
# Universidad Politécnica de Valencia

## Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información

## Tesis Fin de Máster

## Asistente Personal de Seguridad: Infraestructura y Cliente Android

**Ángel Ruiz Zafra**

Bajo la dirección académica de:

José Hilario Canós Cerdá

Departamento de Sistemas Informáticos y Computación

Septiembre 2012

# Resumen

El presente trabajo de investigación se sitúa dentro del contexto de la gestión de emergencias.

Se propone un diseño y desarrollo de un sistema que sea capaz no solo de dar soporte a gestionar toda la información referente a una situación de emergencia, sino además de ayudar al usuario final a actuar correctamente. Ayudando así al usuario al indicarle correctamente la serie de acciones a realizar para poder evacuarse a si mismo correctamente.

Para conseguir dicho objetivo, se han diseñado y desarrollado 3 elementos de cara a dotar al sistema de todas las funcionalidades establecidas como objetivos. Estos elementos son: una herramienta web para la generación de contextos de emergencia, una arquitectura orientada a servicios y una aplicación para Android que haga uso de los elementos anteriores. Comprobando así que todo el sistema funciona y se integra correctamente para cumplir con los objetivos expuestos en esta tesis.

# Agradecimientos

Quisiera poner de manifiesto mi más sincero agradecimiento al Dr. D. José Hilario Canós Cerdá, por toda la ayuda prestada y por un continuo seguimiento y apoyo durante todo el desarrollo de la presente tesis.

También agradecer la colaboración del profesor Francisco Javier Jaén Martínez y al investigador Kamil Krynicki, miembros del grupo de investigación de Ingeniería del Software y Sistemas de Información (ISSI). Tanto por la cesión del componente que contiene el motor de optimización y búsqueda de caminos basados en algoritmos de colonias de hormigas, como por toda la ayuda prestada a lo largo del año para acoplar correctamente dicho componente en el sistema que se presenta en esta tesis.

# Índice

# Parte I: Descripción del proyecto

# I.1

# Introducción

## I. 1.1. Descripción

Actualmente los sistemas de gestión de emergencias actúan en función de protocolos de comportamiento que prácticamente no han variado en los últimos años. Los planes de emergencia incluyen instrucciones para los equipos de respuesta, así como para las personas potencialmente afectadas [1]. Este último caso es el dominio de trabajo de nuestro proyecto. Por lo general, las instrucciones de seguridad ofrecidas a los usuarios de servicios como hoteles, aeropuertos, y otros consisten en recomendar al usuario que abandone la instalación por la salida de emergencia más próxima, sin usar ascensores, usando las correspondientes escaleras de incendios, etc. La localización de las salidas se muestra en diagramas poco accesibles (por ejemplo, tras la puerta de la habitación del hotel), y que, en caso de necesidad, son de efectividad discutible.

Además, las mencionadas instrucciones son de tipo genérico, ignorando las posibles condiciones personales de un usuario. Por ejemplo, una persona con discapacidad no puede seguir las indicaciones por defecto o generales, en cuyo caso debería actuarse de otra forma. O las personas que se encuentran en la planta de un edificio que corresponde con la planta donde se ha declarado el incendio: su plan de evacuación debe ser obviamente distinto al resto.

La necesidad de personalizar los planes de evacuación de acuerdo al contexto abre la puerta a la implantación de nuevas soluciones que permitan actuar en función de las necesidades de cada usuario, obteniendo, si es posible, datos a priori para poder actuar en consecuencia cuando sea necesario.

Aunque existen sistemas de gestión de alertas informatizados, por lo general su objetivo es indicar de una manera más clara y eficaz el protocolo de evacuación, sin

tener en cuenta las necesidades o condiciones de un determinado usuario como las antes comentadas. Dichos sistemas por lo general pertenecen a grandes superficies o empresas privadas, por lo que no están disponibles para todo el mundo y son difícilmente extensibles.

## I. 1.2. Motivación

Dentro del contexto de gestión de planes de emergencia, la orientación que ha seguido el grupo de investigación de Ingeniería del Software y Sistemas de Información (ISSI) del Departamento de Sistemas Informáticos y Computación (DSIC) ha sido aunar por un lado la Ingeniería de Documentos junto con las Líneas de Producto de Software para generar una nueva propuesta llamada Líneas de Producto de Documentos (*Document Product Lines*, DPL) [2].

Dicha rama tiene el principal objetivo de poder generar documentos totalmente personalizables, llegando hasta el caso utópico de generar un documento por cada usuario final en función de las necesidades del mismo. De esta manera un usuario final únicamente percibiría la información relacionada con el o con su interés. DPL es perfectamente aplicable al contexto de emergencias, donde la extrapolación de dicha idea es definir a priori una serie de posibles evacuaciones para que cada usuario reciba su plan de evacuación particular [3]. Se evitaría así la tediosa tarea de tener que mirar todo el documento donde está representado el plan de evacuación descrito por un ingeniero del dominio. Lo cual en situaciones de emergencia no es nada aconsejable, debido a que dificulta la tarea debido al contexto y retrasa totalmente la evacuación del paciente.

Haciendo uso de nuevas tecnologías como dispositivos móviles de última generación, y aplicando dicha tecnología al contexto de los sistemas de emergencia, se posibilitaría crear un sistema completo de gestión de emergencias en tiempo real.

Actualmente cada vez más gente usa Smartphone, los cuales debido a su gran potencia en relación a su tamaño y su fácil portabilidad posibilitan tener un sistema de control portable que pueda ayudar al usuario a seguir una serie de pasos de cara a poder realizar un plan de evacuación correctamente. Cumpliendo así el objetivo de poner a salvo al usuario a través de su Smartphone

# I. 1.3. Objetivos Globales

El proyecto "The Personal Safety Assistant" tiene el propósito de diseñar y desarrollar una plataforma software compuesta de varios elementos: arquitectura software, aplicaciones de uso, aplicaciones de gestión, etc., con el que se pueda hacer una gestión total de una emergencia orientada a los usuarios que se vean implicados en dichas situaciones.

Con esta premisa, el sistema debe cumplir los siguientes objetivos:

- **Facilidad y simplicidad de uso:** El sistema debe ser lo más amigable posible, no solo para facilitar su utilización al usuario, sino que durante una emergencia, el software no debe generar la duda por parte del usuario.
- **Robustez, eficiencia y estabilidad:** en este ámbito el margen de error debe reducirse en todo lo posible, ya que un fallo en un momento crítico puede ser desastroso. Además el sistema debe ser eficiente para dar un mejor servicio.
- **Portabilidad**: interesa que el sistema pueda operar en diversos dispositivos móviles con diferentes características. También hay que considerar otros dispositivos intermedios, como pueden ser los *Tablets*, donde también se puede usar este sistema. Ello también redundará en una mayor adaptación al usuario permitiendo la elección de dispositivos en función, por ejemplo, de las dimensiones de la pantalla de cada tipo de dispositivo.
- **Utilización de formatos estándares y abiertos:** Se usarán estos para facilitar así el tratamiento y futuro uso de la información por aplicaciones independientes. Se usarán tecnologías libres y ampliamente extendidas como XML, MySQL, Java, etc., con lo que conseguiremos la interoperabilidad de aplicaciones entre usuarios y la facilidad para los futuros desarrolladores que trabajen con el sistema.
- **Configuración del sistema sin usar herramientas externas:** Es importante centralizar la configuración de las distintas aplicaciones que puedan formar parte del sistema con el objetivo de facilitar la realización de este tipo de tareas.
- **Escalabilidad:** Hay que diseñar el sistema considerando que puede haber un elevado número de usuarios haciendo uso del mismo.
- **Gestionar las emergencias:** No es suficiente con que al usuario se le indique que hay una emergencia, sino que se le indique cuál debe ser su forma de actuar. Debido a que en función del escenario y de la emergencia en concreto el comportamiento debe cambiar, el sistema debe diferenciar entre los distintos grados de emergencia que pueda haber e interactuar con el usuario indicándole qué hacer.

- **Gestión de notificaciones:** También puede ser muy útil que el sistema informe a los distintos usuarios con un mensaje o notificación concreta, ya sea una emergencia, un mensaje informativo, etc.
- **Sistema de respuesta en tiempo real:** La respuesta en tiempo real del sistema es totalmente prioritaria por el ámbito del problema y por las desastrosas consecuencias del mismo en caso de no responder adecuadamente.
- **Almacén de datos:** Se deberá almacenar toda la información no solo de los usuarios y entidades sino también todas las alertas que ocurran y las notificaciones que se indiquen, con el objetivo de poder usar dichos datos para su posterior análisis y uso.

## I. 1.4. Objetivo de la tesis

El principal objetivo es definir el diseño y desarrollo de un sistema capaz de gestionar situaciones de emergencia correctamente, esto es, ser capaz de actuar en una situación de emergencia de una manera suficientemente automática y que permita al usuario final hacer uso de dicho sistema para conseguir actuar correctamente. Por lo general, el sistema guiará al usuario a través de una serie de pasos hasta que esté en lugar seguro.

Todo el sistema se va a dividir en dos grandes partes representadas en dos tesis fin de Máster diferentes. Una de las dos partes es la tesis expuesta en este documento. La cual tiene como principales tareas diseñar y desarrollar tres partes del sistema que se explican a continuación:

- **Herramienta de creación de planes de evacuación**: Cada plan de evacuación es distinto, ya que depende del usuario que lo pida así como de la posición en la que se encuentre y el contexto en concreto. Debido a esto se hace necesario definir un mecanismo de generación de planes de evacuación automático que sea capaz de dar soporte a cualquier usuario en cualquier situación posible. Dicha herramienta es la mostrada en el apartado 3.1, en la parte 2 del documento, sección 3 (Implementación).
- **Arquitectura Software**: Es importante el desarrollo de una arquitectura software sobre la que se base todo el sistema y que permita al desarrollador hacer uso de ella de cara a crear un buen sistema para la generación de planes de evacuación. Sobre esta arquitectura software (apartado 3.2 de la parte 2) se definirá el esquema de información correspondiente a la segunda tesis.
- **Personal Safety Assistant**: Haciendo uso de los dispositivos móviles, los cuales gracias a su potencia y escaso tamaño potencian su uso en un contexto de este tipo, se diseñará e implementará una aplicación que pueda usar cualquier cliente y que le permita automáticamente haciendo uso de ella actuar correctamente en una situación de emergencia. Dicha aplicación, desarrollada en Android

(apartado 3.3), hará uso de los servicios definidos en la arquitectura software antes comentado: obtener planes de evacuación, obtener información de algún cliente, ir almacenando información en el log, etc.

La segunda Tesis, que complementa el sistema, se centrará en definir el esquema de información capaz de dar soporte a toda la información a almacenar en un sistema de este tipo: datos de usuarios, estados de emergencias, etc.

Dicho esquema de información tendrá una implementación concreta que será usada por la arquitectura software (más concretamente por los servicios de la misma) para acceder a la información.

Además de esto, en esta segunda Tesis también se aborda el desarrollo de software para dispositivos móviles pero orientado a plataformas iOS (iPhone, iPad) implementado tanto la aplicación final del usuario como otras aplicaciones para otros tipos de usuarios (bomberos, personal médico).

## I. 1.5. Estructura del documento

El presente documento tiene cuatro grandes bloques/partes. En la parte 1 (donde este apartado está incluido) se expone una breve descripción y motivación. Y pone en contexto al lector sobre el tema que se va a tratar en la tesis.

El segundo bloque es el de mayor extensión e importancia. Es el bloque donde se plasman los resultados de la fase de análisis, diseño e implementación. La fase de análisis y de diseño muestran varios diagramas para comprender mejor tanto la estructura del sistema así como su comportamiento.

La parte de conclusiones (tercer bloque) expone un breve resumen o conclusión sobre todo el trabajo. Además se proponen diversas mejoras para el sistema. Pudiéndolo mejorarlo considerablemente tanto en funcionalidad como en eficiencia y eficacia. Dicha mejoras se aportan como trabajo futuro.

El último y gran bloque es el formado por los Anexos, donde se muestran información adicional o bien para entender mejor el sistema o bien como usar la funcionalidad de este.

# I.2

# El *Personal Safety Assistant*

## I. 2.1. Esquema general

La Figura 1 muestra una vista general de la interacción de los distintos actores o participantes dentro de una situación de emergencia. Vamos a considerar cuatro tipos de elementos fundamentales dentro de este escenario:

- **Clientes**: Tendrán en su propiedad un Smartphone o Tablet con el software correspondiente para poder interactuar con el sistema. La función de este será establecer la comunicación con el servidor de un edificio o entidad en concreto una vez que se encuentre dentro del edificio. Una vez establecida la conexión y reconocido por el servidor se registrará, si es posible automáticamente, en el mismo pudiendo interactuar con dicho servidor en caso de que se declare una emergencia.
- **Servidor Global**: Es un servidor común a todos los servidores locales y tiene varios objetivos, entre los que cabe destacar:
    - Almacenamiento de la información referente a los distintos servidores locales que forman el sistema, a modo de backup (temporal o no)
    - Gestión de emergencias: Aunque sean los servidores locales los que interaccionen con dichos usuarios, este servidor podrá cumplir con los mismos objetivos con el fin de suplir una posible caída de servidores locales.
    - Gestión de incidencias: Este servidor, por lo general ,será el encargado de enviar la incidencia después de que se declare una alerta a los distintos

servicios de gestión de emergencias, como puede ser al departamento de bomberos local o provincial, a un servicio médico que puede ser un hospital o centro médico, una notificación a la policía local, etc.

- **Edificio**: Un edificio estará definido por la pareja Servidor Local (Bank Server, Hospital Server, etc. en la Figura 1) y sistema de alerta de emergencias, o EAS por sus siglas en inglés. El servidor, permitirá la conexión de varios usuarios y sus peticiones, se encargará de gestionar los distintos usuarios que se encuentran en el edificio y las correspondientes alertas o notificaciones personalizadas para cada uno de los usuarios.
- **Gestión de emergencia externo:** Además de notificar y gestionar correctamente las incidencias, se hace necesario poder notificar de dichas incidencias a los organismos públicos o privados encargados de éstos. Este último grupo, que puede ser un departamento de policía, de bomberos o sencillamente un hospital, tendrá la responsabilidad de atender la petición como si se tratase de una emergencia estándar.

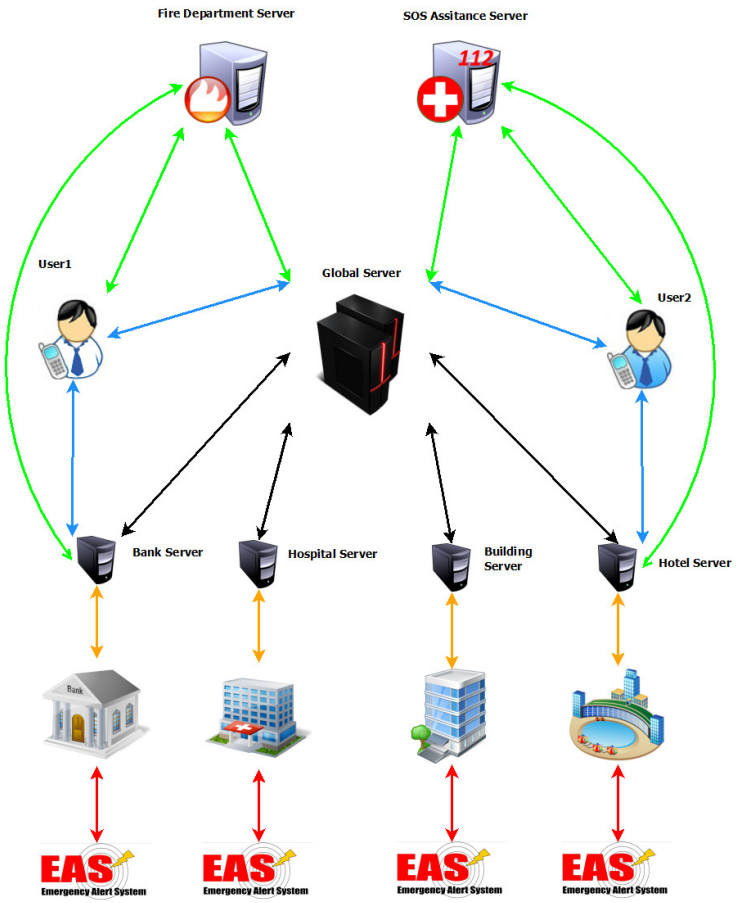



Figura 1 – Esquema general

El propósito de definir diversos tipos de usuarios, así como diversos elementos dentro de la arquitectura como por ejemplo varios tipos de servidores, es no solo el de conseguir un sistema capaz de soportar cualquier tipo de emergencia y que dicho procesamiento sea totalmente automático, transparente y fácilmente interpretable por el usuario final; también, el poder establecer algún tipo de relación con entidades externas para que realicen su trabajo sin cambiar, en principio, el protocolo de actuación.

De esta manera el usuario sencillamente tendrá que seguir las indicaciones que se muestran en el dispositivo móvil, las cuales, junto con las acciones realizadas por los equipos de emergencia, garantizarán una mejor actuación.

## I. 2.2. Escenarios alternativos

Aunque si bien es cierto que la Figura 1 presenta un escenario típico y general dentro de un sistema de gestión de emergencias, no deja de ser una representación concreta de un caso real.

Debido a la multitud de posibilidades que puede haber en el mundo real, será necesario que nuestro sistema sea capaz de dar soporte a escenarios concretos, con el fin de que se pueda implantar en cualquier situación siempre y cuando se posea la arquitectura adecuada.

Un escenario particularmente crítico es el que se representa en la figura 2. Supongamos que se ha producido un incendio en un edificio (en este caso concreto un banco), con la mala suerte de que el incendio se ha iniciado en la sala donde se encontraba el servidor local Con lo que no se puede establecer comunicación con el servidor y por lo tanto no se puede notificar una emergencia.

De esta manera, las relaciones del servidor tanto con los usuarios del edificio cómo con el servidor general quedan rotas, por lo que en este escenario concreto y gracias a la aplicación implantada en el dispositivo móvil es el usuario el que puede dar la notificación de alerta tanto al servidor general, que será el encargado a partir de ahora de atender las peticiones de los distintos usuarios del sistema, como de enviar las notificaciones al servidor global.

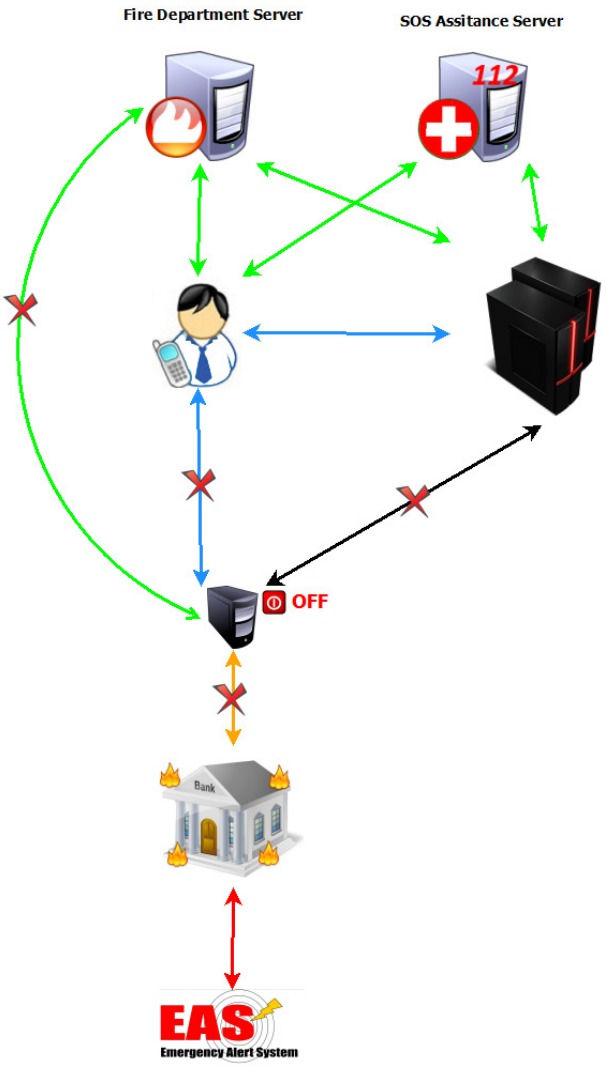



Figura 2. Escenario alternativo (I)

Otro escenario posible puede ser el que se representa en la Figura 3. En él, aunque el sistema no ha sufrido daño alguno, existen varios planes de actuación dentro de mismo edificio/entorno. Este caso en concreto es un *resort*, o complejo hotelero, que está formado por varios edificios, como puede ser el propio hotel, un cine con varias salas, un centro comercial o un centro médico u hospital para atender emergencias satinarías. Con el fin de hacer el sistema más versátil se introduce un servidor por cada edificio, coordinados localmente por un servidor general del resort.

Aunque en la figura no se muestran todas las relaciones para hacer la figura más sencilla, existen relaciones entre todos los servidores locales y el servidor principal y servidores externos de emergencias, con el fin de que el funcionamiento interno de cada edificio pueda continuar aunque se produzcan fallos en el resto. De igual manera, los usuarios tendrán las mismas relaciones ofreciendo todos los servicios representados en el esquema general (Figura 1).

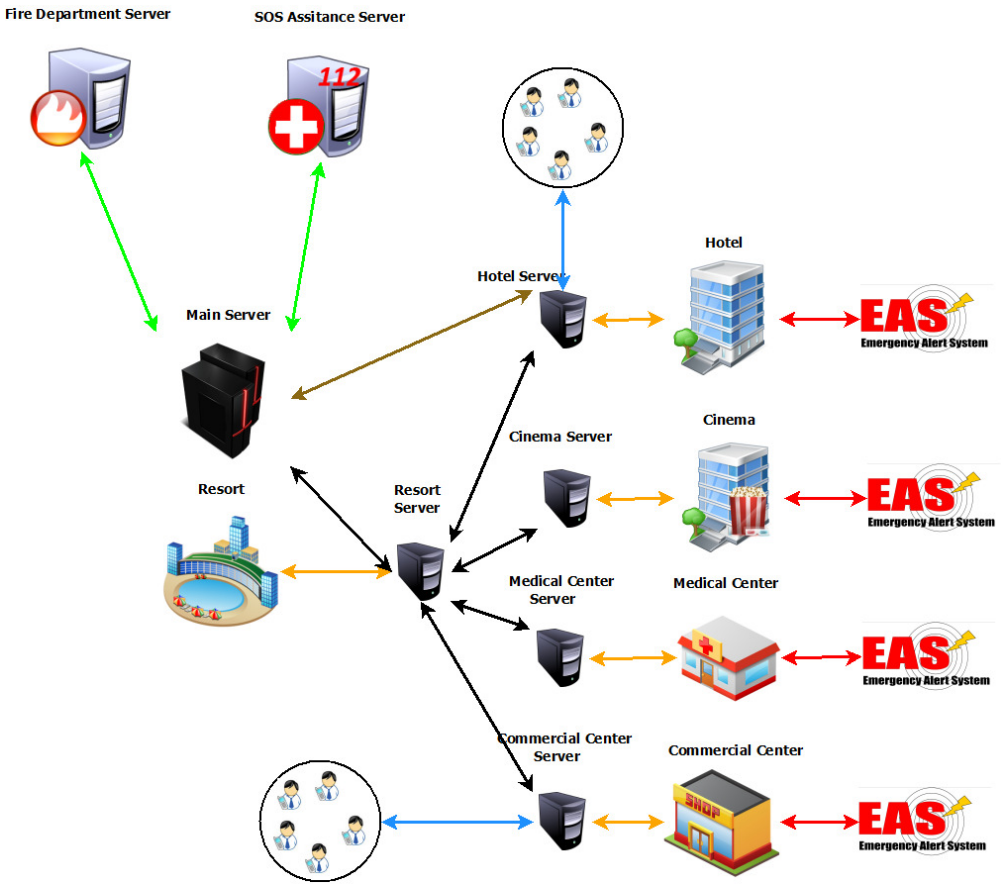



Figura 3 – Escenario alternativo (II)

## I. 2.3. Funcionalidades

Las distintas funcionalidades que posee el sistema son:

**Gestión de usuarios**

Aunque por la naturaleza del sistema no es imprescindible la información personal de un usuario, siempre es interesante registrar cualquier dato que pueda repercutir en el bienestar del mismo. Como puede ser si posee algún tipo de discapacidad o estado de salud. Es por esto que se considera necesaria la gestión de los distintos usuarios del sistema.

- **Nuevo usuario**: A Cualquier usuario que instale el software se le dará la opción rellenar su información personal.
- **Eliminar usuario**: Un usuario puede solicitar darse de baja del sistema.
- **Modificar usuario**: Un usuario podrá modificar sus datos personales cuando lo crea necesario.
- **Información del usuario**: Será posible ver la información de cualquier usuario por parte de los equipos de respuesta en caso de necesidad.

**Gestión de planes de evacuación**

Uno de los principales objetivos del sistema es dar al usuario unas pautas de acción cuando se declara una emergencia y por tanto el consiguiente seguimiento de un plan de evacuación. Estas pautas son el plan de evacuación, el cual se genera para cada cliente.

- **Nuevo plan de evacuación:** se generarán las pautas o acciones que se deben seguir si se define un tipo de alerta y en función de la información personal de un usuario.
- **Información del plan de evacuación:** Se podrá ver la información referente a un plan de evacuación.

**Gestión de posicionamiento en interiores**

Debido a que el plan de evacuación puede variar en función de la planta del edificio o incluso de la zona, resulta imprescindible saber la localización del usuario dentro del edificio en el momento en el que se quiere evacuar el edificio. De esta manera, conociendo su localización se puede actuar en consecuencia y ajustar el plan de evacuación a la situación en la que se encuentre el usuario cuando dicha emergencia sea declarada.

Se usará algún método de posicionamiento en interiores que de cara a obtener la posición inicial del usuario, para conseguir así dirigirlo correctamente a una posición final.

**Gestión de entidades**

Se entiende por entidad cualquier organismo público o privado que desee usar el sistema. Para esto deberá, además de tener el hardware necesario y el software instalado, registrarse como "nodo" del sistema en el servidor principal.

- **Nuevo entidad:** Será la propia entidad la que se dé de alta en el sistema.
- **Modificar datos de la entidad:** La modificación de datos de información la podrá realizar la propia entidad o bien una entidad externa autorizada como el servidor global.
- **Dar de baja una entidad:** Una entidad que no desee formar parte del sistema podrá darse de baja.
- **Ver información personal:** Se podrá ver la información personal referente a una entidad.

**Gestión de Históricos (Log)**

Debido al contexto crítico en el que se encuentra la aplicación se hace necesario registrar todas y cada una de las acciones que ocurren en el sistema por parte de los distintos usuarios en un registro que a posteriori pueda ser consultado por actores expertos (bomberos, policías, etc.) para obtener información sobre el estado actual de la emergencia, el número de personas que aún no han sido evacuadas, la última posición de algún usuario, etc. Por esto las distintas funcionalidades son:

- **Nueva entrada en el Log:** Se podrá añadir entradas en el histórico cada vez que se realice una acción en el sistema: cuando se declara una emergencia, la última posición de un usuario, etc.
- **Borrar información del Log de un cliente:** Debido al gran volumen de datos que puede tener el Log es comprensible que de la declaración de una emergencia a otra se quiera limpiar el log para dejarlo totalmente vacío.
- **Ver información del Log de un cliente:** Se podrá recuperar información del Log en función a ciertos parámetros para poder así recuperar datos particulares: mostrar todos los usuarios que hicieron alguna acción, mostrar el recorrido que ha seguido un usuario, etc.

**Gestión de Escenarios**

Para poder generar planes de evacuación de una manera totalmente personalizada y en cierta medida automática es necesario ser capaz de poder diseñar un escenario que albergue, en cierta medida, todos los posibles planes de evacuación posibles. De esta manera en función de la posición del usuario y del contexto se generará el plan de evacuación correspondiente. El sistema permite, a través de una herramienta:

- **Generar Escenarios:** Después de que el usuario correspondiente diseño el escenario, la propia herramienta debe permitir generar dicho escenario como un proyecto para su posterior tratamiento en un futuro.
- **Cargar Escenarios:** La herramienta debe permitir carga escenarios previamente creados, facilitando la modificación del diseño para si se desea poder guardarlo a posteriori.
- **Generar una implementación del diseño:** Para poder usar dicho diseño (totalmente visual) en nuestro sistema, se hace necesario traducir dicho diseño a una implementación concreta que pueda ser usada y manejada por el sistema. La herramienta posibilita generar una implementación a partir de un diseño o un proyecto/escenario previamente cargado.
- **Generar Zonas:** La herramienta permitirá exportar o generar partes del diseño como zonas independientes, para así poder ser usadas en otros diseños.
- **Cargar Zonas:** Se posibilita que se carguen en un diseño concreto zonas ya definidas, acoplándolas al diseño para su adaptación como se considere oportuno.

# I.3

# Tecnologías en generación de planes de evacuación

Una vez especificado el escenario o contexto del problema, se hace necesario especificar cómo va a ser posible llevar los objetivos ya mencionados a cabo. Para esto se hará un repaso de las tecnologías ya existentes y a considerar como futuras tecnologías a usar para este proyecto, así como algunos ejemplos de proyectos similares ya desarrollados o en desarrollo.

## I. 3.1. Estudio Tecnológico

### I. 3.1.1. Sistemas de gestión de emergencias

En los últimos años han sido desarrollados numerosos sistemas relacionados con situaciones de emergencia. Esto está totalmente relacionado con la aparición de los dispositivos móviles, tanto por su potencia y tamaño como por su expansión y aceptación social. Así como por el fácil acceso a internet.

Esta potencia de los dispositivos móviles junto con la posibilidad de guardar y recuperar cualquier tipo de información en cualquier momento de tiempo hace que el contexto de emergencias pase a ser un contexto fácilmente procesable, desde un punto de vista de gestión de la información.

Debido al amplio espectro que abarca el contexto de emergencias: ¿Cómo organizar la información? ¿Cómo interactuar con el dispositivo móvil? ¿Cuándo y cómo declarar una emergencia? ¿Cómo reaccionar ante una emergencia? Han aparecido numerosos proyectos los cuales por lo general se centran en algún aspecto concreto dentro de todo el contexto de gestión de emergencia.

Algunos proyectos como el mostrado en [4] se centran en como coordinar correctamente los flujos de información en una situación de emergencia. Es decir, que

información recibe cada uno de los actores del sistema y como. Además de centrarse en garantizar que dicho flujo de información sea correcto.

El proyecto que se presenta en [5] sin embargo se centra en proporcionar al usuario en tiempo real información contextual de emergencias producidas por eventos naturales (volcanes, huracanes, etc.). Dicho sistema tiene una utilidad muy concreta: ser consciente de que hay una emergencia. Pero no dice al usuario del sistema que hacer, o que recomendaciones seguir.

El último proyecto a comentar y que se muestra en [6], es un aplicación que permite al usuario final definir una serie de datos a considerar en una situación de emergencia: teléfono de emergencia, teléfono de familiares, mensajes de alerta a alguna entidad o familiar, etc. Dicha aplicación está conectada a un servidor externo que es el encargado de gestionar dicha información.

Haciendo uso del GPS y cuando se detecte una situación extraña, dicha aplicación enviará las notificaciones o mensajes a quien corresponda en función a su configuración.

Aunque habrá otros muchos proyectos centrados en otras muchas funcionalidades del contexto de las emergencias, como se ha podido ver en los 2 últimos ejemplos, los sistemas reaccionan siempre de la misma manera. Es decir, independientemente de la situación personal del usuario así como de la situación concreta de la emergencia, la información es la misma para todos los usuarios.

El sistema propuesto en esta tesis pretende proponer un diseño general que sea capaz de dar soporte a cualquier tipo de emergencia (tanto en interiores como en exteriores) y que sea totalmente consciente del contexto. Pudiendo así ofrecer la mejor opción a cada usuario en una situación de emergencia.

## I. 3.1.2. Plataformas móviles

A día de hoy existen numerosas plataformas móviles, cada una con sus ventajas e inconvenientes, pero suficientemente potentes y con las tecnologías y características adecuadas para soportar nuestro proyecto sin ningún tipo de problema.

A esto, hay que añadirle el rápido crecimiento de los Tablets que se pueden considerar también dentro de las plataformas móviles, debido a su arquitectura y características.

Lo que realmente hace distinto a una plataforma móvil de otra, además de las características de cada una, es el software que esta usa, siendo el más importante y relevante el sistema operativo de cada una.

### iPhone y iPad

Apple es sin duda uno de los grandes ganadores en la batalla por el mercado de las plataformas móviles creando una gama de productos que sin duda ha conseguido una cuota de mercado envidiable.

Todos los dispositivos móviles de Apple funcionan con el sistema operativo iOS, el cual es propiedad de Apple y el cual incorpora numeroso software que ha hecho que los productos de Apple destaquen de los demás, como puede ser: *multitouch*, esto es, todo táctil, control de acelerómetros, etc.

Para desarrollar aplicaciones para estos dispositivos es necesario poseer un sistema operativo MacOs y conocimientos de Objetive-C.

## Android

Sistema operativo basado en el Kernel de Linux y creado y soportado por un consorcio de empresas lideradas por Google. Android es el principal competidor de iOSteniendo una cuota de mercado muy amplia debido a que son muchos los fabricantes (HTC, Samsung, LG, etc.) que usan este sistema operativo, consiguiendo un mayor número de usuarios y por lo tanto una mayor expansión.

Debido a que es software libre, es un sistema muy robusto y completo con multitud de aplicaciones para prácticamente cualquier cosa que se nos ocurra ya que tiene una comunidad de usuarios amplia, la cual proporciona ayuda y apoyo para cualquier tipo de problema.

Esto es una grandísima ventaja, ya que es un sistema donde si tienes que hacer algo, y no está hecho, puedes hacértelo tú, cosa que no era viable con iOS por ejemplo por sus limitaciones.

El entorno de desarrollo está basado en Java y se puede integrar con entornos de desarrollo como Eclipse o Netbeans lo que facilita mucho de cara al desarrollador su uso.

## Symbian

Sistema operativo para dispositivos móviles y empotrados creado por una alianza de empresas. Aunque actualmente están más en auge otros sistemas operativos para dispositivos móviles como iOS o Android, sigue siendo uno de los más usados y extendidos debido a la popular venta de teléfonos móviles de Nokia, donde es el principal sistema operativo de todas sus terminales.

## Blackberry

Dispositivos móviles considerados vertientes de los PDAs desarrollados por la empresa Research in Motion (RIM).

Estos dispositivos móviles usan un sistema operativo propio, conocido como BlackBerry Os (o BlackOs en el argot popular) el cual según ciertos expertos es bastante inestable, con respecto por ejemplo a los otros 2 grandes competidores directos: iOS y Android.

Así mismo una característica de estos dispositivos y que para muchos es una ventaja es su enorme teclado QWERTY, el cual puede llegar a ocupar la mitad del dispositivo móvil.

### Windows 7

Sucesor de Windows Mobile y adaptación del original para equipos de sobremesa que aunque aún no está muy extendido pretende serlo debido al reciente acuerdo entre Microsoft y Nokia.

Es una plataforma interesante debido a su posible extensión en un futuro y su potencia, lo que garantiza que se pueda realizar cualquier tipo de aplicación para él.

### WebOS

Sistema operativo de los antiguos Palm y actualmente adquirido por HP. Aunque es un sistema operativo versátil y con muchas posibilidades para el desarrollador, se descartó ya que a día de hoy hay poca variedad de productos que lo soportan además de haber sido su proyecto abandonado por Hewlett-Packard.

### MeeGo

Sistema operativo de Nokia para Smartphone y netbooks surgido de la unión entre Maemo (de Nokia) y Moblin (de Intel).

Aunque era un sistema operativo bastante prometedor, ya que las aplicaciones para el mismo estaban desarrolladas en C++ junto con Qt, lo que proporcionaba cierta reusabilidad entre distintas plataformas no llego a ver la luz más allá de la versión 1.1 para netbooks y 1.0 para dispositivos móviles.

Dicho sistema operativo está basado en Debian y es libre, lo que era un punto a favor de cara a ser uno de los candidatos. Pero debido a las pruebas iniciales que se realizaron con el mismo y los problemas que ya surgieron, junto con el futuro desconcertante hicieron que se eligiera otro sistema operativo más extendido y estable.

El proyecto MeeGo se abandonó después de la alianza entre Microsoft y Nokia para que todas las terminales usen Windows 7.

## I. 3.2. Tecnologías

En este capítulo se mencionarán las distintas tecnologías que se han estudiado y utilizado para poder desarrollar este sistema.

### Android

Aunque tal y como se ha comentado anteriormente existen alternativas interesantes. Nos decantamos por Android por varias razones:

- Software Libre
- Dispositivos con precios muy asequibles. Los más baratos suelen rondar los 100€.

- La pareja Dispositivo + Sistema Operativo Android puede hacer un sistema muy potente con muchas posibilidades tanto por las que otorga Android como por la tecnología implementada en el dispositivo móvil. Como puede ser pantallas capacitivas con multitouch, Wifi, Bluetooth, puertos USB, etc.
- Gran variedad de dispositivos móviles que usan Android: móviles, tablets, notebook, etc.
- Facilidad y comodidad para el desarrollador. Se usa Java como lenguaje de desarrollo además del completo SDK donde se incluyen una máquina virtual (Dalvik) con un emulador (AVD) para probar aplicaciones.
- Plataforma en auge. El número de dispositivos móviles con Android va en aumento debido a su gran versatilidad y a su implantación por varios fabricantes (al contrario que iOS, que únicamente es usado en dispositivos Apple).

## Java

Es el lenguaje predominante del sistema. Usado tanto en la herramienta Web como en la implementación de los servicios web, además de ser el lenguaje oficial sobre el que se soporta el desarrollo de aplicaciones en Android.

Debido a la extensión del mismo, contamos con innumerables bibliotecas así como con una extensa documentación para resolver cualquier tipo de problema. Lo que hace, además de la facilidad del lenguaje, poder solucionar cualquier contratiempo a lo largo del proyecto.

## XML

El estándar [7] más usado para la representación e intercambio de información es una de las bases de Android. Aunque su uso puede ser muy variable se usa en Android para representar las interfaces.

En el presente sistema se usará como lenguaje de definición para el intercambio de información entre el cliente y todos los servicios del servidor, además de ser el usado para representar los planos y proyectos generados por la herramienta de generación de grafos en planes de evacuación.

## zXing

zXing es una librería capaz de reconocer e interpretar distintos tipos de códigos y obtener la información que estos representan. En este proyecto se va a usar como procesador para los códigos QR. Gracias a esta librería podremos traducir dicho código en su equivalente a cadena de texto, dirección web o la información que represente [8].

## Neo4J

Neo4j es un software libre de Base de datos orientada a grafos [9], implementado en Java. Los desarrolladores describen a Neo4j como un motor de persistencia embebido, basado en disco, completamente transaccional Java que almacena datos estructurados en grafos más que en tablas [10].

En nuestro contexto, la base de datos principal donde se almacenarán todos los nodos y relaciones de cara a posibilitar la generación de evacuaciones será una base de datos orientada a grafos que usará este motor como principal componente para insertar, modificar y recuperar información.

## MySQL

Es un sistema de gestión de bases de datos gratuitos, ahora propiedad de Oracle, y que es uno de los más usados junto a Oracle. Se usará como SGBD tanto en los servidores de locales y globales para almacenar cualquier información contextual del sistema.

## Tecnología Web

En el proyecto el paradigma web es bastante relevante, o bien por la implementación de servicios web o bien por la herramienta web de generación de grafos para gestionar planes de evacuación.

Para esta última se ha usado:

- **JSP:** tecnología web basada en el lenguaje de programación Java que permite generar contenido web de una forma dinámica. Al ser Java el lenguaje principal de dicha tecnología, y al restar el resto del sistema implementado en Java se consigue una total interoperabilidad y cohesión entre los distintos elementos del sistema.
- **HTML**: Lenguaje de marcas predominante en el mundo web.
- **Javascript**: Lenguaje usado para gestionar y controlar los eventos de los usuarios, posibilitando poder guardar temporalmente estructuras de datos y conseguir potencia en la herramienta a la vez que un mayor abanico de funcionalidades.
- **CSS**: Lenguaje usado para definir estilos.
- **JQuery**: Librería Javascript que permite de una manera muy fácil y rápida poder trabajar directamente con el árbol de objetos DOM, gestionar eventos, añadir efectos. Su única función es facilitar la tarea al programador en tediosas tareas a bajo nivel [11].
- **AJAX**: Tecnología usado en nuestro caso para poder pasar información del cliente al servidor sin necesidad de recargar la página, de esta manera se puede generar la base de datos, generar imágenes de códigos qr, cargas planos, etc.

sin necesidad de recargar la página y de una manera totalmente transparente al usuario.

- **jsPlumb**: Librería Javascript usada para la generación de nodos y aristas. Se ha usado y adaptado dicha librería para usarla en nuestro contexto de emergencias [12].

## REST

REST es un estilo de arquitectura software diseñado por Roy Fielding en su tesis doctoral en el año 2000 [13]. La idea principal de REST es como conseguir implementar servicios web haciendo uso de la tecnología en la que se disponía en ese momento, sin necesidad de inventar una nueva tecnología. De esta manera el protocolo de intercambio de comunicación que se usa en REST es el HTTP haciendo uso de cuatro de los verbos del protocolo de HTTP: Crear (POST), Leer (GET), Actualizar (PUT) y Eliminar (DELETE).

En REST predomina el concepto de Recurso. Un recurso es un elemento de información al que se puede ser accedido directamente a través de un identificador único (URI) y sobre la que se aplican una serie de operaciones (las definidas anteriormente) para modificar dichos recursos. El objetivo de REST es definir recursos, modificarlos e intercambiar la representación de estos recursos en un formato concreto: XML, JSON, HTML, Texto plano, etc.

Por lo que con una API bien definida (conjunto de URIS (recursos) junto con algún dato adicional como el verbo a usar, parámetros en caso de ser necesarios) y haciendo uso de dichos verbos se puede hacer un sistema de intercambio de información basado en servicios que permita desde cualquier dispositivo actuar sobre estos recursos. Ya que solo necesitaría un pequeña aplicación cliente o entorno web que transforme esta representación en XML por ejemplo y la muestre con otra diseminación.

Los principales motivos por los que se ha optado por REST y no por RPC o SOAP son:

- El contexto en el que se centra el sistema, donde el uso de dispositivos móviles va a ser esencial, hace necesario que la cantidad de información entre dichos dispositivos y el servidor sea mínima, debido a la posible limitada capacidad de ancho de banda en los dispositivos móviles. Con SOAP esta cantidad de información se incrementa notablemente, al ser necesario especificar la descripción del servicio (WSDL) así como las cabeceras de intercambio de información, que hacen que incremente el tamaño de archivo considerablemente [14].
- Aunque en la práctica con REST y RPC se obtiene el mismo resultado, desde un punto de vista conceptual REST se adaptaba mejor a la arquitectura del sistema,

siendo así más fácilmente implementable o ampliable de cara al desarrollador o futuros desarrolladores.

El creciente uso de REST en Internet hace que sea una apuesta segura de cara a una futura orquestación o composición de servicios con otros organismos al ser un estilo de arquitectura muy extendido.

# Parte II: Proceso de Desarrollo

# II. 1

# Análisis y Modelado del Sistema

## II. 1.1. Características del sistema

De cara a cumplir con las expectativas del sistema, este debe cumplir una serie de características u objetivos.

El hecho de que el sistema tenga una característica o grupo de características garantiza cierta funcionalidad. Lo que al final del desarrollo garantiza que se cumplen las expectativas propuestas del sistema.

Las características que tendrá que tener nuestro sistema para cumplir con los objetivos a alcanzar son las siguientes:

O1 El sistema debe proporcionar al usuario un constante servicio en cuanto a respuesta ante situaciones de emergencia sin que el usuario sea consciente de ello, de forma automática.

O2 El sistema debe ser autogestionado, actualizando la distinta información que considere y detectando los nuevos usuarios así como las distintas peticiones.

O3 El usuario podrá obtener un plan de evacuación haciendo uso de un código QR y dentro del edificio u organización en el que se encuentre físicamente. Siempre y cuando esté dado de alta en el sistema.

O4 El sistema debe ser capaz de responder en tiempo real a las distintas peticiones de los clientes, y actuar con el menor tiempo de respuesta posible a las mismas.

O5 El sistema debe ser configurable para adaptarnos a las necesidades de los distintos y posibles escenarios

O6 El sistema debe ser capaz de compartir información con otros sistemas externos que estén correctamente autorizados.

O7 El sistema debe ser escalable, para que se puedan añadir en cualquier momento nuevos elementos (como servidores locales, planes de emergencia, conexión de nuevos usuarios, etc.). Estos nuevos elementos conectados no deben afectar al rendimiento del sistema.

O8 Al ser un sistema en tiempo real, debe ofrecer una respuesta o procesamiento de los datos en el menor tiempo posible y con una alta precisión. Es decir, garantizar la fiabilidad de los datos o lo que es lo mismo, proporcionar un plan de evacuación correcto.

O9 El sistema debe proporcionar una interfaz lo más sencilla posible.

O10 Debido a que algunos componentes del sistema serán usados en dispositivos móviles, se optimizará todo lo posible para que el gasto de batería sea mínimo.

O11 El sistema debe estar disponible en varios idiomas de cara a dar soporte a más usuarios.

O12 El sistema usará tecnologías de software libre en la medida de lo posible, para reducir el coste lo máximo posible.

## II. 1.2. Actores

En el sistema hay, hasta la fase de análisis y diseño actual, cuatro actores claramente identificables. Los cuales se pueden ver en las siguientes tablas.

| **Actor** | **Cliente** |
|---|---|
| **Descripción** | Será el usuario final del sistema y que recibirá los distintos planes de emergencia y notificaciones. |
| **Tipo** | Principal |
| **Comentarios** | Será el usuario estándar que usará el sistema para obtener los distintos servicios que este ofrece. Sería un huésped de un hotel, un usuario que va al cine o cualquier persona que entra en un edificio donde está instalado el sistema. |

| **Actor** | **Administrador local** |
|---|---|
| **Descripción** | Es el encargado de definir las notificaciones y planes de emergencia a nivel local. |
| **Tipo** | Principal |
| **Comentarios** | Por lo general habrá uno por edificio/institución, y será el encargado del mantenimiento del sistema además de definir su funcionalidad. |

| **Actor** | **Administrador Global** |
|---|---|
| **Descripción** | Es el administrador global del servidor global |
| **Tipo** | Principal |
| **Comentarios** | Su tarea principalmente es la de mantenimiento. Además de esto tiene privilegios sobre diversas opciones. Pero no deja de ser un actor secundario al no ser necesario (excepto para la instalación del sistema) para el funcionamiento del mismo (excepto para tares de mantenimiento, pero no de funcionalidad) |

| Actor | Personal externo autorizado |
|---|---|
| **Descripción** | Serán usuarios externos al sistema que en momentos puntuales participaran en el mismo. |
| **Tipo** | Principal |
| **Comentarios** | Debido a que el sistema está conectado con servicios de emergencias externos (como asistencia médica, departamento de bomberos, etc.), es conveniente que estos usuarios puedan acceder a cierta información únicamente dentro de un ámbito. Por ejemplo ver los distintos usuarios que han salido de un edificio donde se ha declarado una emergencia, para conocer a priori la situación al completo y actuar en consecuencia. O bien en caso médico poder consultar la información médica facilitada por el paciente para actuar correctamente. |

La distancia clara entre este tipo de actores tiene como razón principal la funcionalidad o el uso que cada uno hace del sistema así como el nivel de privilegios de cada uno.

Siendo el administrador global el que tiene el mayor número de privilegios: desde consultar datos de los usuarios hasta declarar emergencias. El administrador local es el segundo actor con mayor número de privilegios, al tener que gestionar todo lo relacionado con su entidad. Puede declarar emergencias, o controlar el log pero sin embargo no puede ver los datos personales de un usuario o su perfil sanitario. Ya que dicha información es privada y el actor usuario por lo general estará relacionado con una entidad durante un espacio de tiempo determinado.

Por último, los actores usuario y personal externo autorizado tienen el mismo número de privilegios pero repartidos entre las distintas funcionalidades. Mientras que el usuario puede declarar emergencias, el personal externo como tal y con su aplicación personalizada no podría pero si podría consultar el estado de cualquier usuario. Acción que no puede realizar un usuario.

# II. 1.3. Modelo de casos de uso

## II. 1.3.1. Modelado Conceptual

La Figura 4 muestra a nivel conceptual el modelo de casos de uso. En dicha figura se muestran los distintos actores del sistema así como el conjunto de funcionalidades o acciones que tiene cada uno de ellos organizado por temática o contexto.

Aunque en esta tesis únicamente se va a considerar el desarrollo de usuario o cliente, conviene dejar bien definido en el diseño cuales son las funcionalidades que

tendrá cada uno de los actores. Facilitando así en cierta manera la comprensión del sistema a futuros desarrolladores que continúen con este trabajo.

Cada uno de los casos de uso conceptuales está internamente compuesto por varios casos de uso, siendo dicho número casos de uso dinámico. Es decir, en esta tesis se ha optado por definir un cierto número de casos de uso así como su desarrollo. Pero en un futuro un caso conceptual podría representar más casos de uso de los aquí expuestos o bien incluso añadir nuevos casos de uso contextuales que se consideren necesarios por la propia evolución del sistema.

La relación entre actores y casos de uso se rige en principio por cuestiones de privacidad y privilegios. De esta manera el administrador global es el que tendría un mayor grado de privilegios, estando relacionado con todos los casos de uso conceptuales. Y siendo el usuario y personal externo de seguridad (bombero, policía, etc.) los que tienen menos privilegios. Ya que por cuestiones de privacidad así como seguridad para el sistema estos dos últimos actores solo deben hacer uso del sistema, y no influir en su configuración o composición.

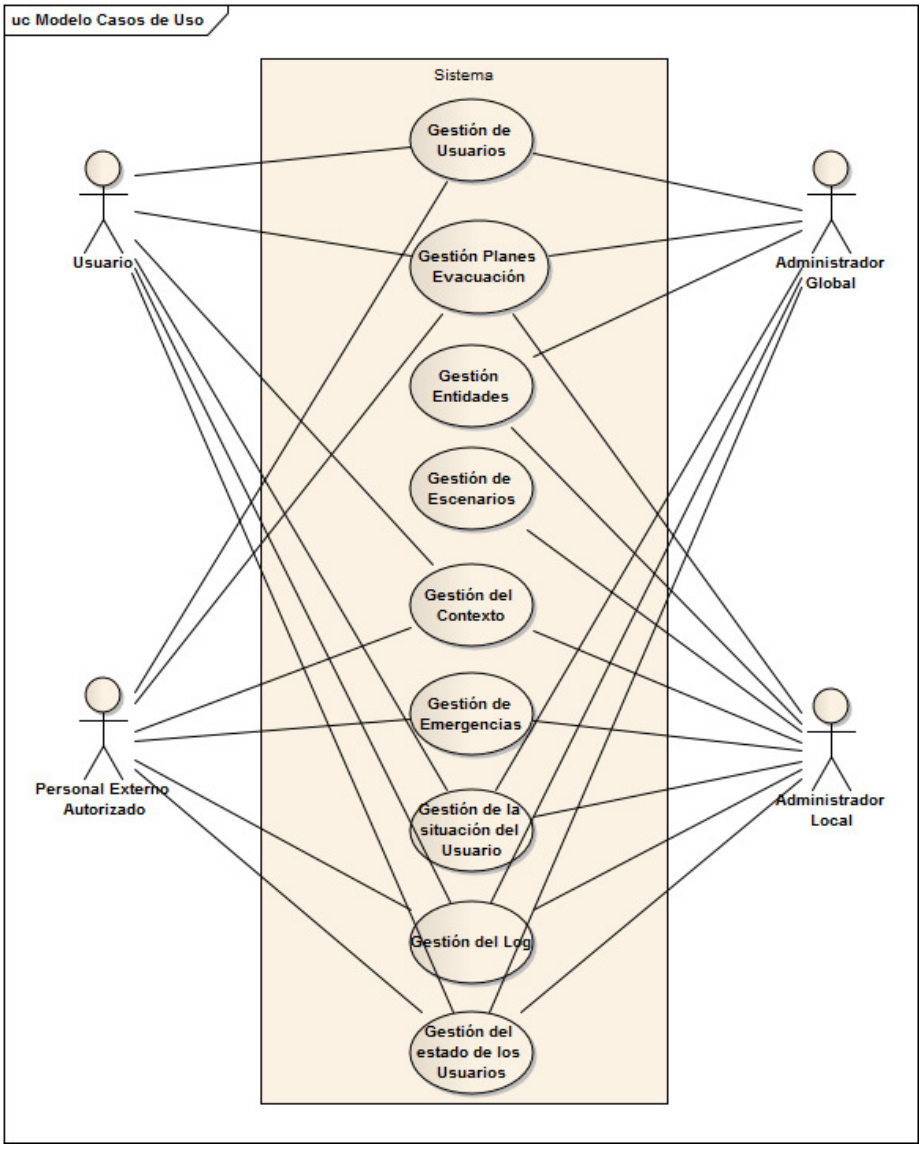


Figura 4 – Modelo Conceptual de Casos de Uso

## II. 1.3.2. Modelado de Casos de Uso

Los casos de uso especifican las distintas funcionalidades que ofrece el sistema, quién las realiza y para qué.

En este apartado se presentan los diagramas de casos de uso que tienen relación directa con los actores sin considerar al sistema.

### Caso de Uso del Actor Usuario

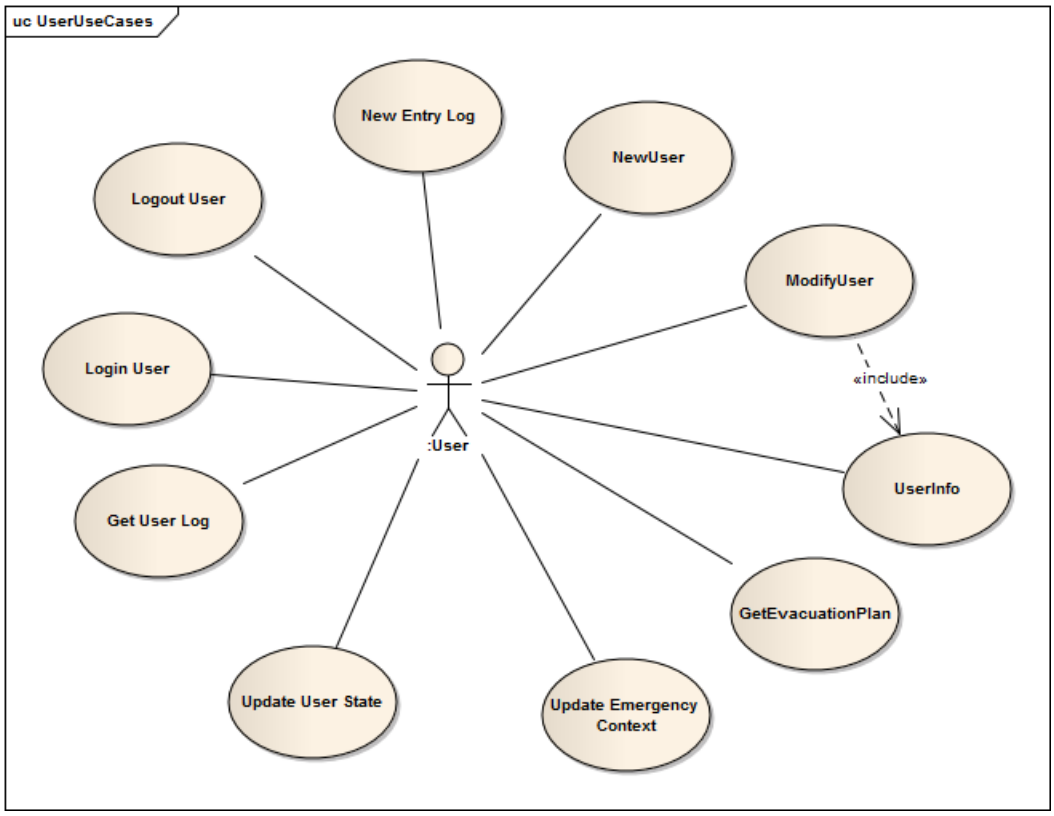



Figura 5 – Caso de Uso del Actor Usuario

## Caso de Uso del Actor Administrador Local

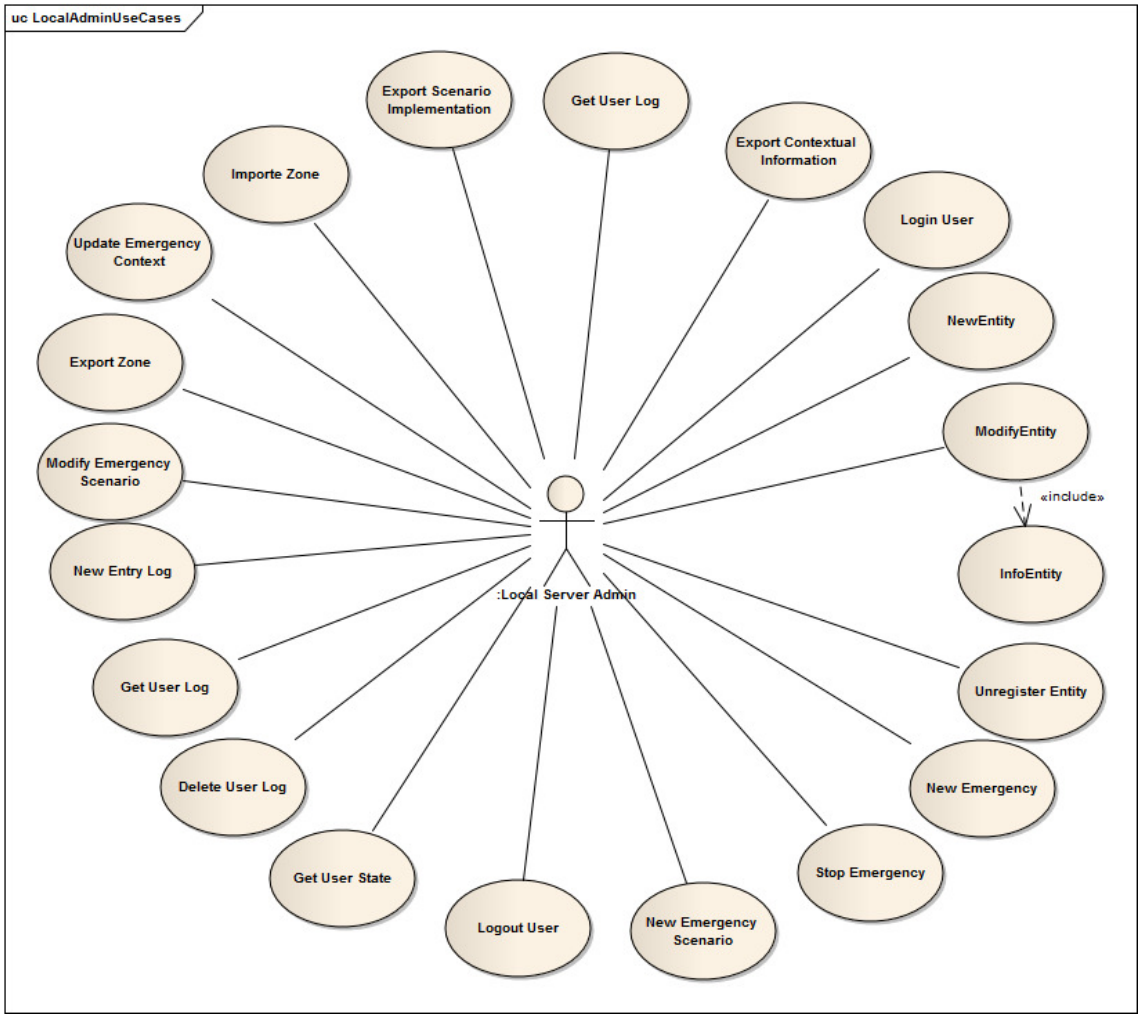



Figura 6 – Caso de Uso del Administrador Local

### Caso de Uso del Actor Administrador Global

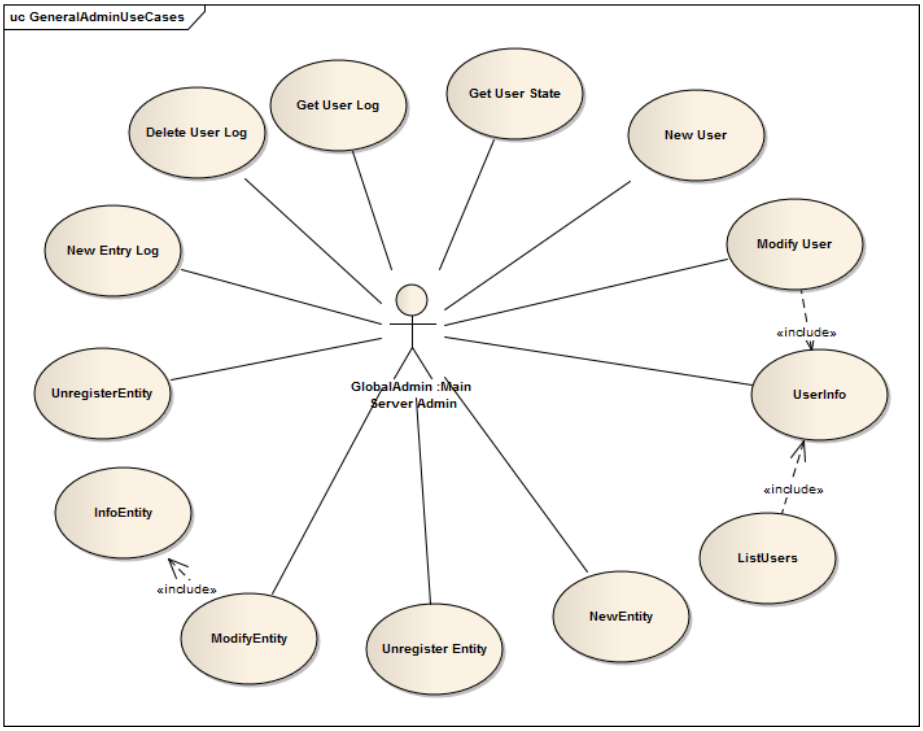



Figura 7 – Caso de Uso del Administrador Global

### Caso de Uso del Actor Personal del Equipo de Respuesta

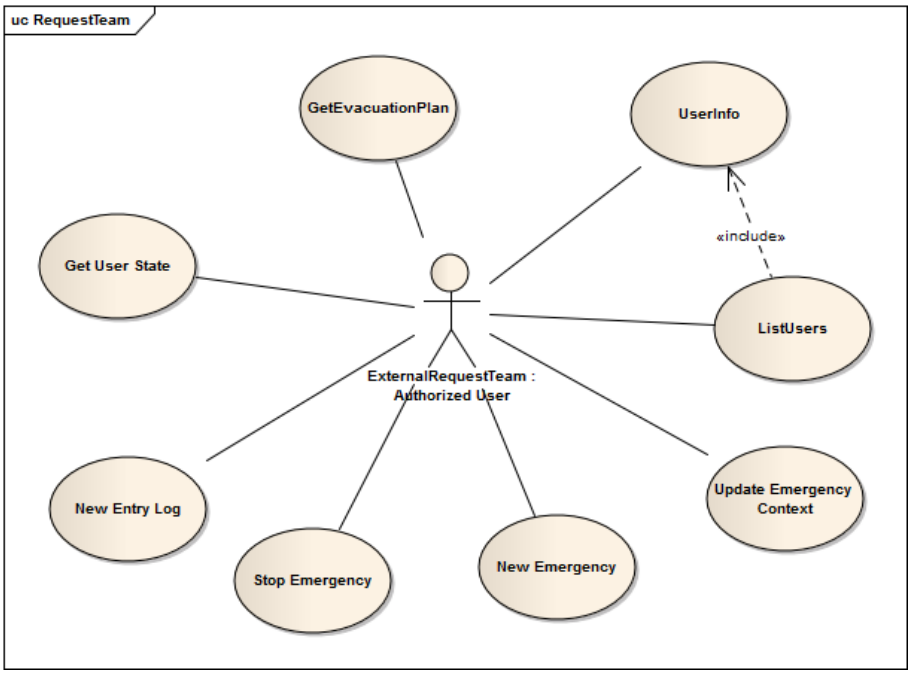



Figura 8 – Caso de Uso del Actor Personal del equipo de respuesta

# II. 1.4. Especificación de casos de Uso

**Gestión de Usuarios**

| **Nombre** | Nuevo Usuario |
|---|---|
| **Identificador** | 001 |
| **Actores** | Usuario, Administrador Global |
| **Resumen** | Será el propio usuario el encargado de rellenar sus datos personales en la aplicación. |
| **Dependencias** | |
| **Precondición** | Que dicho usuario no esté registrado. |
| **Postcondición** | El usuario quedará registrado. |
| **Curso Normal** | 1º El usuario iniciará la aplicación.<br>2º Dentro de la zona correspondiente, podrá crear su perfil. Rellenando una serie de campos obligatorios.<br>3º Una vez rellenados los campos y guardado se creará dicho usuario. |
| **Curso alternativo** | 3º Si hay algún problema con alguno de los datos o bien con el registro del usuario, se notificará con un mensaje de error. |
| **Comentarios** | La opción de dar de alta un nuevo usuario se puede considerar global, ya que queda registrado en el servidor global. |

| **Nombre** | Modificar usuario |
|---|---|
| **Identificador** | 002 |
| **Actores** | Usuario Administrador Global |
| **Resumen** | Será el propio usuario el encargado de modificar los datos que considere. |
| **Dependencias** | <<Include>> a 003 |
| **Precondición** | El usuario debe estar previamente registrado. |
| **Postcondición** | Los nuevos datos del usuario quedarán actualizados. |
| **Curso Normal** | 1º El usuario iniciará la aplicación.<br>2º Dentro de la zona correspondiente, podrá modificar su perfil o datos personales.<br>3º Una vez modificados los campos se actualizará dicho usuario. |
| **Curso alternativo** | 3º Si hay algún problema con alguno de los datos o bien con la actualización del usuario, se notificará con un mensaje de error. |
| **Comentarios** | La opción de dar de modificar un usuario se puede considerar global, que quede registrado en el servidor global. |

| **Nombre** | Ver datos de Usuario |
|---|---|
| **Identificador** | 003 |
| **Actores** | Usuario, Administrador Global, , Personal externo autorizado |
| **Resumen** | Se podrán ver los datos personales de un usuario que este previamente registrado en el sistema y solo por parte de personal autorizado. |
| **Dependencias** | |
| **Precondición** | El usuario debe estar previamente registrado y el actor autorizado |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán su correspondiente herramienta de gestión.<br>2º Dentro de la zona correspondiente, cada actor podrá ver los datos personales de un usuario en concreto. |
| **Curso alternativo** | 2º Si hay algún problema con la función de recuperar y mostrar información se notificará con un mensaje de error. |
| **Comentarios** | No todos los actores antes mencionados podrán ver en cualquier momento del tiempo los datos personales de un usuario. Excepto el propio Usuario y el Administrador General. El resto de actores deben recibir algún tipo de autorización que por lo general será concedida en momentos de necesidad. |

| **Nombre** | Listar usuarios |
|---|---|
| **Identificador** | 004 |
| **Actores** | Administrador Global, Personal Externo Autorizado |
| **Resumen** | Se podrá ver una lista de usuarios en función de una serie de parámetros o bien dentro de un contexto. |
| **Dependencias** | <<Include>> a 003 |
| **Precondición** | Debe haber usuarios dados de alta en el sistema. |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán su correspondiente herramienta de gestión.<br>2º Dentro de la zona correspondiente, cada actor podrá mostrar los distintos usuarios que hay en el sistema. Mostrándose todos o solamente aquellos que se ajusten a unos parámetros de búsqueda. |
| **Curso alternativo** | 2º Si hay algún problema con la función de recuperar y mostrar información se notificará con un mensaje de error. |
| **Comentarios** | El administrador general tendrá poder para ver cualquier usuario |

| | |
|---|---|
| | que haya formado parte de su sistema. Sin embargo el Administrador local solo podrá ver aquellos usuarios que formen parte de su sistema en ese momento. De igual modo que el personal externo autorizado, que solo podrá ver aquellos usuarios a los que se les de autorización. |

**Gestión de Planes de Evacuación**

| **Nombre** | Obtener plan de evacuación |
|---|---|
| **Identificador** | 005 |
| **Actores** | Usuario, Personal Externo Autorizado |
| **Resumen** | El propio usuario podrá solicitar voluntariamente un plan de evacuación con el fin de conocer el proceso de evacuación. Así mismo dicho proceso se usará por el sistema automáticamente cuando se declaré una emergencia. |
| **Dependencias** | |
| **Precondición** | Además de haber iniciado la aplicación y la opción correspondiente, que haya definido un plan de emergencia por parte de la entidad. |
| **Postcondición** | |
| **Curso Normal** | 1º El usuario se situará en el mismo emplazamiento que un QR code.<br>2º Usando dicha aplicación y enfocándola hacía el QR code obtendrá dicho código.<br>3º Dicho código será enviado al servidor el cual responderá con el plan de evacuación concreto en función de su posición. |
| **Curso alternativo** | 2º Si no se consigue capturar bien el QR code se le notificará al usuario con un mensaje.<br>3º Si hay algún problema con el envío y transmisión de los datos se notificará con un mensaje. |
| **Comentarios** | |

**Gestión de Entidades**

| **Nombre** | Dar de alta una nueva entidad |
|---|---|
| **Identificador** | 006 |
| **Actores** | Administrador Local, Administrador Global |
| **Resumen** | Se podrá dar de alta una nueva entidad que desee formar parte del sistema. |
| **Dependencias** | |

| | |
|---|---|
| **Precondición** | Tener instaladas las correspondientes herramientas. |
| **Postcondición** | La nueva entidad quedará registrada en el sistema global. |
| **Curso Normal** | 1º Los actores iniciarán su correspondiente herramienta.<br>2º Dentro de la zona correspondiente, el actor podrá rellenar unos simples cuestionarios para dar de alta dicha entidad.<br>3º Una vez rellenados dichos formularios se enviará la petición al servidor general. |
| **Curso alternativo** | 3º Si hay algún problema a la hora de recibir o enviar la solicitud o bien algún problema en el proceso de registro se le notificará al actor. |
| **Comentarios** | El concepto de dar de alta una nueva entidad es útil de cara a conocer información sobre un edifico o zona donde está la propia entidad. De esta manera y por ejemplo para los servicios de actuación externos es información útil a la hora de actuar. |

| | |
|---|---|
| **Nombre** | Modificar los datos de una entidad |
| **Identificador** | 007 |
| **Actores** | Administrador Local, Administrador Global |
| **Resumen** | Se podrá modificar los datos relacionados a una entidad. |
| **Dependencias** | <<Include>> a 009 |
| **Precondición** | Que dicha entidad esté registrada previamente. |
| **Postcondición** | La entidad quedará actualizada en el sistema global y local. |
| **Curso Normal** | 1º Los actores iniciarán su correspondiente herramienta.<br>2º Dentro de la zona correspondiente, el actor podrá ver la información referente a la entidad y modificar los campos que desee.<br>3º Una vez modificados dichos campos se actualizará tanto en el servidor local como en el servidor general. |
| **Curso alternativo** | 3º Si hay algún problema con el proceso de actualización se notificará con un mensaje de error. |
| **Comentarios** | El único que puede modificar los datos de una entidad en modo local es el administrador local. No obstante se posibilita al administrador general poder modificar dicha entidad en el servidor general. |

| **Nombre** | Dar de baja una entidad |
|---|---|
| **Identificador** | 008 |
| **Actores** | Administrador Local, Administrador Global |
| **Resumen** | Se podrá dar de baja a una entidad del sistema. |
| **Dependencias** | |
| **Precondición** | Que dicha entidad esté registrada previamente. |
| **Postcondición** | La entidad quedará dada de baja. |
| **Curso Normal** | 1º Los actores iniciarán su correspondiente herramienta.<br>2º Dentro de la zona correspondiente, el actor podrá dar de baja la entidad. |
| **Curso alternativo** | 2º Si hay algún problema con el proceso de actualización se notificará con un mensaje de error. |
| **Comentarios** | Al darse de baja una entidad sencillamente se hace a nivel local. En el servidor general siguen permaneciendo los distintos datos de cara a ser reusables en un futuro. |

| **Nombre** | Ver información de una entidad |
|---|---|
| **Identificador** | 009 |
| **Actores** | Administrador Local, Administrador Global |
| **Resumen** | Se podrá consultar la información referente a una entidad. |
| **Dependencias** | |
| **Precondición** | Que dicha entidad esté registrada previamente. |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán su correspondiente herramienta.<br>2º Dentro de la zona correspondiente, el actor podrá ver la información referente a la entidad. |
| **Curso alternativo** | 2º Si hay algún problema con el proceso de recuperación y visionado de la información se notificará con un error. |
| **Comentarios** | |

**Gestión de Escenarios**

| **Nombre** | Crear un proyecto de escenario de emergencia |
|---|---|
| **Identificador** | 010 |
| **Actores** | Administrador Local |
| **Resumen** | Se podrán diseñar y generar escenarios asociados a edificios donde se quieren automatizar la generación de planes de evacuación |
| **Dependencias** | |
| **Precondición** | |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta de generación de escenarios de emergencia.<br>2º Dentro de esta y haciendo uso de los elementos de la herramienta se podrá diseñar el escenario general.<br>3º Una vez terminado el diseño, y haciendo uso de la opción correspondiente el actor podrá exportar un modelo de dicho diseño. |
| **Curso alternativo** | 3º Si hay algún problema con el proceso de generación se informará mediante un error. |
| **Comentarios** | |

| **Nombre** | Modificar un proyecto de escenario de emergencia |
|---|---|
| **Identificador** | 011 |
| **Actores** | Administrador Local |
| **Resumen** | Se podrán modificar escenarios asociados a edificios donde se quieren automatizar la generación de planes de evacuación |
| **Dependencias** | |
| **Precondición** | Que dicho proyecto este previamente creado |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta de generación de escenarios de emergencia.<br>2º Dentro de esta cargaran el proyecto previamente creado.<br>3º Una vez cargado este proyecto retocarán el diseño y modificarán lo que deseen.<br>3º Una vez terminado el diseño, y haciendo uso de la opción correspondiente el actor podrá exportar el modelo de dicho diseño ya modificado. |
| **Curso alternativo** | 2º Si hay un problema con el proceso de carga del proyecto se |

| | |
|---|---|
| | notificará mediante un mensaje de error.<br>3º Si hay algún problema con el proceso de generación se informará mediante un error. |
| **Comentarios** | |

| **Nombre** | Exportar zonas de un diseño |
|---|---|
| **Identificador** | 012 |
| **Actores** | Administrador Local |
| **Resumen** | Se podrá exportar una zona o zonas de un diseño en concreto, con el fin de facilitar la composición de escenarios. |
| **Dependencias** | |
| **Precondición** | Que dicho diseño este previamente realizado |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta de generación de escenarios de emergencia.<br>2º Dentro de esta y haciendo uso de los elementos de la herramienta se exportará la zona seleccionada. |
| **Curso alternativo** | 2º Si hay algún problema con el proceso de generación se informará mediante un error. |
| **Comentarios** | |

| **Nombre** | Importar zonas de un diseño |
|---|---|
| **Identificador** | 013 |
| **Actores** | Administrador Local |
| **Resumen** | Se podrán cargar en un diseño nuevas zonas previamente definidas |
| **Dependencias** | |
| **Precondición** | Que dicho zona esté previamente creada y exportada |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta de generación de escenarios de emergencia.<br>2º Dentro de esta y haciendo uso de los elementos de la herramienta se cargará la zona correspondiente. Acoplándose al diseño pero definiéndose las funcionalidades por el usuario. |
| **Curso alternativo** | 2º Si hay algún problema con el proceso de carga se informará mediante un error. |
| **Comentarios** | |

| **Nombre** | Exportar una implementación final de un escenario |
|---|---|
| **Identificador** | 014 |
| **Actores** | Administrador Local |
| **Resumen** | Se generará una implementación final en base a un diseño/proyecto para poder usarse en la AOS. |
| **Dependencias** | |
| **Precondición** | Que dicho diseño esté creado o bien cargar algún proyecto o zona. |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta de generación de escenarios de emergencia.<br>2º Dentro de esta y haciendo uso de los elementos de la herramienta podrán:<br>2.1. Cargar un proyecto o zona previamente creada<br>2.2. Diseñar desde el principio un escenario nuevo<br>3º Una vez realizada algunas de las tareas anteriores la herramienta generará una implementación del modelo. |
| **Curso alternativo** | 3º Si hay algún problema con el proceso de generación se informará mediante un error. |
| **Comentarios** | |

| **Nombre** | Exportar información contextual del escenario |
|---|---|
| **Identificador** | 015 |
| **Actores** | Administrador Local |
| **Resumen** | Se permitirá exportar información contextual del escenario. En este caso las imágenes asociadas a los códigos QR usados en el diseño |
| **Dependencias** | |
| **Precondición** | Que dicho diseño esté creado o bien cargar algún proyecto o zona. |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta de generación de escenarios de emergencia.<br>2º Dentro de esta y haciendo uso de los elementos de la herramienta podrán:<br>2.1. Cargar un proyecto o zona previamente creada<br>2.2. Diseñar desde el principio un escenario nuevo<br>3º Una vez realizada algunas de las tareas anteriores la |

| | |
|---|---|
| | herramienta generará dicha información contextual. |
| **Curso alternativo** | 3º Si hay algún problema con el proceso de generación se informará mediante un error. |
| **Comentarios** | |

**Gestión del Contexto**

| **Nombre** | Actualizar el contexto de una emergencia |
|---|---|
| **Identificador** | 016 |
| **Actores** | Usuario, Administrador Local, Personal Externo de Seguridad |
| **Resumen** | En una situación de emergencia el contexto de esta cambia continuamente. Desde un punto de vista computacional este debe permitir actualizarse para generar así planes de evacuación o datos de acuerdo a este cambio contextual. |
| **Dependencias** | |
| **Precondición** | Que se haya declarado una emergencia |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta o aplicación correspondiente.<br>2º En el apartado correspondiente el actor puede indicar alguna información contextual que él está percibiendo y notificársela al sistema. |
| **Curso alternativo** | 2º Si hay algún error a la hora de notificársela al sistema se mostrar un mensaje de error |
| **Comentarios** | |

**Gestión de Emergencias**

| **Nombre** | Notificar de una nueva emergencia |
|---|---|
| **Identificador** | 017 |
| **Actores** | Usuario, Administrador Local, Personal Externo de Seguridad |
| **Resumen** | Cuando algún actor descubra o encuentre alguna situación que pueda suponer una emergencia para el resto de usuarios, podrá notificar al sistema una situación de emergencia. |
| **Dependencias** | |
| **Precondición** | |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta o aplicación correspondiente.<br>2º En el apartado correspondiente el actor puede indicar que quiere declarar una emergencia así como información relevante a esta (donde se encuentra, gravedad de la emergencia, una descripción, etc.). |
| **Curso alternativo** | 2º Si hay algún error a la hora de notificársela al sistema se mostrar un mensaje de error |
| **Comentarios** | |

| **Nombre** | Suspender una emergencia |
|---|---|
| **Identificador** | 018 |
| **Actores** | Administrador Local, Administrador Global, Personal Externo de Seguridad |
| **Resumen** | Se podrá suspender una situación de emergencia una vez que el personal externo de seguridad notifique que no hay peligro o bien que algún administrador observe por los datos la situación general. |
| **Dependencias** | |
| **Precondición** | |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta o aplicación correspondiente.<br>2º En el apartado correspondiente el actor puede indicar que quiere declarar el fin de una emergencia. |
| **Curso alternativo** | 2º Si hay algún error a la hora de notificársela al sistema se mostrar un mensaje de error |
| **Comentarios** | |

**Gestión de la situación del Usuario**

| **Nombre** | Notificar a un usuario como parte de un sistema(entidad) |
|---|---|
| **Identificador** | 019 |
| **Actores** | Usuario, Administrador Local |
| **Resumen** | Cuando un usuario se encuentre en algún emplazamiento que de soporte a la gestión de emergencias, debe notificarse para indicar que quiere formar parte de ese sistema. |
| **Dependencias** | |
| **Precondición** | |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta o aplicación correspondiente.<br>2º En el apartado correspondiente el actor:<br>2.1. Si es el usuario se dará de alta a través de la aplicación<br>2.2. Si es otro actor, podrá dar de alta a un usuario a través de su identificador. |
| **Curso alternativo** | 2º Si hay algún error a la hora de notificársela al sistema se mostrar un mensaje de error |
| **Comentarios** | |

| **Nombre** | Notificar que un usuario deja de formar parte de un sistema (entidad) |
|---|---|
| **Identificador** | 020 |
| **Actores** | Usuario, Administrador Local |
| **Resumen** | Una vez que el usuario deje la zona que soporta gestión de emergencias debe desvincularse de la misma para que así el sistema no le notifique en un futuro. |
| **Dependencias** | |
| **Precondición** | |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores iniciarán la herramienta o aplicación correspondiente.<br>2º En el apartado correspondiente el actor:<br>2.1. Si es el usuario se dará de baja a través de la aplicación<br>2.2. Si es otro actor, podrá dar de baja a un usuario a través de su identificador. |
| **Curso alternativo** | 2º Si hay algún error a la hora de notificársela al sistema se |

| | |
|---|---|
| | mostrar un mensaje de error |
| **Comentarios** | |

**Gestión del Log**

| **Nombre** | Creación de un nuevo registro |
|---|---|
| **Identificador** | 021 |
| **Actores** | Usuario, Administrador Local, Administrador Global, Personal Externo de Seguridad |
| **Resumen** | El sistema debe almacenar todas las acciones que se consideren relevantes por parte de todos los usuarios. |
| **Dependencias** | |
| **Precondición** | |
| **Postcondición** | |
| **Curso Normal** | 1º Cualquier actor que use el sistema y al realizar una acción definida como relevante: solicitar un plan de evacuación, modificar los datos de un usuario, etc. Creará un registro en el servidor local (o global en caso de error o a modo de backup) en el que se encuentre. |
| **Curso alternativo** | 1º Si hay problemas para guardar dicho registro se notificará con un mensaje de error. |
| **Comentarios** | |

| **Nombre** | Eliminar los registros de un usuario |
|---|---|
| **Identificador** | 022 |
| **Actores** | Administrador Local, Administrador Global |
| **Resumen** | Debido a cuestiones de privacidad, un usuario determinado puede desear que todas sus acciones en un sistema en concreto sean borradas. |
| **Dependencias** | |
| **Precondición** | Que dicho usuario tenga algún registro en el sistema |
| **Postcondición** | |
| **Curso Normal** | 1º Cualquier actor de los autorizados abre la herramienta autorizada.<br>2º Seleccionará el usuario sobre el que desea realizar acciones<br>3º Una vez seleccionado el usuario podrá seleccionar la opción de eliminar todo su log para un escenario/servidor local/entidad en concreto. |

| | |
|---|---|
| **Curso alternativo** | 2º Si no existe dicho usuario se notificará con un mensaje<br>3º Si dicho usuario no tiene entradas en el registro o bien hay algún problema se notificará correspondientemente. |
| **Comentarios** | |

| | |
|---|---|
| **Nombre** | Obtener el log de un usuario |
| **Identificador** | 023 |
| **Actores** | Administrador Local, Administrador Global, Personal Externo de Seguridad |
| **Resumen** | Los actores autorizados podrán ver todo el log de un usuario en concreto. |
| **Dependencias** | |
| **Precondición** | Que dicho usuario tenga entradas en el registro. |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores accederán a la herramienta correspondiente.<br>2º Buscarán dicho usuario por su identificador o cualquier otro campo.<br>3º Una vez que lo hayan encontrado y seleccionado tendrán opción de ver todo el log de este usuario. |
| **Curso alternativo** | 2º Si dicho usuario no existe se notificará con un mensaje.<br>3º Si hay algún problema con la recuperación de los datos se notificará con un mensaje de error. |
| **Comentarios** | |

**Gestión del Estado de los Usuarios**

| | |
|---|---|
| **Nombre** | Actualizar el estado de un usuario |
| **Identificador** | 024 |
| **Actores** | Usuario |
| **Resumen** | Los actores autorizados podrán definir el estado de un usuario. Si está a salvo o bien necesita ayuda. |
| **Dependencias** | |
| **Precondición** | Que se haya declarado una emergencia |
| **Postcondición** | |
| **Curso Normal** | 1º El actor, en una situación de emergencia, accederá a la parte correspondiente para notificar su estado.<br>2º Podrá indicar si está a salvo o bien si necesita algún tipo de ayuda. |

| | |
|---|---|
| **Curso alternativo** | 2º Si hay algún problema a la hora de registrar en el servidor el nuevo estado del usuario se notificará correspondientemente. |
| **Comentarios** | |

| | |
|---|---|
| **Nombre** | Obtener el estado de los usuarios |
| **Identificador** | 025 |
| **Actores** | Administrador local, Administrador global, Personal Externo de Seguridad |
| **Resumen** | Los actores podrán obtener o bien el estado de un usuario o bien el estado de todos los usuarios dentro de una situación de emergencia. |
| **Dependencias** | |
| **Precondición** | Que se haya declarado una emergencia |
| **Postcondición** | |
| **Curso Normal** | 1º Los actores, en una situación de emergencia, accederá a la parte correspondiente de la herramienta.<br>2º Podrán o bien buscar un usuario y ver su estado o bien listar todos los usuarios junto con su estado que hay en esa situación de emergencia. |
| **Curso alternativo** | 2º Si hay algún problema a la hora de obtener los datos se notificará correctamente. |
| **Comentarios** | |

## II. 1.5. Modelo Conceptual

El diagrama de clases conceptual que se muestra en la Figura 9 refleja la organización e información de todo el sistema.

En este diagrama cada una de las clases represente conceptualmente una o varias clases correspondientes en el diagrama de clases obtenido en la fase de diseño. El cual se encuentra y se puede ver en la tesis complementaria a esta.

En el diagrama de clases de diseño se muestran los atributos de cada clase, una especificación más completa así como las relaciones y cardinalidades entre las distintas clases.

En el Anexo III se puede ver una descripción de cada una de las clases conceptuales, ayudando así a entender mejor no solo dicho diagrama si no también la organización del sistema.

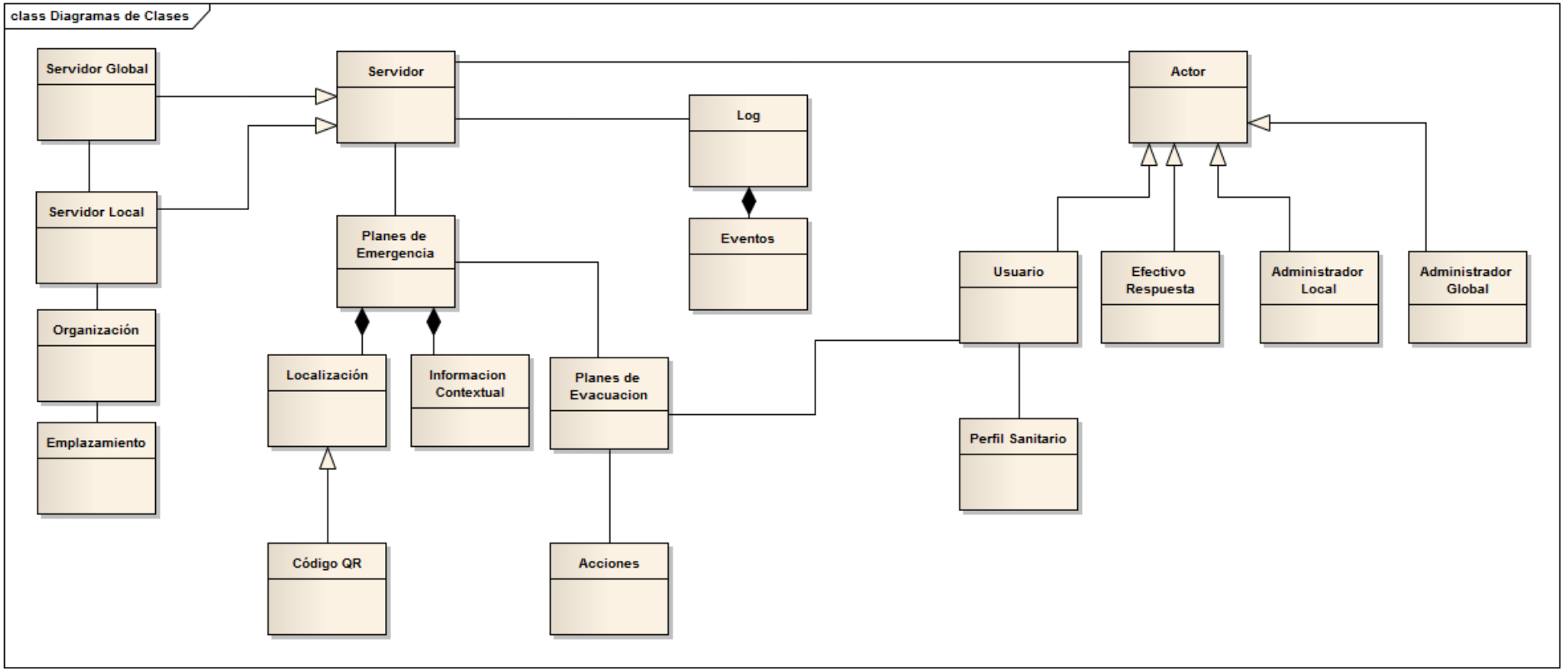



Figura 9 – Diagrama de Clases Conceptual

## II. 1.6. Diagramas de Secuencia

En el diagrama que se puede ver a continuación, Figura 10, se muestra la secuencia de eventos que van sucediendo y que elementos intervienen en esta cuando un usuario o cliente se da de alta en una entidad y solicita un plan de evacuación como entrenamiento para posibles situaciones de emergencia.

Como se puede apreciar el proceso de registro es un proceso típico y rutinario pero el proceso de obtener un plan de evacuación tiene dos posibles vertientes.

Dichas vertientes derivan del algoritmo que se use para generar el plan de evacuación correctamente.

Cada algoritmo trabaja exclusivamente con un tipo de base de datos, por la naturaleza del mismo. Mientras el algoritmo basado en optimizaciones de colonias de hormigas trabaja con una base de datos orientada a grafos (generada con la herramienta web y basada en el motor Neo4J [9]); el algoritmo de Floyd-Warshall accede a la base de datos relacional y consulta la tabla de caminos para obtener el camino del nodo de origen al nodo de destino. En ambas vertientes el resultado debe ser el mismo.

En última instancia y una vez que el usuario ha recibido el plan de evacuación descrito como en los modelos de datos del capítulo de implementación, usara su Asistente Personal de Seguridad para interpretarlo y poder actuar correctamente según las indicaciones.

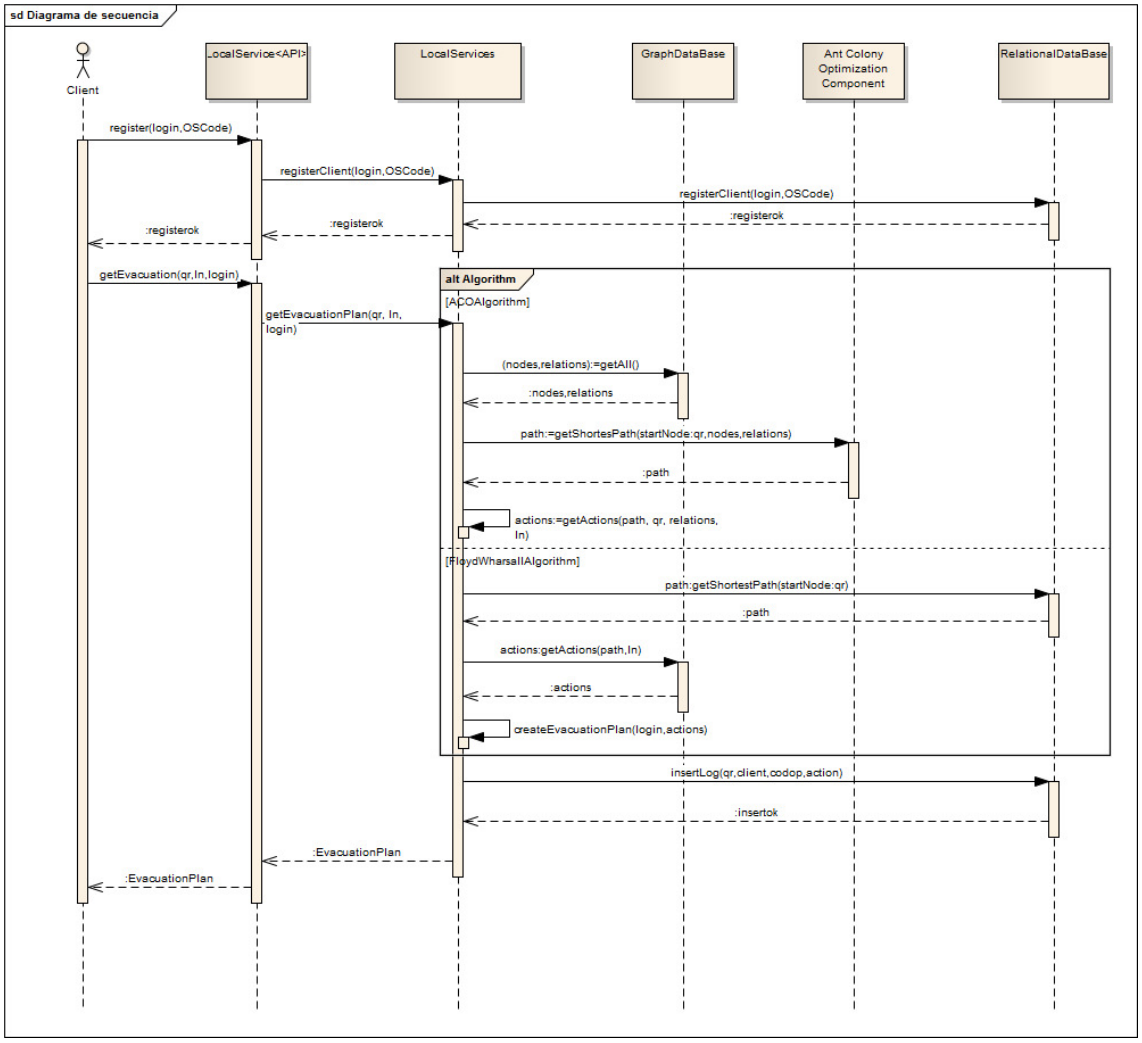



Figura 10 – Diagrama de Secuencia de una petición de evacuación

La Figura 11, mostrada a continuación, representa todo el conjunto de eventos que se producen desde que un usuario decreta una emergencia hasta que reciben la notificación todos los usuarios en el sistema. En este caso el mismo cliente que la notifica y otro llamado Actor 1.

La primera acción que realizan es registrarse o darse de alta como usuarios del servidor que representa una organización u organización. Para ello facilitan su login (número de teléfono) y un código o token necesario para poder recibir notificaciones push.

Una vez que ambos actores están dados de alta en el servidor local, el Actor1 notifica que hay una emergencia. En caso de que se opte por usar el algoritmo Floyd-Warshall el sistema generará las tablas de caminos y distancias para usar cuando se necesite generar un plan de evacuación.

Después de esto se obtienen los token de todos los usuarios datos de alta y se les envía una notificación push haciendo uso de los servidores de Google (C2DM) [15] o de Apple (APNS) [16] dependiendo del dispositivo móvil.

Esta notificación push es un texto XML, que se puede ver en el anexo II, y que representa una estructura y que interpretará al Asistente Personal de Seguridad (PSA).

Cuando el asistente detecte que se trata de una emergencia, hará uso de los servicios web y pedirá su plan de evacuación pasándole un código qr (capturado a través de su cámara del dispositivo móvil) obteniendo así un texto XML a interpretar por el PSA.

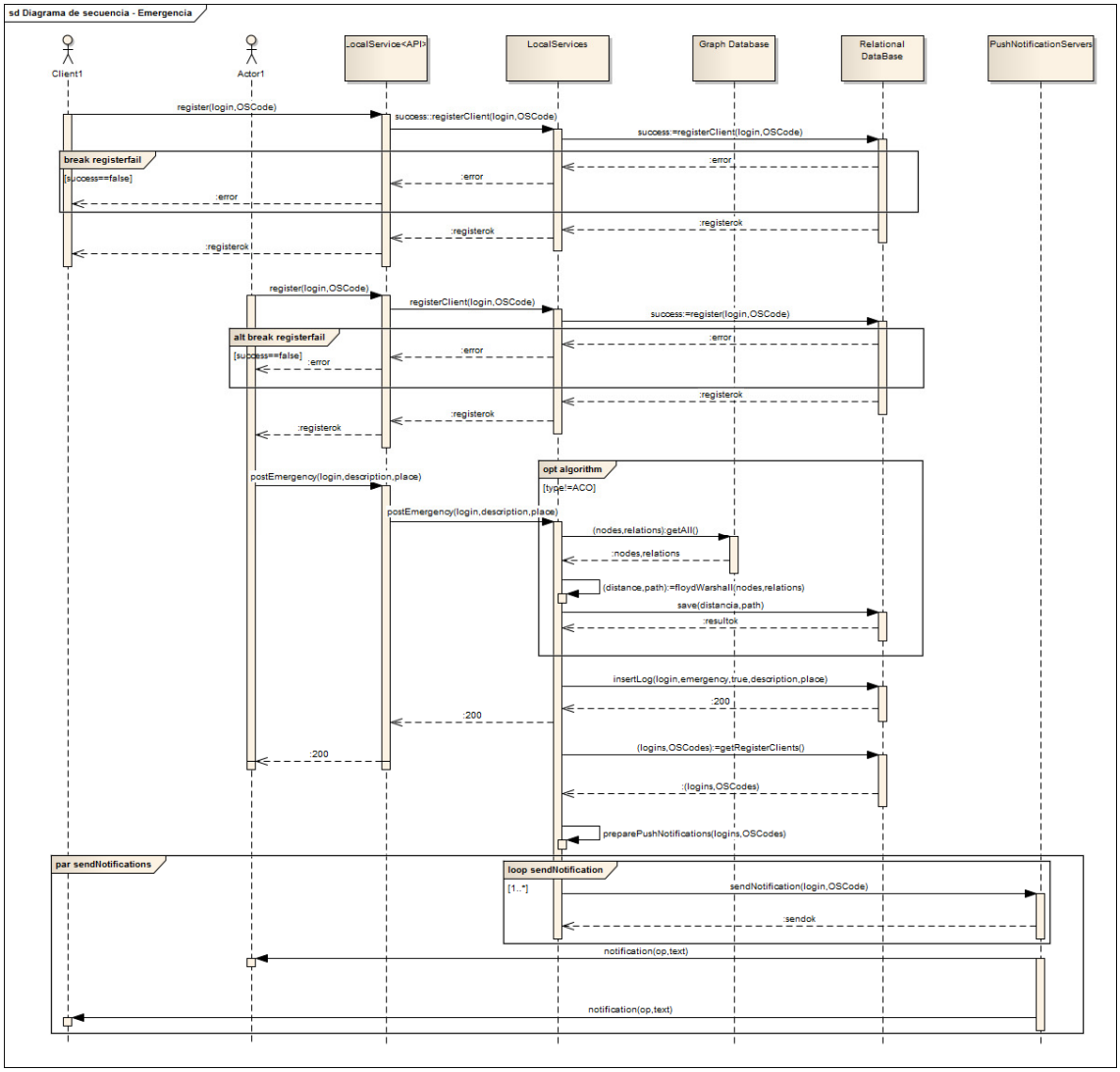



Figura 11 – Diagrama de secuencia de notificación de una emergencia

# II. 1.7. Modelo Conceptual del Grafo de una Emergencia

La Figura 12 muestra el modelo conceptual de la representación de planes de evacuación a través de una estructura de grafo. Una de las posibles implementaciones concretas de este modelo se puede ver en el modelado de los datos en el apartado de diseño. Donde como se verá un plan de evacuación corresponderá únicamente a un conjunto de acciones.

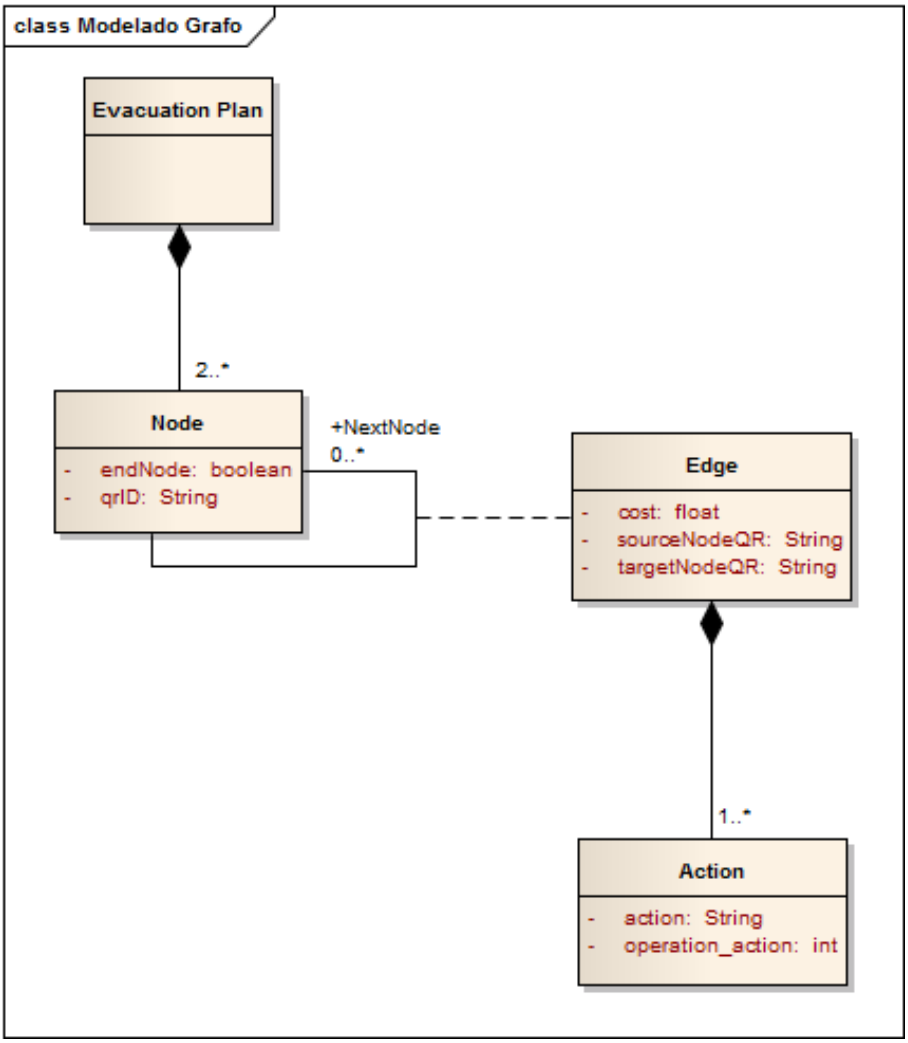



Figura 12 - Modelo conceptual de un Grafo de Emergencia

A cada acción se le asigna un código de operación que simboliza un código perteneciente a una de las acciones básicas predefinidas: girar a la izquierda, seguir recto, girar a la derecha, etc. De esta manera esta información podrá ser procesada por el cliente y usarla para facilitar al usuario final mediante imágenes, texto adicional, videos, etc. la correcta evacuación del sitio en emergencia.

# II. 2

# Diseño

## II. 2.1. Arquitectura General

La Figura 13 muestra la arquitectura general por bloques de todo el sistema. Como se puede apreciar existen 4 bloques generales:

- **Servidores Locales:** Aunque en el diagrama se represente únicamente uno, a modo de ejemplo ilustrativo. El sistema general puede estar formado por un número indeterminado de servidores locales asociados a entidades. Todos con los mismos componentes pero cada uno con un estado concreto.
- **Servidor Global:** Únicamente hay uno, y es el encargado de proporcionar servicios a los distintos tipos de usuarios en función de su rol. Además de servir de respaldo en caso fallo técnico en alguno o algunos de los servidores locales.
- **Usuarios:** Es un único bloque, representando a los distintos actores que hacen uso de los servicios y elementos del sistema. En el diagrama dicho bloque se ha descompuesto en los 4 tipos de actores que hay: Administrador local, Administrador Global, Usuario y Personal del Equipo de Respuesta o Personal Externo Autorizado.
- **Servidores de Servicios para las notificaciones:** Aunque no forman parte integra l del sistema, son vítales para garantizar el principal objetivo del mismo: notificar a los usuarios en situaciones de emergencia. Debido a que actualmente se está trabajando con las plataformas Android y iOS los dos servidores que se usan son los de Google y Apple respectivamente.

La correcta interacción de estos bloques se garantiza a través de los servicios. Los cuales tienen una API bien definida y un modelado de datos acorde a la misma. Para garantizar que es fácil acceder a la información y procesarla.

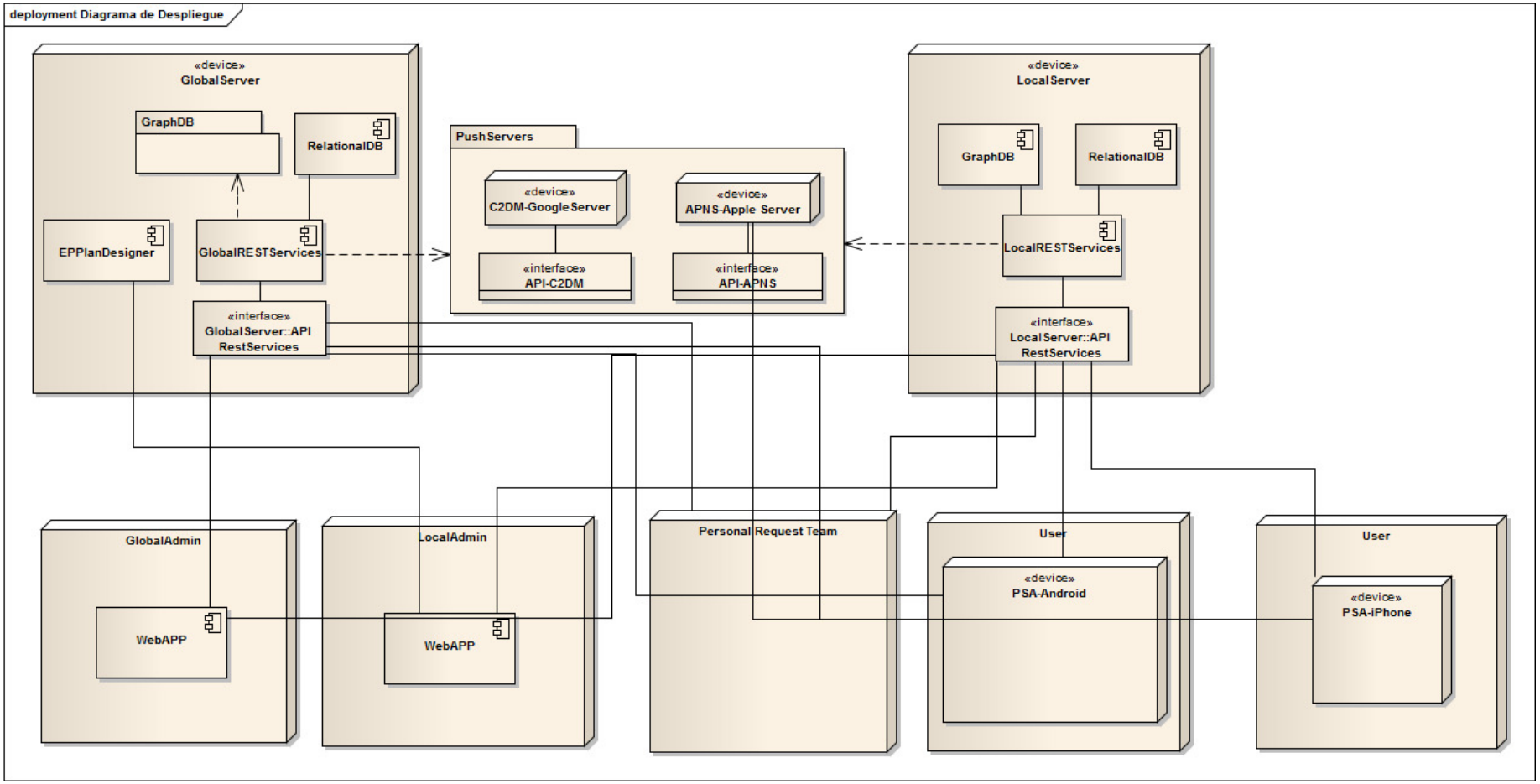



Figura 13 – Arquitectura General

## II. 2.2. Interfaces de Comunicación

Mirando la Figura 13 se observa que se hace necesario establecer diversos mecanismos para conseguir una correcta comunicación entre los distintos elementos del sistema (usuario-servidor local, servidor local-servidor general, etc.) con el fin de conseguir una total interoperabilidad.

Con el fin de ofrecer una funcionalidad variada se definen en esta sección los verbos del protocolo de comunicación de cara a poder ser usados por los distintos elementos del sistema, pudiendo verse modificadas a lo largo del tiempo. Con el objetivo de ir cubriendo diversas necesidades.

Al haber 3 canales de comunicación principales y distintos formados por las parejas cliente o usuario y servidor local, servidor local y servidor global y por último servidor global y usuario se definirán distintos elementos de comunicación en función del canal de comunicación, siendo posible que compartan algún verbo común.

Dichos protocolos de comunicación deben ser comunes para todos los elementos que formen parte del sistema, para permitir que cualquier cliente pueda acceder a los servicios del mismo en cualquier instalación.

Además y debido a que se deben garantizar una comunicación correcta dichos verbos se aplicarán en una comunicación orientada a conexión, donde se controlará el intercambio de mensajes para que no haya problema de pérdida de datos o similares.

### II. 2.2.1. Cliente / Servidor Local

Se definen distintos verbos para cada una de las partes y que podrán ser usados por el otro elemento de la pareja o por otro elemento del sistema, excepto entre clientes. La implementación concreta correspondiente a esta definición se puede consultar en el Anexo I.

**Cliente**

- *alert*
  Sirve para permitir al cliente o usuario recibir notificaciones o avisos de cualquier tipo por parte del servidor.
  En el caso de las emergencias dicho mensaje tendrá una estructura concreta que representará las acciones que el cliente debe realizar.

**Servidor Local**

- *connect*
  Permite a un cliente conectarse al servidor. El cual establecerá el modo de conexión dependiendo de la infraestructura, o bien automáticamente sin

necesidad de previo conocimiento de la dirección del servidor o bien haciendo uso del servidor global para conocer dicha dirección.

Cuando dicho usuario establezca la conexión suministrará al servidor información del mismo, como puede ser información personal, datos relevantes que puedan ser importantes como el sistema (como si se ha estado antes conectado con dicho servidor local, datos médicos, etc.).

- *identify*

  Una vez que el usuario se ha conectado y el servidor ha recibido información del mismo, el usuario procederá a identificarse usando este protocolo. Donde el servidor le enviará información relevante para el uso adecuado del resto de verbos del protocolo. Como puede ser la información relacionada con un plan de evacuación o bien un fichero XML donde se defina dicho plan (capítulo 3, Modelado de Datos).

- *subscribe*

  Permite estar suscrito a diferentes tipos de información, siendo algunas de ellas de carácter obligatorio (como puede ser la de control de emergencias) y otras de carácter opcional (notificaciones de publicidad, notificaciones informativas, etc.).

- *hist_info_items*

  Debido a que el servidor tendrá un log donde se almacenará cualquier tipo de acción en o sobre el mismo, cualquier usuario tendrá acceso a los mensajes que le envío al servidor o los mensajes/peticiones que le envío al servidor.

- *get_info_item*

  Un usuario podrá obtener un ítem (plan de evacuación, fichero con otro contenido, etc.) por parte del servidor. Estos ítems podrán definirse en el servidor y tendrán una diseminación particular por parte del usuario.

- *navigate*

  Este verbo tendrá como argumento la imagen o bien algún tipo de información referente y unívoca a un QR code. El servidor al interpretar esta información generará una ruta de evacuación codificada.

- *disconnect*

  El mecanismo de desconexión debe ser automático, el cual provoca que el servidor declare fuera del sistema a un usuario cuando este abandone el edificio.

## II. 2.2.2. Servidor Local/Servidor Global

**Servidor Global**

- *register*
  Permite registrar un nuevo servidor como parte del sistema.
- *alert*
  Aviso de emergencia o cualquier otro tipo de notificación que envía un servidor local.
- *info_user*
  Permite que el servidor global reciba información de un usuario. De esta manera es posible que el servidor global conozca todos los usuarios conectados a un servidor local concreto junto con información importante de estos, la cual podrá ser usada en caso de ser necesario. Como por ejemplo en situaciones donde el servidor local deja de funcionar. Para que la información permanezca actualizada se realizará esta opción cada cierto intervalo de tiempo.
- *set_info_item*
  Un servidor local podrá subir planes de evacuación o ficheros con información importante o relevante a un servidor global.

**Servidor Local**

- *alert*
  Aviso de emergencia o cualquier otro tipo de notificación que envía un servidor global a un servidor local para que este lo transmita a los usuarios que están conectados a ese servidor local.
- *synchronize*
  Sirve para comprobar que dicho servidor local sigue activo y por tanto formando parte del sistema

## II. 2.2.3. Cliente/Servidor Global

**Cliente**

- *alert*
  Sirve para permitir al cliente o usuario recibir notificaciones o avisos de cualquier tipo por parte del servidor global.

**Servidor Global**

- *get_info_item*
  Un usuario podrá obtener un ítem (plan de evacuación, fichero con otro contenido, etc.) por parte del servidor global. Estos ítems ya estarán definidos, ya que fueron definidos por el servidor local y enviados al global.
- *navigate*
  ~~Este verbo tendrá como argumento la imagen o bien algún tipo de información referente y unívoca a un QR code.~~ El servidor al interpretar esta información permitirá identificar la ruta de evacuación previamente generada por el servidor local.
- *alert*
  Aviso de emergencia o cualquier otro tipo de notificación que envía un usuario.

La Figura 14, muestra gráficamente los verbos de los protocolos que usan los elementos entre sí así como la relación de estos y la tabla inmediatamente a continuación, Tabla 1, muestra que elementos interaccionan entre sí.

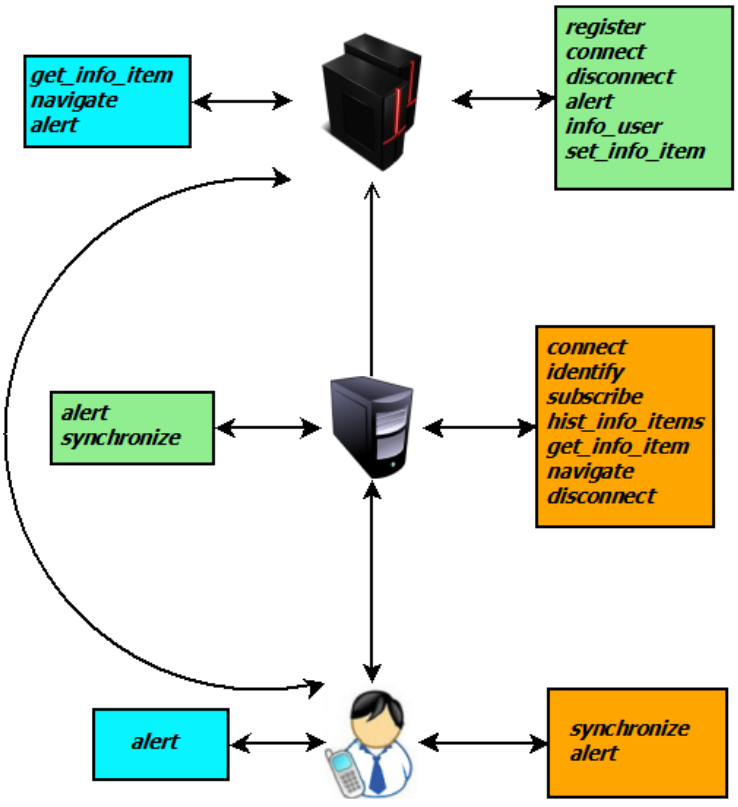



Figura 14 – Esquema general de la interfaz de comunicación

| | Cliente | Servidor Local | Servidor Global |
|---|---|---|---|
| **Cliente** | | | |
| **Servidor Local** | | | |
| **Servidor Global** | | | |

Tabla 1 – Tabla-resumen de la interacción entre elementos

## II. 2.3. Generación de planes de evacuación

Con el objetivo de poder generar una ruta de evacuación concreta para un determinado usuario se optado por un modelo basado en una estructura de grafos donde tenemos los dos elementos básicos de un grafo pero adaptados a este contexto:

- **nodos**: representan elementos físicos dentro del contexto de evacuación y que pueden ser o bien nodos iniciales (donde se encuentra un cliente), nodos finales (donde debe llegar) o nodos intermedios (que nodo debe visitar).
- **aristas**: los caminos entre estos nodos, que a su vez determinan la ruta de evacuación al obtener el camino mínimo entre nodos.

Cada nodo tiene 2 atributos básicos y necesarios para poder dar soporte a la funcionalidad de evacuación:

- **Localización**: en nuestro contexto en concreto y a corto plazo vamos a usar un código QR unívoco (para cada nodo de emergencia) que servirá al usuario, a través de su procesamiento con su dispositivo móvil, para obtener una ruta de evacuación mínima y segura tomando dicho código QR como posición inicial (o "nodo padre" en nuestro contexto)
- **Nodo final**: si es un nodo final o no, esto es, un nodo que indica una situación en la que el usuario se encuentra a salvo: la salida del edificio, un lugar controlado por personal de seguridad y totalmente fiable, etc.

A las aristas se le han asignado:

- **costes**: representarán la distancia en una unidad métrica entre un nodo y otro. Este atributo puede ir cambiando durante una situación de emergencia, al incrementarse antes caminos intransitables o bien reducirse antes nuevos caminos abiertos.
- **acciones**: serie de pasos que tiene que realizar un usuario para poder moverse entre nodos QR hasta llegar a un nodo final. De esta manera una simple lista de acciones (de cara al usuario final) le permitirían evacuar correctamente el edificio o lugar donde se haya declarado la emergencia.
- **Nodo inicial y nodo final**: nodo inicial y nodo final de la arista correspondiente

# II. 2.4. Arquitectura del Sistema

El diagrama de la Figura 15 muestra la arquitectura del sistema haciendo uso de un diagrama basado en componentes, donde se muestran por paquetes los distintos elementos del sistema así como los distintos componentes que forman parte de cada uno y su interacción con el resto.

La elección de diseñar una arquitectura basada en componentes reside extensibilidad futura, de cara a poder añadir nuevos componentes a los sistemas arriba definidos sin que interfiera en el resto de elementos. O bien modificar internamente algún componente sin que eso interfiriera en el resto de componentes.

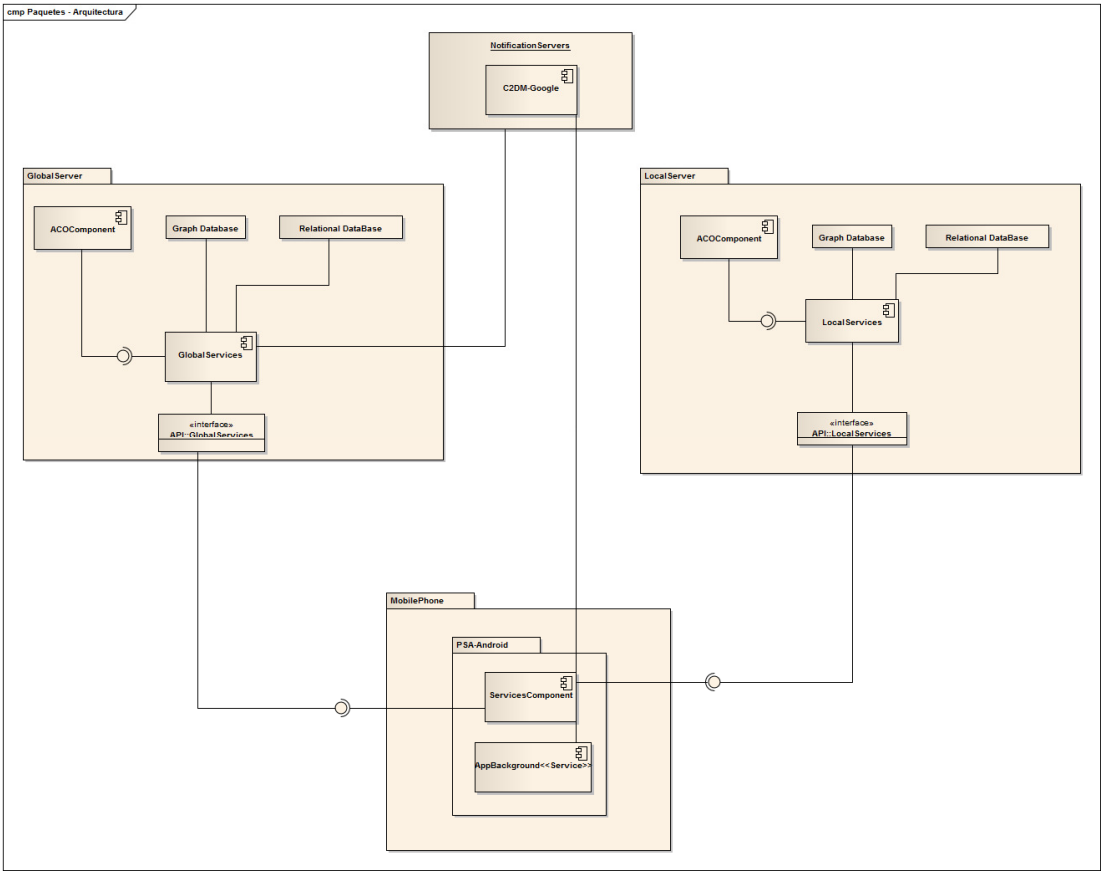



Figura 15 – Arquitectura del Sistema

# II. 2.5. Modelado de los datos

Debido a la orientación a servicios de este sistema se necesita definir correctamente la estructura correcta de los tipos de datos que se van a intercambiar entre cliente y servidor. De cara a poder conseguir una total interoperabilidad entre dispositivos y tener una estructura bien definida para que pueda usar por cualquier dispositivo que haga uso de esta API. Ejemplos así como una pequeña descripción de cada modelo se puede consultar en el Anexo I. Correspondiente a la API del sistema.

En este apartado se muestran a alto nivel y desde un punto de vista más amigable el prototipo de llamada, es decir: nombre, parámetros y valor de retorno. En el anexo IV se puede consultar los esquemas XML correspondiente al modelado de los datos. Los cuales tendrán una implementación concreta en el sistema final.

## II. 2.5.1 Servidor Local

### II. 2.5.1.1. Nueva emergencia

| | |
|---|---|
| **Nombre** | Nueva Emergencia |
| **Parámetros de Entrada** | String Date, String login, String location, String description |
| **Valor de Retorno** | --- |

### II. 2.5.1.2. Obtener plan de evacuación

| | |
|---|---|
| **Nombre** | Obtener Plan de Evacuación |
| **Parámetros de Entrada** | String qr, String ln, String login, String alg |
| **Valor de Retorno** | String date, String login, String[] actions |

### II. 2.5.1.3. Notificar de variación contextual

| | |
|---|---|
| **Nombre** | Notificar Tramo Cortado |
| **Parámetros de Entrada** | String login, String startqr, String endqr |
| **Valor de Retorno** | --- |

### II. 2.5.1.4. Dar de alta un cliente en un servidor local

| | |
|---|---|
| **Nombre** | Dar de alta un cliente en una entidad |
| **Parámetros de Entrada** | String login, String id_push, String op |
| **Valor de Retorno** | --- |

#### II. 2.5.1.5. Insertar información en el Log

| **Nombre** | Insertar información en el log |
|---|---|
| **Parámetros de Entrada** | String login, String op, String text |
| **Valor de Retorno** | --- |

#### II. 2.5.1.6. Obtener información de Log de un usuario

| **Nombre** | Obtener información del log de un usuario |
|---|---|
| **Parámetros de Entrada** | String login, String sdate, String edate, String stime, String etime |
| **Valor de Retorno** | String login, String[] op, String [] text |

#### II. 2.5.1.7. Notificar que un usuario está a salvo

| **Nombre** | Notificar que un usuario está a salvo |
|---|---|
| **Parámetros de Entrada** | String op=0, String login |
| **Valor de Retorno** | --- |

#### II. 2.5.1.8. Notificar que un usuario necesita ayuda

| **Nombre** | Notificar que un usuario necesita ayuda |
|---|---|
| **Parámetros de Entrada** | String op=1, String login |
| **Valor de Retorno** | --- |

#### II. 2.5.1.9. Actualizar el último sitio visitado por un usuario

| **Nombre** | Notificar el último sitio visitado |
|---|---|
| **Parámetros de Entrada** | String op=2, String login, String qr |
| **Valor de Retorno** | --- |

#### II. 2.5.1.10. Obtener todos los usuarios que no están a salvo

| **Nombre** | Obtener todos los usuarios que no están a salvo |
|---|---|
| **Parámetros de Entrada** | String op=0 |
| **Valor de Retorno** | String[] users_id |

#### II. 2.5.1.11. Obtener el estado de un usuario en concreto

| **Nombre** | Obtener el estado de un usuario en concreto |
|---|---|
| **Parámetros de Entrada** | String op=1, String login |
| **Valor de Retorno** | String help, String safe, String qr |

#### II. 2.5.1.12. Obtener todos los usuarios a salvo

| **Nombre** | Obtener el estado de un usuario en concreto |
|---|---|
| **Parámetros de Entrada** | String op=2 |
| **Valor de Retorno** | String[] users_id |

#### II. 2.5.1.13. Obtener información de un código qr

| **Nombre** | Obtener información de un código qr |
|---|---|
| **Parámetros de Entrada** | String op=0, String qr |
| **Valor de Retorno** | String description |

### II. 2.5.2 Servidor Global

#### II.2.5.2.1. Nueva emergencia

| **Nombre** | Notificar una nueva emergencia |
|---|---|
| **Parámetros de Entrada** | String login, String description, String location, String server |
| **Valor de Retorno** | --- |

#### II. 2.5.2.2. Obtener un plan de evacuación

| **Nombre** | Obtener un plan de evacuación |
|---|---|
| **Parámetros de Entrada** | String login, String qr, String alg, String ln, String server |
| **Valor de Retorno** | String[] actions |

#### II. 2.5.2.3. Notificar de variación contextual

| **Nombre** | Obtener un plan de evacuación |
|---|---|
| **Parámetros de Entrada** | String login, String snode, String enode, String server |
| **Valor de Retorno** | --- |

#### II. 2.5.2.4. Registrar un cliente

| | |
|---|---|
| **Nombre** | Dar de alta a un usuario |
| **Parámetros de Entrada** | String login, String op=0, String name, String lastname, String familyphone |
| **Valor de Retorno** | --- |

#### II. 2.5.2.6. Obtener información de un cliente

| | |
|---|---|
| **Nombre** | Obtener información de un cliente |
| **Parámetros de Entrada** | String login |
| **Valor de Retorno** | String name, String lastname, String telephone_family |

#### II. 2.5.2.7. Dar de alta un cliente en un servidor local a través del global

| | |
|---|---|
| **Nombre** | Dar de alta un cliente en un servidor local a través del global |
| **Parámetros de Entrada** | String login, String op=1, String phoneid, String server |
| **Valor de Retorno** | --- |

#### II. 2.5.2.8 Insertar información en el Log

| | |
|---|---|
| **Nombre** | Insertar información en el log |
| **Parámetros de Entrada** | String login, String op, String text, String server |
| **Valor de Retorno** | --- |

#### II. 2.5.2.9. Dar de alta un servidor local en un servidor global

| | |
|---|---|
| **Nombre** | Dar de alta un servidor local en un servidor global |
| **Parámetros de Entrada** | String id, String name, String location |
| **Valor de Retorno** | --- |

#### II. 2.5.2.10. Obtener información de un servidor local

| | |
|---|---|
| **Nombre** | Obtener información de un servidor local |
| **Parámetros de Entrada** | String id |
| **Valor de Retorno** | String id, String name, String location |

#### II. 2.5.2.11. Dar de alta el perfil sanitario de un usuario

| **Nombre** | Dar de alta el perfil sanitario de un usuario |
|---|---|
| **Parámetros de Entrada** | String login, String sip, String bloodgroup, String allergies, String anticoagulant, String breathdes, String info |
| **Valor de Retorno** | --- |

#### II. 2.5.2.12. Obtener el perfil sanitario de un usuario

| **Nombre** | Obtener el perfil sanitario de un usuario |
|---|---|
| **Parámetros de Entrada** | String login |
| **Valor de Retorno** | String login, String sip, String bloodgroup, String allergies, String anticoagulant, String breathdes, String info |

# II. 3

# Implementación

En esta sección se van a explicar los distintos elementos (herramientas o aplicaciones) que se han implementado siguiendo los diseños antes mencionados.

## II. 3.1. Tratamiento de Planes de Emergencia

### II. 3.1.1. Herramienta de Generación de Grafos

Como vimos en el apartado 1.7 la estructura elegida y que más se adecua a la generación de planes de evacuación es una estructura de grafo. En esta estructura los nodos representan o bien los posibles puntos de partida inicial de los usuarios, nodos intermedios en una ruta o bien nodos finales y sirven para mostrar el camino, a través de una serie de aristas. A estas aristas se les asignaba una serie de acciones, entre otros datos, que serían las acciones que se mostrarían al usuario final de cara a poder seguir dichas acciones a modo de instrucciones para poderse evacuar correctamente en una situación de emergencia.

Para poder generar de una manera fácil y rápida los grafos generales que representan la estructura de un posible edificio o corporación se ha realizado una herramienta web de generación de grafos, como se puede ver en la Figura 16 y 17

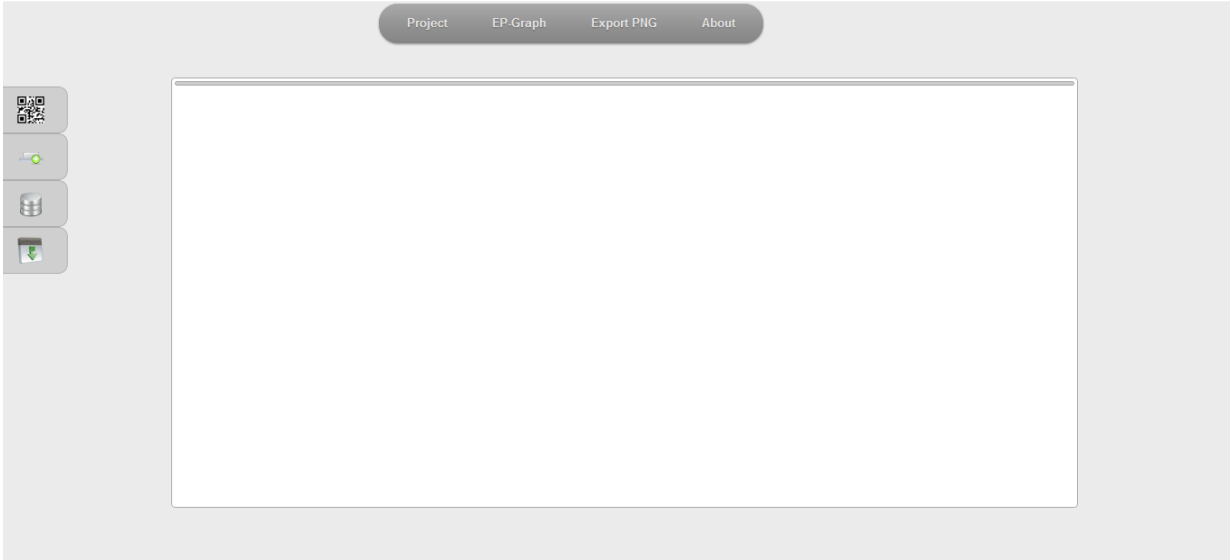



Figura 16 – Herramienta Web para la generación de Grafos (I)

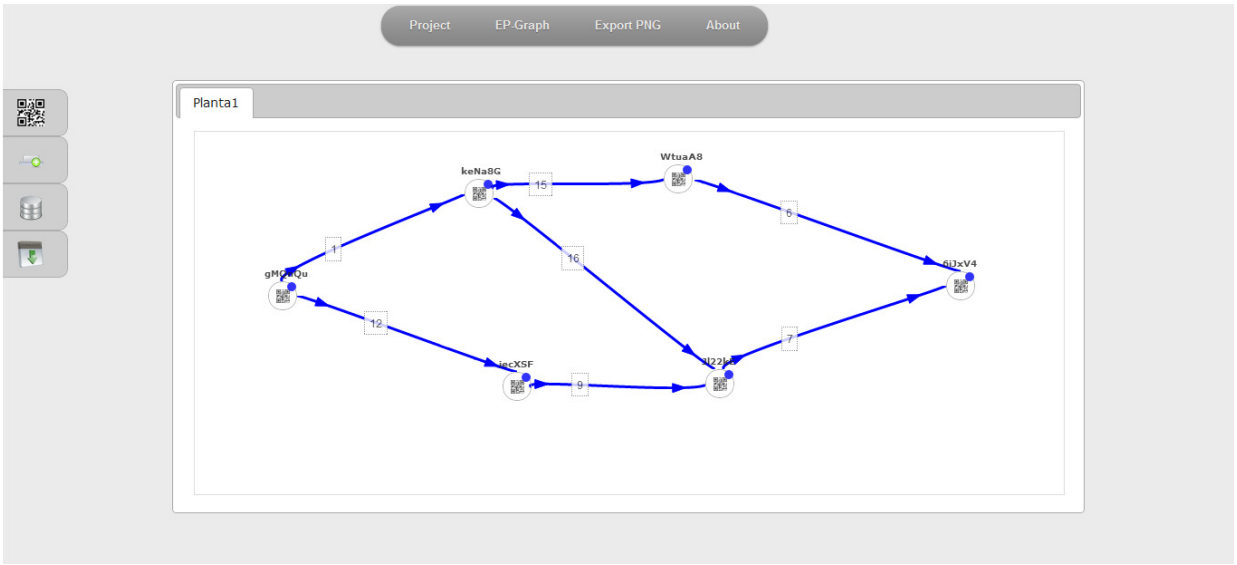



Figura 17 – Herramienta Web para la generación de Grafos (II)

Dicha herramienta tiene como principal objetivo que el usuario pueda diseñar toda la estructura de nodos y aristas que representan un edificio o emplazamiento de una manera rápida, fácil e intuitiva. Sencillamente iría añadiendo nuevas plantas o zonas, y sobre estas añade los distintos nodos que representan en la realidad distintos nodos QR. Una vez definidos los nodos QR añade los enlaces entre nodos con las acciones correspondientes. Además, la herramienta permite en primera instancia generar automáticamente el identificador unívoco de los nodos QR que se van creando y exportar luego cada nodo QR como una imagen, para facilitar la tarea a las organizaciones que usen dicha herramienta.

Una vez que el diseño esta completo: todos los nodos con su código QR definido, los enlaces entre los nodos correspondientes todos con sus respectivas acciones, los nodos finales (nodos que indican que es el nodo objetivo para estar a salvo)y las distintas plantas o zonas conectadas, la herramienta permite generar y descargar una base de datos orientada a grafos (usando el motor Neo4J [9])que sigue una estructura similar a la que se puede ver en la figura 17, la cual será la usada en el servidor local de la institución o corporación correspondiente. Esta base de datos será la que usarán los servicios para poder obtener las distintas rutas, obtener información de acciones, etc.

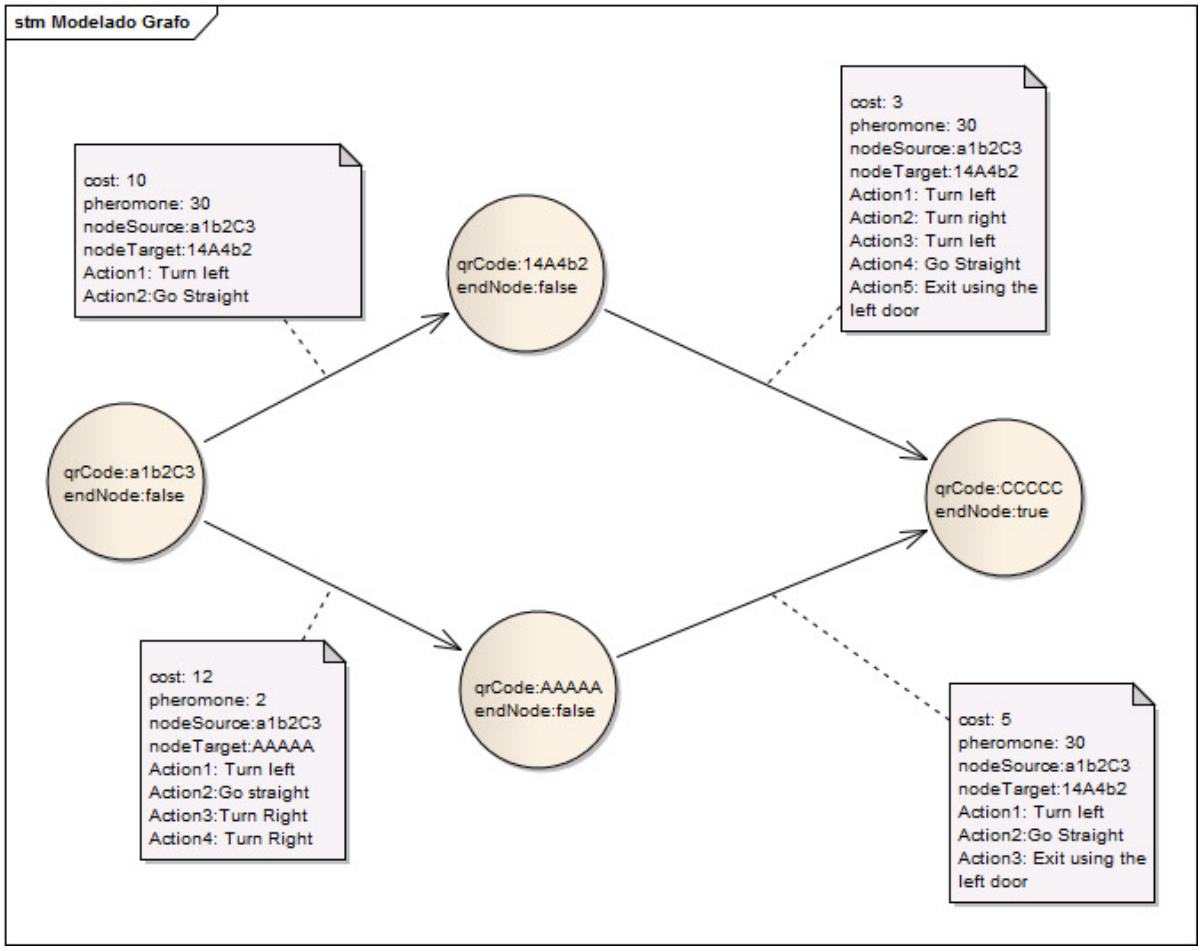



Figura 17 – Modelo representativo de la implementación del modelo del Grafo

La herramienta además permite exportar e importar planos concretos (plantas de edificios, zonas comunes, etc.) así como guardar y abrir proyectos previamente creados. De esta manera una entidad que tenga que modelar varios edificios puede sencillamente unir las distintas plantas de ambos proyectos ya definidos y enlazarlas, o bien abrir proyectos ya creados para modificar alguna parte del diseño que ha de ser cambiado. Generando de nuevo la base de datos para poder actualizar en el servidor correctamente los cambios realizados.

La estructura de dichos modelos de datos es un fichero XML con una estructura bien definida y con soporte para poder representar cualquier plano.

## II. 3.1.2. Cálculo de planes de evacuación

Partiendo del grafo antes mencionado, se deberá calcular una ruta de evacuación partiendo de un nodo inicial facilitado por cliente y unos nodos finales ya previamente definidos, aunque pueden cambiar a lo largo del tiempo si el contexto lo requiere. De cara a poder generar planes de evacuación capaces de evacuar a los clientes en el menor tiempo posible y de la manera más rápida posible, se han usado dos algoritmos distintos pero igualmente válidos de cara a obtener esta ruta o camino mínimo.

**Algoritmo de optimización basado en Colonias de Hormigas**

Esta primera alternativa se basa en usar algoritmos probabilísticos de optimización basados en colonias de hormigas. Es decir, partiendo del nodo inicial hay que llegar a un nodo final simulando el comportamiento real que usan las hormigas.

Debido que las hormigas son ciegas van realizando movimientos aleatorios, y en los caminos que recorren van dejando unas feromonas que pueden ser percibidas por otras hormigas. Este nivel de feromonas es el que siguen otras hormigas de cara a elegir un camino u otro. Debido a que el nivel de feromona va decrementando debido a la evaporación de la misma, a lo largo del tiempo únicamente quedará "marcado" el camino óptimo desde un punto A hasta un punto B.

En nuestro contexto la aplicación es similar, en vez de ir en busca de comida vamos a identificar el nodo objetivo como el nodo de salida que representa la última posición para poder estar a salvo en una situación de emergencia.

Para poder adaptar y usar esta solución teórica hemos usado el motor de optimización basados en colonias de hormigas creado por el profesor Javier Jaén Martínez del DSIC [16] y adaptado por él y su equipo para nuestro contexto en particular.

Debido a la potencia de dicho motor, el cual está desarrollado en un plano muy general para poder adaptarse a cualquier contexto, ha sido realmente fácil y directo poder aplicarlo a nuestro contexto. El procedimiento para adaptar y usar dicho motor a nuestro sistema consta de los siguientes pasos:

1. Obtener de nuestra base de datos orientada a grafos todos los nodos, y su información (el código QR y si es nodo final o no), y todas las aristas pero únicamente obteniendo el coste y el nivel de feromona de esta.
2. Por cada nodo de nuestra base de datos se crea un tipo de nodo-hormiga del motor de hormigas (AntNode)
3. Por cada relación de nuestros nodos se crea una relación entre nodos AntNode. Si no existe la relación en sentido inverso se define una con un coste elevado. Esto se realiza ya que las hormigas para recorrer los distintos nodos en todos sus posibles caminos a lo largo del tiempo tienen que poder volver al nodo anterior. Entonces se define dicha relación aunque en la práctica no esté reflejada para que puedan volver al nodo visitado, pero al tener un coste muy

elevado no lo consideran como arista-solución. De esta manera garantizamos que el motor de hormigas puede funcionar correctamente y nosotros obtener el camino mínimo sin ciclos.

4. A cada una de estas relaciones se le asigna el coste y el nivel de feromona obtenido en el punto 1. En caso de no definir ninguno se le asignar el valor por defecto del motor.
5. Por cada nodo de nuestra base de datos que este definido como nodo final, en el correspondiente nodo-hormiga se le añade un recurso "SALIDA". El recurso es el objetivo que van buscando las hormigas. Sería el homólogo a la comida en un contexto real.
6. Se identifica el nodo inicial a través del código QR y en su correspondiente AntNode se lanzan (4*X) + 100 consultas de búsqueda siendo X el número de nodos totales del sistema. Ya que al ser un algoritmo probabilístico y al haber variables aleatorias se necesitan repetidas iteraciones para conseguir el resultado correcto.
7. Una vez obtenida la ruta mínima (conjunto de identificadores de nodos) se actualizan en la base de datos el nivel de feromona de cada relación.
8. Con el conjunto de identificadores (código QR) se accede a la base de datos y se obtiene las acciones asociadas a cada relación dos a dos y consecutivas ( X y X+1, X+1 y X+2,…., X+(N-1) y X+N)
9. Con estas acciones y otra serie de metadatos se genera el plan de evacuación que se le enviará al cliente.

Este proceso se realiza por cada consulta o petición de un cliente para obtener un plan de evacuación, de esta manera los resultados irán siendo coherentes en función de las peticiones de otros clientes, al ir actualizando entre todos el nivel de feromonas el elegir el camino correcto.

El algoritmo internamente usa una ponderación entre coste y nivel de feromona para obtener el camino mínimo y correcto por lo que realmente obtendría la solución óptima desde la primera petición, en este caso el nivel de feromonas es un ayuda de cara a reforzar la toma de decisiones por parte del sistema en un futuro.

**Floyd-Warshall**

El algoritmo de Floyd-Wharsall es un algoritmo de análisis sobre grafos que sirve para encontrar el camino mínimo en grafos dirigidos y ponderados [17]. Es decir, permite encontrar el camino mínimo entre cada par de vértices del grafo en función de un coste asociado a cada arista.

La aplicación de dicho algoritmo en nuestro contexto es directa, entendiendo el concepto de ponderación de arista como coste de distancia.

Los pasos implementados en nuestro sistema son los siguientes:

1. Obtener de nuestra base de datos los nodos(únicamente su identificador) y las relaciones (únicamente los costes)

2. Definir dos estructuras de datos que serán la matriz de distancias y matriz de caminos de tamaño NxN siendo N el número de nodos.
3. Inicializar en la matriz de distancias el elemento (i, i) con valor 0 y con valor INFINITO (un valor relativamente alto y superior a cualquier coste real) en la matriz de caminos mínimos.
4. Ir recorriendo las distintas relaciones obtenidas de nuestra base de datos e ir asignando el coste entre cada pareja de nodos en la matriz de distancias y el nodo-fuente en la matriz de distancias.
5. Para las relaciones no existentes identificarlas como INFINITO en ambas matrices.
6. Aplicar el algoritmo de Floyd-Warshall [18] para obtener las matrices finales de distancia y camino.
7. Portar dichas matrices a una estructura de datos persistente (en nuestro caso tablas en un modelo relacional)

Aunque todo este proceso tenga una eficiencia computacional O(n^3) la principal ventaja es que esto únicamente tiene que realizarse una vez, en nuestro caso sería al declarar una emergencia.

Una vez aplicado todo esto tendremos en nuestro esquema relacional dos tablas correspondientes a las matrices de distancias y camino, ahora cada vez que un cliente solicite una ruta mínima en función de su código QR únicamente habrá que ir a la tabla correspondiente de matrices de caminos y obtener de ahí el camino mínimo en un tiempo lineal.

Con este camino mínimo se accedería a nuestra base de datos orientada a grafos y se obtendría la lista de acciones asociadas a cada par de nodos.

Si se produjese alguna notificación que alterase el coste entre un camino, se modificaría en la tabla/matriz de distancias y se volvería a aplicar el algoritmo de Floyd-Warshall para obtener las matrices de distancia y camino mínimo correspondientes.

La principal ventaja de este algoritmo respecto al algoritmo basado en colonias de hormigas reside en que es mucho más simple y más eficiente, computacionalmente hablando, a lo largo del tiempo.

## II. 3.2. Arquitectura Orientada a Servicios

Para dar soporte a todas las operaciones o acciones definidas en el punto 1.1 se ha optado por implementar una arquitectura orientada a servicios basada en una arquitectura REST para implementar servicios web, los cuales serán usados por los distintos usuarios del sistema.

Los motivos de elegir dicha tecnología en contraposición a las otras dos disponibles, SOAP y RPC, han sido:

- El contexto en el que se centra el sistema, donde el uso de dispositivos móviles va a ser esencial, hace necesario que la cantidad de información entre dichos dispositivos y el servidor sea mínima, debido a la posible limitada capacidad de ancho de banda en los dispositivos móviles. Con SOAP esta cantidad de información se incrementa notablemente, al ser necesario especificar la descripción del servicio (WSDL) así como las cabeceras de intercambio de información, que hacen que incremente el tamaño de archivo considerablemente.
- Aunque en la práctica con REST y RPC se obtiene el mismo resultado, desde un punto de vista conceptual REST se adaptaba mejor a la arquitectura del sistema, siendo así más fácilmente implementable o ampliable de cara al desarrollador o futuros desarrolladores.
- El creciente uso de REST en Internet hace que sea una apuesta segura de cara a una futura orquestación o composición de servicios con otros organismos al ser un estilo de arquitectura muy extendido.

Con el objetivo de ajustar dicha arquitectura de servicios a la infraestructura física del sistema, se han desarrollado dos paquetes de servicios distintos (figura 18). Un paquete que se encontrará en cada uno de los servidores locales perteneciente a una organización o entidad y otro que irá en el servidor global.

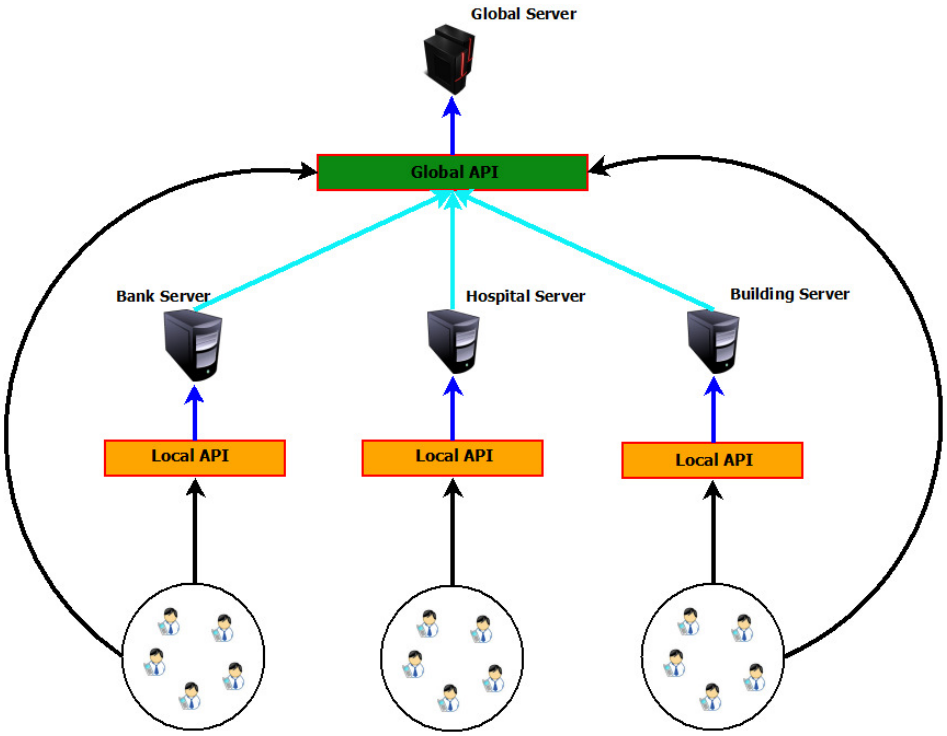



Figura 18 – Arquitectura REST

## II. 3.3. Asistente Personal de Seguridad

La aplicación final a usar por el usuario será una aplicación para Smartphone (Android) que permita al usuario a través de su dispositivo móvil ser avisado y recibir ayuda en caso de emergencia.

Además el paciente con esta aplicación podrá realizar todas las funcionalidades especificadas anteriormente: registrarse en el sistema, pedir planes de evacuación aunque no haya una emergencia, notificar de una emergencia, recibir notificaciones (aunque por ahora solo de emergencia), etc.

Las siguientes figuras muestran dicha aplicación y las distintas funcionalidades que tiene:

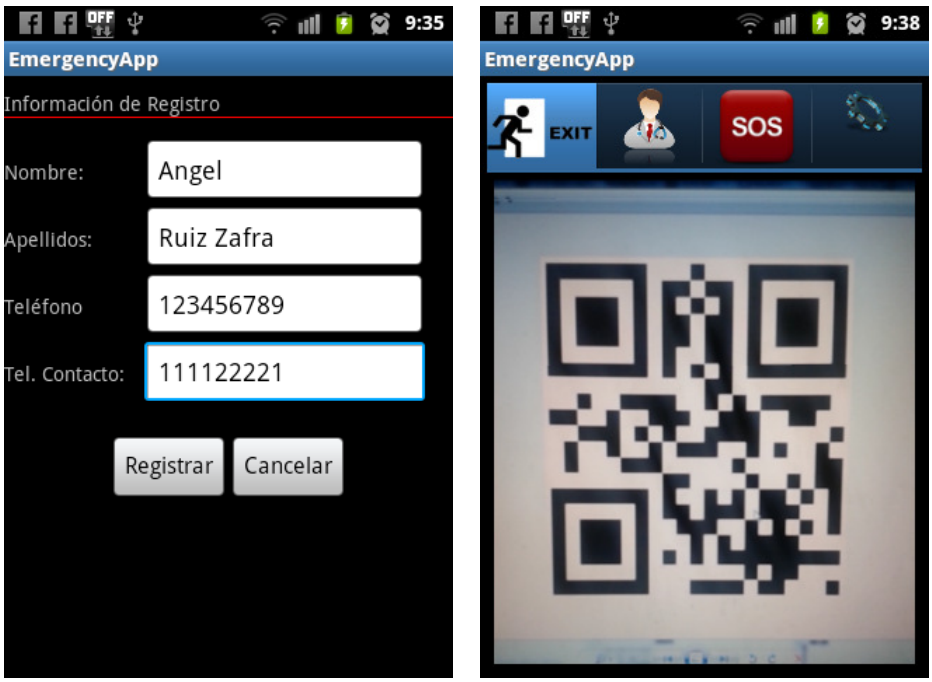



Figura 19 – Pantalla de registro (Izquierda) y de obtención de un plan de evacuación (derecha)

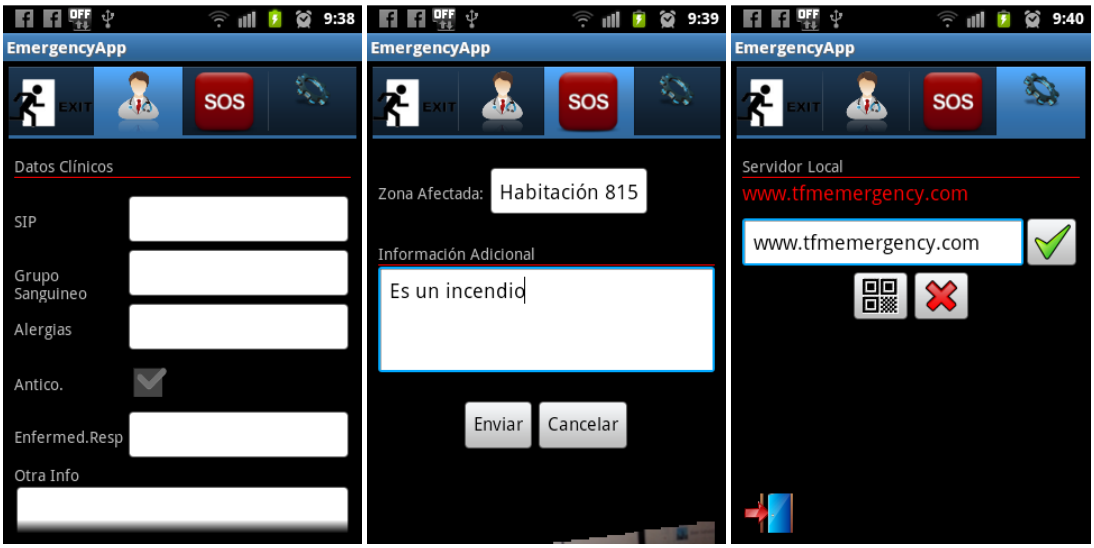



Figura 20 – Datos sanitarios (izquierda), declaración de emergencia (central) y parámetros de configuración (derecha)

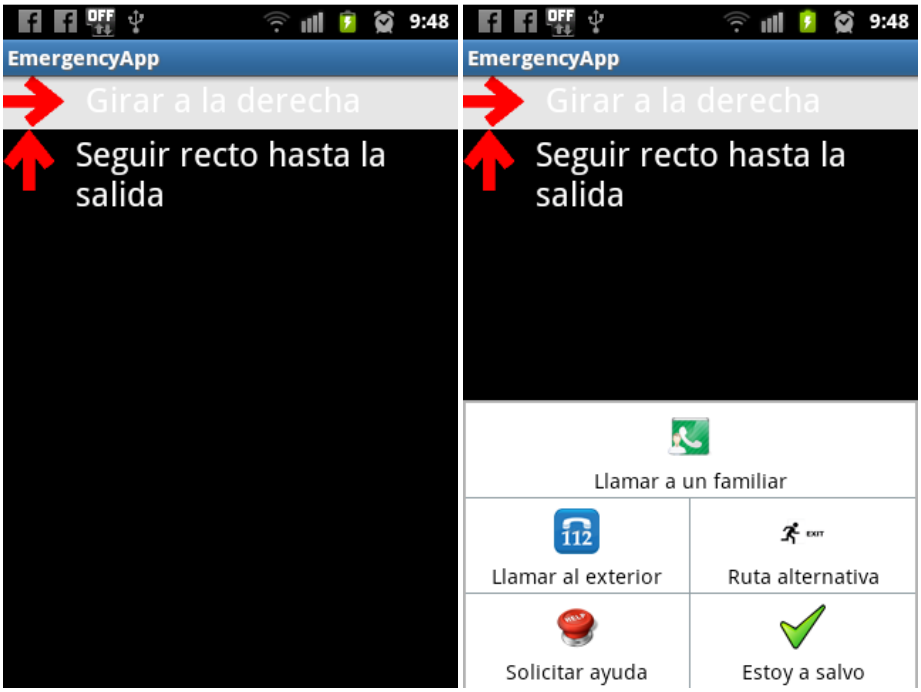



Figura 21 – Ventana de evacuación (izquierda) y misma ventana con menú(derecha)

# Parte III:
# Conclusiones

# III. 1

# Conclusiones

## III. 1.1. Conclusiones

Se ha presentado el diseño y parte del desarrollo de un sistema completo para gestionar totalmente una situación dentro de un contexto de emergencia. Un sistema que permite a usuarios actuar correctamente en una situación de emergencia, gracias a su dispositivo móvil y a la generación personalizada de planes de evacuación por parte del sistema. Y que facilita a administradores del sistema controlar en todo momento el estado de una emergencia así como todo el contexto de esta: usuarios que están a salvo, zonas declaradas como afectadas, usuarios que están atrapados, etc.

Así mismo las propias características derivadas del diseño planteado posibilitan que dicho sistema sea totalmente extensible, escalable y robusto. En cuanto a garantizar en todo momento el acceso a los servicios por parte de los usuarios finales así como equipos de respuesta, que son en definitiva, los usuarios finales que necesitan del sistema para actuar correctamente en una situación de emergencia.

Buscando conseguir un mayor grado de extensibilidad, funcionalidad y personalización se han definido 4 actores básicos: Usuario final, Personal Externo de Respuesta (Bombero, Policía, etc.), Administrador Local y Administrador Global. Que junto con un diseño orientado a servicios y una herramienta de generación de emergencias posibilita suplir sin ningún problema las características principales definidas como objetivos para el sistema.

Además, al usar una arquitectura orientada a servicios, se posibilita que en un futuro nuevos desarrolladores que continúen con el trabajo modifiquen, añadan o eliminen dichos servicios de cara a satisfacer características que vayan surgiendo a lo largo del tiempo. Facilitando además que cualquier usuario con cualquier Smartphone, sea del sistema operativo que sea, pueda hacer uso del sistema gracias a la

interoperabilidad entre dispositivos gracias al estándar XML [7] y a una API bien definida.

Debido a que el diseño propuesto excede con creces el objetivo de esta tesis, no todo se ha implementado y finalizado. Pero se ha dejado el diseño hecho y las bases implementadas para facilitar la continuación de dicho proyecto, y que se mencionan en el apartado siguiente como trabajo futuro.

Los elementos que se han desarrollado como parte del diseño propuesto han sido:

- **Arquitectura orientada a servicios**: Se ha propuesto en el diseño y se ha desarrollado una arquitectura orientada a servicios basada en tecnología REST [13] que dota de funcionalidad al sistema en función de los requisitos definidos en esta tesis. Dicha arquitectura es totalmente extensible y se le pueden añadir nuevos servicios en un futuro. Todas las funcionalidades que esta arquitectura soporta son las definidas en la API de la misma (Anexo I) y que son: registrar usuarios, registrar emergencias, obtener plan de evacuación, notificar de un tramo cortado, entre otras.

  Dicha arquitectura tiene servicios como respaldo, debido a la naturaleza de la gestión de una emergencia, para garantizar que aunque haya problemas el usuario y equipos de respuesta puedan seguir usando el sistema.
- **Generación de planes de emergencia**: Para poder generar planes de evacuación personalizados es necesario representar un contexto general de emergencia. Se ha desarrollado una herramienta web completa que posibilita el diseño de un contexto/escenario general de emergencia que permite generar una implementación de dicho diseño, como una base de datos orientada a grafos. Y que después junto con determinados servicios genera un plan de evacuación personalizado.
- **Asistente Personal de Seguridad**: Es una aplicación para Android que hace uso de la arquitectura orientada a servicios y que sirve de primera versión como aplicación real para el usuario final. Dicha aplicación posibilita el registro de un usuario, darlo de alta en un servidor, pedir un plan de evacuación, notificar una emergencia, notificar un tramo afectado, gestionar su perfil sanitario, etc. Actualmente da soporte en español e inglés.

# III. 1.2. Trabajo Futuro

Aunque el sistema sea completamente funcional, en un futuro se le pueden implementar y añadir un sinfín de posibilidades adicionales, no solo consiguiendo un sistema mucho más completo si no adecuándose a las distintas necesidades que vayan surgiendo a lo largo del tiempo.

Es por esto que aquí se proponen una serie de cambios o elementos nuevos que se podrían realizar en un futuro:

- Adaptar toda esta arquitectura al paradigma de computación en la nube (Cloud Computing): Se hace necesario dar soporte en tiempo real con total fiabilidad a las distintas peticiones de los distintos usuarios en situaciones de emergencia. Dichas peticiones crecen considerablemente en una situación de emergencia, y si además algún servidor local sufre problemas es el servidor global quien debe atender dichas demandas. La computación en la nube [18] suple con creces esta necesidad, pudiendo automáticamente gestionar la carga para atender cualquier número de peticiones. Además, esto podría simplificar la arquitectura y eliminar el elemento servidor local. Ya que al no haber problemas de carga y una capacidad de espacio y computo muy elevado, no se precisa redistribuir la carga y las funcionalidades entre servidores. Existen muchas plataformas a nivel SaaS que pueden dar soporte fácilmente para portar todo el sistema a la nube. Entre las que destacan Google Engine, Windows Azure, iCloud o G-Technology, de la compañía Española Gnúbila[19]
- Portar el PSA a otras plataformas: En esta tesis se ha desarrollado únicamente para Android, mientras que para la tesis complementaria también ha sido desarrollada para iPhone. De cara a tener un sistema completo para cualquier tipo de usuario sería recomendable implementar distintas versiones del PSA para las distintas móviles actuales y futuras.
- Herramientas de Gestión de los Administradores: Aunque son actores omnipresentes en el diseño, lo cierto es que no han participado en el desarrollo. Se precisa de herramientas para poder mostrar una diseminación de toda la información relevante en una emergencia. Es por esto que un trabajo futuro es desarrollar las herramientas tanto para el administrador local como el servidor global.
- Añadir nuevos algoritmos de posicionamiento: En el diseño se contempla la posibilidad de usar distintos algoritmos de posicionamiento aunque en el desarrollo y por simplicidad y eficacia se ha usado código QR. En un futuro sería necesario acoplarle al sistema posicionamiento GPS (para emergencias en exteriores) y otros métodos de posicionamiento en interiores como los basados en "*fingerprinting*" [20] o filtros de partículas y redes bayesianas [21], siendo estos últimos los que dan mejores resultados.

Esto posibilitaría no solo localizar la posición inicial de un usuario y a partir de ahí obtener su plan de evacuación, si no seguir su recorrido en tiempo real y determinar correctamente donde está su posición en cada momento. Mejorando la experiencia del usuario y facilitando a los equipos de respuesta futuras labores de rescate.

# Bibliografía

[1] Servicio Integrado de Prevención en Riesgos Laborales. Universidad Politécnica de Valencia. http://www.upv.es/entidades/SIPRL/infoweb/sprl/info/766491normalc.html
[2] Document product lines: variability-driven document generation. María del Carmen Penadés, José Hilario Canós, Marcos R. S. Borges, Manuel Llavador. ACM Symposium on Document Engineering 2010. PPs: 203-206
[3] A Product Line Approach to the Development of Advanced Emergency Plans. María del Carmen Penadés, José Hilario Canós, Marcos R. S. Borges, Adriana S. Vivacqua
[4] Patricia Gómez Bello, Ignacio Aedo, Fausto Sainz, Paloma Díaz, Jennifer Munnelly, and Siobhán Clarke. “Improving Communication for Mobile Devices in Disaster Response”.
[5] K.-T. Chang, B.-C. Lo, M. C. Teng, Y.-Y. Jiang. “Developing An Android-based Emergency Broadcasting System for Natural Hazards”
[6] Yuanyuan Du, Yu Chen, Dan Wang, JinzhaoLiu, Yongqiang Lu. “An Android-Based Emergency Alarm and Healthcare Management System”
[7] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan, Extensible Markup Language (XML). W3C
[8] http://code.google.com/p/zxing/
[9] http://neo4j.org/
[10] David W. Williams, Jun Huan, Wei Wang. "Graph Database Indexing Using Structured Graph Decomposition". IEEE 23rd International Conference on Data Engineering
[11] www.jquery.com
[12] http://jsplumb.org/
[13] Roy Fielding. "Architectural Styles and the Design of Network-based Software Architectures". 2000
[14] Francisco Curbera, Matthew Duftler, Rania Khalaf,William Nagy,Nirmal Mukhi, and Sanjiva Weerawarana. "An Introduction to SOAP"
[15] https://developers.google.com/android/c2dm/
[16] MoMo: Una Infraestructura basada en Grids para Museos Híbridos, Javier Jaén Martínez.
[17] Teoría de Algoritmos - José Luis Verdegay. Departamento de Ciencias de la Computación e Inteligencia Artificial (DECSAI). Universidad de Granada
[18] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. "A View of Cloud Computing".
[19] http://gnubila.com/
[20] Tomás Ruiz-López, José Luis Garrido, Kawtar Benghazi, and Lawrence Chung. "A Survey on Indoor Positioning Systems: Foreseeing a Quality Design"
[21] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz,Gaetano Borriello. "Bayesian Filtering for Location Estimation"

# Anexo I

## API

### *Local Server*

#### Emergency

**Acción:**

Notificar una nueva emergencia (Enviada por un cliente pero que debe ser autorizada por un personal autorizado)

**Verbo:**

POST

**Entrada:**

```
<?xml version="1.0" encoding="UTF-8"?>
<emergency>
<login> [login] <login>
<location> [location] <location>
<description> [description] </description>
</emergency>
```

**Llamada General:**

/rest/Emergency

**Retorno:**

200 = OK

!200 = Error

**Ejemplo**

http://www.prueba.com/rest/Emergency

http://192.168.1.1/rest/Emergency

**Acción:**

**Notificar el fin de una emergencia (Solo personal autorizado)**

**Verbo:**

**DELETE**

**Entrada:**

**Llamada General:**

**/rest/Emergency**

**Retorno:**

**200 = OK**

**!200 = Error**

**Ejemplo:**

**http://www.prueba.com/rest/Emergency**

**http://192.168.1.1/rest/Emergency**

## Evacuation Plan

**Acción:**

**Obtener un plan de evacuación**

**Verbo:**

**GET**

**Parámetros:**

***qr:*** Código qr leído por el usuario con su dispositivo móvil

***ln***: Idioma del dispositivo móvil o usuario

***id*:** Identificador único del usuario (login, teléfono, etc)

***alg***: Parámetro para definir que método de búsqueda usar. 0 para algoritmos basados en optimización de colonias de hormigas y 1 para Floyd-Wharshall.

**Llamada General:**

**/rest/Evacuation?qr=<qr>&ln=<ln>&id=<id>**

**Retorno:**

```
<?xml version="1.0" encoding="UTF-8"?>
<evacuation>
<login>[login]</login>
<actions>
        <action op=[x] sqr=[qr]  eqr=[qr]>[Action 1]</action>
        <action op=[y] sqr =[qr] eqr =[qr]> [Action 2] </action>
        <action op=[z] sqr =[qr] eqr =[qr]> [Action N] </action>
</actions>
</evacuation>
```

**Ejemplo:**

http://192.168.1.1/rest/Evacuation?qr=qrID24&ln=fr&id=12345678&alg=1

**Acción:**

Dar información de un tramo cortado / con fuego / etc etc . Por donde no se puede pasar o no es aconsejable

**Verbo:**

POST

**Parámetros:**

```
<?xml version="1.0" encoding="UTF-8"?>
<context>
<login> [login] </login>
<startnode>[qrStartNode]</startnode>
<endnode>[endNode]</ennode>
</context>
```

**Llamada General:**

/rest/EVACUATION

**Retorno:**

200 = OK

!200 = Error

**Ejemplo:**

http://www.prueba.com/rest/Evacuation

http://192.168.1.1/rest/Evacuation

## Client

**Acción:**

**"Registrar" un cliente en el servidor local (Dar de alta para que pueda recibir notificaciones)**

**Verbo:**

**POST**

**Entrada:**

```
<?xml version="1.0" encoding="UTF-8"?>
<register>
<op>1</op>
<login> [login] </login>
<id> id </id>
</register>
```

**Llamada General:**

**/rest/Client**

**Retorno:**

**200 = OK**

**¡200 = Error**

**Ejemplo:**

**http://www.prueba.com/rest/Client**

**http://192.168.1.1/rest/Client**

**Acción:**

**“Eliminar” un cliente en el servidor local (Dar de baja un cliente para que pueda recibir notificaciones)**

**Verbo:**

**DELETE**

**Parámetros:**

***login****:* Teléfono

**Llamada General:**

**/rest/Client?id=[id]**

**Retorno:**

**200 = OK**

**!200 = Error**

**Ejemplo:**

**http://www.prueba.com/rest/Client**

**http://192.168.1.1/rest/Client**

## Log

**Acción:**

Añadir nueva entrada al LOG

**Verbo:**

POST

**Entrada:**

```
<?xml version="1.0" encoding="UTF-8"?>
<log>
<login>[login]</login>
<op>[op]</op>
<text>[text]</text>
</log>
```

**Llamada General:**

/rest/Log

**Retorno:**

200 = OK

¡200 = Error

**Ejemplo:**

http://www.prueba.com/rest/Log

http://192.168.1.1/rest/Log

**Acción:**

Eliminar el log de un usuario

**Verbo:**

DELETE

**Parámetros:**

login: identificador del usuario (teléfono)

**Llamada General:**

/rest/Log

**Retorno:**

200 = OK

!200 = Error

**Ejemplo:**

http://www.prueba.com/rest/Log

**Acción**

Obtener el log de un usuario

**Verbo**

GET

**Parámetros:**

login: identificador del usuario (teléfono)

sdate : fecha de inicio (dd-mm-yyyy)

edate : fecha de fin (dd-mm-yyyy)

stime : hora de inicio

etime : hora de fin

**Llamada General:**

/rest/Log?login=<login>&sdate=<sdate>&edate=<edate>&stime=<stime>&etime=<etime>

**Retorno:**

```
<?xml version="1.0" encoding="UTF-8"?>
<log>
<login>[login]</login>
<actions>
        <action date=<date> time=<time>LogAction1</action>
        <action date=<date> time=<time>LogActionN</action>
</actions>
</log>
```

**Ejemplo:**

http://www.prueba.com/rest/Log?login=12345&sdate=10-10-2000&edate=10-10-2000&stime=10:00&etime=11:00

## Status

**Acción:**

**Indicar el estado de un usuario : a salvo, que necesita ayuda, etc**

**Verbo:**

**POST**

**Tipos:**

Estar a salvo (Código de operación = 0)

```
<?xml version="1.0" encoding="UTF-8"?>
<status>
<op>0 </op>
<login> [login] </login>
</ status >
```

Necesita ayuda (OP = 1) [Avisar a bomberos, policía, etc]

```
<?xml version="1.0" encoding="UTF-8"?>
< status >
<op>1 </op>
<login> [login] </login>
</ status >
```

Actualizar QR (OP = 2)

```
<?xml version="1.0" encoding="UTF-8"?>
< status >
<op>2</op>
<login> [login] </login>
<qr>[qr]</qr>
</ status >
```

**Llamada General:**

**/rest/Status**

**Resultado:**

**200 = OK**

**!200 = Error**

**Ejemplo:**

http://www.prueba.com/rest/Status

http://192.168.1.1/rest/Status

**Acción:**

Obtener el estado de la situación (general) de emergencia

**Verbo:**

GET

**Parámetros:**

op = <op>

- ➔ Op = 0 -> Obtener TODOS los usuarios que aún siguen dentro
  /rest/Status?op=0
- ➔ Op = 1 && login=<login> -> Obtener el estado de un usuario en concreto
  /rest/Status?op=0&login=123456
- ➔ Op = 2 -> Obtener todos los usuarios salvados
  /rest/Status?op=2

**Llamada General:**

/rest/Status?op=<op>[&login=<login>]

**Retorno:**

```
<?xml version="1.0" encoding="UTF-8"?>
<clients>
        <client>[client1]</client>
        <client>[clientN]</client>
</clients>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<information>
    <qr>[qr]</qr>
    <safe>[salvado]</safe>
    <help>[help]</help>
</information>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<clients>
        <client>[client1]</client>
        <client>[clientN]</client>
</clients>
```

**Ejemplo:**

http://www.prueba.com/rest/Status?op=0

http://192.168.1.1/rest/Status?op=1&login=12345

## QRCode

**Acción:**

Obtener información de un código QR

**Verbo:**

GET

**Parámetros:**

op = <op>

➔ Op = 0 & qr=<qr> -> Obtiene la descripción de dicho código QR

**Llamada General:**

/rest/QRCode?op=<op>&qr=<qr>

**Retorno:**

```
<?xml version="1.0" encoding="UTF-8"?>
<information>
        <qr>[qr]</qr>
        <description>[description]</description>
</information>
```

**Ejemplo:**

http://192.168.1.1/rest/QRCode?op=1&qr=12345

# *Global Server*

## Emergency

**Acción:**

**Notificar una nueva emergencia (Enviada por un cliente pero autorizada por un personal autorizado)**

**Verbo:**

**POST**

**Entrada:**

```
<?xml version="1.0" encoding="UTF-8"?>
<emergency>
<login> [login] <login>
<location> [location] <location>
<description> [description] </description>
<server>[server]</server>
</emergency>
```

**Llamada General:**

**/rest/Emergency**

**Retorno:**

**200 = OK**

**!200 = Error**

**Ejemplo:**

**http://www.global.com/rest/Emergency**

**Acción:**

Notificar el fin de una emergencia (Solo personal autorizado)

**Verbo:**

DELETE

**Parámetros:**

server=<server>

**Llamada General:**

/rest/Emergency

**Retorno:**

200 = OK

!200 = Error

**Ejemplo:**

http://www.global.com/rest/Emergency

## Evacuation Plan

**Acción:**

**Obtener un plan de evacuación**

**Verbo:**

**GET**

**Parámetros:**

***qr:*** Código qr leído por el usuario con su dispositivo móvil

***ln***: Idioma del dispositivo móvil o usuario

***id*****:** Identificador único del usuario (login, teléfono, etc)

***alg***: Algoritmo a usar

***server***: Servidor del entorno en el que se encuentra el usuario

**Llamada General:**

**/rest/Evacuation?qr=<qr>&ln=<ln>&id=<id>&alg=<alg>&server=<server>**

**Retorno:**

```
<?xml version="1.0" encoding="UTF-8"?>
<evacuation>
<date>[date]</date>
<login>[login]</login>
<actions>
        <action op=[x] qrstart=[qr] qrend=[qr]>[Action 1]</action>
        <action op=[y] qrstart=[qr] qrend=[qr]> [Action 2] </action>
        <action op=[z] qrstart=[qr] qrend=[qr]> [Action N] </action>
</actions>
</evacuation>
```

**Ejemplo:**

http://www.global.com/rest/Evacuation?qr=A1b2C3&ln=es&id=902202122&server=192.168.1.1

**Acción:**

Dar información de un tramo cortado / con fuego / etc etc . Por donde no se puede pasar o no es aconsejable

**Verbo:**

POST

**Parámetros:**

```
<?xml version="1.0" encoding="UTF-8"?>
<evacuation>
<login> [login] </login>
<startnode>[qrStartNode]</startnode>
<endnode>[endNode]</ennode>
<server>[server]</server>
</evacuation>
```

**Llamada General:**

/rest/EVACUATION

**Retorno:**

200 = OK

!200 = Error

**Ejemplo:**

http://www.global.com/rest/Evacuation

## Client

**Acción:**

**"Registrar" un cliente en el servidor global (Dar de alta para que pueda recibir notificaciones) y registrarlo en el sistema global**

**Verbo:**

**POST**

**Parámetros:**

```
<?xml version="1.0" encoding="UTF-8"?>
<register>
<op>0</op>
<login> [login] </login>
<name> [name] </name>
<lastname> [lastname] </lastname>
< familyphone>[familyphone]</ familyphone>
<phoneid>[phoneid]</phoneid>
<operatingsystem>[os]</operatingsystem>
</register>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<register>
<op>1</op>
<identificador> [login] </identificador>
<phoneid> [phoneid] </phoneid>
<server>[server]</server>
</register>
```

**Llamada General:**

**/rest/Client**

**Retorno:**

**XML que representa un plan de evacuación estándar**

**Ejemplo:**

**http://www.global.com/rest/Client**

**http://192.168.1.1/rest/Client?id=12345678910111&ln=es**

**Acción:**

**“Eliminar” un cliente en el servidor global (Dar de baja un cliente para que no reciba más notificaciones)**

**Verbo:**

**DELETE**

**Parámetros:**

***login**:* Teléfono

***server***: Servidor local donde se va a encontrar

**Llamada General:**

**/rest/Client?id=[id]&server=[server]**

**Retorno:**

**200 = OK**

**!200 = Error**

**Ejemplo:**

**http://www.prueba.com/rest/Client**

**http://192.168.1.1/rest/Client**

**Acción:**

**Obtener información de un cliente**

**Verbo:**

**GET**

**Parámetros:**

***id:*** Android/Iphone id

***op:*** Código de operación

**Llamada General:**

**/rest/Client?id=<id>&op=<op>**

**Retorno:**

**op = 0 -> Información personal**

```
<?xml version="1.0" encoding="UTF-8"?>
<client>
<name>[name]</name>
<lastname>[lastname]</lastname>
<familyphone> [familyphone] </ familyphone >
</client>
```

**Ejemplos:**

http://www.global.com/rest/Cliente?op=0&id=123456

## Log

**Acción:**

Añadir nueva entrada al LOG

**Verbo:**

POST

**Entrada:**

```
<?xml version="1.0" encoding="UTF-8"?>
<log>
<op>[op]</op>
<login>[login]</login>
<text>[text]</text>
<server> [server] </server>
</log>
```

**Llamada General:**

/rest/Log

**Retorno:**

XML que representa un plan de evacuación estándar

**Ejemplo:**

http://www.global.com/rest/Log

**Acción:**

Eliminar el log de un usuario

**Verbo:**

DELETE

**Parámetros:**

login: identificador del usuario (teléfono)

server: [server] (Si es vacío borrar todo el log)

**Llamada General:**

/rest/Log

**Retorno:**

200 = OK

!200 = Error

**Ejemplo:**

http://www.global.com/rest/Log

**Acción:**

Obtener el log de un usuario

**Verbo:**

GET

**Parámetros:**

login: identificador del usuario (teléfono)

sdate : fecha de inicio (dd-mm-yyyy)

edate : fecha de fin (dd-mm-yyyy)

stime : hora de inicio

etime : hora de fin

server: para un server en concreto (Opcional)

**Llamada General:**

/rest/Log?login=<login>&sdate=<sdate>&edate=<edate>&stime=<stime>&etime=<etime>&server=192.168.1.1

**Retorno:**

```
<?xml version="1.0" encoding="UTF-8"?>
<login>[login]</login>
<actions>
        <action date=<date> time=<time>LogAction1</action>
        <action date=<date> time=<time>LogActionN</action>
</actions>
```

**Ejemplo:**

http://www.prueba.com/rest/Log?login=12345&sdate=10-10-2000&edate=10-10-2000&stime=10:00&etime=11:00&server=192.168.1.1

## Status

**Acción:**

**Indicar el estado de un usuario: a salvo, que necesita ayuda, etc**

**Verbo:**

**POST**

**Tipos:**

Estar a salvo (Código de operación = 0)

```
<?xml version="1.0" encoding="UTF-8"?>
< status>
<op>0 </op>
<login> [login] </login>
< /status>
```

Necesita ayuda (Código de operación = 1)

```
<?xml version="1.0" encoding="UTF-8"?>
< status>
<op>1 </op>
<login>[login]</login>
</ status>
```

Actualizar código qr

```
<?xml version="1.0" encoding="UTF-8"?>
< status>
<op>1 </op>
<login>[login]</login>
<qr>[qr] </qr>
< /status>
```

**Llamada General:**

**/rest/Status**

**Retorno:**

**200 = OK**

**!200 = Error**

**Ejemplo:**

http://www.global.com/rest/Status

**Acción:**

Obtener el estado de la situación general de emergencia

**Verbo:**

GET

**Parámetros:**

server= <server>

op = <op>

- ➔ Op = 0 -> Obtener TODOS los usuarios que aún siguen dentro
  /rest/Status?op=0&server=192.168.1.1
- ➔ Op = 1 && login=<login> -> Obtener el estado de un usuario en concreto
  /rest/Status?op=0&login=123456&server=192.168.1.1
- ➔ Op = 2 -> Obtener todos los usuarios a salvo
  /rest/Status?op=2&server=192.168.1.1

**Llamada General:**

/rest/Status?op=<op>[&login=<login>]&server=<server>

**Retorno:**

```
<?xml version="1.0" encoding="UTF-8"?>
<clients>
        <client>[client1]</client>
        <client>[clientN]</client>
</clients>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<information>
    <qr>[qr]</qr>
    <safe>[salvado]</safe>
    <help>[help]</help>
</information>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<clients>
        <client>[client1]</client>
        <client>[clientN]</client>
</clients>
```

**Ejemplo:**

http://www.global.com/rest/Status?op=0&server=192.168.1.1

## Server

**Acción:**

Dar de alta un servidor local en el servidor global

**Verbo:**

POST

**Entrada:**

```
<?xml version="1.0" encoding="UTF-8"?>
<server>
<id>[ip address or domain]</id>
<name>[name]</name>
<location>[location]</location>
</server>
```

**Llamada General:**

/rest/Server

**Retorno:**

XML que representa un plan de evacuación estándar

**Ejemplo:**

http://www.global.com/rest/Server

**Acción:**

Obtener información de un servidor

**Verbo:**

GET

**Parámetros:**

server: <server>

**Llamada general:**

/rest/Server?server=192.168.1.1

**Retorno:**

```
<?xml version="1.0" encoding="UTF-8"?>
<server>
<id>[ip address or domain]</id>
<name>[name]</name>
<location>[location]</location>
</server>
```

**Ejemplo:**

http://www.global.com/rest/Server?server=192.168.1.1

**Acción:**

Dar de baja un servidor local del servidor global

**Verbo:**

DELETE

**Parámetros:**

server: <server>

**Llamada general:**

/rest/Server

**Retorno:**

200 = OK

!200 = Error

**Ejemplo**

http://www.globa.com/rest/Server

## HealthProfile

**Acción:**

Registrar el perfil sanitario de un usuario

**Verbo:**

POST

**Entrada:**

```
<?xml version="1.0" encoding="UTF-8"?>
<health>
<login>[login]</login>
<sip>[sip]</sip>
<bloodgroup>[bloodgroup]</bloodgroup>
<allergy>[allergy]</allergy>
<antico>[antico]</antico>
<breathdes> [breathdes ]</ breathdes >
<info>[info]</info>
</health>
```

**Llamada General:**

/rest/Server

**Retorno:**

200 = OK

¡200 = Error

**Ejemplo:**

http://www.global.com/rest/Health

**Acción:**

Obtener el perfil sanitario de un usuario

**Verbo:**

GET

**Parámetros:**

login: <login>

**Llamada general:**

/rest/Health?login=123456

**Retorno:**

```
<?xml version="1.0" encoding="UTF-8"?>
<health>
<login>[login]</login>
<sip>[sip]</sip>
<bloodgroup>[bloodgroup]</bloodgroup>
<allergy>[allergy]</allergy>
<antico>[antico]</antico>
<breathdes> [breathdes ]</ breathdes >
<info>[info]</info>
</health>
```

**Ejemplo:**

http://www.global.com/rest/Health?login=123456

# Anexo II

## NOTIFICACIONES HACÍA EL CLIENTE

**Código:** 0
**Concepto:** Representa que el plan de evacuación asociado indica que el sistema **NO** está en estado de evacuación/emergencia, si no en modo entrenamiento

```
<?xml version="1.0" encoding="UTF-8"?>
<op>0<op>
<text>[text]</text>
```

**Código:** 1
**Concepto:** Representa que el plan de evacuación asociado indica que el sistema **SI** está en estado de evacuación/emergencia.

```
<?xml version="1.0" encoding="UTF-8"?>
<op>1<op>
<text>[text]</text>
```

**Código:** 2
**Concepto:** Representa que el servidor está enviando notificaciones ajenas a contextos de emergencia: Información de interés, publicidad, novedades, etc

```
<?xml version="1.0" encoding="UTF-8"?>
<op>2<op>
<text>[text]</text>
```

# Anexo III

## ESPECIFICACIÓN DE LAS CLASES CONCEPTUALES

| **Clase** | **Actor** |
|---|---|
| **Descripción** | Clase que representa el concepto de elemento que interactúa con el sistema. Dicha clase tiene varias subclases, una por cada tipo de actor, y está relacionada principalmente con la clase conceptual “Servidor”. |
| **Superclase** | --- |
| **Relacionadas** | Servidor |

| **Clase** | **Usuario** |
|---|---|
| **Descripción** | Subclase de la clase Actor que representa al actor más común dentro del sistema, el usuario final. Dicha clase está relacionada con un plan de evacuación (que puede cambiar) y un perfil sanitario. |
| **Superclase** | Actor |
| **Relacionadas** | Planes de Evacuación, Perfil Sanitario |

| **Clase** | **Efectivo Respuesta** |
|---|---|
| **Descripción** | Subclase de la clase Actor que representa al actor identificado como personal externo autorizado. Es decir, actores como policías, bomberos, etc. que únicamente formarán parte del sistema cuando se encuentre el mismo en estado de emergencia. |
| **Superclase** | Actor |
| **Relacionadas** | --- |

| **Clase** | **Administrador Local** |
|---|---|
| **Descripción** | Subclase de la clase Actor que representa al actor que tiene la tarea de gestionar una entidad. |
| **Superclase** | Actor |
| **Relacionadas** | --- |

| Clase | Administrador Global |
|---|---|
| Descripción | Subclase de la clase Actor que representa al actor que tiene la tarea de poder gestionar todo el sistema al completo. Desde entidades a usuarios pasando por configuración y recopilación de información del sistema. |
| Superclase | Actor |
| Relacionadas | --- |

| Clase | Perfil Sanitario |
|---|---|
| Descripción | Clase que representa toda la información médica relacionada asociada a un actor del tipo Usuario. |
| Superclase | --- |
| Relacionadas | Usuario |

| Clase | Servidor |
|---|---|
| Descripción | Clase que representa el concepto de elemento que se encarga de almacenar información, realizar tareas, facilitar el acceso a la información a los distintos actores, etc. |
| Superclase | --- |
| Relacionadas | Actor, Log, Planes de Emergencia |

| Clase | Servidor Local |
|---|---|
| Descripción | Clase que representa el concepto de servidor local. Será donde esté almacenada toda la información. Dicho servidor será gestionado por un Administrador Local (y ocasionalmente uno Global) y será usado por cualquier usuario. |
| Superclase | Servidor |
| Relacionadas | Organización, Servidor Global |

| Clase | Servidor Global |
|---|---|
| Descripción | Clase que representa el concepto de servidor global. Dicha clase conceptual deberá mostrar cómo es posible gestionar toda la información de todos los servidores locales: tanto sus datos, como su información, como su gestión de emergencia. |
| Superclase | Servidor |
| Relacionadas | Servidor Local |

| Clase | Organización |
|---|---|
| Descripción | Esta clase representa el organismo que usa el sistema: un hotel, una empresa privada, un organismo público, etc. Dicho organismo está relacionado con un servidor local, que en la práctica pueden ser uno o varios, y tiene asociado uno o varios emplazamientos. |
| Superclase | --- |
| Relacionadas | Servidor Local, Emplazamiento |

| Clase | Emplazamiento |
|---|---|
| Descripción | Representa conceptualmente una zona o posición física asociada a una entidad/organización. Está clase posibilita representar organizaciones distribuidas en varias localizaciones. |
| Superclase | --- |
| Relacionadas | Servidor Local, Emplazamiento |

| Clase | Log |
|---|---|
| Descripción | La clase log representa conceptualmente toda la gestión o monitorización del sistema. Es decir, registrar todo lo que se hace en el sistema: desde que usuario se ha registrado, hasta cuando se notifica una emergencia.<br>Debido a que este concepto es ambiguo, conceptualmente esta clase estará compuesta de eventos que pueden ser de diversos tipos. |
| Superclase | --- |
| Relacionadas | Servidor |

| Clase | Eventos |
|---|---|
| Descripción | Clase que representa conceptualmente una acción a registrar en el sistema. Esta clase conceptual se debería dividir en subclases de cara a poder representar cualquier situación a monitorizar, ya que no es lo mismo los datos a guardad cuando un usuario se registrar a cuando un bombero pide información. |
| Superclase | --- |
| Relacionadas | --- |

| Clase | Planes de Emergencia |
|---|---|
| Descripción | Clase que representa el concepto de plan de emergencia. Es decir, la definición de alguna u otra manera de cómo actuar ante una situación de emergencia. Dicha clase tiene como elementos principales elementos de localización y otra información contextual: zonas habilitadas o inhabilitadas, zonas de escaleras, etc. |
| Superclase | --- |
| Relacionadas | Servidor, Planes de Evacuación |

| Clase | Localización |
|---|---|
| Descripción | Sirve para comprender, dentro del contexto de los planes de emergencia, una posición concreta que tiene cierta relevancia. En este caso más concreto, una situación definida en el plan de emergencia pero a usar por el usuario final como es la posición inicial desde la que parte un usuario. Hemos usado código qr para determinar dicha posición inicial pero en la fase de diseño esta clase puede tener varias subclases: posicionamiento en interiores, localización GPS, etc. |
| Superclase | --- |
| Relacionadas | --- |

| Clase | Código QR |
|---|---|
| Descripción | Subclase de Localización y que representa una posición concreta dentro del plan de emergencia. El usuario usa estos códigos qr como posición inicial y para calcular así una ruta de evacuación hasta la salida que está representada por otro código qr. |
| Superclase | Localización |
| Relacionadas | --- |

| Clase | Información Contextual |
|---|---|
| Descripción | Clase conceptual que forma parte de un plan de emergencia y que representa información contextual que puede ser útil tanto para generar planes de evacuación como para actuar correctamente por parte de los equipos de respuesta. |
| Superclase | --- |
| Relacionadas | --- |

| Clase | Planes de Evacuación |
|---|---|
| Descripción | Clase conceptual que representa un conjunto de elementos que definen una serie de acciones a seguir por un usuario para ser evacuado correctamente en una situación de emergencia.<br>Un plan de evacuación es un conjunto de localizaciones a seguir por el usuario junto con un conjunto de información contextual. |
| Superclase | --- |
| Relacionadas | Planes de emergencia, Usuario |

| Clase | Acciones |
|---|---|
| Descripción | La clase Acciones es una clase conceptual que representa que debe hacer un usuario en cada plan de evacuación. Dicha clase en una fase de diseño puede dividirse y especificarse aún más. Aunque en la práctica se usarán sencillas acciones de texto puede entenderse como un video, audio o cualquier otro elemento que pueda ayudar en un plan de evacuación. |
| Superclase | --- |
| Relacionadas | Planes de evacuación |

# Anexo IV

## MODELADO DE DATOS

### Servidor Local

#### Nueva emergencia

Modelo (instanciado en XML) que enviará el usuario al servidor a través de la API de servicios cuando quiera notificar de una emergencia.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Emergency">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:date" name="date"/>
                <xs:element type="xs:string" name="login"/>
                <xs:element type="xs:string" name="location"/>
                <xs:element type="xs:string" name="description"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

#### Obtener plan de evacuación

Se corresponde con la estructura del fichero que recibirá el usuario y que representa un plan de evacuación a seguir.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Evacuation">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="login"/>
                <xs:element name="actions">
                    <xs:complexType id="actions">
                        <xs:sequence minOccurs="1" maxOccurs="unbounded">
                            <xs:element name="action" type="xs:string" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Notificar de variación contextual**

Este modelo representa el esquema de un fichero XML que debería enviar un usuario al servidor para notificar de que hay algún elemento del contexto/entorno que ha variado: un camino cortado, una zona por donde no se puede volver, etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Context">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="login"/>
                <xs:element type="xs:string" name="startnode"/>
                <xs:element type="xs:string" name="endnode"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Dar de alta un cliente en un servidor local**

Estructura de datos a enviar al servidor local por parte de un cliente para notificar que el cliente forma a partir de ahora parte del sistema. Y que deberá ser notificado si se declara una emergencia o cuando se considere necesario.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Register">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="login"/>
                <xs:element type="xs:string" name="id"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Insertar información en el Log**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Log">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="login"/>
                <xs:element type="xs:string" name="text"/>
                <xs:element type="xs:string" name="op"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Obtener información de Log de un usuario**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="LogClient">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="login"/>
                <xs:element name="actions">
                    <xs:complexType id="actions">
                        <xs:sequence minOccurs="1" maxOccurs="unbounded">
                            <xs:element name="actions" type="xs:string"  />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Notificar que un usuario está a salvo o necesita ayuda**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Client">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="login"/>
                <xs:element type="xs:integer" name="op"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Obtener todos los usuarios que no están a salvo**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Clients">
        <xs:complexType id="actions_log">
            <xs:sequence minOccurs="1" maxOccurs="unbounded">
                <xs:element name="client" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Obtener todos los usuarios que  están a salvo**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Clients">
        <xs:complexType id="actions_log">
            <xs:sequence minOccurs="1" maxOccurs="unbounded">
                <xs:element name="client" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Obtener el estado de un usuario en concreto**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Clients">
        <xs:complexType id="actions_log">
            <xs:sequence>
                <xs:element name="lastqr" type="xs:string" />
                <xs:element name="safe" type="xs:string" />
                <xs:element name="help" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

## Servidor Global

### Nueva emergencia

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Emergency">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="login"/>
                <xs:element type="xs:string" name="location"/>
                <xs:element type="xs:string" name="description"/>
                <xs:element type="xs:string" name="server"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

### Notificar de variación contextual

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Context">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="login"/>
                <xs:element type="xs:string" name="startnode"/>
                <xs:element type="xs:string" name="endnode"/>
                <xs:element type="xs:string" name="server"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

### Dar de alta un cliente en un servidor local a través del servidor global

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Register">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="login"/>
                <xs:element type="xs:string" name="id"/>
                <xs:element type="xs:string" name="op"/>
                <xs:element type="xs:string" name="server"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

### Insertar información en el Log

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Log">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="login"/>
                <xs:element type="xs:string" name="text"/>
                <xs:element type="xs:string" name="op"/>
                <xs:element type="xs:string" name="server"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

### Dar de alta un servidor local en un servidor global

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="SetLocal">
        <xs:complexType id="actions_log">
            <xs:sequence>
                <xs:element name="id" type="xs:string" />
                <xs:element name="name" type="xs:string" />
                <xs:element name="location" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

### Registrar nuevo usuario

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="SetLocal">
        <xs:complexType id="actions_log">
            <xs:sequence>
                <xs:element name="login" type="xs:string" />
                <xs:element name="name" type="xs:string" />
                <xs:element name="lastname" type="xs:string" />
                <xs:element name="familyphone" type="xs:string" />
                <xs:element name="op" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Obtener información de un cliente**

Estructura del fichero que se devolverá a los servicios de emergencias o usuario que solicite dicha información.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="Client">
        <xs:complexType>
            <xs:sequence>
                <xs:element type="xs:string" name="name"/>
                <xs:element type="xs:string" name="lastname"/>
                <xs:element type="xs:string" name="familyphone"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

**Dar de alta/obtener perfil sanitario**

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="SetLocal">
        <xs:complexType id="actions_log">
            <xs:sequence>
                <xs:element name="login" type="xs:string" />
                <xs:element name="sip" type="xs:string" />
                <xs:element name="bloodgroup" type="xs:string" />
                <xs:element name="allergies" type="xs:string" />
                <xs:element name="anticoagulant" type="xs:string" />
                <xs:element name="breathdes" type="xs:string" />
                <xs:element name="info" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

# Anexo V

## MANUAL DEL ADMINISTRADOR

De cara a poder implantar y usar todo el sistema mostrado en esta tesis, se hace necesario definir los pasos a seguir. Para facilitar en la medida de lo posible a cualquier administrador la puesta en funcionamiento del sistema.

Los pasos que se muestran a continuación son los pasos necesarios y en orden para poner en funcionamiento todo el sistema. La parte de la aplicación de usuario (Android, iPhone) se considera relativamente fácil y no precisa de manual. Ya que basta con instalarla como cualquier otra aplicación y usarla, siendo relativamente fácil debido a la amigable interfaz.

### 0 – Requisitos Técnicos

Para poder usar todo este sistema, se presupone que se dispone de la siguiente tecnología y que está en funcionamiento:

- **Tomcat 7.0 o superior**: Debido que tanto los servicios Web como las librerías usadas para el uso de los algoritmos basados colonias de hormigas están programados en Java, se ha usado Tomcat, de Apache. Con los cambios adecuados podría adaptarse para Glassfish de Sun Microsystem (Oracle).
- **Java 1.6.20 o inferior:** El paquete de servicios web está compilado con la versión 1.6.20 de Java, por lo que una versión superior daría problemas de compatibilidad y no lograría hacer funcionar bien dichos servicios.
- **MySQL:** Alguna base de datos, ya sea en el mismo servidor o en otro externo, sobre la que implementar el diseño del esquema relacional que se puede ver en la tesis complementaria.

A partir de aquí, presuponemos que el servidor Tomcat está en funcionamiento y que tenemos las tablas del esquema de base de datos ya creadas.

### 1 – Generación del Contexto de Emergencia

Haciendo uso de la herramienta web vista en el capítulo de implementación, el administrador (o persona responsable) deberá diseñar al completo el contexto de emergencia para todas y cada una de las zonas del edificio o zona donde se quiere gestionar la emergencia: añadidos los costes en las aristas, añadidas las acciones en las aristas y definir que nodo o nodos son considerados como nodos finales.

También es posible repartir dicha tarea entre varias personas y después que una única persona importe todos los diseños en uno solo.

Una vez que el diseño está completo los pasos a seguir son:

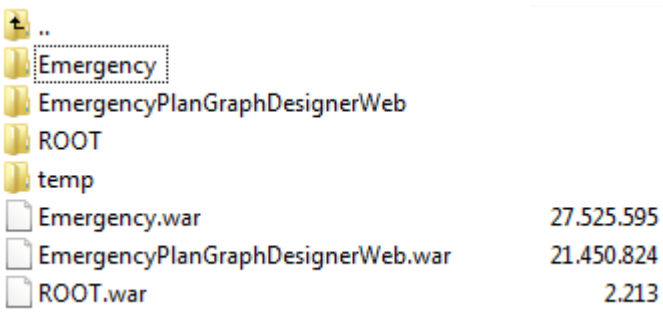
- **Guardar el proyecto:** Aunque no es obligatorio si es muy recomendable, de esta manera en un futuro, si se habilita o se añade una nueva zona se podrá modelar a partir de ese diseño sin tener que empezar desde nuevo. Para realizar esta tarea hay que pulsar en el menú superior en **Project->Save**.
- **Generar imagen de los códigos QR:** Otra opción recomendable es exportar las imágenes de los códigos QR que hay en el diseño. De esta manera se pueden imprimir y pegar directamente en las zonas, facilitando así la tarea al personal de mantenimiento. Para realizar esta tarea hay que pulsar en el botón inferior del panel izquierdo, el que tiene el icono 
- **Generar la implementación del diseño:** Esta parte si es obligatoria para el funcionamiento del sistema. Pulsando en el tercer botón del panel izquierdo  , se generar una base de datos orientada a grafos en un fichero .zip (database.zip a partir de ahora). El cual contiene la implementación de nuestro diseño con todos los datos. La guardamos para usarla posteriormente.

## 2 – Carga de los servicios Web

En la página oficial del proyecto o en cualquier otro sitio será posible descargarse el paquete de servicios que dará soporte a toda la funcionalidad del sistema. El nombre de este paquete es **Emergency.war.** Una vez que lo tengamos, independientemente de donde lo hayamos descargo u obtenido, tenemos que acceder mediante algún cliente FTP o cualquier otro gestor de ficheros web a la FTP de nuestro servidor, el cual ya tiene Tomcat.

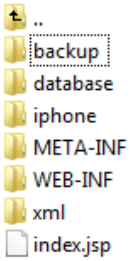
Una vez accedido a dicha FTP copiaremos dicho paquete de servicios, **Emergency.war,** en el directorio **webapps** de Tomcat.

Una vez hecho esto reiniciaremos el servidor Tomcat para ver como aparece una carpeta Emergency en dicho directorio.



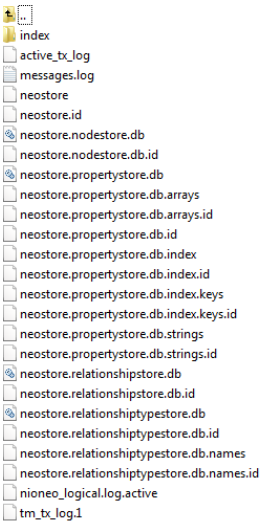
## 3 – Configuración del Sistema

En la carpeta webapps/Emergency, en nuestro servidor, tendremos una estructura similar a la siguiente:




En este directorio hay que realizar las siguientes tareas:

➔ Extraer el contenido de nuestra implementación del diseño (database.zip) y copiar su contenido tanto en la carpeta **backup.** De esta manera cada vez que se declare una emergencia toda la base de datos se copiar al directorio **database**, que será donde se modificará en función del contexto de la emergencia. Así cada vez que se declara una emergencia la base de datos sobre la que se actúa es la original.




➔ En la carpeta **XML** hay un fichero llamado **database.xml** que debemos modificar. En este fichero hay que especificar la cadena de conexión para la base de datos relacional así como el usuario y contraseña de la misma.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <url>jdbc:mysql://50.31.138.79:3306/angelr0_emergency</url>
    <user>usuario</user>
    <password>password</password>
</root>
```

➔ En la carpeta **iphone** hay que subir el certificado de Apple para poder enviar notificaciones push. Dicho fichero debe llamarse **clave.p12**


Es importante mantener el nombre de directorios y ficheros para mantener la integridad y el buen funcionamiento del sistema.

## 4 – Uso del Sistema

Una vez realizados todos los pasos anteriores, el sistema ya está listo para usar. A partir de ahora se podrán usar todas las funcionalidades implementadas en las aplicaciones para dispositivos móviles especificando únicamente desde el dispositivo móvil la dirección del servidor. Por lo general dicha dirección será un dominio fácil de recordar o vendrá representado por un código QR.