
null)
                  recentMediaFile = tmpPicture;

            if(tmpPicture == null && tmpVideo != null && tmpAudio ==
null)
                  recentMediaFile = tmpVideo;

            if(tmpPicture == null && tmpVideo == null && tmpAudio !=
null)
                  recentMediaFile = tmpAudio;

            if(recentMediaFile != null)
                  Log.e("The most recent media file is: ",
recentMediaFile.getAbsolutePath());

            return recentMediaFile;
      }

private File getTheMostRecentPictureOfMediaFolderIfExists(){

            File recentPictureFile = null;

            Calendar calendar = Calendar.getInstance();
            calendar.add(Calendar.SECOND, -15);
            Timestamp waitingTime = new
Timestamp(calendar.getTimeInMillis());


            Uri mediaUri =
MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
            String[] selectedColumns = {

      MediaStore.Images.Media.DATA,

      MediaStore.Images.Media.DATE_MODIFIED
                                                      };
            String orderBy = MediaStore.Images.Media.DATE_MODIFIED + "
DESC LIMIT 1";
```

```
Cursor c = this.getContentResolver().query(mediaUri, selectedColumns,
null, null, orderBy);

            if(c.moveToFirst()){ // retornará false si el cursor está
vacio

                  File lastPictureFile = new
File(c.getString(c.getColumnIndex(MediaStore.Images.Media.DATA)));
                  Timestamp lastModified = new
Timestamp(lastPictureFile.lastModified());

                  Log.e("Image: ",
c.getString(c.getColumnIndex(MediaStore.Images.Media.DATA)));
                  Log.e("Last image:", lastModified.toString());
                  Log.e("Waiting time: ", waitingTime.toString());

                  //Verifica si el último archivo modificado de la
carpeta multimedia es de hace a lo más 15 segundos atrás
                  if(lastModified.after(waitingTime)){     //
lastModified > (now() - 15 seg)

                        recentPictureFile = lastPictureFile;

                        Log.e("Image", "ok");

                  }
                  c.close();

            }

            return recentPictureFile;

      }


      private File getTheMostRecentVideoOfMediaFolderIfExists(){

            File recentVideoFile = null;

            Calendar calendar = Calendar.getInstance();
            calendar.add(Calendar.SECOND, -15);
            Timestamp waitingTime = new
Timestamp(calendar.getTimeInMillis());


            Uri mediaUri =
MediaStore.Video.Media.EXTERNAL_CONTENT_URI;
            String[] selectedColumns = {

      MediaStore.Video.Media.DATA,

      MediaStore.Video.Media.DATE_MODIFIED
                                                      };
```

```
            String orderBy = MediaStore.Video.Media.DATE_MODIFIED + "
DESC LIMIT 1";

            Cursor c = this.getContentResolver().query(mediaUri,
selectedColumns, null, null, orderBy);

            if(c.moveToFirst()){ // retornará false si el cursor está
vacio

                  File lastVideoFile = new
File(c.getString(c.getColumnIndex(MediaStore.Video.Media.DATA)));
                  Timestamp lastModified = new
Timestamp(lastVideoFile.lastModified());

                  Log.e("Video: ",
c.getString(c.getColumnIndex(MediaStore.Video.Media.DATA)));
                  Log.e("Last video: ", lastModified.toString());
                  Log.e("Waiting time: ", waitingTime.toString());

                  //Verifica si el último archivo modificado de la
carpeta multimedia es de hace a lo más 15 segundos atrás
                  if(lastModified.after(waitingTime)){     //
lastModified > (now() - 15 seg)

                        recentVideoFile = lastVideoFile;

                        Log.e("Video", "ok");

                  }

                  c.close();

            }

            return recentVideoFile;

      }
private File getTheMostRecentAudioOfMediaFolderIfExists(){

            File recentAudioFile = null;

            Calendar calendar = Calendar.getInstance();
            calendar.add(Calendar.SECOND, -15);
            Timestamp waitingTime = new
Timestamp(calendar.getTimeInMillis());


            Uri mediaUri =
MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
            String[] selectedColumns = {

      MediaStore.Audio.Media.DATA,
```

```
      MediaStore.Audio.Media.DATE_MODIFIED
                                                        };
            String orderBy = MediaStore.Audio.Media.DATE_MODIFIED + "
DESC LIMIT 1";

            Cursor c = this.getContentResolver().query(mediaUri,
selectedColumns, null, null, orderBy);

            if(c.moveToFirst()){ // retornará false si el cursor está
vacio

                  File lastAudioFile = new
File(c.getString(c.getColumnIndex(MediaStore.Audio.Media.DATA)));
                  Timestamp lastModified = new
Timestamp(lastAudioFile.lastModified());

                  Log.e("Audio: ",
c.getString(c.getColumnIndex(MediaStore.Audio.Media.DATA)));
                  Log.e("Last audio:", lastModified.toString());
                  Log.e("Waiting time: ", waitingTime.toString());

                  //Verifica si el último archivo modificado de la
carpeta multimedia es de hace a lo más 15 segundos atrás
                  if(lastModified.after(waitingTime)){     //
lastModified > (now() - 15 seg)

                        recentAudioFile = lastAudioFile;

                        Log.e("Audio", "ok");

                  }
                  c.close();

            }

            return recentAudioFile;

      }
```

```
      private void showSendMessage(final String filePath, final double
senderLatitude, final double senderLongitude, final boolean
isUpdatedSenderLocation){

                  if(sendMessageDialog == null){

                        sendMessageDialog = new
AlertDialog.Builder(this);

                        sendMessageDialog.setCancelable(false);

sendMessageDialog.setMessage(LABEL_DO_YOU_WANT_SHARE_THIS_DIGITAL_RESO
URCE);
```

```
                }

      sendMessageDialog.setPositiveButton(CommonLabels.LABEL_YES, new
DialogInterface.OnClickListener() {

                                    @Override
                                public void onClick(DialogInterface
dialog, int id) {


                                        if(isUpdatedSenderLocation){

                                              Intent intent = new
Intent(DigitalResourcesMaker.this, DigitalResourcesForm.class);

      intent.putExtra(StringsOfExtras.EXTRA_ID_EMERGENCY,
emergency.getIdEmergency());

      intent.putExtra(StringsOfExtras.EXTRA_FILE_PATH, filePath);

      intent.putExtra(StringsOfExtras.EXTRA_DIGITAL_RESOURCE_LATITUDE,
senderLatitude);

      intent.putExtra(StringsOfExtras.EXTRA_DIGITAL_RESOURCE_LONGITUDE
, senderLongitude);

      intent.putExtra(StringsOfExtras.EXTRA_IS_FILE_IN_DIGITAL_RESOURC
ES_FOLDER, isFileInDigitalResourcesFolder);

                                        startActivity(intent);


                                        }
                                        else{

                                              tmpMessage =
Toast.makeText(DigitalResourcesMaker.this,
CommonLabels.LABEL_YOUR_LOCATION_COULD_NOT_BE_GOT, Toast.LENGTH_LONG);
                                              tmpMessage.show();

                                        }

                                    }

                                });


      sendMessageDialog.setNegativeButton(CommonLabels.LABEL_NO, new
DialogInterface.OnClickListener() {

                                    @Override
                                    public void onClick(DialogInterface
dialog, int id) {
```

```
      if(!isFileInDigitalResourcesFolder)

      moveFileToUserDigitalResourceFolder(new File(filePath),
compManager);

                                    }

                                });

                  sendMessageDialog.create().show();

      }

      public static void moveFileToUserDigitalResourceFolder(final
File fileToMove, final ComponentsManager compManager){

            new Thread(new Runnable() {

                  @Override
                  public void run() {

                        DigitalResourceManager drrManager =
compManager.getBasicComponents().getDigitalResourceManager();
                        DigitalResourceTypes digitalResourceTypes =
compManager.getBasicComponents().getDigitalResourceTypes();

                        String filePath = null;

                        FileInputStream videoIS = null;
                        FileOutputStream videoOS = null;

                        byte[] bufferData = new byte[1024];
                        int length;

                              try {

                                    videoIS = new
FileInputStream(fileToMove);

                                    filePath =
drrManager.getSavePathForUserDigitalResource(digitalResourceTypes.getF
ormatFromFile(fileToMove));
                                    videoOS = new
FileOutputStream(filePath);

                                    while((length =
videoIS.read(bufferData, 0, bufferData.length)) != -1)

      videoOS.write(bufferData, 0, length); // Almacenando archivo en
la carpeta de recursos digitales del usuario

                                    videoIS.close();
```

```
                                        videoOS.close();

                                        fileToMove.delete(); //
Eliminando archivo

                                    } catch (IOException e) {
                                        e.printStackTrace();
                                    }

                    }

              }).start();

        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item){

              switch (item.getItemId()) {

                    case android.R.id.home: // Home/Up button

                          onBackPressed(); // Simulando BackButton
                          return true;

                default:
                    return super.onOptionsItemSelected(item);

            }

        }

        private void backToThePreviousActivity(){

              super.onBackPressed();

        }

        @Override
        public void onConfigurationChanged(Configuration newConfig) {

            super.onConfigurationChanged(newConfig);

            if (newConfig.orientation ==
Configuration.ORIENTATION_LANDSCAPE)
                Log.d("On Config Changed","Landscape");

            if (newConfig.orientation ==
Configuration.ORIENTATION_PORTRAIT)
              Log.d("On config Changed","Portrait");

        }
```

```
	@Override
	protected void onPause(){

		super.onPause();

	}

	@Override
	protected void onStop(){

		super.onStop();

	}

	@Override
	protected void onDestroy(){

		super.onDestroy();

	}

}
```

2. *ConcreteOptionalComponentsA*.java

```
package
cl.ucsc.projectoftitle.newmobilemap.components.compmanagers.optional;

import
cl.ucsc.projectoftitle.newmobilemap.components.compmanagers.basic.Basi
cComponents;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.PictureMa
ker;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.VideoMake
r;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.AudioMake
r;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.ConcreteP
ictureMakerA;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.ConcreteV
ideoMakerA;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.ConcreteA
udioMakerA;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.PictureO
pener;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.VideoOpe
ner;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.AudioOpe
ner;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.Concrete
PictureOpenerA;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.Concrete
VideoOpenerA;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.Concrete
AudioOpenerA;

public class ConcreteOptionalComponentsA extends
DefaultOptionalComponents{

      private PictureOpener pictureOpener;
      private VideoOpener videoOpener;
      private AudioOpener audioOpener;
```

```
	private PictureMaker pictureCreator;
	private VideoMaker videoCreator;
	private AudioMaker audioCreator;

	public ConcreteOptionalComponentsA(BasicComponents
basicComponents) {

		super(basicComponents);

	}

	@Override
	protected PictureOpener getPictureOpener() {

		if(pictureOpener == null)
			pictureOpener = new ConcretePictureOpenerA();

		return pictureOpener;

	}

	@Override
	protected VideoOpener getVideoOpener() {

		if(videoOpener == null)
			videoOpener = new ConcreteVideoOpenerA();

		return videoOpener;

	}

	@Override
	protected AudioOpener getAudioOpener() {

		if(audioOpener == null)
			audioOpener = new ConcreteAudioOpenerA();

		return audioOpener;

	}

	@Override
	public PictureMaker getPictureMaker() {

		if(pictureCreator == null)
			pictureCreator = new ConcretePictureMakerA();

		return pictureCreator;

	}
```

```
	@Override
	public VideoMaker getVideoMaker() {

		if(videoCreator == null)
			videoCreator = new ConcreteVideoMakerA();

		return videoCreator;

	}

	@Override
	public AudioMaker getAudioMaker() {

		if(audioCreator == null)
			audioCreator = new ConcreteAudioMakerA();

		return audioCreator;

	}

}
```

3. DefaultOptionalComponents.java

```
package
cl.ucsc.projectoftitle.newmobilemap.components.compmanagers.optional;

import java.io.File;
import java.util.ArrayList;

import
cl.ucsc.projectoftitle.newmobilemap.components.compmanagers.basic.Basi
cComponents;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.DigitalR
esourcesOpener;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.Concrete
DigitalResourcesOpener;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitald
ownloader.DigitalResourceDownloader;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitald
ownloader.ConcreteDigitalResourceDownloader;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitald
ownloader.DigitalResourceDownloader.DownloadObserver;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitals
ender.DigitalResourceSender;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitals
ender.ConcreteDigitalResourceSender;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitals
ender.DigitalResourceSender.SendObserver;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.location
sender.UserLocationSender;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.lrsynchr
onizer.LocalRepositorySynchronizer.OptionalTask;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.statusse
nder.UserStatusSender;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.statusse
nder.ConcreteUserStatusSender;
import
cl.ucsc.projectoftitle.newmobilemap.components.entities.DigitalResourc
e;
import
cl.ucsc.projectoftitle.newmobilemap.components.entities.Emergency;
```

```
import
cl.ucsc.projectoftitle.newmobilemap.components.layers.single.SingleLay
er;
import
cl.ucsc.projectoftitle.newmobilemap.components.layers.single.Emergency
DirectionLayer;
import cl.ucsc.projectoftitle.newmobilemap.components.map.Map;

public abstract class DefaultOptionalComponents extends
OptionalComponents{

	protected BasicComponents basicComponents;

	protected ArrayList<OptionalTask> optionalTasks;
	private OptionalTask locationSender;

	private DigitalResourceDownloader drrDownloader;
	private DigitalResourceSender drrSender;
	private UserStatusSender statusSender;

	private SingleLayer emergencyDirectionLayer;

	private DigitalResourcesOpener drrOpener;


	public DefaultOptionalComponents(BasicComponents
basicComponents){

		this.basicComponents = basicComponents;

	}

	public ArrayList<OptionalTask> getAllOptionalTasks(){

		if(optionalTasks == null)
			optionalTasks = new ArrayList<OptionalTask>();

		OptionalTask tmp1 = getUserLocationSender();

		if(tmp1 != null){

			if(!optionalTasks.contains(tmp1))
				optionalTasks.add(getUserLocationSender());

		}

		return optionalTasks;

	}
```

```
	private OptionalTask getUserLocationSender(){

		locationSender =
UserLocationSender.getInstance(basicComponents.getUserLocationReader()
, basicComponents.getUserData(),
basicComponents.getLocalRepositorySynchronizer(),
basicComponents.getDebugHelper(), basicComponents.getJSONParser());

		return locationSender;

	}

	public DigitalResourceDownloader
getDigitalResourceDownloader(DigitalResource selectedDigitalResource,
DownloadObserver downloadObserver) {

		drrDownloader = new
ConcreteDigitalResourceDownloader(selectedDigitalResource,
downloadObserver, basicComponents.getUserData(),
basicComponents.getInfoDevice(), basicComponents.getInternetChecker(),
basicComponents.getEventDispatchThread(),
basicComponents.getDebugHelper(), basicComponents.getJSONParser(),
basicComponents.getDigitalResourceManager());

		return drrDownloader;

	}

	public DigitalResourceSender getDigitalResourceSender(File
fileToSend, double resourceLatitude, double resourceLongitude, String
resourceDescription, ArrayList<String> resourceAddressees,
SendObserver sendObserver) {

		drrSender = new ConcreteDigitalResourceSender(fileToSend,
resourceLatitude, resourceLongitude, resourceDescription,
resourceAddressees, sendObserver, basicComponents.getUserData(),
basicComponents.getInfoDevice(), basicComponents.getInternetChecker(),
basicComponents.getEventDispatchThread(),
basicComponents.getDebugHelper(), basicComponents.getJSONParser(),
basicComponents.getDigitalResourceTypes());

		return drrSender;

	}

	public UserStatusSender getUserStatusSender() {

		statusSender =
ConcreteUserStatusSender.getInstance(basicComponents.getUserData(),
basicComponents.getInfoDevice(), basicComponents.getInternetChecker(),
basicComponents.getEventDispatchThread(),
basicComponents.getDebugHelper(), basicComponents.getJSONParser());
```

```
            return statusSender;

      }

      public SingleLayer getEmergencyDirectionLayer(Map decoratedMap,
Emergency emergency){

            emergencyDirectionLayer = new
EmergencyDirectionLayer(decoratedMap, emergency,
basicComponents.getUserData(),
basicComponents.getUserLocationReader(),
basicComponents.getPointOfInterestCreator(),
basicComponents.getIconsManager(),
basicComponents.getCoordinateUtilities());

            return emergencyDirectionLayer;

      }

      public DigitalResourcesOpener getDigitalResourcesOpener() {

            drrOpener =
ConcreteDigitalResourcesOpener.getInstance(basicComponents.getDigitalR
esourceManager(), getPictureOpener(), getVideoOpener(),
getAudioOpener(), basicComponents.getDigitalResourceTypes());

            return drrOpener;

      }
}
```

4. OptionalComponents.java

```
package
cl.ucsc.projectoftitle.newmobilemap.components.compmanagers.optional;

import java.io.File;
import java.util.ArrayList;

import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.PictureMa
ker;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.VideoMake
r;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers.AudioMake
r;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.DigitalR
esourcesOpener;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.PictureO
pener;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.VideoOpe
ner;
import
cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners.AudioOpe
ner;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitald
ownloader.DigitalResourceDownloader;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitald
ownloader.DigitalResourceDownloader.DownloadObserver;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitals
ender.DigitalResourceSender;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.digitals
ender.DigitalResourceSender.SendObserver;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.lrsynchr
onizer.LocalRepositorySynchronizer.OptionalTask;
import
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.statusse
nder.UserStatusSender;
import
cl.ucsc.projectoftitle.newmobilemap.components.entities.DigitalResourc
e;
import
cl.ucsc.projectoftitle.newmobilemap.components.entities.Emergency;
```

```
import
cl.ucsc.projectoftitle.newmobilemap.components.layers.single.SingleLay
er;
import cl.ucsc.projectoftitle.newmobilemap.components.map.Map;

public abstract class OptionalComponents {

	public abstract ArrayList<OptionalTask> getAllOptionalTasks();
	public abstract DigitalResourceDownloader
getDigitalResourceDownloader(DigitalResource selectedDigitalResource,
DownloadObserver downloadObserver);
	public abstract DigitalResourceSender
getDigitalResourceSender(File fileToSend, double resourceLatitude,
double resourceLongitude, String resourceDescription,
ArrayList<String> resourceAddressees, SendObserver sendObserver);
	public abstract UserStatusSender getUserStatusSender();
	public abstract SingleLayer getEmergencyDirectionLayer(Map
decoratedMap, Emergency emergency);
	public abstract DigitalResourcesOpener
getDigitalResourcesOpener();
	protected abstract PictureOpener getPictureOpener();
	protected abstract VideoOpener getVideoOpener();
	protected abstract AudioOpener getAudioOpener();
	public abstract PictureMaker getPictureMaker();
	public abstract VideoMaker getVideoMaker();
	public abstract AudioMaker getAudioMaker();

}
```

5. *AudioMaker.java*

```
package cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers;

import
cl.ucsc.projectoftitle.newmobilemap.components.utilities.UIInstance;

public interface AudioMaker {

      public void make(String savePath, UIInstance UIInstance);

}
```

6. *ConcreteAudioMakerA.java*

```
package cl.ucsc.projectoftitle.newmobilemap.components.digitalmakers;

import java.io.File;

import
cl.ucsc.projectoftitle.newmobilemap.components.utilities.UIInstance;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.provider.MediaStore;


public class ConcreteAudioMakerA implements AudioMaker {


      public void make(String savePath, UIInstance UIInstance){

            if(savePath != null && UIInstance != null){

                  if(UIInstance instanceof Activity){

                        Uri audioUriPath = Uri.fromFile(new
File(savePath));

                        Intent takeAudioIntent = new Intent();

      takeAudioIntent.setAction(android.provider.MediaStore.Audio.Medi
a.RECORD_SOUND_ACTION);


      takeAudioIntent.putExtra(MediaStore.EXTRA_OUTPUT, audioUriPath);
                        ((Activity)
UIInstance).startActivityForResult(takeAudioIntent, 1);

                  }

            }

      }

}
```

7. AudioOpener.java

```
package cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners;

import java.io.File;

import
cl.ucsc.projectoftitle.newmobilemap.components.utilities.UIInstance;

public interface AudioOpener{

      public void open(File file, UIInstance UIInstance);

}
```

8. *ConcreteAudioOpenerA.java*

```
package cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners;

import java.io.File;

import
cl.ucsc.projectoftitle.newmobilemap.components.utilities.UIInstance;


import android.app.Activity;
import android.content.Intent;
import android.net.Uri;


public class ConcreteAudioOpenerA implements AudioOpener{

      public void open(File file, UIInstance UIInstance){

            if(file != null && UIInstance != null){

                  if(UIInstance instanceof Activity){

                        Uri uriPath = Uri.fromFile(file);

                        Intent openIntent = new Intent();
                        openIntent.setAction(Intent.ACTION_VIEW);
                        openIntent.setDataAndType(uriPath,"audio/*");

                        ((Activity)
UIInstance).startActivity(openIntent);

                  }

            }

      }

}
```

9. *ConcreteDigitalResourcesOpener.java*

```
package cl.ucsc.projectoftitle.newmobilemap.components.digitalopeners;

import java.io.File;

import
cl.ucsc.projectoftitle.newmobilemap.components.digitalmanager.DigitalR
esourceManager;
import
cl.ucsc.projectoftitle.newmobilemap.components.entities.DigitalResourc
e;
import
cl.ucsc.projectoftitle.newmobilemap.components.entities.types.DigitalR
esourceTypes;
import
cl.ucsc.projectoftitle.newmobilemap.components.utilities.UIInstance;

public class ConcreteDigitalResourcesOpener implements
DigitalResourcesOpener{

	private static ConcreteDigitalResourcesOpener sInstance;

	private DigitalResourceManager drrManager;

	private PictureOpener pictureOpener;
	private VideoOpener videoOpener;
	private AudioOpener audioOpener;

	private DigitalResourceTypes digitalResourceTypes;


	private ConcreteDigitalResourcesOpener(DigitalResourceManager
drrManager, PictureOpener pictureOpener, VideoOpener videoOpener,
AudioOpener audioOpener, DigitalResourceTypes digitalResourceTypes){

		this.drrManager = drrManager;

		this.pictureOpener = pictureOpener;
		this.videoOpener = videoOpener;
		this.audioOpener = audioOpener;

		this.digitalResourceTypes = digitalResourceTypes;

	}

	public static ConcreteDigitalResourcesOpener
getInstance(DigitalResourceManager drrManager, PictureOpener
pictureOpener, VideoOpener videoOpener, AudioOpener audioOpener,
DigitalResourceTypes digitalResourceTypes){

		if(sInstance == null)
```

```
                    sInstance = new
ConcreteDigitalResourcesOpener(drrManager, pictureOpener, videoOpener,
audioOpener, digitalResourceTypes);

            return sInstance;
      }

      public void open(DigitalResource drr, UIInstance UIInstance){

            if(drr != null){

                  String format = drr.getFormat();
                  String type =
digitalResourceTypes.getTypeFromFormat(format);

                  if(type != null){

                        String filePath =
drrManager.getSavePathForExternalDigitalResource(drr);

                        File file = new File(filePath);

                        open(file, UIInstance);

                  }

            }

      }

      public void open(File file, UIInstance UIInstance){

            String type = digitalResourceTypes.getTypeFromFile(file);

            if(type != null){

      if(type.equalsIgnoreCase(DigitalResourceTypes.Picture.TYPE_NAME)
&& pictureOpener != null)
                        pictureOpener.open(file, UIInstance);


      if(type.equalsIgnoreCase(DigitalResourceTypes.Video.TYPE_NAME)
&& videoOpener != null)
                        videoOpener.open(file, UIInstance);


      if(type.equalsIgnoreCase(DigitalResourceTypes.Audio.TYPE_NAME)
&& audioOpener != null)
                        audioOpener.open(file, UIInstance);

            }
      }
}
```

10. DigitalResourceTypes.java

```
package cl.ucsc.projectoftitle.newmobilemap.components.entities.types;

import java.io.File;

public interface DigitalResourceTypes extends ResourceTypes{

	public String getTypeFromFormat(String format);
	public String getTypeFromFile(File file);
	public String getFormatFromFile(File file);
	public String getNameOfDownloadPOIIcon(String type);
	public String getNameOfListIcon(String type);

	// El valor de las constantes "TYPE_NAME" debe ser igual a como
está establecido en el dominio "file_types" de la BD externa, mientras
que el valor de las constantes "FORMAT_" debe ser igual al señalado en
las tuplas correspondientes de la tabla "format"
	public interface Picture{
	  public static final String TYPE_NAME = "Fotografía";
	  public static final String TYPE_NAME_IN_PLURAL_FOR_UI =
"Fotografías";
	  public static final String TYPE_POI_ICON_NAME =
"ic_poi_picture";
	  public static final String TYPE_DOWNLOAD_POI_ICON_NAME =
"ic_poi_downloaded_picture";
	  public static final String TYPE_LIST_ICON_NAME =
"ic_list_picture";

	  public static final String FORMAT_GIF = "gif";
	  public static final String FORMAT_JPG = "jpg";
	  public static final String FORMAT_JPEG = "jpeg";
	  public static final String FORMAT_BMP = "bmp";
	  public static final String FORMAT_PNG = "png";
	  public static final String FORMAT_TIFF = "tiff";
	}

	public interface Video{
	  public static final String TYPE_NAME = "Video";
	  public static final String TYPE_NAME_IN_PLURAL_FOR_UI =
"Videos";
	  public static final String TYPE_POI_ICON_NAME =
"ic_poi_video";
	  public static final String TYPE_DOWNLOAD_POI_ICON_NAME =
"ic_poi_downloaded_video";
	  public static final String TYPE_LIST_ICON_NAME =
"ic_list_video";

	  public static final String FORMAT_3GP = "3gp";
	  public static final String FORMAT_MPG = "mpg";
	  public static final String FORMAT_MPEG = "mpeg";
	  public static final String FORMAT_AVI = "avi";
```

```
        public static final String FORMAT_MP4 = "mp4";
        public static final String FORMAT_MKV = "mkv";
    }

    public interface Audio{
        public static final String TYPE_NAME = "Audio";
        public static final String TYPE_NAME_IN_PLURAL_FOR_UI =
"Audios";
        public static final String TYPE_POI_ICON_NAME =
"ic_poi_audio";
        public static final String TYPE_DOWNLOAD_POI_ICON_NAME =
"ic_poi_downloaded_audio";
        public static final String TYPE_LIST_ICON_NAME =
"ic_list_audio";

        public static final String FORMAT_MP3 = "mp3";
        public static final String FORMAT_AAC = "aac";
        public static final String FORMAT_WAV = "wav";
        public static final String FORMAT_REC = "rec";
        public static final String FORMAT_WMA = "wma";
        public static final String FORMAT_AMR = "amr";
        public static final String FORMAT_M4A = "m4a";
    }
}
```

B2. Componente de ruta hacia la emergencia

Los archivos modificados/creados para el componente de ruta hacia la emergencia fueron:

1. EmergencyMenu.java (modificado)
2. CommonLabels.java (modificado)
3. RouteToEmergencyMap.java (creado)
4. DirectionsJSONParser.java (creado)

Los cuales se muestran a continuación:

1. *EmergencyMenu.java*

```
package cl.ucsc.projectoftitle.newmobilemap.application.ui;

import java.util.ArrayList;

public class EmergencyMenu extends ActionBarActivity implements
OnItemClickListener {

	private ComponentsManager compManager;

	private LocalRepositoryReader lrReader;
	private Emergency emergency;

	private LocalRepositorySynchronizer localRSync;

	private UserLocationReader userLocation;
	private UserData userData;

	private ListView emergencyMenu;

	private UserStatusSender statusSender;
	private ReflectionUtilities reflectionUtilities;

	private AlertDialog backButtonDialog;

	private Toast tmpMessage;


	private static final String LABEL_ASSIST_EMERGENCY = "Asistir
esta emergencia";
	private static final String LABEL_SHOW_CALLS = "Llamados";

	private static final String LABEL_CALLS_SOON = "Llamados:
próximamente";
	private static final String
LABEL_DO_YOU_WANT_STOP_ASSIST_THIS_EMERGENCY = "¿Desea abandonar la
emergencia asistida?";

	@SuppressLint("ShowToast")
	@Override
	protected void onCreate(Bundle savedInstanceState) {

		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_emergency_menu);

		CheckBox checkBox = (CheckBox)
findViewById(R.id.emergency_checkbox);
		checkBox.setText("    " + LABEL_ASSIST_EMERGENCY);

                                .
                                .
                                .
```

```
                                    .
                                    .
                                    .


      private void setMenu(){

            String[] listItems = setUpListItems();

            emergencyMenu.setAdapter(new ArrayAdapter<String>(this,
R.layout.emergency_menu_childs, listItems));
          emergencyMenu.setOnItemClickListener(this);

      }

      private String[] setUpListItems(){

            String[] result = null;

            ArrayList<String> tmp = new ArrayList<String>();


      if(reflectionUtilities.checkExistence(UserStatusForm.class) &&
userData.getAssistedEmergency() == emergency.getIdEmergency())

                  tmp.add(CommonLabels.LABEL_SHOW_USER_STATUS_FORM);

            tmp.add(CommonLabels.LABEL_SHOW_EMERGENCY_MAP);
            tmp.add(CommonLabels.LABEL_SHOW_ROUTE_TO_EMERGENCY_MAP);
            tmp.add(CommonLabels.LABEL_SHOW_DYNAMIC_RESOURCES_LIST);


      if(reflectionUtilities.checkExistence(DigitalResourcesList.class
))

      tmp.add(CommonLabels.LABEL_SHOW_DIGITAL_RESOURCES_LIST);


      if(reflectionUtilities.checkExistence(DigitalResourcesMaker.clas
s))

      tmp.add(CommonLabels.LABEL_SHOW_DIGITAL_RESOURCES_MAKER);

            tmp.add(LABEL_SHOW_CALLS);

            result = (String[]) tmp.toArray(new String[tmp.size()]);

            return result;

      }
```

```
      @Override
      public void onItemClick(AdapterView<?> adapterView, View view,
int position, long id) {

            TextView textView = (TextView)
view.findViewById(android.R.id.text1);
            String currentRow = textView.getText().toString();

            if(currentRow == CommonLabels.LABEL_SHOW_EMERGENCY_MAP)

                  showEmergencyMap(EmergencyMenu.this, emergency,
emergency.getAddress());

            if(currentRow ==
CommonLabels.LABEL_SHOW_ROUTE_TO_EMERGENCY_MAP)
                  showRouteToEmergencyMap(EmergencyMenu.this,
emergency, emergency.getAddress());

            if(currentRow ==
CommonLabels.LABEL_SHOW_DYNAMIC_RESOURCES_LIST)
                  showDynamicResourcesList(EmergencyMenu.this,
emergency);

            if(currentRow == LABEL_SHOW_CALLS)
                  showCalls(EmergencyMenu.this, emergency, tmpMessage);

            if(currentRow == CommonLabels.LABEL_SHOW_USER_STATUS_FORM)

                  showUserStatusForm(EmergencyMenu.this, emergency,
userData, reflectionUtilities, tmpMessage);

            if(currentRow ==
CommonLabels.LABEL_SHOW_DIGITAL_RESOURCES_LIST)
                  showDigitalResourcesList(EmergencyMenu.this,
emergency, userData, reflectionUtilities, tmpMessage);

            if(currentRow ==
CommonLabels.LABEL_SHOW_DIGITAL_RESOURCES_MAKER)
                  showDigitalResourcesMaker(EmergencyMenu.this,
emergency, userData, reflectionUtilities, tmpMessage);

      }
```

.
.
.

```
      public static void showRouteToEmergencyMap(Activity activity,
Emergency emergency, String nameOfFirstPOIToFocusOnMap){

      Intent intent = new Intent(activity, RouteToEmergencyMap.class);
      intent.putExtra(StringsOfExtras.EXTRA_ID_EMERGENCY,
emergency.getIdEmergency());

      intent.putExtra(StringsOfExtras.EXTRA_NAME_OF_FIRST_POI_TO_FOCUS
_ON_MAP, nameOfFirstPOIToFocusOnMap);
      activity.startActivity(intent);

      }

      public static void showDynamicResourcesList(Activity activity,
Emergency emergency){

      Intent intent = new Intent(activity,
DynamicResourcesList.class);
      intent.putExtra(StringsOfExtras.EXTRA_ID_EMERGENCY,
emergency.getIdEmergency());
      activity.startActivity(intent);

      }

      public static void showCalls(Activity activity, Emergency
emergency, Toast tmpMessage){

            tmpMessage.setText("\n " + LABEL_CALLS_SOON + " \n");
            tmpMessage.setDuration(Toast.LENGTH_LONG);
            tmpMessage.setGravity(Gravity.NO_GRAVITY,
Gravity.CENTER_HORIZONTAL, Gravity.CENTER_VERTICAL);
            tmpMessage.show();

      }
```

.
.
.

2. *CommonLabels.java*

```
package cl.ucsc.projectoftitle.newmobilemap.application.utilities;


public final class CommonLabels {

	public static final String LABEL_SHOW_USER_STATUS_FORM =
"Actualizar status";
	public static final String LABEL_SHOW_EMERGENCY_MAP = "Mostrar
ubicación";
	public static final String LABEL_SHOW_DYNAMIC_RESOURCES_LIST =
"Recursos que la asisten";
	public static final String LABEL_SHOW_DIGITAL_RESOURCES_LIST =
"Recursos digitales asociados";
	public static final String LABEL_SHOW_DIGITAL_RESOURCES_MAKER =
"Crear recurso digital";
	public static final String LABEL_SHOW_ROUTE_TO_EMERGENCY_MAP =
"Mostrar ruta hacia la emergencia";

	public static final String LABEL_DISTANCE_NO_AVAILABLE = "No
disponible";
	public static final String
LABEL_YOU_ARE_NOT_ASSIST_THIS_EMERGENCY = "Usted no está asistiendo
esta emergencia";
	public static final String LABEL_YOUR_LOCATION_COULD_NOT_BE_GOT
= "Su ubicación no pudo ser obtenida...";
	public static final String LABEL_OPENING_DIGITAL_RESOURCE =
"Abriendo...";

	public static final String LABEL_PLEASE_WAIT = "Espere por
favor.";
	public static final String LABEL_YES = "Sí";
	public static final String LABEL_NO = "No";

}
```

3. RouteToEmergencyMap.java

```
package cl.ucsc.projectoftitle.newmobilemap.application.ui;

import java.io.BufferedReader;...

public class RouteToEmergencyMap extends ActionBarActivity implements
LocationListener, TasksOfConcreteInfoBubbleManager, UIInstance,
UserLocationObserver, ObserverOfSynchronizer, DownloadObserver{
      GoogleMap gMap;
    ArrayList<LatLng> mMarkerPoints;
    double mLatitude=0;
    double mLongitude=0;

    private ComponentsManager compManager;

      private LocalRepositoryReader lrReader;
      private Emergency emergency;

      private LocalRepositorySynchronizer localRSync;

      private UserLocationReader userLocation;
      private UserData userData;

      private SingleLayer emergencyLayer;
      private DigitalResourceDownloader drrDownloader;
      private ProgressDialog downloaderProgress;

      private DigitalResourcesOpener drrOpener;

      private ReflectionUtilities reflectionUtilities;
      private CoordinateUtilities coordinateUtilities;
      private DistanceUtilities distanceUtilities;

      private boolean disableSyncIcon = false;
      private Toast tmpMessage;


      private static final String LABEL_SHOW_USER_LOCATION = "Ver mi
ubicación";

      @SuppressLint("ShowToast")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_route_emergency_map);


getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

        compManager =
ConcreteComponentsManagerA.getInstance(getApplicationContext());
```

```
            lrReader =
compManager.getBasicComponents().getLocalRepositoryReader();

          Intent intent = getIntent();
          emergency =
lrReader.getParticularEmergency(intent.getIntExtra(StringsOfExtras.EXT
RA_ID_EMERGENCY,0));


intent.getStringExtra(StringsOfExtras.EXTRA_NAME_OF_FIRST_POI_TO_FOCUS
_ON_MAP);

          setupActionBarUpButton();


            if(emergency != null){

                  getUserLocationOnMap();
                  setupActionBarTitle();

                  localRSync =
compManager.getBasicComponents().getLocalRepositorySynchronizer();

                  userLocation =
compManager.getBasicComponents().getUserLocationReader();
                  userData =
compManager.getBasicComponents().getUserData();


      compManager.getBasicComponents().getInfoBubbleManager(this);


                  drrOpener =
compManager.getOptionalComponents().getDigitalResourcesOpener();

                  reflectionUtilities =
compManager.getBasicComponents().getReflectionUtilities();
                  coordinateUtilities =
compManager.getBasicComponents().getCoordinateUtilities();
                  distanceUtilities =
compManager.getBasicComponents().getDistanceUtilities();

                  tmpMessage = Toast.makeText(getApplicationContext(),
"", Toast.LENGTH_SHORT);
            }
    }
```

```
//Submenu, barra superior, sincronización

    @TargetApi(Build.VERSION_CODES.HONEYCOMB)
      private void setupActionBarUpButton() {

            if (Build.VERSION.SDK_INT <=
Build.VERSION_CODES.HONEYCOMB)

      getSupportActionBar().setDisplayHomeAsUpEnabled(true);

            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB)
                getActionBar().setDisplayHomeAsUpEnabled(true);

      }

      @TargetApi(Build.VERSION_CODES.HONEYCOMB)
      private void setupActionBarTitle() {

            String emergencyAddress = emergency.getAddress();

            if (Build.VERSION.SDK_INT < Build.VERSION_CODES.HONEYCOMB)

      getSupportActionBar().setTitle(Emergency.LABEL_EMERGENCY + ": "
+ emergencyAddress);

            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.HONEYCOMB)
                  getActionBar().setTitle(Emergency.LABEL_EMERGENCY +
": " + emergencyAddress);

      }

      @Override
      protected void onResume(){

            super.onResume();
            EmergenciesList.commonBehavoirInOnResumeMethod(localRSync,
userLocation, this);


            userLocation.attachLocationObserver(this);
            localRSync.attachObserverOfSynchronizer(this, true);

      }

      @Override
      public void doBeforeSync() {

            disableSyncIcon = true;
            EmergenciesList.commonBehavoirInDoBeforeSync(this);

      }
```

```
	@Override
	public void doAfterSync(String successMessage) {

		disableSyncIcon = false;

	EmergenciesList.commonBehavoirInDoAfterSync(successMessage,
this, tmpMessage);

		setEmergencyMap();

	}

	@Override
	public void doAfterNoSync(String messageAboutTheFail) {

		disableSyncIcon = false;

	EmergenciesList.commonBehavoirInDoAfterNoSync(messageAboutTheFai
l, this, tmpMessage);

		setEmergencyMap(); // permite actualizar la información
que cambia constantemente, como el tiempo transcurrido asociado a
diferentes puntos de interés

	}

	private void setEmergencyMap(){

		gMap.clear();
		getUserLocationOnMap();

	}

	@Override
	public void showDigitalResourcesList() {

		EmergencyMenu.showDigitalResourcesList(this, emergency,
userData, reflectionUtilities, tmpMessage);

	}

	@Override
	public void downloadDigitalResource(DigitalResource
selectedDigitalResource) {


	if(reflectionUtilities.checkExistence(DigitalResourceDownloader.
class)){

			drrDownloader =
compManager.getOptionalComponents().getDigitalResourceDownloader(selec
tedDigitalResource, this);
```

```
                    downloaderProgress =
DigitalResourcesList.getProgressBarOfDownload(this, drrDownloader);

                    downloaderProgress.show();

                    drrDownloader.executeTask();

              }

       }

       @Override
       public void openDigitalResource(DigitalResource
selectedDigitalResource) {

              if(drrOpener != null){


       tmpMessage.setText(CommonLabels.LABEL_OPENING_DIGITAL_RESOURCE);
                    tmpMessage.setDuration(Toast.LENGTH_SHORT);
                    tmpMessage.setGravity(Gravity.NO_GRAVITY,
Gravity.CENTER_HORIZONTAL, Gravity.CLIP_VERTICAL);
                    tmpMessage.show();

                    drrOpener.open(selectedDigitalResource, this);

              }

       }

       private String getDirectionsUrl(LatLng origin,LatLng dest){

          String str_origin =
"origin="+origin.latitude+","+origin.longitude;

          String str_dest =
"destination="+dest.latitude+","+dest.longitude;

          String sensor = "sensor=false";

          String parameters = str_origin+"&"+str_dest+"&"+sensor;

          String output = "json";

          String url =
"https://maps.googleapis.com/maps/api/directions/"+output+"?"+paramete
rs;

          return url;
     }
```

```
    // Método para descargar los datos json desde el url
    private String downloadUrl(String strUrl) throws IOException{
        String data = "";
        InputStream iStream = null;
        HttpURLConnection urlConnection = null;
        try{
            URL url = new URL(strUrl);

            // Creando una conexion http para comunicarse con el url
            urlConnection = (HttpURLConnection) url.openConnection();

            // Conectando al url
            urlConnection.connect();

            // leyendo datos desde el url
            iStream = urlConnection.getInputStream();

            BufferedReader br = new BufferedReader(new
InputStreamReader(iStream));

            StringBuffer sb  = new StringBuffer();

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

            data = sb.toString();

            br.close();

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

```
        /** Clase para descargar datos desde el Google Directions URL
*/
    private class DownloadTask extends AsyncTask<String, Void,
String>{

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

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

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

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

            ParserTask parserTask = new ParserTask();

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

/** Una clase para analizar las direcciones de Google en formato JSON
*/
    private class ParserTask extends AsyncTask<String, Integer,
List<List<HashMap<String,String>>> >{

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

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

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

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

// Executes in UI thread, after the parsing process
        @Override
        protected void onPostExecute(List<List<HashMap<String,
String>>> result) {
            ArrayList<LatLng> points = null;
            PolylineOptions lineOptions = null;

            // Recorriendo todas las rutas
            for(int i=0;i<result.size();i++){
                points = new ArrayList<LatLng>();
                lineOptions = new PolylineOptions();

                // Obteniendo la i-ésima ruta
                List<HashMap<String, String>> path = result.get(i);

                // Obteniendo todos los puntos en la i-ésima ruta
                for(int j=0;j<path.size();j++){
                    HashMap<String,String> point = path.get(j);

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

                    points.add(position);
                }

          // Añadiendo todos los puntos a la ruta con LineOptions
                lineOptions.addAll(points);
                lineOptions.width(8); //Grosor linea de la ruta
                lineOptions.color(Color.BLUE);
            }

            // Dibujando el polyline en el Mapa
            gMap.addPolyline(lineOptions);
        }
    }
private void drawMarker(LatLng point){
        mMarkerPoints.add(point);

        // Creating MarkerOptions
        MarkerOptions options = new MarkerOptions();
        // Setting the position of the marker
        options.position(point);

    }
```

```
      @Override
    public void onLocationChanged(Location location) {
        // Draw the marker, if destination location is not set
        if(mMarkerPoints.size() < 2){

            mLatitude = location.getLatitude();
            mLongitude = location.getLongitude();
            LatLng point = new LatLng(mLatitude, mLongitude);

            gMap.moveCamera(CameraUpdateFactory.newLatLng(point));
            gMap.animateCamera(CameraUpdateFactory.zoomTo(15)); //zoom
del mapa inicial

            drawMarker(point);
        }
    }

      public void getUserLocationOnMap(){

// Obteniendo el estado de disponibilidad de Google Play Services
                              int status =
GooglePlayServicesUtil.isGooglePlayServicesAvailable(getBaseContext())
;

                              if(status!=ConnectionResult.SUCCESS){ //
Google Play Services no está disponible
                                    int requestCode = 10;
                                    Dialog dialog =
GooglePlayServicesUtil.getErrorDialog(status, this, requestCode);
                                    dialog.show();
                              }else { // Google Play Services está
disponible
                                    // Inicializando
                                    mMarkerPoints = new
ArrayList<LatLng>();

                                    // Obteniendo referencia a
SupportMapFragment del activity_main
                                    SupportMapFragment fm =
(SupportMapFragment)getSupportFragmentManager().findFragmentById(R.id.
map);

                                    // Obteniendo el mapa para el
SupportMapFragment
                                    gMap = fm.getMap();

                                    // Obteniendo el objeto
LocationManager del servicio del sistema LOCATION_SERVICE
                                    LocationManager locationManager =
(LocationManager) getSystemService(LOCATION_SERVICE);
```

```
      // Creando un objeto de criterio para recuperar un proveedor
                                    Criteria criteria = new Criteria();

            // Obteniendo el nombre del mejor proveedor
                                    String provider =
locationManager.getBestProvider(criteria, true);

            // Obteniendo la ubicación actual desde el GPS
                                    Location location =
locationManager.getLastKnownLocation(provider);

                                    if(location!=null){
                                          onLocationChanged(location);
                                    }


      locationManager.requestLocationUpdates(provider, 20000, 0,
this);

                                    if(mMarkerPoints.size()>1){
                                          mMarkerPoints.clear();
                                          gMap.clear();

                                          LatLng startPoint = new
LatLng(mLatitude, mLongitude);


      mMarkerPoints.add(startPoint);
                                      MarkerOptions optionStart = new
MarkerOptions();

       // Configurando la posición del marcador de origen
                                      optionStart.position(startPoint);

optionStart.icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_po
i_available_user));
                                      gMap.addMarker(optionStart);

                                    }
LatLng destinyPoint = new LatLng(emergency.getLatitude(),
emergency.getLongitude());

                                    mMarkerPoints.add(destinyPoint);

                                  MarkerOptions optionsDestiny = new
MarkerOptions();

            // Configurando la posición del marcador de destino
                                  optionsDestiny.position(destinyPoint);

optionsDestiny.icon(BitmapDescriptorFactory.fromResource(R.drawable.ic
_poi_emergency));
                                    gMap.addMarker(optionsDestiny);
```

```
// Comprueba si las ubicaciones inicial y final son capturadas
                                    if(mMarkerPoints.size() >= 2){
                                          LatLng origin =
mMarkerPoints.get(0);
                                          LatLng dest =
mMarkerPoints.get(1);

                                          // Obteniendo el URL para el
API de Google Directions
                                          String url =
getDirectionsUrl(origin, dest);

                                          DownloadTask downloadTask =
new DownloadTask();

                                          // Comienza a descargar datos
de json del API de Google Directions
                                          downloadTask.execute(url);
                                    }
                              }

      }

      @Override
      public void doWhenDownloadProgresses(int progressValue) {


      DigitalResourcesList.commonBehavoirInDoWhenDownloadProgresses(pr
ogressValue, downloaderProgress);

      }

      @Override
      public void doAfterDownload(String successMessage,
DigitalResource downloadedDigitalResource) {


      DigitalResourcesList.commonBehavoirInDoAfterDownload(successMess
age, downloadedDigitalResource, downloaderProgress, tmpMessage,
drrOpener, this);

            setEmergencyMap();

      }



                                    .
                                    .
                                    .
```

4. *DirectionsJSONParser.java*

```
package
cl.ucsc.projectoftitle.newmobilemap.components.edbconnections.json;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

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

public class DirectionsJSONParser {
      /** Recibe un JSONObject y devuelve un listado que contiene
latitudes y longitudes */
    @SuppressWarnings({ "rawtypes", "unchecked" })
      public List<List<HashMap<String,String>>> parse(JSONObject
jObject){

        List<List<HashMap<String, String>>> routes = new
ArrayList<List<HashMap<String,String>>>() ;
        JSONArray jRoutes = null;
        JSONArray jLegs = null;
        JSONArray jSteps = null;

        try {

            jRoutes = jObject.getJSONArray("routes");

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

                /** Traversing all legs */
                for(int j=0;j<jLegs.length();j++){
                    jSteps = (
(JSONObject)jLegs.get(j)).getJSONArray("steps");

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

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

        } catch (JSONException e) {
            e.printStackTrace();
        }catch (Exception e){
        }
        return routes;
    }

    /**
    * Method to decode polyline points
    * Courtesy : jeffreysambells.com/2010/05/27/decoding-polylines-
from-google-maps-direction-api-with-java
    * */
    private List<LatLng> decodePoly(String encoded) {

        List<LatLng> poly = new ArrayList<LatLng>();
        int index = 0, len = encoded.length();
        int lat = 0, lng = 0;

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

            int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result
     >> 1));
            lat += dlat;

            shift = 0;
            result = 0;
```

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

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

}
```

C. Manual de usuario

En esta sección del anexo, se darán a conocer los pasos que se deben seguir al momento utilizar las nuevas funcionalidades para que resulten exitosas. Se mostrará además todos los mensajes que arroja la aplicación durante cada procedimiento.

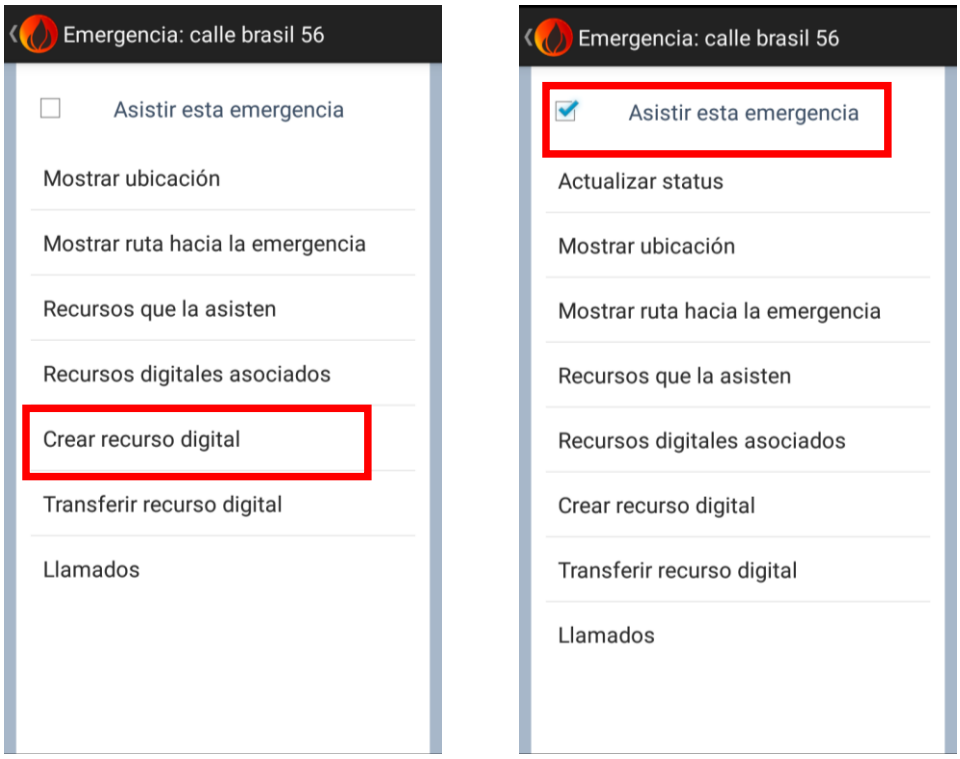
Cabe mencionar que para acceder al listado de todas las emergencias, el usuario, en este caso el bombero, debe estar previamente registrado en la base de datos.

C1. Manual de uso de del componente de audio

Al abrir la aplicación móvil, se mostrará un listado con todas las emergencias disponibles para ser atendidas, tal como se ve en la Figura 27, mostrando para cada emergencia la ubicación del incendio, la distancia en la que se encuentra con respecto a la ubicación actual del bombero, la cantidad de días en la que se inició dicha emergencia y la cantidad de carros bomba disponibles para cada emergencia. Es posible actualizar el listado, presionando el botón de sincronización marcado con rojo en la Figura 27.

Figura 27. Menú inicial con el listado de emergencias.

Si el bombero selecciona una emergencia del listado a la cual quiera acceder, se desplegará un menú de opciones para la emergencia escogida. Dentro de estas opciones se encuentra la nueva funcionalidad para crear el recurso digital de audio seleccionando la opción “Crear recurso digital” (ver Figura 28(a), pero para hacer uso de ella, el bombero deberá estar asistiendo a la emergencia marcando la casilla disponible para ello (marcado en rojo en Figura 28(b)).



a) *b)*

Figura 28. Menú de opciones antes/después de activar asistencia.

Por el contrario, si se selecciona “Crear recurso digital” sin asistir a la emergencia, el sistema no permitirá la creación del recurso y el bombero recibirá un mensaje como en la Figura 29.

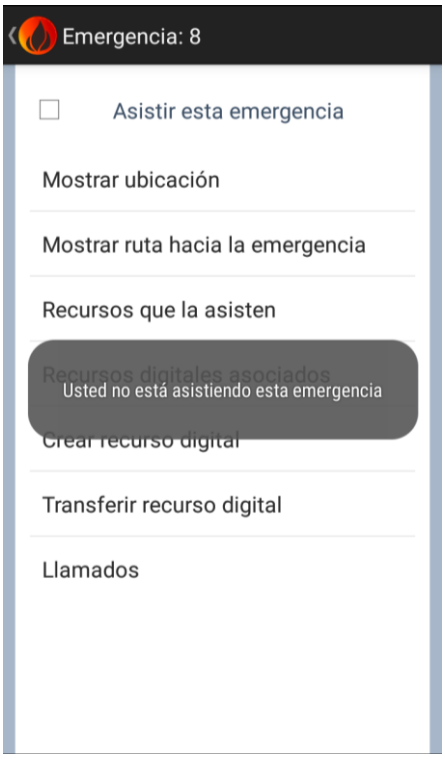


Figura 29. Mensaje por no asistencia a una emergencia.

Si el bombero asiste a la emergencia y elige la opción “Crear recurso digital”, se desplegarán 3 opciones para la creación del recurso y deberá seleccionar la opción “Grabar audio” (ver Figura 30).

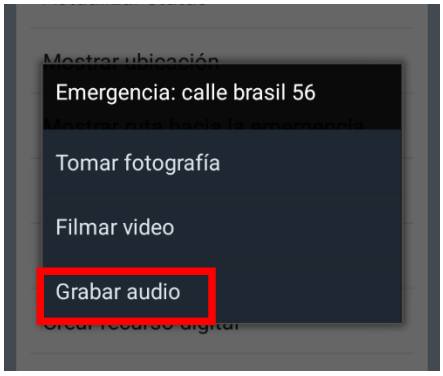
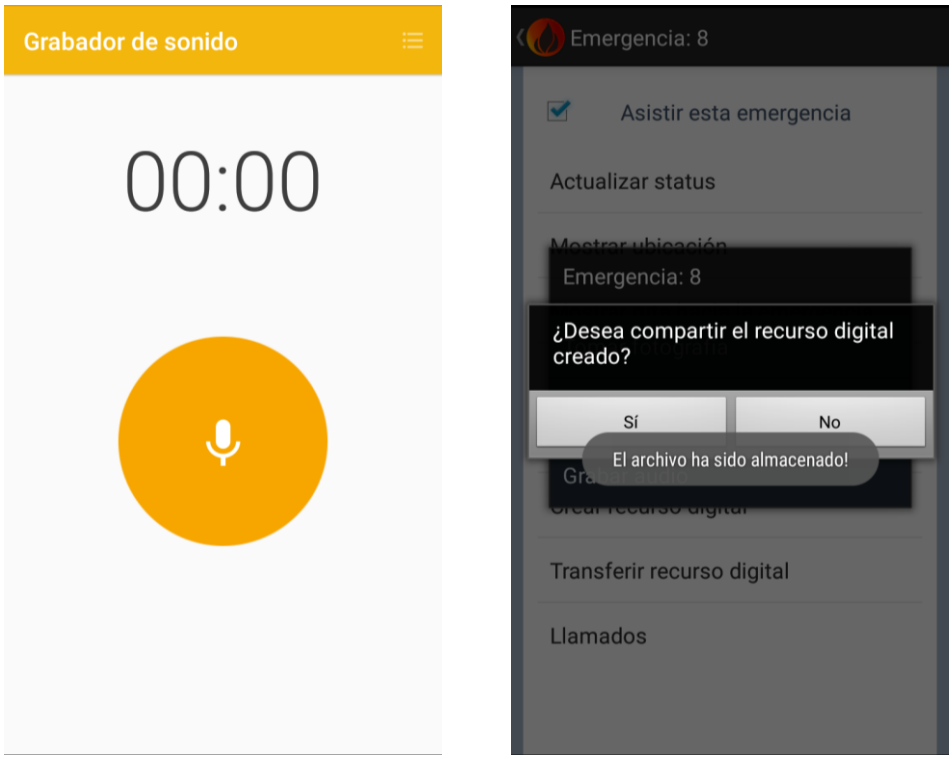


Figura 30. Menú de opciones para la creación de un recurso digital.

Inmediatamente después de seleccionar “Grabar audio” se abrirá el grabador de audio predeterminado de Android (ver Figura 31(a)) y el bombero deberá grabar su mensaje. Luego de la grabación, la aplicación móvil desplegará un mensaje informando que “El archivo ha sido almacenado!” y también preguntará al bombero si éste desea compatir el archivo digital recién creado, como se ve en la Figura 31(b). Si se escoge la opción “No” se volverá al menú según la Figura 30.



a) *b)*

Figura 31. Grabación de audio exitoso exitoso para compartirlo.

Por otro lado, al seleccionar “Sí” en la ventana de diálogo “¿Desea compartir el recurso digital creado?”, se desplegará un formulario que contiene el tipo de recurso digital, los destinatarios y de forma opcional, una descripción para der agregada al recurso a enviar (ver Figura 32).

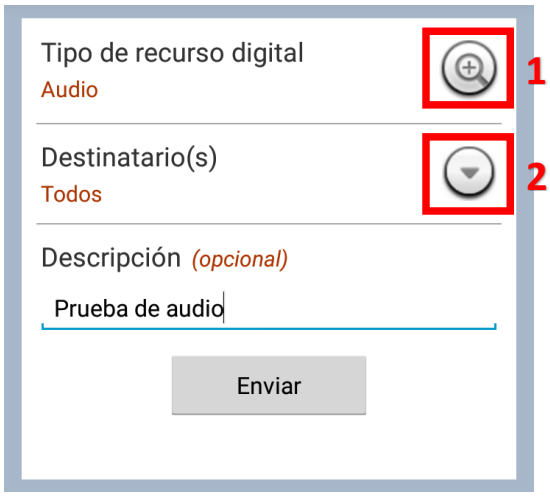
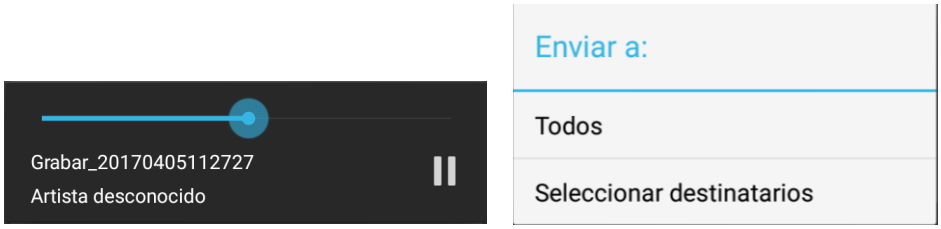


Figura 32. Formulario disponible para enviar un recurso digital

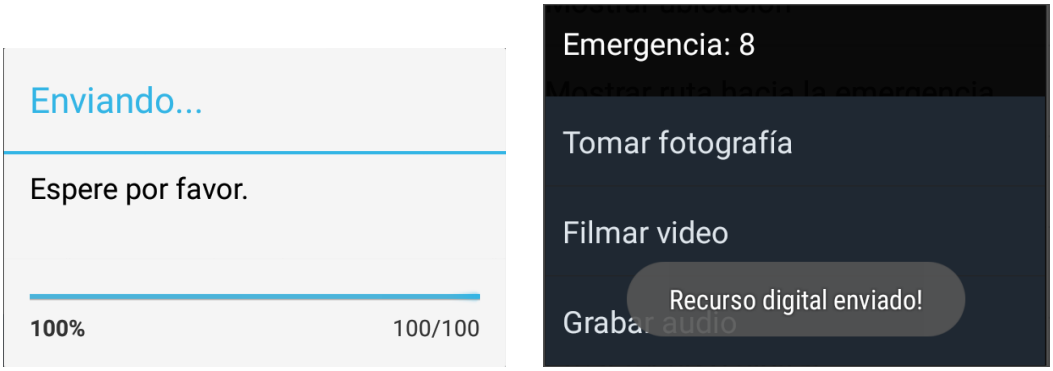
Del formulario de la Figura 32, se despliegan dos ventanas adicionales. El ícono de lupa marcado en rojo al costado derecho de el "Tipo de recurso digital" (marcado con un "1") permite ver el recurso digital creado, en este caso, reproduce el audio recién grabado y que será enviado (ver Figura 33(a)), y el íc ono de flecha al costado derecho de "Destinatario(s)" (marcado con un "2") permite seleccionar a quién enviar el recurso digital creado (ver Figura 33(b)).



a) *b)*

Figura 33. Reproducción del audio y selección de destinatarios.

Después de llenar el formulario, se debe presionar “Enviar” y el recurso digital de audio será enviado, mostrando una ventana de progreso con el envío hasta completar el 100%, tal como muestra la Figura 34(a). Finalmente, cuando el envío fue exitoso se desplegará el mensaje “Recurso digital enviado!” como se ve en la Figura 34(b).



a) *b)*

Figura 34. Progreso de envío de recurso digital y mensaje de éxito.

C2. Manual de uso del componente de ruta hacia la emergencia

Al abrir la aplicación móvil, se mostrará un listado con todas las emergencias disponibles para ser atendidas, tal como se muestra en la Figura 35, mostrando para cada emergencia la ubicación del incendio, la distancia en la que se encuentra con respecto a la ubicación actual del bombero, la cantidad de días en la que se inició dicha emergencia y la cantidad de carros bomba disponibles para cada emergencia. Es posible actualizar el listado, presionando el botón de sincronización marcado con rojo en la Figura 35.

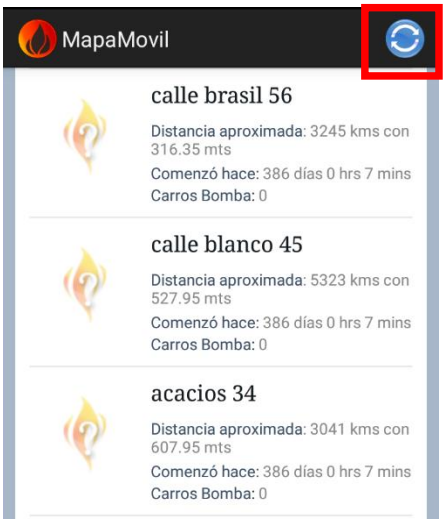


Figura 35. Botón de sincronización en el menú inicial.

Si el bombero selecciona una emergencia del listado a la cual quiera acceder, se desplegará un menú de opciones para la emergencia escogida. Dentro de estas opciones se encuentra la nueva funcionalidad para trazar la ruta hacia la emergencia seleccionando la opción "Mostrar ruta hacia la emergencia" (ver Figura 36), y a diferencia del componente de audio mostrado anteriormente, no es necesario asistir a la emergencia para hacer uso de esta funcionalidad.

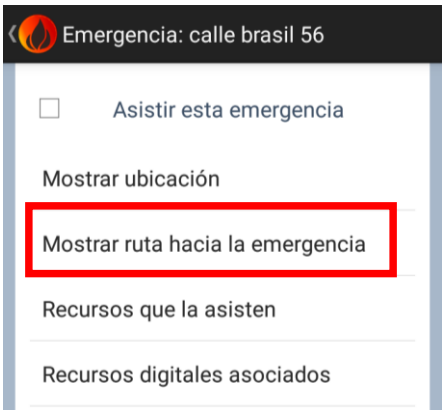


Figura 36. Mostrar ruta de una emergencia.

Inmediatamente después de seleccionar la opción marcada en rojo en la Figura 36, se mostrará al bombero el mapa con el trazado de la ruta desde la ubicación actual del bombero hasta la ubicación del incendio (ver Figura 37).

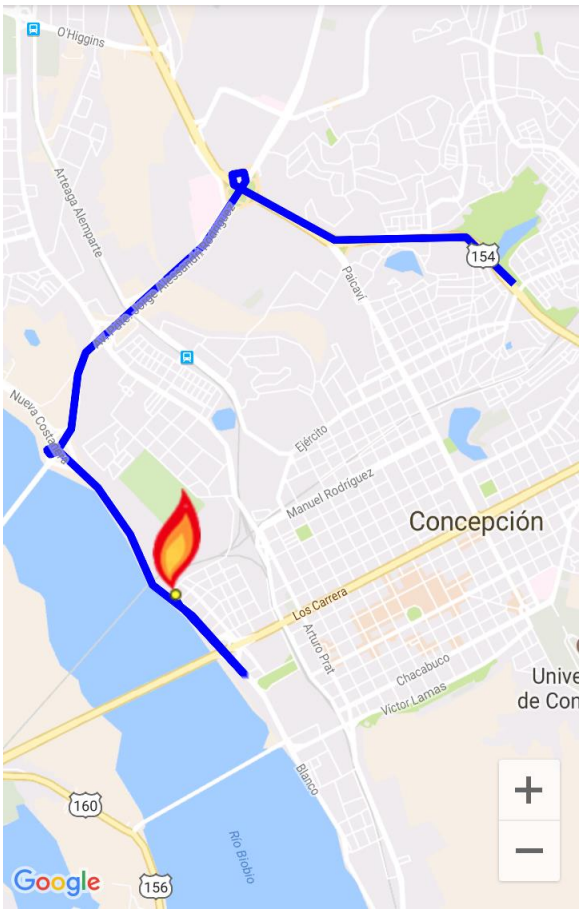


Figura 37. Trazado de ruta hacia una emergencia.

Finalmente, a medida que el bombero cambie su ubicación inicial, la aplicación actualizará este punto en el mapa formando la nueva ruta para la emergencia. Esta actualización se puede llevar a cabo cuando la aplicación genera una sincronización de forma automática o también, presionando el botón de sincronización mostrado en la Figura 35.