UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Departamento de Sistemas Informáticos y Computación

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

Tesis Final de Máster

Asistente Personal de Seguridad: Cliente para Iphone

Javier Flores Font

Director de Tesis: José Hilario Canos Cerdá

27 de Septiembre de 2012

RESUMEN

Este trabajo de investigación presenta el análisis y diseño de un sistema de información para la gestión de emergencias. El sistema dará soporte al almacenamiento de la información y de la gestión de toda la información referente al contexto de las emergencias.

Para hacer uso de esta información se accederá a la misma por servicios web diseñados e implementados por la tesis complementaria a esta.

Con el fin de probar estos servicios web, se han diseñado e implementado dos aplicaciones para iPhone y iPad que utilizan dichos servicios.

Ambas aplicaciones han sido estudiadas y verificadas por varios psicólogos pertenecientes al Observatorio Psicosocial de Recursos en Situaciones de Desastre (OPSIDE) perteneciente a la Universidad Jaume I de Castellón.

Agradecimientos:

Primero de todo me gustaría dar las gracias a mi director de tesis de master José Hilario Canos Cerda por ofrecerme este trabajo de investigación y por darme soporte cuando lo necesitaba.

He de expresar mi profundo agradecimiento también al grupo de psicólogos pertenecientes al Observatorio Psicosocial de Recursos en Situaciones de Desastre (OPSIDE) perteneciente a la Universidad Jaume I de Castellón por mantener varias reuniones con nosotros las cuales me han servido de gran ayuda a la hora de darle un mayor grado de usabilidad a la aplicación.

Agradecer también el trabajo desarrollado por Ángel Ruiz Zafra en su tesis de master puesto que, sin la infraestructura de los servidores desarrollada, esta tesis no tendría sentido.

Por último, en el apartado personal, mi gratitud a mi familia por apoyarme en todo momento para seguir adelante.

ÍNDICE:

Sección 1: Introducción

I. Descripción

La historia está marcada por sucesos en los que una multitud de personas, ante una situación crítica de emergencia (incendio, explosión, derrumbe, actuaciones masivas desordenadas de individuos sin aparentes motivos que las justifiquen...), provocan con su comportamiento que dicha emergencia no se resuelva de la mejor forma esperada. Especialmente en caso de incendios y terremotos, el número de víctimas tiende a ser elevado, no tanto como consecuencia directa del desastre sino, sobre todo, a causa de la actuación inadecuada de las personas involucradas.

Al sentir el temblor de un seísmo o al percatarse de la presencia de humo y fuego, las personas se ponen muy nerviosas y siguen su instinto de conservación, comienzan a correr hacia las escaleras empujándose de tal manera, que las salidas quedan bloqueadas y por tanto, la mayoría de victimas pierden la oportunidad de salvarse. Este nerviosismo normalmente es acentuado debido a que los desastres suelen presentarse sin previo aviso, de modo sorpresivo, y en muchas ocasiones la probabilidad de salvación ya no depende de ellos.

Si bien es cierto que existen varios protocolos de gestión de emergencias conocidos y estandarizados ,por lo general, la mayoría de ellos, son tan simples como indicarle al usuario por medio de señales la dirección de la salida más próxima. Este modo de actuar frente a una emergencia puede resultar útil en espacios pequeños, donde rápidamente encontraríamos la salida en cuestión de minutos. Pero, ¿Qué ocurre si se produce una emergencia en edificios con más de 10 plantas, naves industriales, centros comerciales, hoteles de grandes dimensiones, etc.?. Y es mas, ¿Qué ocurriría si siguiendo las indicaciones, llegamos a un punto afectado por el cual no podemos continuar?

Además, uno de los errores más grande a destacar que poseen los protocolos de evacuación actuales, es no considerar a las personas discapacitadas. Consideran las mismas medidas para todo el mundo, y esto es un inconveniente ya que, su discapacidad puede impedirles bajar escaleras e incluso moverse del lugar.

Por tanto, podríamos llegar a la conclusión de que hay varios condicionantes a la hora de que una persona realice correctamente el plan de emergencia:

- La ubicación de dicha persona. (En que lugar se encuentra y que tramos están afectados).
- El estado físico de la persona. (invalidez, diabetes, si posee alguna enfermedad respiratoria ,etc..).
- El estado psicológico de la persona. (Esta asustada, nerviosa, ...etc.).

Llegados a este punto, podríamos hacernos la idea de que cada persona debería de tener su propio plan de evacuación en caso de emergencia, y esto es algo que no es considerado por los actuales protocolos de evacuación ya que este aspecto era inviable e impensable años atrás.

Los grandes avances informáticos en la última década junto con el auge de la telefonía móvil, ha abierto la puerta a la posibilidad de crear un sistema de emergencias capaz de manejar información tal como, localización de las personas, perfil sanitario de las mismas, etc. con el fin de proporcionar alertas de emergencias , planes de evacuación individuales, posibilidad de monitorizar a los afectados, etc.

II. Motivación

La evolución de los últimos años en el ámbito de la investigación junto con la creciente evolución tecnológica ha posibilitado que dichas líneas de investigación puedan tener unos objetivos distintos sobre la investigación inicial y puedan ser aplicables en otros contextos gracias a la expansión, facilidad de uso y potencia asociada a dicha evolución tecnológica.

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 Producción de Software para generar una nueva rama llamada Líneas de Producción de Documentos[X].

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.

Esta rama con este mismo objetivo 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. De esta manera se evitaría la tediosa y actual 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.

Este contexto aplicado y haciendo uso de nuevas tecnologías como dispositivos móviles de última generación posibilitaría crear un sistema completo de gestión de emergencias en tiempo real. Debido a que actualmente cada vez más gente usa Smartphones, 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. Consiguiendo poner a salvo al usuario del mismo.

III. Objetivo

Las tecnologías inalámbricas han tenido mucho auge y desarrollo en estos últimos años. Una de las que ha tenido un gran desarrollo ha sido la telefonía móvil. Desde sus inicios a finales de los 70 ha revolucionado enormemente las actividades que realizamos diariamente. Los teléfonos móviles se han convertido en una herramienta primordial para la gente común y de negocios; las hace sentir más seguras y las hace más productivas.

A pesar de que la telefonía móvil fue concebida estrictamente para la voz, la tecnología móvil de hoy es capaz de brindar otro tipo de servicios, como datos, audio y video con algunas limitaciones.

No solo se a avanzado desde el punto de vista del hardware, además, se han desarrollado multitud de sistemas operativos, así como aplicaciones para los mismos.

IOS es uno de los sistemas operativos para teléfonos móviles (iPhone) y tablets (iPad) más comúnmente utilizados hoy en día. Uno de sus puntos fuertes es la AppStore. Desde que Apple lanzara su tienda virtual de aplicaciones, ésta no ha parado de crecer día a día, añadiendo nuevos contenidos a diario, tanto juegos como aplicaciones de productividad y utilidades diversas.

Existen una serie de aplicaciones en la AppStore que nos resultarían de gran ayuda en caso de que se declarara una emergencia como es el caso de WhatsApp, Kik, Tu Me,etc... aunque únicamente se limitan a establecer comunicación rápida con el exterior vía oral o mediante mensajería instantánea.

Los objetivos básicos de este proyecto son dos: primeramente aprender un lenguaje y una tecnología de desarrollo totalmente nuevos y segundo desarrollar dos aplicaciones para esta nueva plataforma (iOS) , una que tiene como objetivo gestionar las emergencias del usuario proporcionándole planes de evacuación personalizados, comunicación vía oral y escrita con el exterior así como la de monitorizar al usuario para que así de algún modo , en la segunda aplicación los bomberos puedan saber en todo momento la posición exacta del mismo.

Además, se deben de garantizar una serie de requisitos de calidad del software fundamentales puesto que dicha calidad será clave a la hora de disminuir el número de heridos y víctimas mortales. A continuación mostraremos una lista de cada uno de los requisitos de calidad que requeriremos.

Requisitos de Calidad:

Corrección: La corrección es la cualidad principal. Si un sistema no hace lo que se supone que debe hacer y más en el caso de las emergencias, poco importan el resto de consideraciones que hagamos sobre él – si es rápido, si tiene una bonita interfaz de usuario…etc.

Robustez: La robustez complementa la corrección. La corrección tiene que ver con el comportamiento de un sistema en los casos previstos por su especificación; la robustez caracteriza lo que sucede fuera de tal especificación. En nuestro caso, por ejemplo supongamos que se declara una emergencia y el usuario recibe un plan de evacuación determinado. Podría darse el caso que siguiendo dicho plan de emergencia, se encontrara con una ruta afectada, ya que no había sido registrada a priori en el sistema, y no pudiera continuar el plan de evacuación. El sistema debería de poder resolver dicha incidencia de forma correcta.

Eficiencia: Es la capacidad de un sistema software para exigir la menor cantidad posible de recursos hardware, tales como tiempo del procesador, espacio ocupado de memoria interna y externa o ancho de banda utilizado en los dispositivos de comunicación. La eficiencia es uno de los factores de calidad mas importante que deberá de tener nuestra aplicación y más aun cuando nuestro hardware es un dispositivo móvil (iPhone) limitado de recursos. La eficiencia esta directamente relacionada con el tiempo de respuesta.

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. Lo que ocasionaría un incremento del tiempo de reacción del usuario.

Facilidad de uso: el sistema va a ser usado por personal ajeno a la informática. El sistema debe ser lo más amigable posible, no solo para facilitar su uso al usuario, si no que en momentos como los ocurridos durante una emergencia, el software no debe permitir la duda por parte del usuario. Dada la importancia de este factor de calidad, hemos contactado con psicólogos pertenecientes al Observatorio Psicosocial de Recursos en Situaciones de Desastre (OPSIDE) perteneciente a la Universidad Jaume I de Castellón, los cuales han dado su palabra de que nos ayudaran en este aspecto.

Funcionalidad: es el conjunto de posibilidades que ofrecerá nuestro sistema. Como hemos comentado anteriormente, proporcionar planes de evacuación a los usuarios, monitorización de los mismos, alertas y avisos de emergencias...etc.

Integridad: El software deberá proteger sus diversos datos (perfil sanitario, datos del log,etc...) contra modificaciones y accesos no autorizados.

IV. Alcance

Una vez estudiados todos los requisitos que debe reunir ambas aplicaciones, vamos a continuar haciendo una descripción del funcionamiento de las mismas:

Funcionamiento del asistente personal de seguridad:

El cliente se ejecutará en un teléfono móvil o Smartphone. Descripción de las etapas de su uso:

- **Instalación**

  Se instalará como toda aplicación para la plataforma iOS, desde la AppStore.

- **Primera ejecución**

  Al ejecutar por vez primera, el servidor global debe solicitar al usuario dos tipos de información:

  - Datos personales: nombre, apellidos, número de teléfono (para posible contacto de los equipos de respuesta) y un teléfono de un familiar o amigo al que poder comunicar en caso de necesidad.
  - Perfil sanitario (anonimizado/confidencial).

  Los datos se guardan, y pueden modificarse en cualquier momento desde el menú principal.

- **Uso habitual**

  Se supone que la aplicación estará activa en background, para recibir avisos.

  El primer paso es el establecimiento de la conexión con el servidor de seguridad (servidor global) . En ese momento, el cliente queda registrado en el servidor global y tanto sus datos personales como su perfil sanitario, datos técnicos ,etc.. podrán ser consultados por este mismo.

De forma opcional, también podríamos guardarnos la posición del usuario por defecto.

Tras la conexión, el sistema está en reposo hasta que ocurra uno de los eventos siguientes:

1) Se recibe un aviso de emergencia.
   a) La aplicación se abre de forma automática y se le informa al usuario que esta en modo emergencia y que debe posicionarse en el código QR mas cercano a el.
   b) Una vez realizada la lectura del código QR más próximo, el servidor (local o global) le devolverá el plan de evacuación más adecuado.
   c) Durante la evacuación, el usuario sigue las instrucciones de la ruta, pero en cualquier momento, éste puede pedir una nueva ruta, explicando el motivo (si se pierde, o si hay zonas impracticables) mediante los botones correspondientes. También puede pedir hablar por teléfono con alguna persona determinada.
   d) Si la detección de que una ruta es inválida la realiza otro cliente, hay que actualizar la ruta de todos los que han sido dirigidos por el tramo impracticable.

2) El usuario descubre una emergencia. En ese caso, pulsa el botón “Aviso de emergencia” en la pantalla principal .

Modos de funcionamiento del cliente:

   a) Modo en reposo o training: cuando no hay emergencia activa. Durante este tiempo, el usuario puede *jugar* con la aplicación (por ejemplo, calcular ruta desde un QR, ver instrucciones, etc.) .
   b) Modo emergencia: En este modo se le informa al usuario del problema y el plan de evacuación generado tiene en cuenta tramos afectados.

- **<u>Funcionamiento del asistente para los equipos de rescate:</u>**

La aplicación se ejecutará en un Tablet iPad. Descripción de las etapas de su uso:

- **Instalación**

  Se instalaría como toda aplicación para la plataforma iOS, desde la AppStore.

- **Primera ejecución**

  Contemplará la misma funcionalidad que su uso habitual. Los datos del personal de respuesta deben estar almacenados a priori en el servidor de emergencias global.

- **Uso habitual**

  Se supone que la aplicación estará activa en background, para recibir avisos, del mismo modo que con el cliente para los usuarios.

1) Se recibe un aviso de emergencia.
2) Se informa al equipo de respuesta que ha habido una emergencia en un edificio, indicándole el lugar.
3) Durante el trayecto, sabiendo el edificio donde se ha producido la emergencia, el bombero introduce la dirección del mismo
4) Una vez introducida la dirección se le muestra una lista que contiene información actualizada sobre los usuarios que aun residen en el edificio.
5) Si el bombero se encuentra en peligro o no sabe como salir del edificio puede solicitar un plan de evacuación como sucedía en el caso de nuestra primera aplicación.
6) Si por el contrario el bombero selecciona un afectado, la aplicación deberá mostrarle toda su información, incluyendo una opción que le permita obtener el camino más corto hacia el mismo.
7) En caso de que el bombero no encuentre al afectado bien por que se ha perdido o no lo encuentra o algún otro imprevisto tendrá la opción de pedir al servidor que le proporcione otra plan de rescate adicional.
8) Una vez el bombero consigue llegar hasta al afectado, este mismo debe de confirmar dicha acción.

Sección 2: Tecnología empleada

Tecnologia Software:

I. Estudio sobre la Plataforma

Hoy en día existen diversas plataformas que cumplen con los requisitos requeridos para llevar a cabo aplicaciones como las que se nos proponen.

En principio, ambas aplicaciones están pensadas para servir como prototipo a una posible aplicación final que se desarrolle en un futuro.

Si queremos que ambas aplicaciones sean útiles, deberemos elegir la plataforma móvil más adecuada que posea multitud de usuarios y a la vez, ofrezca los recursos necesarios para poder desarrollarlas. En la siguiente figura se observa un estudio de las plataformas moviles indicandonos la relacion existente entre el volumen de ventas, ingresos y beneficios entre los años 2011 y 2012 de cada una de ellas:

Figura 1. Economía de las plataformas móviles en la actualidad

En la figura 1 podemos observar que a día de hoy, a pesar de que hay un gran volumen de ventas por parte de Nokia,LG, HTTC, RIM , etc..., Apple y Samsung son las que generan muchos mas beneficios y según indica la figura, a la larga se impondrán al resto.

Aun partiendo de la idea de la imposición de Apple y Samsung frente al resto, a continuación vamos realizar un estudio comparativo de la mayoría de ellas:

1. Plataforma Nokia:

Nokia Corporation es una empresa transnacional , además de una de las principales empresas del sector de las telecomunicaciones. Con sede en Keilaniemi de Espoo (Finlandia), Nokia es una de las marcas más conocidas dentro y fuera de la Unión Europea.

En 2011, Stephen Elop, el nuevo patrón de Nokia, escribió un informe interno demoledor sobre la situación de la compañía. En el comentaba que Nokia estaba sobre una plataforma en llamas (Symbian) debido al éxito de Apple y Android.

Debido a esto en 2011, se alió con Microsoft para combatir a Apple y Android

El fabricante finlandés de móviles incorpora desde entonces de forma mayoritaria el sistema operativo Windows Phone 7 y el navegador Bing en sus teléfonos moviles.

Esta alianza intenta evitar que ambas compañías queden marginadas ante el éxito de sus competidores, Apple y el sistema operativo para móviles de Google, Android.

En resumen, podemos decir que Nokia a pesar de que en el pasado dominaba el mercado de los teléfonos, ha perdido cuota de mercado en la gama alta, la de los teléfonos inteligentes. Dado que nuestras aplicaciones móviles requieren de la más avanzada tecnología, por este motivo hemos descartado a Nokia.

Como dato interesante, parece que Nokia en la actualidad no se rinde y ha desarrollado una nueva plataforma llamada Nokia Belle FP1, un sistema operativo de de nueva generación que veremos que novedades trae , pero todo parece indicar que Apple y Android no tendrán rivales.

2. Windows Phone 8

Es la nueva versión de su sistema operativo móvil, para una nueva generación de equipos. Este sistema operativo a sido anunciado el 20 de Junio del 2012.

Windows Phone 8 supone un cambio de rumbo en la filosofía de Microsoft en dispositivos móviles.

Todos los dispositivos Windows Phone anteriores utilizaban un sólo núcleo y, por lo tanto, el sistema operativo estaba optimizado para utilizar sólo un core. Sin embargo, Windows Phone 8 está optimizado para tener soporte de varios núcleos (2 y 4).

Además, la llegada de esta nueva plataforma ha incorporado un aire fresco en el desarrollo de aplicaciones móviles gracias a la experiencia de uso completamente renovada que ha presentado Microsoft.

A continuación se presentan algunas de las características nuevas que ha traido este sistema operativo:

- NTKernel: Es el cambio más grande que ha tenido el OS. Así, Windows Phone 8 compartirá el mismo sistema de archivos, kernel drivers, y partes del modelo de seguridad de Windows 8. Esto significa que ahora pasar una aplicación de Windows Phone 8 a Windows 8, o vice-versa, es mucho más sencillo. Por otra parte, ahora tienen acceso directo a librerías nativas de C y C++, además de soporte DirectX y SQLite.
- Múltiples Núcleos y más cambios en hardware: Gracias al salto a la arquitectura NT, ahora Windows trae soporte para múltiples núcleos. Luego los dispositivos podrán incorporar hasta 64 cores, aunque en la actualidad con un intervalo de [2-4] cores bastaría. Por otra parte, el dispositivo tendra soporte para tres tipos de resolución. Las resoluciones seran del tipo WVGA (800×480, la que utilizaban los telefonos Windows Phone anteriores), WXGA (1280×768) y 720p (1280×720). Además, tendremos soporte para tarjetas microSD para el almacenamiento de fotos, música, video, e incluso la instalación de aplicaciones.
- Nueva interfaz: La interfaz que trae Windows Phone 8 se asemeja ahora más la versión de escritorio. La pantalla de inicio tiene ahora tres tamaño de “tiles” o cuadrículas, incluyendo una más pequeña que se asemeja más a un ícono, lo que permitirá concentrar más información en la pantalla de inicio.
- Nokia Maps:Como hemos comentado anteriormente, la alianza entre Microsoft y Nokia ha producido que Windows Phone 8 incorpore Nokia Maps como aplicación por defecto en el sistema operativo para la gestión de mapas. Nokia Maps supondra un cambio radical con respecto a su antiguo predecesor Bing Maps. Además de poseer una cobertura de nivel mundial, supondra una experiencia superior en todo y dispondremos de mapas offline.
-

Los primeros fabricantes en disponer de teléfonos con Windows Phone 8 serán Nokia, Huawei, Samsung y HTC.

Desde nuestro punto de vista, parece ser que Windows Phone 8 será una de las plataformas mobiles más competitivas pero siempre irá por detras tanto de Apple como de Android.

3. Android

Android es el sistema operativo de Google, basado en Linux. A diferencia de iOS, es de codigo abierto. Esto una gran gran ventaja con respecto a iOS que posee una serie de limitaciones en el desarrollo de aplicaciones. Por otra parte, ademas al ser software libre, es un sistema muy robusto

Con Android se puede utilizar todas las aplicaciones de Google que conocemos. En la actualidad hay disponibles en el Market más de 700.000 aplicaciones.

El 27 de Junio de 2012, Android sacó la nueva versión de su sistema operativo para móviles, Android 4.1 Jelly Bean.

A continuación se presentan las novedades que aporta dicho sistema operativo:

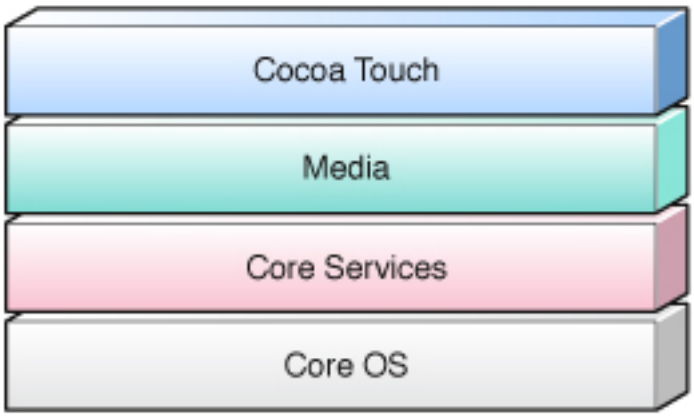
- La mejora de Ice Cream Sandwich produce que el sistema sea mas fluido y rapido.
- Se ha mejorado el dictado por voz y se ha eliminado la necesidad de estar conectado a Internet. Sin embargo, en un primer momento esta característica solo esta disponible en inglés estadounidense.
- AndroidBeam permite ahora la conexión con otros dispositivos mediante NFC con tan solo tocarlos.
- Las notificaciones también han sido mejoradas añadiendo acciones comunes para acceder de una forma más rápida, por ejemplo, a la hora de responder a una llamada perdida o consultar un evento en la agenda
- La función Google Now permitirá una mayor inmediatez a la hora de obtener resultados así como los mostrará de forma más inteligente en función de la hora y localización del usuario. Así, podrá consultar de una forma más rápida a qué hora pasa el siguiente autobús para ir a un destino concreto dependiendo del lugar donde estemos en ese momento.

4. Apple iOS

Es el sistema operativo móvil de Apple. Originalmente fue desarrollado para el iPhone, y posteriormente utilizado también en dispositivos como el iPod Touch, iPad y el Apple TV. Actualmente esta disponible la versión 5.1.

La interfaz de usuario de iOS está basada en el concepto de manipulación directa, usando gestos multitáctiles. Los elementos de control consisten de deslizadores, interruptores y botones. La respuesta a las órdenes del usuario es inmediata y provee de una interfaz fluida. La interacción con el sistema operativo incluye gestos como deslices, toques, pellizcos, los cuales tienen definiciones diferentes dependiendo del contexto de la interfaz.

El **Sistema Operativo** de los dispositivos de Apple (iOS), está formado por un conjunto de capas, que conforman el conjunto de servicios ofrecidos por el dispositivo.
Arquitectura:



- Cada capa de la arquitectura está compuesta por un conjunto de *frameworks*
- La capa **Core OS** es la base del sistema operativo. Se encarga de realizar la gestión de memoria, el sistema de ficheros, conexión a la red y procesos que interactúan con el hardware
- **Core Services** nos permite el acceso a los servicios básicos, tales como la agenda, el acceso a la base de datos, preferencias, conexión a servidores y procesamiento de URLs, etc…
- La capa **Media** nos permite la ejecución de tareas multimedia. Entre ella el acceso al Audio, OpenGL, Imágenes y PDF, Animaciones, etc…
- **Cocoa Touch** nos permite acceder al acelerómetro, los eventos y controles táctiles, la jerarquía de vistas, alertas, etc…gestiona la interacción visual con el usuario

Las caracterísiticas principales de este sistema operativo son:

Pantalla principal:

En la pantalla principal es donde se presentan la mayoría de las aplicaciones que vienen instaladas por defecto. Además en la parte inferior de dicha pantalla se encuentra el Dock donde se pueden anclar aplicaciones de uso frecuente. La pantalla tiene una barra de estado en la parte superior para mostrar datos, tales como la hora, el nivel de batería, y la intensidad de la señal.

Carpetas:

Con iOS 4 se introdujo un sistema simple de carpetas en el sistema. Podemos de esta forma organizarnos el escritorio del iPhone por carpetas, similar al escritorio del PC.

Cada carpeta tiene como limitación hasta 12 y 20 aplicaciones en el iPhone y iPad respectivamente.

Centro de notificaciones

Con iOS 5, el sistema de notificaciones se ha modificado notablemente. Podremos acceder a las notificaciones deslizando el dedo desde la barra de estado hacia abajo, de forma idéntica al sistema de notificaciones ofrecido por el sistema Android. Al hacer click sobre una notificación, el sistema abrirá la aplicación que envió la notificación.

Aplicaciones

La pantalla inicial de iOS contiene diversas aplicaciones. Puede darse el caso de que algunas de ellas esten ocultas como por ejemplo Nike + iPod.

Explicadas las diferentes plataformas y como ya sabíamos a priori, las plataformas más indicadas para el desarrollo de nuestras dos aplicaciones por su competencia en el mercado y por su probabilidad de que en un futuro sigan siendo competitivas son iOS y Android.
Dado que la tesis de Angel Ruiz Zafra incorporará una aplicación para la gestión de emergencias en los edificios basada en Android, hemos decidido implementar ambas aplicaciones en iOS, de este modo, los usuarios dispondrán de dicho sistema de emergencias en las dos plataformas mas competentes existentes hoy en dia en el mundo.

II. XCODE

Xcode es el entorno de desarrollo integrado (IDE, en sus siglas en inglés) de Apple Inc. y se suministra gratuitamente junto con Mac OS X. Xcode trabaja conjuntamente con Interface Builder, una herencia de NeXT, una herramienta gráfica para la creación de interfaces de usuario.

La versión de XCode con la que trabajaremos es la Versión 4.3.1 (4E1019).

• Xcode IDE

Esta herramienta de desarrollo permite trabajar en un ambiente como eclipse o NetBeans, permitiendo programar, compilar y depurar cualquier aplicacion que use este programa.
Además esta herramienta proporciona una gran facilidad para utilizar cualquiera de los frameworks incluidos en la API Cocoa Touch.

• Apple LLVM Compiler

Este compilador es una version del compilador GNU, un compilador de codigo abierto compatible con C,C++,C#,Objective-C, Java, Fortran, Python, Ruby,etc. De este modo, en nuestro codigo programado en Objetive-C podremos añadir funciones adicionales y metodos escritos en otro lenguaje . Esta característica sera esencial para nosotros puesto que habrá parte del codigo que requeriremos utilizar C++ en vez de objective-C.
No hay que olvidar de que dispone de un gran depurador el cual nos permite situar puntos de ruptura("breakpoints") en cualquier parte del codigo para comprobar el valor de todas las variables y que zona de la memoria estan alojadas. De esta forma encontraremos errores en nuestras aplicaciones de una forma facil y sencilla.

• Interface Builder

Es una herramienta que fue incluida en el SDK un poco después de que surgiera el XCode IDE. Anteriormente la interfaz gráfica de las aplicaciones debía de crearse de forma manual programando todo el código. Esta herramienta nos facilita esta tediosa tarea. Con la ultima versión del XCode IDE, existe un archivo llamado .Storyboard que se encarga de gestionar la navegación entre los controladores de la aplicación, asi como la creación de las vistas de cada controlador.

• Simulador de iPhone

Además de la herramientas descritas en la página anterior. El SDK de iPhone contiene un simulador integrado con el cual podemos testear y depurar el codigo de nuestras aplicaciones facilmente sin la necesidad de disponer del dispositivo físico.
Sin embargo, el simulador posee algunas limitaciones. Por ejemplo, la arquitectura usada para compilar las aplicaciones esta basada en ARM7 o ARM6 , es decir , para que las aplicaciones funcionen en un intel X86 y no en dipositivos iPhone o iPad.
Por otra parte, para depurar una aplicación de iPhone o iPad correctamente, es necesario que se use simultaneamente al simulador un depurador para encontrar los fallos. Usando

ambas herramientas juntas podemos pausar la aplicación, volver a la instrucción anterior ejecutada, etc... con el fin de detectar fallos.

• API Cocoa Touch

El SDK de iPhone mencionado anteriormente tambien incluye la API Cocoa Touch usada en Mac OS X.
Cocoa Touch incorpora una asbstracción de ella misma (Mac OS X) pero enfocado al iOS.
De este modo, nosotros tenemos acceso a las siguientes características:

1. Control de eventos y "Multi-Touch"
2. Acelerometros.
3.Estructura organizada por vistas
4. Localización
5. Cámara

Muchos de estos frameworks estan agrupados en dos framework llamados UIKit y Foundation.
En UIKit, nosotros encontramos todos los accesorios como botones, tablas, etc. los cuales son identificados por las sigas "UI" al principio ,por ejemplo "UIButton".
Por otra parte, Foundation ofrece todas las clases básicas personalizadas por Apple como NSString (tipico string en C), NSObject, y unas cuantas más. Todas ellas son identificadas por el prefijo "NS" al principio.

Por último comentar, que La API Cocoa Touch esta programada en Objective-C .

III. OBJECTIVE-C

Objective-C esta considerado como una extension de C, esto significa que nuestro compilador de objective-C puede compilar C sin ningun problema. La sintaxis es la misma que presentan los codigos orientados a objetos, usando la misma construcción de variables, expresiones, punteros, etc... que C, y recoge en gran parte el estilo de los mensajes de SmallTalk.

Objective C se basa en enviar mensajes a instancias de objetos. La llamada a los metodos de un objeto se produce enviando un mensaje a la instancia de dicho objeto. Este envio, incluye enviar el nombre del metodo del objeto y este objeto es responsable de interpretar el mensaje en tiempo de ejecución.

Para clarificar esto, se presenta el siguiente codigo tanto en C++ como en Objective-C:

C++: *object ->method (parameter);*

Objective-C: *[object method: parameter];*

Por ejemplo:

C++: *object ->count(0);*

Objective-C: *[object count:0];*

IV. GESTIÓN DE MEMORIA

Al tratarse de dos aplicaciones cuya finalidad es la de mostrar información al usuario, existen muchas posibilidades a la hora gestionar en memoria dicha información.

En nuestro caso, la aplicación trabajará con información que dependerá del contexto de la emergencia y portanto variará con el tiempo. Por tanto, esta información no precisará ser almacenada en el iPhone por ninguna base de datos sino que serán las propias instancias de los objetos las que almacenen los datos actuales.

Únicamente nos almacenaremos los datos de registro, el perfil sanitario y información sobre la configuración de la aplicación y no requeriremos de una base de datos puesto que ambas aplicaciones estan pensadas para que cada una de ellas sea utilizada por un único usuario y por tanto no habrían multiples tuplas en la base de datos.

Para almacenar los datos comentados anteriormente nos basaremos en una clase llamada NSUserDefaults la cual trabaja como un diccionario. Para un key dado, puedes proveer algunos tipos primitivos y algunos tipos de objetos como por ejemplo: NSData, NSString, NSNumber, NSDate, NSArray, o NSDictionary.

Estos datos se guardarán en disco y podrán ser accedidos a posteriori aunque la aplicación se cierre por completo.

Para guardar datos en el disco utilizando la clase NSUserDefaults basta con escribir las siguientes lineas de código:

```
1.    // Get the shared defaults object
2.    NSUserDefaults *settings = [NSUserDefaults standardUserDefaults];
3.
4.    // Save the Phone
5.    [settings setInteger:@"66523453" forKey:@"TelefonoUsuario"];
6.
7.    // Save them to disc
8.    [settings synchronize];
```

En cambio, para acceder a los valores guardados :

```
1.    // Get the settings and the phone number.
2.     NSUserDefaults *settings = [NSUserDefaults standardUserDefaults];
3.     Int x= [settings objectForKey:@"TelefonoUsuario"];
4.
5.     }
```

I. NOTIFICACIONES PUSH:

Utilizaremos la tecnología que ofrece Apple, en concreto el servidor APNS (Apple Push Notification Service). Se trata de un servicio robusto y altamente eficiente para la propagación de información a dispositivos como iPhone, iPad y iPod touch. Cada dispositivo establece una conexión IP acreditada y cifrada con el servicio y recibe notificaciones través de esta conexión persistente. Si una notificación de una solicitud llega cuando dicha aplicación no se está ejecutando, el dispositivo avisa al usuario que la aplicación tiene datos esperando por ella.

El sistema funcionará de la siguiente forma:

Nuestro servidor global al registrarse nuevos usuarios almacenará el identificador del terminal de cada uno de ellos, dicho identificador es una tira de 70 caracteres alfanuméricos, por ejemplo:

```
<a7f76aae5ff140bfbcd2ebfdc399341254d9fb6b626f35a6c21bb5e83e079f09
>
```

De esta forma en la clase "Usuario" quedarán almacenados todos los terminales de aquellos usuarios que tienen la aplicación instalada y se han registrado en ella.

Una vez disponemos de los identificadores de todos los terminales, deberemos desarrollar el código necesario para que el sistema envíe notificaciones a una lista de dispositivos.

Este código, deberá ser implementado en el servidor, luego deberemos utilizar lenguaje Java para ello.

Utilizaremos una librería de Java que permite a los desarrolladores las notificaciones push para dispositivos iOS (iPhone, iPod y iPad) a través del servicio de notificación push de Apple utilizando un conjunto simple y poderoso de herramientas. Dicha librería recibe el nombre de JavaPNS.

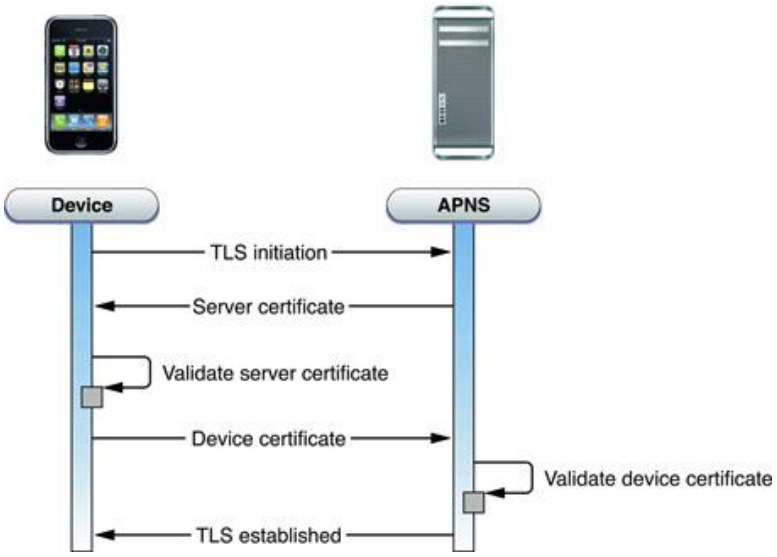
Concretamente, la función que utilizaremos de la librería JavaPNS para el envio de notificaciones será la siguiente:

```
Push.combined("texto de la notificación","código","sonido de la
alerta", "dirección clave privada", "contraseña clave
privada","key del terminal");
```

Donde el código variará dependiendo del tipo de notificación:

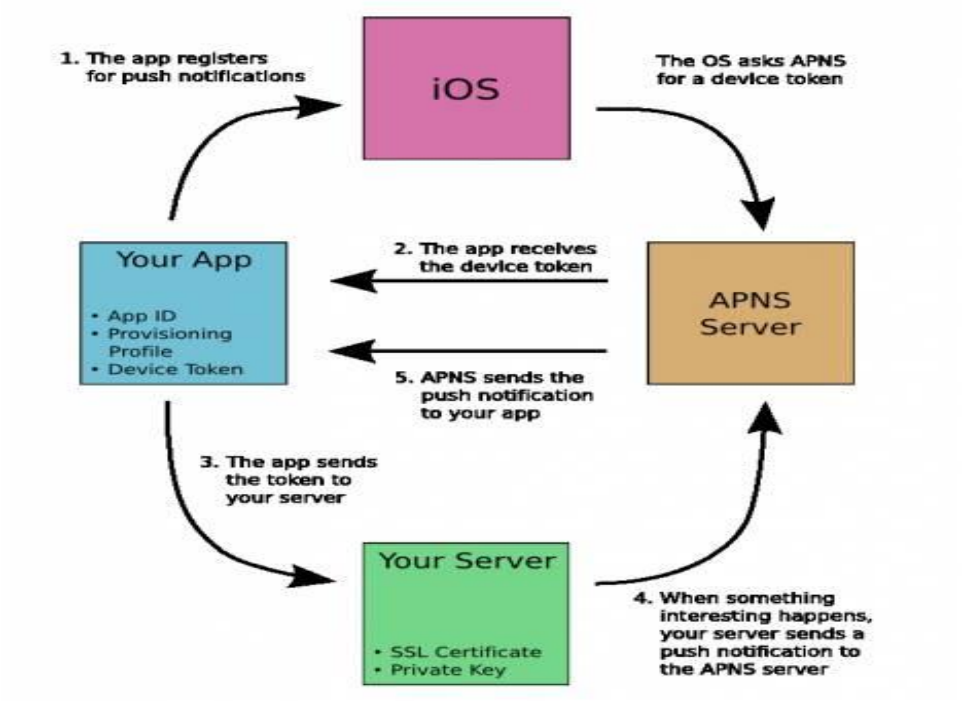
código	Descripción
0	Representa que el plan de evacuación asociado indica que el sistema NO está en estado de evacuación/emergencia, si no en modo entrenamiento.
1	Representa que el plan de evacuación asociado indica que el sistema SI está en estado de evacuación/emergencia.
2	Representa que el servidor está enviando notificaciones ajenas a contextos de emergencia: Información de interés, publicidad, novedades, etc.

Una vez puesto en funcionamiento el servidor, debemos establecer comunicación con los servidores de notificaciones Push de Apple. Todo este proceso se realiza a través del portal de desarrolladores del que dispone Apple, mediante el intercambio de certificados y el uso de la licencia correspondiente. El proceso es el siguiente:



Con este fin, en lugar de que cada aplicación se conecte al servidor de emergencias correspondiente, Apple sólo permite que exista una conexión permanente a su propio servidor, de tal forma en que las notificaciones de todas las aplicaciones llegan a través de un único hilo de comunicación, con el ahorro de energía que ello conlleva.

De forma resumida, esta es la arquitectura final que describe el funcionamiento de las APN en Apple:



Cuando nuestro dispositivo móvil realice alguna operación/consulta en alguno de los servidores (global o local) , XML será el formato establecido para intercambiar datos entre sí. De este modo, podrá haber un flujo de datos entre el dispositivo y los servicios web y viceversa.

a. FLUJO DE DATOS ENTRE SERVIDOR WEB Y DISPOSITIVO MOVIL:

El parseo de ficheros XML puede realizarse con la clase NSXMLParser y es una lectura guiada por eventos (*event-driven*).

Para realizar la lectura, hemos optado por crear una clase derivada de NSXMLParser e implementar los métodos de recepción de eventos (didStartElement y didEndElement) y manejo de datos en esa clase.

Además de derivar la clase NSXMLParser deberemos indicar que nuestra clase implementa el protocolo correspondiente, de forma que pueda recibir los eventos durante el parseo del XML.

Durante la lectura, deberemos de definir alguna variable o propiedad para almacenar los datos. En nuestro caso, por ejemplo cuando recibamos un plan de evacuación,

deberemos de tener una variable que almacene las acciones a realizar y otra que almacene el código de la acción para poder representar una imagen de ella.

A continuación podemos ver un ejemplo de ello:

```
-(void) parser:(NSXMLParser *)parser didStartElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName
attributes:(NSDictionary *)attributeDict{
    if([elementName isEqualToString:@"action"]){
        currentaction=[Action alloc];
        currentop=[NSString stringWithString:[attributeDict objectForKey:@"op"]];
    }}
-(void) parser:(NSXMLParser *)parser didEndElement:(NSString *)elementName
namespaceURI:(NSString *)namespaceURI qualifiedName:(NSString *)qName{

    if([elementName isEqualToString:@"action"]){
        currentaction.accion=currentNodeContent;
        currentaction.op=currentop;
        [self.actions addObject:currentaction];
        currentaction=nil;
        currentNodeContent=nil;   }}
```

b. FLUJO DE DATOS ENTRE DISPOSITIVO MOVIL Y SERVIDOR WEB:

Previamente al envio de datos hacia el servidor global y dependiendo del tipo de operación que queramos realizar, estos datos deberán de tener una estructura determinada que se regirá en base a la API proporcionada por la tesis complementaria.

Por ejemplo a la hora de registrar una nueva emergencia deberemos de utilizar dicha estructura:

<?xml version="1.0" encoding="UTF-8"?>
<date> [date] </date>
<login> [login] <login>
<location> [location] <location>
<description> [description de la emergencia] </description>

V. REST:

Ahora bien, llegados a este punto donde hasta ahora hemos comentado el parseo y la estructura a seguir para el envio de datos (XML) deberemos de conocer cual será el protocolo de comunicación con nuestro servidor.

Dada que los Web Service implementados en el servidor son de tipo REST, la comunicación con el servidor deberá adaptarse a este tipo.

REST tiene una serie de ventajas con respecto a SOAP en el ámbito de los Smartphone.

En primer lugar SOAP es simplemente demasiado pesado para las comunicaciones móviles. ¿Por qué tanto trabajo para envolver las solicitudes en una capa XML adicional la cual se tendrá que analizar? Estamos enviando mas datos de los que se necesitan y imponemos una mayor carga de CPU en el cliente y el servidor.

1. Ejemplo de consulta en SOAP:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>  <getProductDetails xmlns="http://warehouse.example.com/">
<productId>827635</productId>  </getProductDetails>  </soap:Body>
</soap:Envelope>
```

2. En cambio con REST sería una única línea:

GET http://warehouse.example.com/Product/827635

No solo implicaría un mayor uso de CPU sino que el ancho de banda es muy importante y necesita estar limitado en este tipo de dispositivos. Por tanto, REST es útil para dispositivos de perfil limitado como PDAs y teléfonos móviles, para los cuales las capas adicionales y los encabezados de los elementos de SOAP son XML son demasiado complejos y deben ser restringidos.

Además REST es más simple, pero depende de HTTP como protocolo que envía peticiones en una dirección y datos en la otra. Esto podría ser una ventaja porque HTTP tiene un grupo de funciones diseñadas y testadas, como la autenticación y el mantenimiento de recursos, mientras que cualquier cosa construida en SOAP necesita reinventar todo eso.

Por último, su Escalabilidad: Debido a que únicamente exige HTTP para funcionar y este protocolo es necesario para funcionar en la WWW. Nos aseguramos que todos los dispositivos *móviles* que vayan surgiendo, van a ser compatibles con REST y la propia infraestructura de la red se encargará su crecimiento.

Por tanto, nos decantaremos por REST en vez de SOAP.

Como hemos comentado anteriormente, REST exige HTTP luego nosotros utilizaremos un wrapper para el framework CFnetwork llamado ASIHTTPRequest el cual hace que algunos de los aspectos más tediosos de comunicación con los servidores web se vean simplificados. Está escrito en Objective-C y funciona tanto en Mac OS X y las aplicaciones del iPhone.

Las características mas importantes de ASIHTTP request:

-La posibilidad de enviar archivos de las unidades locales como parte de los datos POST, compatible con el mecanismo de archivo de entrada HTML
-Fácil acceso a la solicitud y respuesta HTTP headers
-Ancho de banda ilimitado.
-Soporte para conexiones persistentes.
-Soporta solicitudes sincrónicas y asincrónicas.
-Reciba notificaciones sobre cambios en su estado de solicitud a través de la delegación.
-Viene con una amplia gama de pruebas unitarias.

VI. FLITE Text To Speech:

El SDK de iPhone no proporciona ninguna herramienta que nos ayude a dar soporte para transformar texto en voz. Por este motivo, se ha estudiado la posibilidad de insertar un framework externo desarrollado por terceros que aporte dicha funcionalidad.

Una de las opciones más adecuadas era utilizar iSpeech, ya que soporta 15 lenguajes diferentes y es el que utilizan muchas aplicaciones disponibles en la Apple Store, el inconveniente es que no es gratuito. Dado que estamos ante un prototipo, vamos a utilizar un framework llamado FliteTTS que aporta casi las mismas características que el anterior comentado.

FliteTTS es una extensión construida sobre la biblioteca de síntesis de voz Flite y fue desarrollada en la Universidad Carnegie Mellon. Es un rápido y gratuito motor de síntesis de voz que nos va a resultar de gran ayuda a la hora de informarnos vía oral de las acciones que hay que realizar en cada momento en un plan de evacuación.

VII. ZXINGWIDGET

Ningún framework SDK de iPhone da soporte a la lectura y el descifrado de códigos QR.

ZXing para iOS es un subproyecto del proyecto ZXing parcialmente mantenido por desarrolladores independientes. Dicho proyecto aporta dicha funcionalidad que nosotros requerimos.

VIII. Sistemas de Localización

Conocido ya nuestro objetivo de proyecto y el dispositivo que vamos a utilizar, hemos hecho un estudio para definir de que forma vamos a localizar a los usuarios de nuestra aplicación.

Hay diferentes formas de localización, más y menos exactas, que dependiendo del uso que se le vaya a dar son más o menos necesarias. Por eso es muy importante elegir un tipo de posicionamiento que se adecue a las necesidades de la aplicación. En nuestro caso, debemos de localizar a los usuarios de la forma más exacta posible puesto que puede que si tenemos por ejemplo un error de [-2,+2] metros podría localizarnos en el pasillo contiguo, dándonos de este modo un plan de evacuación totalmente erróneo.

Posibles sistemas de localización:

Sistemas GPS:

El terminal del usuario debe tener capacidad para ser usado bajo la red de GPS. El GPS es la solución “casi universal” para obtener localización precisa y rápida en cualquier punto del planeta. Lamentablemente, en determinados entornos exteriores, y en la mayoría de los interiores, el GPS no es operativo. Sin embargo, en dichos lugares transcurre gran parte de la actividad humana.

Sistemas LPS :

Local Positioning Systems, son sistemas de localización alternativos creados para funcionar en entornos locales. La ventaja que tienen con respecto al GPS es que son totalmente operativos en espacios cerrados. Este tipo de sistema seria el más adecuado de usar por no ser por su precisión , la cual es bastante inexacta y en nuestra aplicación hay vidas humanas en juego.

Sistema de localización basado en códigos QR:

Un código QR (quick response code, «código de respuesta rápida») es un módulo para almacenar información en una matriz de puntos. Se caracteriza por los tres cuadrados que se encuentran en las esquinas que permiten detectarlos rápidamente. Estos códigos posicionados estratégicamente en un plano de un edificio, podrían dar a conocer la posición de un usuario de una forma 100% precisa y exacta enviando el identificador de dicho código QR con la información del usuario a un servidor.

Hemos elegido sin lugar a dudas el sistema basado en códigos QR como sistema de localización.

TECNOLOGÍA HARDWARE

I. Dispositivos

Dado que la aplicación trabaja con información de localización desde cualquier lugar del edificio, los ordenadores no son los dispositivos indicados para esta aplicación ya que se requiere que la información sea ofrecida con unos niveles de movilidad que los ordenadores no son capaces de ofrecer. Ambos dispositivos, tanto la aplicación para el cliente como para el personal de salvación tienen que ser muy portables, luego deberían de ser o tablets o smartphones.

Para la implementación de nuestra aplicación para el cliente lo ideal es que el dispositivo utilizado fuera un smartphone , ya que, por una parte es más común entre los usuarios y por otra, el usuario dispondrá de señal telefónica pudiendo efectuar llamadas al exterior. La segunda aplicación que servirá a los equipos de respuesta, estará desarrollada para tablet, ya que en ella se podrá ver y manejar mejor la información de una emergencia que en un Smarphone.

II. iPhone

iPhone es una familia de teléfonos inteligentes multimedia con conexión a Internet, pantalla táctil capacitiva y escasos botones físicos diseñado por la compañía Apple Inc.

Será el dispositivo fisico con el cual testearemos el asistente personal de seguridad para los afectados.

En concreto trabajaremos con el iPhone 4S aunque en hoy en día, ya se a lanzado la versión 5 , la cual es mucho más fina y ligera que el anterior y dispone de una pantalla más grande, un chip más rápido, la última tecnología inalámbrica y una cámara iSight de 8'' mpx.

III. iPad

El iPad es un dispositivo electrónico tipo tableta desarrollado por Apple Inc.

Las funciones son similares al resto de dispositivos portátiles de Apple, como es el caso del iPhone o iPod touch, aunque la pantalla es más grande y su hardware más potente. En la actualidad, la versión de iPad mas moderna es la 3.

En este trabajo de investigación no dispondremos de un iPad para testear el asistente personal de seguridad para los equipos de respuesta, pero dado que sería ideal disponer de

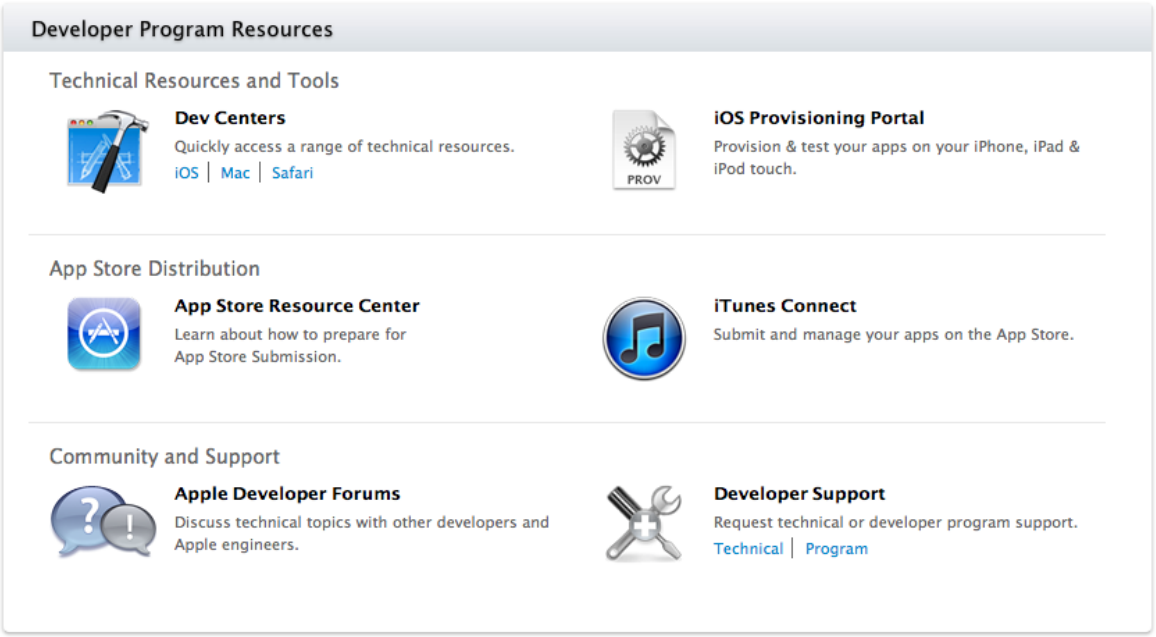
uno, lo hemos añadido en la sección relacionada con las tecnologías. Nostros testearemos el asistente personal de seguridad desde el simulador que ofrece el XCode IDE.

Sección 3: Arquitectura de la solución

I. Primeros pasos

En primer lugar para involucrarse en el desarrollo de aplicaciones en iOS hay que registrarse en el programa de iOS developer en la página oficial de la compañia, el cual tiene un coste de 80 euros al año. Este programa ofrece la capacidad de probar las aplicaciones en diferentes dispositivos físicos, permitiendonos además poder distribuir nuestras aplicaciones dentro de la tienda virtual que ofrece la compañia (AppStore).

Una vez registrados, tendremos acceso a nuestra cuenta de desarrollador donde se nos mostrará un panel con diferentes caracterísiticas:



A continuación comentaremos únicamente los recursos que vamos a utilizar:

Dev Center: Portal desde donde podremos bajarnos el IDE, el SDK que nos proporcionará los elementos necesarios para la programación de aplicaciones para dispositivos iOS ,tutoriales de ayuda para comenzar con el desarrollo,etc.

iOS Provisioning Portal: Este portal servirá para crear nuestros "Provisioning Profiles" y los diferentes certificados, los cuales nos permitirán el uso de notificaciones PUSH en nuestros dispositivos.

Apple Developer Forums: Dispondremos de un foro en el cual podremos compartir problemas con el resto de desarrolladores.

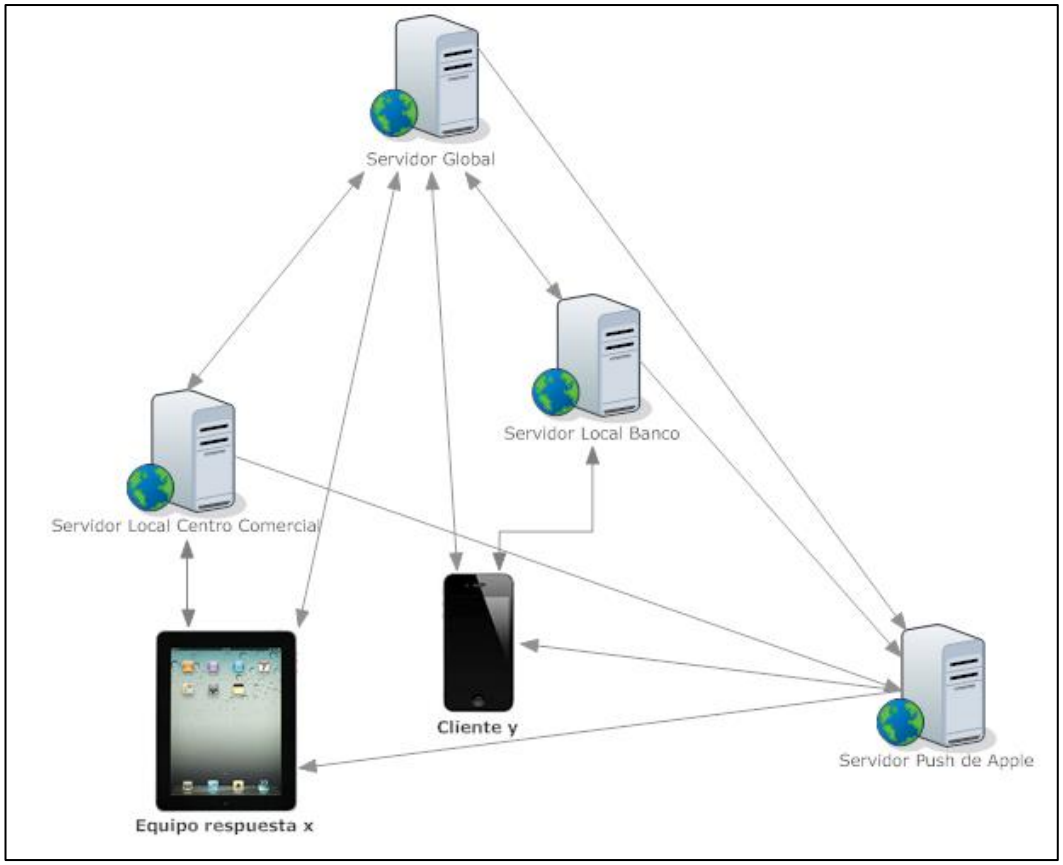
La universidad también dispone de una licencia , la cual me hubiese permitido probar la aplicación en un dispositivo físico sin tener que pagar por mi cuenta la licencia estándar. En cambio, hubiera entrado con el rol de "miembro" y no tendría acceso a los certificados, ni podría crear una nueva aplicación por mi cuenta.

Además, esta licencia me permite en un futuro próximo publicar aplicaciones que desarrolle.

II. Escenario general

Para el correcto funcionamiento de ambos asistentes de seguridad, necesitamos de una serie de elementos que forman la estructura general del sistema.

En la figura se muestran los dispositivos necesarios y el flujo de datos que genera tanto la aplicación como los servidores:



Dispondremos de un Servidor Global que almacenará toda la información del sistema y un conjunto de servidores locales donde cada uno gestionará un edificio en concreto. Por

otra parte utilizaremos el servidor Push de Apple para el envio de notificaciones a los dispositivos.

Como hemos comentado en la sección "tecnología empleada", la comunicación vía dispositivo móvil - servidor (global o local) se realizará utilizando ASIHTTPrequest tanto para el envio como recepción de datos para poder comunicarnos con los webservices de tipo REST.

La comunicación con el servidor APNS será) mediante notificaciones PUSH sera unidireccional en sentido APNS ➔ dispositivo móvil.

III. Características de ambas aplicaciones

Como bien se ha expuesto anteriormente, ambas aplicaciones están enfocadas con el fin o con propósito de proporcionar la información necesaria tanto a los usuarios como a los equipos de rescate, de forma que la emergencia se resuelva de la mejor forma posible. Para conseguir esto, el asistente personal de seguridad deberá reunir las siguientes caracterísitcas:

Las características que tendrá que tener nuestra aplicación para los afectados serán:

- La aplicación deberá 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, avisándole mediante notificaciones Push de que ha habido una alerta en dicho edificio.
- El usuario podrá entrar en la aplicación en modo training y obtener un plan de emergencia concreto cuando lo desee.
- El usuario podrá obtener su posición dentro del edificio cuando lo desee.
- El usuario tendrá la opción de cambiar de servidor local, pudiendo conectarse a cualquier sistema de emergencias de cualquier edificio.
- El usuario podrá ser capaz de configurar la aplicación a su gusto, cambiando propiedades tal como autoconectar aplicación, etc. .
- La aplicación deberá ser capaz de compartir información (datos personales, perfil sanitario, localización...etc.) tanto con el servidor global como con el servidor local al cual esté conectado.
- La aplicación debe ser escalable para que se puedan añadir en cualquier momento nuevos elementos que veamos en un futuro útiles.
- La aplicación debe proporcionar una interfaz lo más sencilla posible.
- El gasto de batería debe ser mínimo.

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

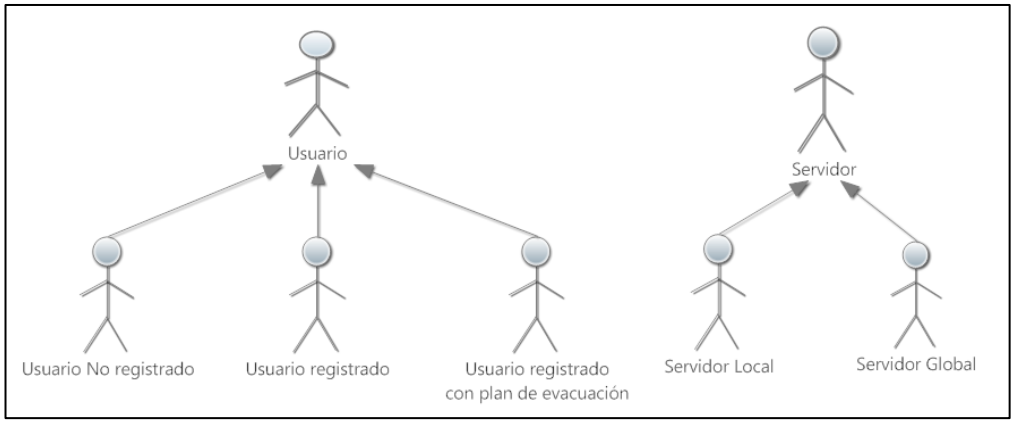
En cambio, para los equipos de respuesta:

- La aplicación deberá 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, avisándole mediante notificaciones Push de que se ha producido una alerta en algún edificio determinado.
- Los equipos de respuesta podrán obtener su posición dentro del edificio cuando lo deseen.
- La aplicación deberá mostrar una lista de personas afectadas dentro del edificio
- La aplicación deberá ser capaz de mostrar información a los equipos de respuesta sobre los perfiles sanitarios de los usuarios que residen en el edificio afectado.
- La aplicación deberá de servir de guía a los equipos de respuesta, proporcionándole el mejor camino hacia cualquiera de los afectados que se encuentre en el edificio.
- aplicación deberá ser capaz de compartir información (datos personales, perfil sanitario, localización...etc.) tanto con el servidor global como con el servidor local al cual esté conectado.
- La aplicación debe ser escalable para que se puedan añadir en cualquier momento nuevos elementos que veamos en un futuro útiles.
- La aplicación debe proporcionar una interfaz lo más sencilla posible.
- El gasto de batería debe ser mínimo.
- El sistema debe estar disponible en varios idiomas de cara a dar soporte a más usuarios.

IV. Actores

En la siguiente página se definirán los actores para nuestro asistente personal de seguridad , diferenciando si se trata de actores para el caso de la aplicacion para los afectados o para los equipos de respuesta.

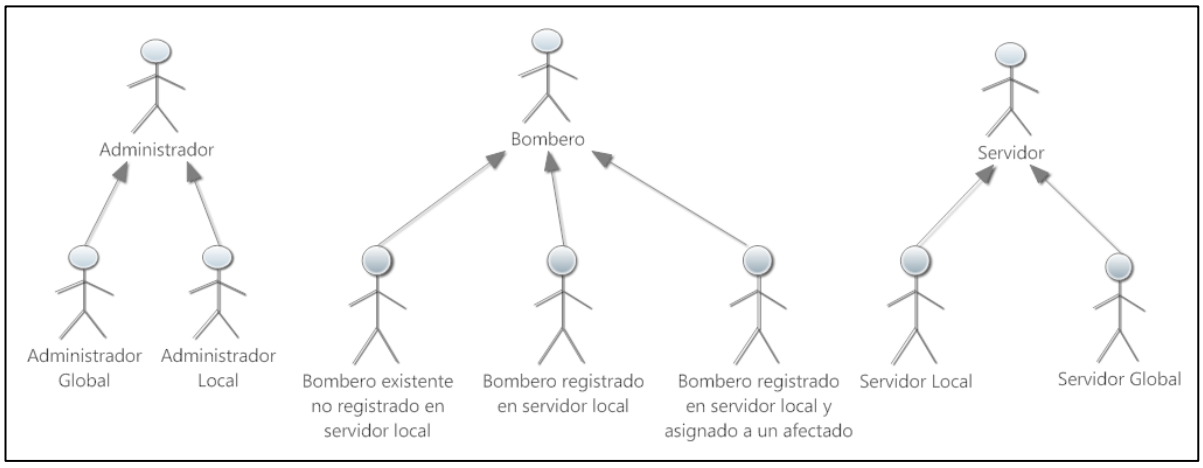
ACTORES - ASISTENTE PERSONAL PARA LOS AFECTADOS:



Cuando un usuario inicie la aplicación por primera vez ,el actor que intervendrá se tratará de un "Usuario No registrado". Cuando re registe y introduzca debidamente el perfil Sanitario estaremos frente a un actor del tipo "Usuario registrado". Los usuarios pueden solicitar planes de evacuación pasando al estado "Usuario registrado con plan de evacuación". En realidad, se tratará de un mismo usuario que va cambiando de estados dependiendo de la situación en la que se encuentre.

El asistente personal de seguridad estará siempre en contacto con el actor Servidor, el cual le suministrará toda la información necesaria dentro del contexto de las emergencias. Normalmente el Actor Usuario contactará directamente con el Actor Servidor Local y éste con el Actor Servidor Global.

ACTORES - ASISTENTE PERSONAL PARA LOS EQUIPOS DE RESPUESTA:



En realidad, el actor Administrador no intervendrá en nuestro asistente personal de seguridad para los equipos de respuesta, pero nos ha parecido razonable introducirlo para

comentar que dicho actor será el encargado de registrar manualmente a cada uno de los actores Bombero. De este modo evitaremos que el asistente personal de seguridad para los equipos de respuesta no registre bomberos que no pertenecen a dicho colectivo.

Una vez iniciamos la aplicación, el actor bombero será del tipo "bombero existente no registrado en servidor local". Cuando se produzca una emergencia y el bomero se registre en dicho servidor local el actor será del tipo "Bombero registrado en local". En caso de que el bombero inicie el plan de rescate de un afectado, el estado del Bombero cambiará y el actor será del tipo "Bombero registrado en servidor local y asignado a un afectado". Se tratará de un mismo bombero que va cambiando de estados dependiendo de su situación.

V. CASOS DE USO

Los casos de uso especifican las distintas acciones que pueden realizar los actores tanto en la aplicación que sirve a los clientes afectados como a la que sirve a los equipos de respuesta. Aquí se presentan los diagramas de casos de uso:

CASOS DE USO DE LA APLICACIÓN PARA LOS AFECTADOS:

Nombre	Registrar Usuario
Identificador	001
Actores	Usuario No registrado
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 cambiará su estado y pasará a ser "Usuario registrado"
Curso Normal	1º El usuario iniciará la aplicación por primera vez. 2º Dentro de la zona correspondiente, rellenará sus datos personales (nombre, apellidos, teléfono y teléfono familiar). Todos los campos serán obligatorios. 3º Una vez rellenados los campos y presionado el botón login se creará dicho usuario.
Curso alternativo	3º Si se ha dejado algún campo vacio o la validación de dicho campo es incorrecta se notificara al usuario con un mensaje de error. En caso de que el usuario exista, no se notificara ningún mensaje y no se registrará dicho usuario, procediendo a continuación de forma habitual.
Comentarios	El usuario quedará registrado en el servidor global.

Nombre	Nuevo Perfil Sanitario
Identificador	002

Actores	Usuario registrado
Resumen	Será el propio usuario el encargado de rellenar perfil sanitario en la aplicación.
Dependencias	
Precondición	
Postcondición	El perfil sanitario del usuario quedará registrado.
Curso Normal	1º El usuario iniciará la aplicación y accederá al menú del perfil sanitario. 2º Dentro de la zona correspondiente, podrá crear su perfil rellenando una serie de campos opcionales. 3º Una vez rellenados los campos y guardado se creará dicho perfil sanitario.
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. 3º Si el perfil sanitario ya existe, se modificará por el anterior.
Comentarios	El perfil sanitario del usuario quedará registrado en el servidor global.

Nombre	Registrar Emergencia
Identificador	003
Actores	Usuario registrado
Resumen	Será el propio usuario el encargado de rellenar los datos de la emergencia (su ubicación y el motivo).
Dependencias	
Precondición	Que dicho usuario este registrado
Postcondición	La emergencia quedara registrada
Curso Normal	1º El usuario iniciará la aplicación por si solo. 2º Dentro del menú principal-Registrar Emergencia, podrá rellenar los datos de dicha emergencia. Ambos campos serán obligatorios. 3º Una vez rellenados los campos guardado se registrará dicha emergencia en el sistema.
Curso alternativo	3º Si hay algún problema con alguno de los datos se notificará con un mensaje de error.
Comentarios	Solo se puede acceder a esta opción habiéndose registrado el usuario previamente, luego podemos tener control de quien envía la emergencia.

Nombre	Registrarse en el Servidor local
Identificador	004

Actores	Usuario registrado
Resumen	Será el propio usuario el encargado de buscar el código QR perteneciente a dicho edificio y registrarse en el.
Dependencias	
Precondición	
Postcondición	El usuario quedará registrado en dicho servidor local.
Curso Normal	1º El usuario iniciará la aplicación. 2º Una vez registrado correctamente accediendo al menú de propiedades, verá la opción "cambiar servidor". Al pulsarla se abrirá el lector de códigos QR. 3º Una vez detectado el código QR correspondiente a dicho edificio dicho usuario quedara registrado en su servidor local.
Curso alternativo	3º Si el servidor esta caído, se le deberá de indicar al usuario de alguna forma.
Comentarios	La acción de registrar un usuario en un servidor local, también quedará reflejada en el servidor global por si de algún modo el servidor local fallara, el global se ocupara de dar servicio a los usuarios que siguen dentro del edicio.

Nombre	Información de ayuda
Identificador	005
Actores	Usuario registrado
Resumen	El propio usuario accediendo al menú de propiedades, tendrá acceso a un video ilustrativo que muestre el funcionamiento de la aplicación
Dependencias	
Precondición	
Postcondición	
Curso Normal	1º El usuario iniciará la aplicación. 2º Una vez registrado correctamente accediendo al menú de propiedades, verá un botón de ayuda. Al pulsarlo se abrirá un video ilustrativo del funcionamiento de la aplicación.
Curso alternativo	
Comentarios	En dicho video se puede apreciar el comportamiento del móvil frente a una emergencia real.

Nombre	Cambiar idioma
Identificador	006
Actores	Usuario registrado
Resumen	El propio usuario podrá cambiar el idioma de la aplicación.

	Este cambio no solo afectará tanto a la interfaz de usuario (se mostrará todo en el idioma elegido) sino que los planes de evacuación se recibirán en el idioma seleccionado.
Dependencias	
Precondición	
Postcondición	
Curso Normal	1º El usuario iniciará la aplicación. 2º Una vez registrado correctamente accediendo al menú de propiedades, verá un botón "switch" que permitirá elegir el idioma que se desee. El idioma quedara registrado de forma local.
Curso alternativo	
Comentarios	Al solicitar un plan de evacuación, el dispositivo le indicara al servidor local el idioma predefinido.

Nombre	Obtener plan de evacuación
Identificador	007
Actores	Usuario registrado
Resumen	El propio usuario podrá solicitar voluntariamente un plan de emergencia con el fin de conocer su proceso de evacuación personal.
Dependencias	
Precondición	Estar registrado en el sistema.
Postcondición	El usuario pasará al estado: Usuario con Plan de Evacuación
Curso Normal	1º Se le notificara al usuario que se situe frente al código QR mas cercano. 2º El usuario se situará en el mismo emplazamiento que un QR code. 3º Usando dicha aplicación y enfocándola hacía el QR code obtendrá dicho código. 4º Dicho código será enviado al servidor el cual responderá con el plan de evacuación concreto en función de su posición.
Curso alternativo	5º Si hay algún problema con el envío y transmisión de los datos se notificará con un mensaje.
Comentarios	Podremos encontrarnos frente a dos situaciones. La primera, que estemos en una situación de emergencia real, luego el plan de evacuación devueltos tendrá en cuenta todos aquellos tramos afectados. La segunda, que estemos en modo training luego el servidor local nos generara un plan de evacuación sin ningún tramo afectado.

Nombre	Refrescar plan de evacuación
Identificador	008

Actores	Usuario registrado con plan de evacuación
Resumen	El propio usuario si durante el seguimiento de un plan de evacuación se perdiera, podría refrescar el plan de evacuación mostrando los pasos anteriores.
Dependencias	
Precondición	Haber obtenido un plan de evacuación
Postcondición	
Curso Normal	1º Aprovechando la característica multitouch del iPhone, el usuario deslizará el dedo hacia arriba de la pantalla sobre el plan de evacuación. 2º El plan de evacuación se refrescará y volverá a mostrar los pasos anteriores.
Curso alternativo	
Comentarios	El sistema, tras este caso de uso, puede que se haya grabado una localización engañosa del usuario. Este error que en principio había sido generado por el usuario, se verá reparado cuando el usuario realice de nuevo otro paso.

Nombre	Confirmar paso con éxito
Identificador	009
Actores	Usuario registrado con plan de evacuación
Resumen	El propio usuario puede confirmar que ha realizado un paso del plan de evacuación y de esta manera , se registrará su nueva localización
Dependencias	
Precondición	Haber obtenido un plan de evacuación
Postcondición	
Curso Normal	1º El usuario deslizará el dedo sobre la celda que contiene el paso realizado. 2º A continuación se le mostrará un botón de confirmación. 3º Si pulsa sobre dicho botón, se actualizará la localización del usuario al código QR más cercano a su lugar.
Curso alternativo	3ºSi el usuario intenta confirmar un paso habiendo pasos anteriores no realizados, se le mostrará una notificación advirtiéndole de la situación. En caso de que pulse aceptar, se confirmaran todos pasos anteriores a este también.
Comentarios	La nueva localización del usuario quedará registrada tanto en el servidor local como en el global.

Nombre	Llamar a familiar
Identificador	010
Actores	Usuario registrado con plan de evacuación

Resumen	El propio usuario puede llamar a un familiar cercano en un plan de evacuación, concretamente al numero de teléfono familiar registrado en su perfil
Dependencias	
Precondición	Estar registrado y haber obtenido un plan de evacuación
Postcondición	
Curso Normal	1º El usuario cuando recibe un plan de evacuación puede en cualquier momento llamar a un familiar sobre el botón Llamar-Familiar
Curso alternativo	
Comentarios	Aunque no es recomendable llamar a un familiar en plena emergencia sino cuando termina, siempre es positivo tener dicha opción en el teléfono.

Nombre	Mensaje Familiar
Identificador	011
Actores	Usuario registrado con plan de evacuación
Resumen	El propio usuario puede enviar un mensaje automático a un familiar cercano durante una emergencia
Dependencias	
Precondición	Haber obtenido un plan de evacuación
Postcondición	
Curso Normal	1º El usuario presionara sobre el botón mensaje familiar 2º A continuación se abrirá una nueva vista que corresponde a la aplicación que viene instalada por defecto en el móvil de mensajería instantánea (SMS). 3º Tanto los campos numero de teléfono como cuerpo serán rellenados automáticamente teniendo en cuenta la información del perfil. 4º Opcionalmente el usuario podrá editar dichos campos. 5º Pulsando sobre el botón enviar, el usuario enviara el mensaje.
Curso alternativo	5ºSi el usuario pulsa cancelar, volverá al plan de evacuación sin haber enviado ningún mensaje.
Comentarios	

Nombre	Llamar a equipo de evacuación
Identificador	012
Actores	Usuario registrado con plan de evacuación

Resumen	El propio usuario puede llamar al equipo de evacuación más cercano y comunicarles su situación por vía oral.
Dependencias	
Precondición	Estar registrado y haber obtenido un plan de evacuación
Postcondición	
Curso Normal	1º El usuario cuando recibe un plan de evacuación puede en cualquier momento llamar a los bomberos pulsando sobre el botón Llamar- Equipo de rescate
Curso alternativo	
Comentarios	

Nombre	Mensaje Equipo de evacuación
Identificador	013
Actores	Usuario registrado con plan de evacuación
Resumen	El propio usuario puede enviar un mensaje automático al equipo de rescate mas cercano durante una emergencia
Dependencias	
Precondición	Haber obtenido un plan de evacuación
Postcondición	
Curso Normal	1º El usuario presionara sobre el botón mensaje equipo rescate 2º A continuación se abrirá una nueva vista que corresponde a la aplicación que viene instalada por defecto en el móvil de mensajería instantánea (SMS). 3º Tanto los campos numero de teléfono como cuerpo serán rellenados automáticamente. 4º Opcionalmente el usuario podrá editar dichos campos. 5º Pulsando sobre el botón enviar, el usuario enviara el mensaje.
Curso alternativo	5ºSi el usuario pulsa cancelar, volverá al plan de evacuación sin haber enviado ningún mensaje.
Comentarios	

Nombre	Marcar tramo afectado
Identificador	014
Actores	Usuario registrado con plan de evacuación

Resumen	El propio usuario si durante un plan de evacuación observa que hay algún tramo impracticable, puede marcarlo como tramo afectado, quedando este registrado en el sistema para que este mismo genere los avisos oportunos a todos los usuarios.
Dependencias	
Precondición	Haber seleccionado un tramo
Postcondición	El tramo será registrado en el servidor como afectado
Curso Normal	1º El usuario presionara sobre un tramo afectado. 2º A continuación pulsara sobre el botón marcar tramo afectado en el apartado de "problemas". 3º Aparecerá una notificación o aviso confirmando la acción solicitada. 4ºEn caso de pulsar SI, el tramo quedara registrado como afectado en el servidor.
Curso alternativo	4ºSi el usuario pulsa cancelar, volverá al plan de evacuación sin haber registrado ningún tramo como afectado.
Comentarios	

Nombre	Buscar otra salida
Identificador	015
Actores	Usuario registrado con plan de evacuación
Resumen	Si el usuario se pierde siguiendo un plan de evacuación, tiene la opción de, o bien volver a los pasos de atrás (ya comentado anteriormente), o bien en este caso si encuentra un código QR cercano, utilizarlo para buscar una nueva salida.
Dependencias	
Precondición	
Postcondición	
Curso Normal	1º Al usuario se le notificará si desea un plan de evacuación distinto 1º En caso afirmativo, el usuario se situará en el mismo emplazamiento que un QR code. 3º Usando dicha aplicación y enfocándola hacía el QR code obtendrá dicho código. 4º Dicho código será enviado al servidor el cual responderá con el plan de evacuación nuevo y concreto en función de su posición.
Curso alternativo	4ºSi el usuario pulsa cancelar, volverá al atrás sin haber obtenido ningún plan de evacuación nuevo.
Comentarios	

Nombre	Solicitar ayuda exterior
Identificador	016
Actores	Usuario registrado con plan de evacuación

Resumen	Si el usuario se ve atrapado siguiendo un plan de evacuación o se ve incapaz de seguirlo, tiene la posibilidad de poder solicitar ayuda a un equipo de rescate.
Dependencias	
Precondición	
Postcondición	El usuario quedará registrado como atrapado.
Curso Normal	1º El usuario presionara sobre el botón solicitar ayuda exterior. 2º Aparecerá un mensaje de alerta que habrá que confirmarlo. 3º Tras la confirmación el estado de dicho usuario quedará actualizado en el servidor. El equipo de rescate deberá tener en cuenta dicha situación.
Curso alternativo	5º Si el usuario pulsa cancelar, volverá al atrás sin haber solicitado ayuda exterior.
Comentarios	

Nombre	Confirmar salida con éxito
Identificador	017
Actores	Usuario registrado con plan de evacuación
Resumen	Los usuarios que hayan salido del edificio correctamente y se encuentren a salvo deberán indicarlo en la aplicación
Dependencias	
Precondición	
Postcondición	El usuario quedará registrado como salvado
Curso Normal	1º El usuario finalizará todos los pasos indicados por el plan de evacuación . 2º A continuación se mostrará un mensaje de confirmación indicando que se encuentra a salvo. 3 º En caso de pulsar SI, el usuario quedará registrado como salvado
Curso alternativo	1º El usuario no ha finalizado los pasos indicados por el plan de evacuación sino que ha confiado en su instinto y a salido por su propio pie por una salida aleatoria. 2ºEn este caso, deberá de confirmar su salida con éxito salida presionando sobre el botón con la imagen de salida con éxito y pulsando aceptar.
Comentarios	

CASOS DE USO DE LA APLICACIÓN PARA LOS EQUIPOS DE RESPUESTA

Nombre	Registrar personal de respuesta en servidor local
Identificador	001
Actores	Bombero existente no registrado en servidor local
Resumen	Será el propio usuario el encargado de introducir "a mano" los datos necesarios y registrarse en un servidor local para poder ver las personas afectadas en su interior.
Dependencias	
Precondición	
Postcondición	El usuario de la aplicación quedará registrado en dicho servidor local.
Curso Normal	1º El usuario iniciará la aplicación 2º Dentro de la zona correspondiente, rellenará sus datos personales (login, password y servidor local). Todos los campos serán obligatorios. 3º Una vez rellenados los campos presionaremos el botón de registrarse.
Curso alternativo	3º Si el campo introducido no es valido, o el servidor está caído, se le informará al usuario mediante una notificación.
Comentarios	

Nombre	Localizar personal de respuesta
Identificador	002
Actores	Personal de respuesta registrado en servidor local
Resumen	Será el propio usuario el encargado de localizarse en un punto del edificio.
Dependencias	
Precondición	
Postcondición	El usuario de la aplicación quedará registrado en dicha ubicación (codigoQR)
Curso Normal	1º El usuario se situará frente al código QR mas cercano. 2º Usando dicha aplicación y enfocándola hacía el QR code obtendrá dicho código. 3º Dicho código será enviado al servidor el cual registrará al usuario en dicha posición.
Curso alternativo	3º Si el usuario presiona cancelar volverá al menú anterior.
Comentarios	

Nombre	Mostrar datos de un afectado

Identificador	003
Actores	Personal de respuesta registrado en servidor local
Resumen	Dada una lista ordenada de personas que aun residen en el edificio afectado, el usuario seleccionara una de ellas, mostrándose a continuación el perfil sanitario de dicha persona .
Dependencias	
Precondición	
Postcondición	
Curso Normal	1º Dada una lista de personas que aun residen en el edificio, el usuario seleccionara uno de los afectados 2º La aplicación preguntara al servidor acerca del perfil sanitario de dicho afectado y éste le responderá con esos datos. 3º El perfil sanitario de dicho afectado se mostrara en una nueva vista.
Curso alternativo	
Comentarios	

Nombre	Obtener camino hacia un afectado
Identificador	004
Actores	Personal de respuesta registrado en servidor local y asignado a un afectado
Resumen	Una vez visualizado el perfil sanitario de un afectado podremos ver cual es el mejor camino y el mas indicado para llegar a dicha persona.
Dependencias	<include a 002>
Precondición	
Postcondición	
Curso Normal	1º En primer lugar se mostrara una notificación advirtiendo si nos encontramos en nuestra localización de origen. 2º En dicho caso la aplicación preguntará al servidor local acerca del camino más corto hasta el afectado. 3º El servidor retornará la respuesta a la aplicación mostrándose ésta misma por pantalla (el mejor camino hasta el usuario afectado).
Curso alternativo	1º Si el usuario responde a la notificación que NO, se mostrará un lector de códigos QR adicional que registrará la posición más cercana del usuario. A continuación se procederá desde el paso 2º.
Comentarios	

Nombre	Registrar punto de encuentro correcto

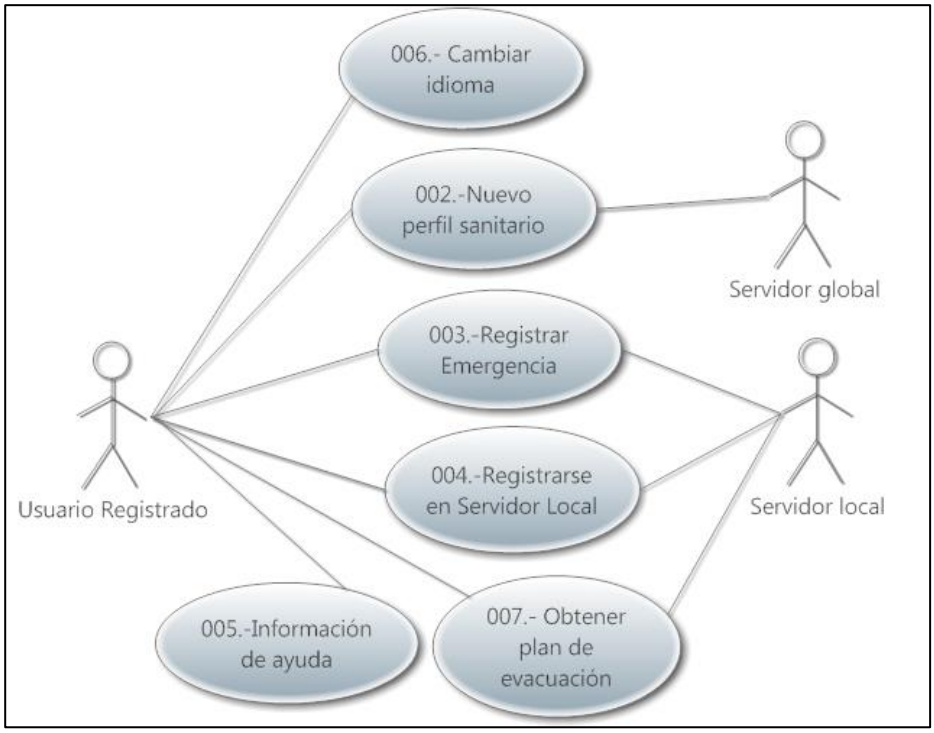
Identificador	005
Actores	Personal de respuesta registrado
Resumen	Si el rescatista encuentra al afectado deberá de indicarlo mediante este caso de uso.
Dependencias	
Precondición	
Postcondición	
Curso Normal	1º Se mostrará una notificación al usuario asegurando su localización 2º En caso de responder SI, se actualizará tanto la localización del rescatista a la posición actual como el estado del usuario (solicita_ayuda=NO)
Curso alternativo	1º En caso de responder NO a la notificación, cancelaremos la operación.
Comentarios	

Nombre	Confirmar afectado a salvo
Identificador	006
Actores	Personal de respuesta registrado en servidor local y asignado a un afectado
Resumen	Si el bombero rescata a una persona deberá confirmar que dicha persona está a salvo.
Dependencias	
Precondición	
Postcondición	
Curso Normal	1º El bombero saldrá del edificio junto con el afectado que tenía asignado 2º El bombero confirmará que dicho usuario esta a salvo. 3º El servidor modificara el estado del usuario poniendolo a salvo.
Curso alternativo	
Comentarios	

Nombre	Solicitar Plan de evacuación
Identificador	007
Actores	Personal de respuesta registrado en servidor local
Resumen	El sistema podrá proporcionar a los bomberos un plan de evacuación idéntico a la aplicación para los afectados.
Dependencias	<include> a 002
Precondición	
Postcondición	

Curso Normal	1º Si el bombero se encuentra en peligro o ya ha encontrado a la persona afectada asignada el sistema marcará la opción Buscar salida. 2º Se mostrará un lector de códigos QR que registrará la posición más cercana del bombero. 3º El servidor le retornará un plan de evacuación al bomber
Curso alternativo	
Comentarios	

DIAGRAMA PARA LOS CASOS DE USO DE LA APLICACIÓN PARA LOS AFECTADOS:



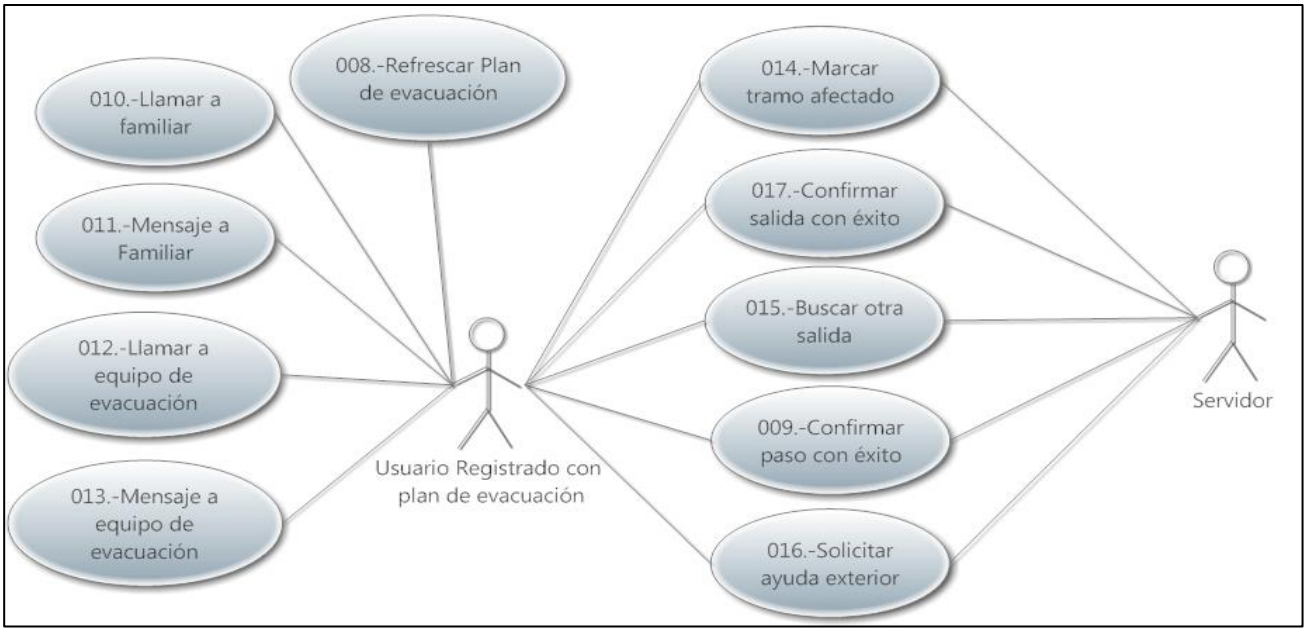
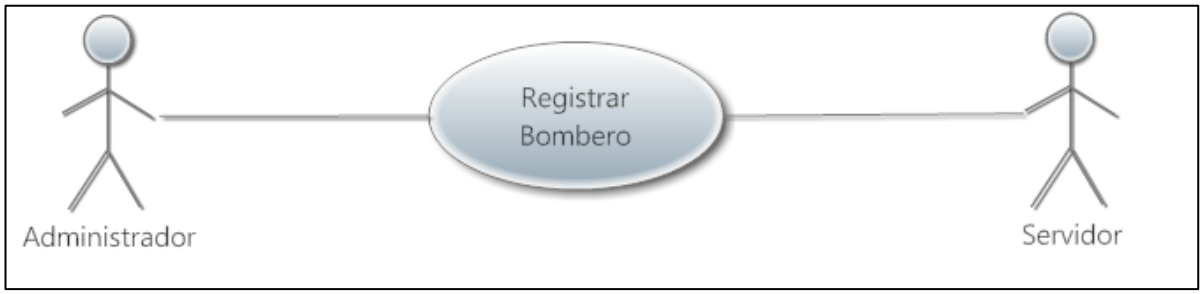
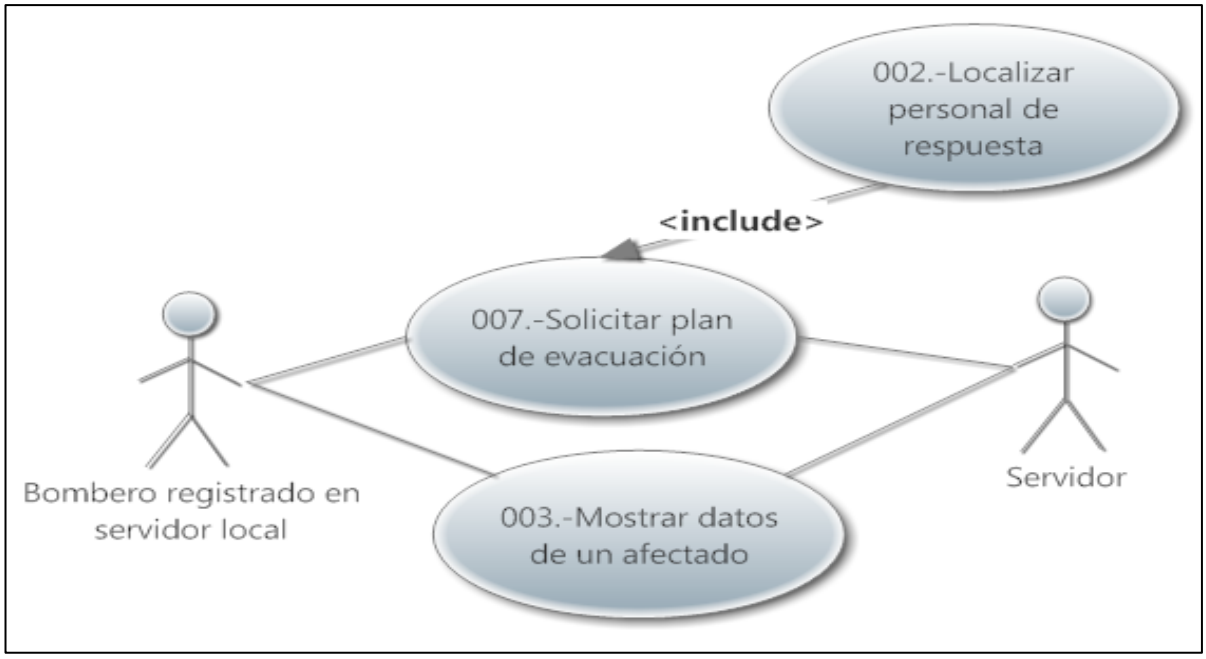
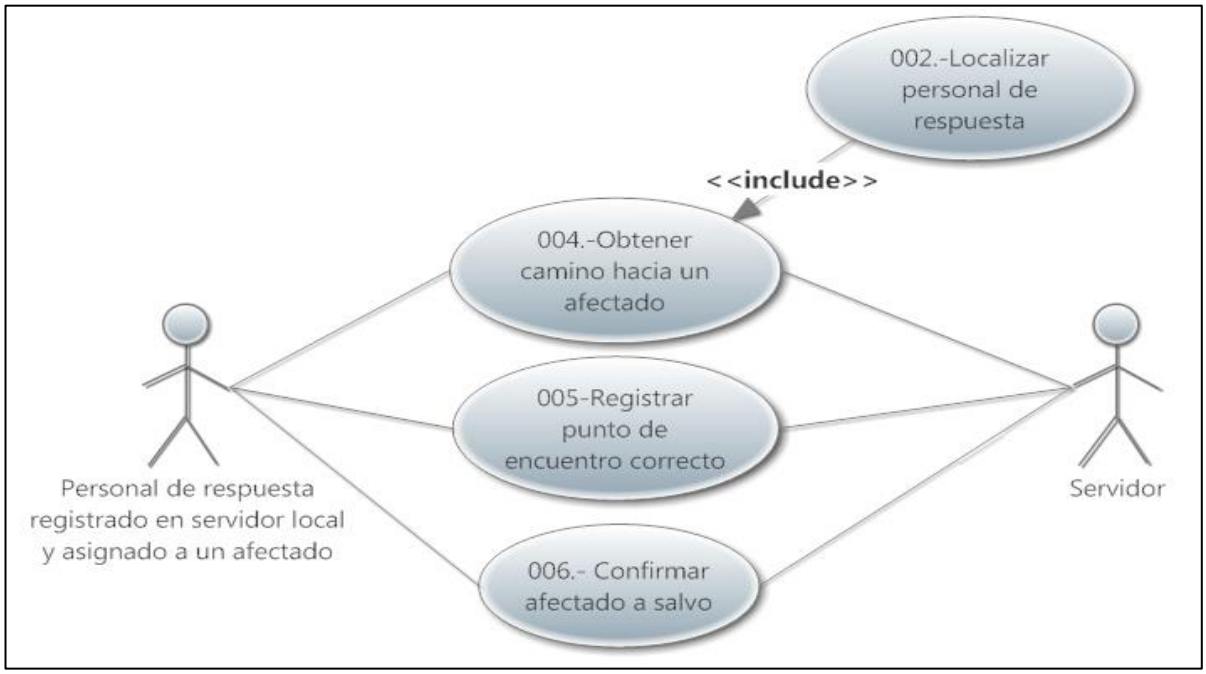


DIAGRAMA PARA LOS CASOS DE USO DE LA APLICACIÓN PARA EL EQUIPO DE RESCATE:



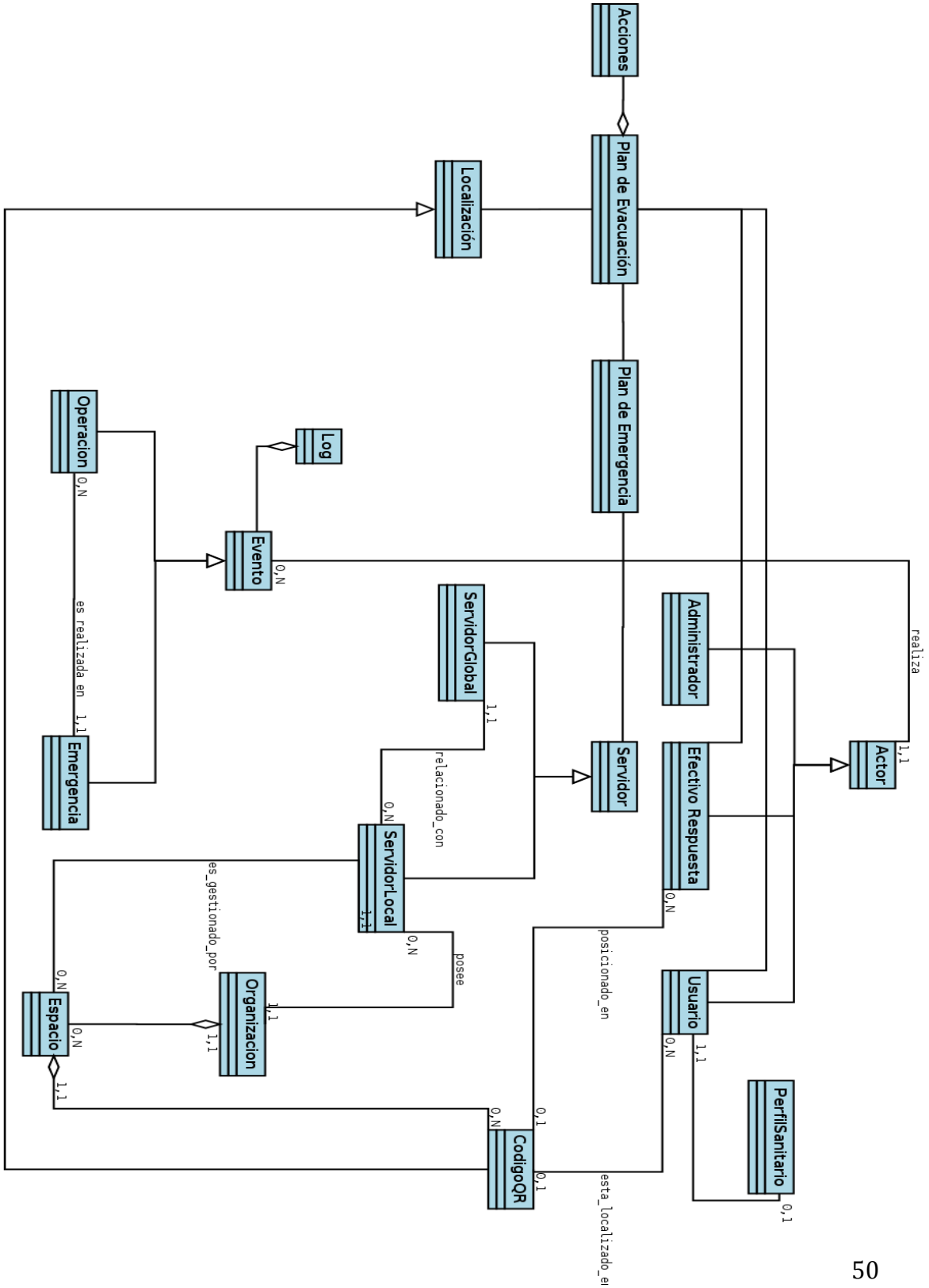

002.-Localizar personal de respuesta
<include>
007.-Solicitar plan de evacuación
Bombero registrado en servidor local
003.-Mostrar datos de un afectado
Servidor


002.-Localizar personal de respuesta
<<include>>
004.-Obtener camino hacia un afectado
005-Registrar punto de encuentro correcto
006.- Confirmar afectado a salvo
Personal de respuesta registrado en servidor local y asignado a un afectado
Servidor

VI. Esquema Conceptual

El esquema conceptual permitirá dar una idea de cómo organiza la información así como las distintas partes del sistema y de la estructura que tendrá el esquema relacional.

Esquema conceptual del sistema de emergencias:



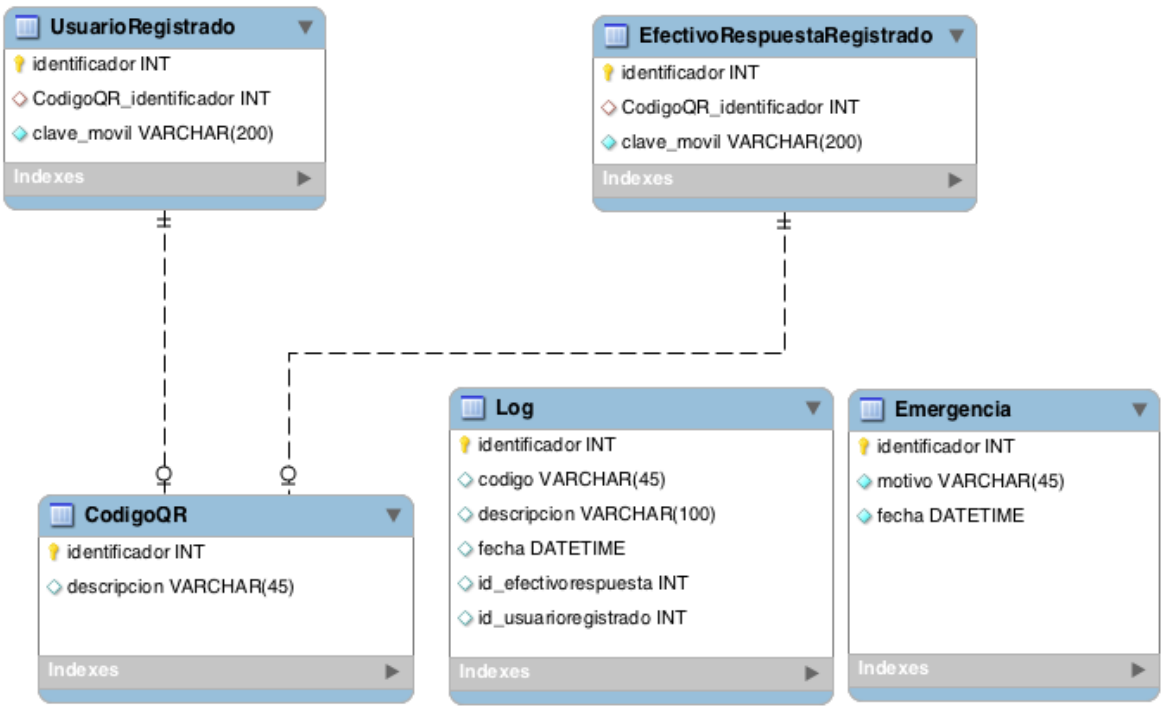
En primer lugar, comentar que dicho esquema conceptual representa todo el sistema de información que gira alrededor de una emergencia y nuestra tesis está enfocada únicamente a realizar el desarrollo de la parte del cliente (aplicaciones para los usuarios y equipos de respuesta) , así como el diseño del esquema relacional junto con el registro de Log, luego habrán conceptos en el esquema conceptual que no se verán representados en el esquema relacional.

VII. Esquema relacional

En este apartado mostraremos la implementación del modelo relacional de datos que hemos llevado a cabo basándonos en el esquema conceptual y que estará implementado en la parte del servidor para que de algún modo, podamos conseguir manejar toda la información que gira alrededor de una emergencia.

En nuestro caso, debemos de diferenciar entre el esquema de base de datos que poseerá el servidor local y el esquema de base de datos que poseerá el servidor global. Habrá información privada como por ejemplo el perfil sanitario de un usuario o otros conceptos que solo deberan ser vistos por el servidor global.

ESQUEMA RELACIONAL DEL SERVIDOR LOCAL:



Para comprender mejor dicho esquema a continuación realizaremos una descripción de cada una de las clases:

Clase	**UsuarioRegistrado**
Descripción	Almacenará todos aquellos usuarios que se registren en la aplicación de gestión de emergencias en ese servidor local en concreto
Parámetros:	• identificador: código que identifica al usuario registrado • codigoQR_identificador: Es una clave ajena a una instancia de la clase Código. Es útil a la hora de conocer la localización exacta de dicho usuario. Dicho parámetro acepta nulos puesto que puede que un usuario no este aún localizado en ninguna posición. • clave_movil: clave para poder enviar notificaciones push al terminal de dicho usuario
Comentarios	El servidor local tendrá la función de preguntar al servidor global acerca de los parámetros introducidos en la interfaz de la aplicación y éste mismo le verificará si son validos o no. En caso de serlos, se creara una nueva instancia de dicho usuario en esta misma clase.

Clase	**Efectivo Respuesta Registrado**
Descripción	Almacenará todos aquellos usuarios que se registren en la aplicación de los equipos de respuesta en ese servidor local en concreto
Parámetros:	• identificador: código que identifica al usuario registrado • codigoQR_identificador: És una clave ajena a una instancia de la clase CodigoQR. Es útil a la hora de conocer la localización exacta de dicho usuario. Dicho parámetro acepta nulos puesto que puede que un usuario no este aún localizado en ninguna posición. • clave_movil: clave para poder enviar notificaciones push al terminal de dicho usuario
Comentarios	El servidor local tendrá la función de preguntar al servidor global acerca de los parámetros introducidos en la interfaz de la aplicación y éste mismo le verificará si son validos o no. En caso de serlos, se creara una nueva instancia de dicho usuario en esta misma clase.

Clase	CodigoQR
Descripción	Almacenará todos aquellos codigosQR que se encuentren en el interior de dicho edificio. Esta clase nos permitirá conocer la ubicación de la mayoría de usuarios que estén registrados en el sistema.
Parámetros:	• identificador: código que identifica al codigoQR • descripción: Descripción detallada del código QR para saber su ubicación, por ejemplo: o Ej.: Escalera A, piso 2º ,habitación 323
Comentarios	Los codigos QR deberán de estar posicionados en lugares estratégicos donde faciliten la información de salida al usuario. Las instancias de los códigos QR deberán ser creadas por el administrador del sistema.

Clase	Emergencia
Descripción	Almacenará todas aquellas emergencias que se hayan producido en dicho edificio, así como la fecha y el motivo de las mismas
Parámetros:	• identificador: código que identifica a la emergencia • motivo: responde a la pregunta de "porqué" se ha originado dicha emergencia • fecha: fecha de declaración de la emergencia
Comentarios	Esta clase es opcional puesto que las emergencia quedarán registradas también en el log y además no está relacionada con ninguna clase. La hemos contemplado por la posibilidad de si cada cierto tiempo se realiza un borrado del log y así quede reflejada la emergencia en algún lugar.

Clase	Log
Descripción	Almacenará todas aquellas operaciones que se van realizando en el servidor local y nos ofrecerá un seguimiento de todo lo que ha ocurrido y esta ocurriendo en el mismo con el fin de poder hacer estudios, estadísticas, trazas, etc. Que nos sirvan en gran medida para conocer los fallos o defectos del sistema.
Parámetros:	• identificador: código que identifica un registro de log • código: identifica la operación perteneciente a dicho registro de log, ya sea el registro de un usuario en el servidor local, el cambio de localización de un usuario, etc.

	• descripción: mostrará información detallada sobre la operación realizada. • fecha: fecha de creación del registro de log. • id_efectivorespuesta: en caso de que la operación la haya realizado un efectivo respuesta contendrá el identificador de dicho usuario. • id_usuarioregistrado: en caso de que la operación haya sido llevada a cabo por un cliente de la aplicación de gestión de emergencias, contendrá el identificador de dicho usuario.
Comentarios	

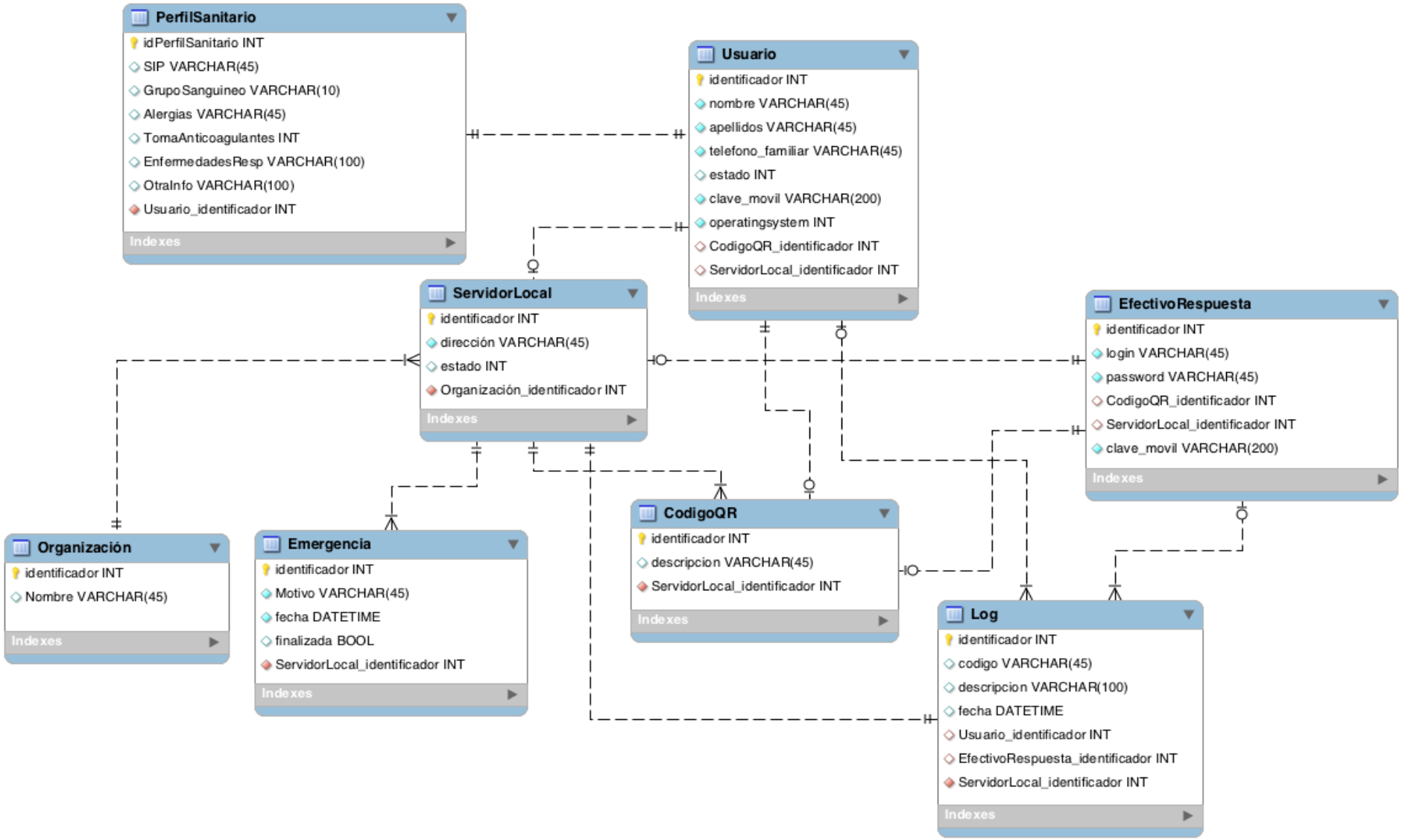
Con el fin de poder diferenciar todas las operaciones registradas en el log en el servidor local, hemos realizado la siguiente tabla:

código	descripción	efectivo_resp	usuario
001	Registro de efectivo respuesta en servidor local	V	×
002	Registro de usuario en servidor local	×	V
003	Cierre de sesión de efectivo respuesta en local	V	×
004	Cierre de sesión de usuario en servidor local	×	V
005	Cambio localización de efec.respuesta en local: "id QR"	V	×
006	Cambio localización de usuario en local: "idqrcode"	×	V
007	Declaración de emergencia "id" .Localización y razones.	×	V
008	Fin de emergencia "id emergencia"	×	×
009	Servidor caído	×	×
010	Notificación enviada a un usuario	×	V
011	Notificación enviada a equipo de respuesta	V	×
012	Usuario salvado	×	V
013	Usuario atrapado	×	V
014	Envio de plan de evacuación a usuario	×	V
015	Envio camino a efectivo respuesta	V	×

Nota: Una V indicará que la operación requiere de dicho campo para poder introducirse en el Log y una × en caso contrario.

Una vez detallado el esquema relacional del servidor local, vamos a adentrarnos en explicar a fondo el mismo esquema relacional para el servidor global , el cual es mucho más extenso puesto que además de contener información privada y contenido adicional respecto a los servidores locales, además debe de dar soporte a los usuarios de éstos mismos en caso de que el servidor local al cual pertenecen se viniera abajo durante una emergencia.

En esta pagina podemos observar el esquema relacional del servidor global.



Para comprender mejor el esquema relacional del servidor global vamos a detallar cada una de las clases que contiene:

Clase	**Usuario**
Descripción	Almacenará todos aquellos usuarios que se registren en la aplicación de gestión de emergencias .

Parámetros:	• identificador : numero de teléfono del usuario.
	• nombre, apellidos, teléfono y teléfono familiar (se sobreentienden).
	• estado: identifica la situación del usuario:
	  ○ Posibles valores:
	  ▪ 0: Fuera del edificio
	  ▪ 1: Dentro del edificio
	  ▪ 2: Solicita Ayuda
	• clavemovil: identificador de notificaciones push
	• operatingsistem: identifica el sistema operativo del Smartphone.
	  ○ Posibles valores:
	  ▪ 0: Android
	  ▪ 1: iPhone
	• codigoqr_identificador: sirve para identificar la localización de un usuario en un servidor local. En caso de que este parámetro sea distinto de null, se deberá introducir el identificador del servidor local asociado.
	• servidorlocal_identificador: identifica a un usuario como registrado en un servidor local.

Comentarios	Una vez registrados, su información quedará grabada para futuros accesos.

Clase	**Perfil Sanitario**
Descripción	Clase que almacena el perfil sanitario de los usuarios.
Parámetros:	• Los parámetros de dicha clase son fácilmente entendibles y no hace falta describirlos.
Comentarios	Dicha información estará protegida y no podrá ser accesible por personal externo y solo podrá ser consultada en caso necesario.

Clase	**Organización**
Descripción	Clase que almacena todas las organizaciones que disponen de nuestro sistema de emergencias instalado.
Parámetros:	• Los parámetros de dicha clase son fácilmente entendibles y no hace falta describirlos.
Comentarios	

Clase	Servidor local
Descripción	Clase que almacena todos los servidores locales contenidos en nuestro sistema

Parámetros:	• identificador: código que identifica al servidor local
	• dirección: dirección web del servidor local
	• estado: parámetro que identifica el estado del servidor
	  ○ Posibles valores:
	  ▪ 0: caído
	  ▪ 1: online
	• organización: organización a la cual pertenece el servidor local.

Comentarios	

Clase	Emergencia
Descripción	Clase que almacena todas las emergencias que se producen en nuestro sistema.

Parámetros:	• identificador: código que identifica a la emergencia
	• motivo: descripción de la emergencia
	• finalizada: parámetro que identifica el estado de la emergencia:
	  ○ Posibles valores:
	  ▪ TRUE: finalizada
	  ▪ FALSE: no finalizada
	• fecha
	• servidorlocal_identificador: parámetro que identifica en que servidor local se ha producido dicha emergencia

Comentarios	

Clase	CodigoQR
Descripción	Clase que almacena todos códigos QR de todos los edificios.
Parámetros:	• identificador: código que identifica al código QR • descripción: Descripción detallada del código QR para saber su ubicación, por ejemplo: ○ Ej.: Escalera A, piso 2º ,habitación 323 • servidorlocal_identificador: código que sirve para asociar dicho código QR a un servidor local en concreto
Comentarios	Los códigos deberán de estar posicionados en lugares estratégicos donde faciliten la información de salida al usuario. Las instancias de los códigos QR deberán ser creadas por el administrador del sistema.

Clase	Efectivo Respuesta
Descripción	Almacenará todos aquellos usuarios que se registren en la aplicación que utilicen los equipos de rescate.
Parámetros:	• identificador: código que identifica usuario • login: Nick del usuario en la aplicación • password: contraseña del usuario • codigoqr_identificador: localización del usuario • servidorlocal_identificador: parámetro que se utiliza para ver en que servidor local está registrado el usuario.
Comentarios	

Clase	Log
Descripción	Almacenará todas aquellas operaciones que se van realizando en todos los servidores locales y nos ofrecerá un seguimiento de todo lo que ha ocurrido y esta ocurriendo en el mismo con el fin de poder hacer estudios, estadísticas, trazas, etc.. Que nos sirvan en gran medida para conocer los fallos o defectos del sistema.
Parámetros:	• identificador: código que identifica un registro de log • código: identifica la operación perteneciente a dicho registro de log, ya sea el registro de un usuario en el servidor local, el cambio de localización de un usuario, etc. • descripción: mostrará información detallada sobre la operación realizada. • fecha: fecha de creación del registro de log. • id_efectivorespuesta: en caso de que la operación la haya realizado un efectivo respuesta contendrá el identificador de dicho usuario. • id_usuarioregistrado: en caso de que la operación haya sido llevada a cabo por un cliente de la aplicación de gestión de emergencias, contendrá el identificador de dicho usuario. • servidor_local_identificador: sirve para localizar la operación es decir, en que servidor local se ha producido.
Comentarios	

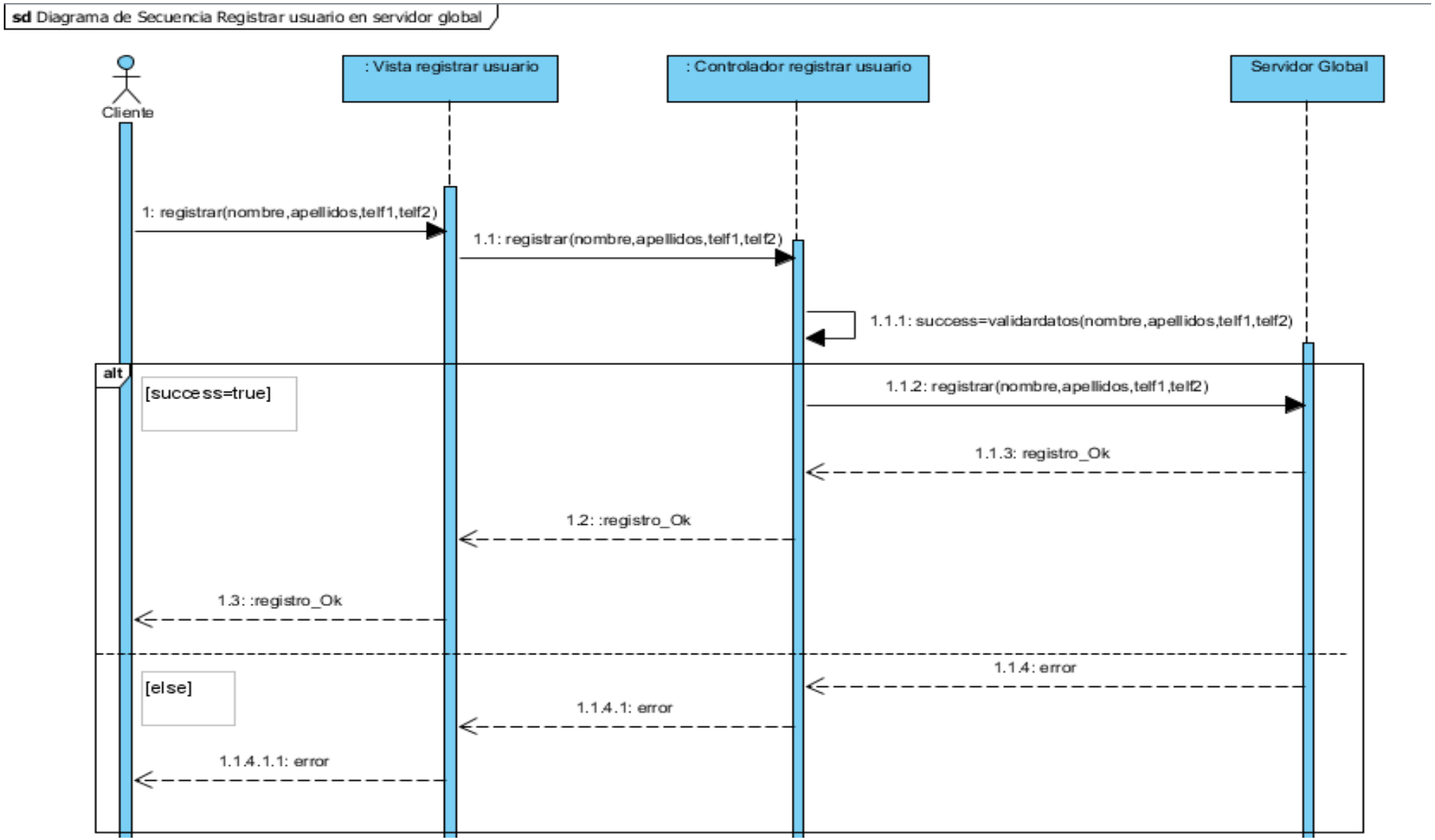
En la siguiente página se podremos observar una tabla que contiene todas las posibles codificaciones del registro log en el servidor global.

código	descripción	efectivo_r	usuario	servidor local
001	Registro de efectivo respuesta en servidor local	V	×	V
002	Registro de usuario en servidor local	×	V	V
003	Cierre de sesión de efectivo respuesta en local	V	×	V
004	Cierre de usuario en servidor local	×	V	V
005	Cambio localización de ef. Respuesta en local: idqrcode	V	×	V
006	Cambio localización de usuario en local: idqrcode	×	V	V
007	Declaración de emergencia :id: Localización y razones.	×	V	V
008	Fin de emergencia :id	×	×	V
009	Servidor caído	×	×	V
010	Notificación enviada a un usuario	×	V	Opcional
011	Notificación enviada a equipo de respuesta	V	×	Opcional
012	Usuario salvado de la emergencia: "id emergencia"	×	V	V
013	Usuario atrapado, emergencia: "id emergencia"	×	V	V
014	Registrado nuevo perfil sanitario en servidor global	×	V	×
015	Registrado nuevo efectivo respuesta en servidor global	V	×	×
016	Registrado nuevo usuario en servidor global	×	V	×
017	Modificado perfil sanitario en servidor global	×	V	×
018	Modificado usuario en servidor global	×	V	×
019	Servidor en marcha	×	×	V
020	Envio de plan de evacuación a usuario	×	V	Opcional
021	Envio camino a efectivo respuesta	V	×	Opcional

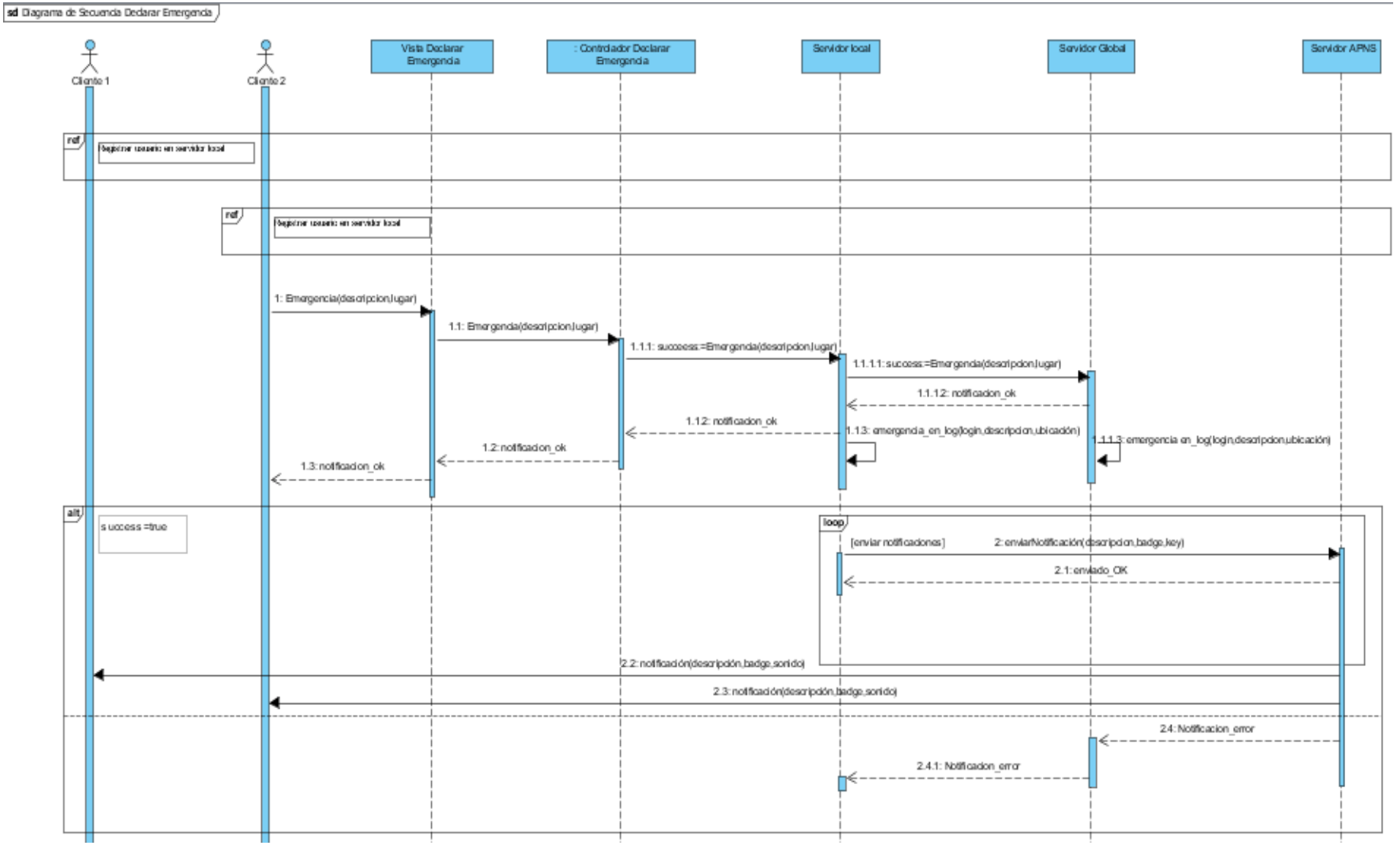
VIII. Diseño del asistente personal de seguridad

A continuación, se mostrarán los distintos diagramas de secuencia que nos ayudarán a comprender como funciona internamente el sistema a la hora de registrar un usuario, declarar una emergencia y solicitar un plan de evacuación.

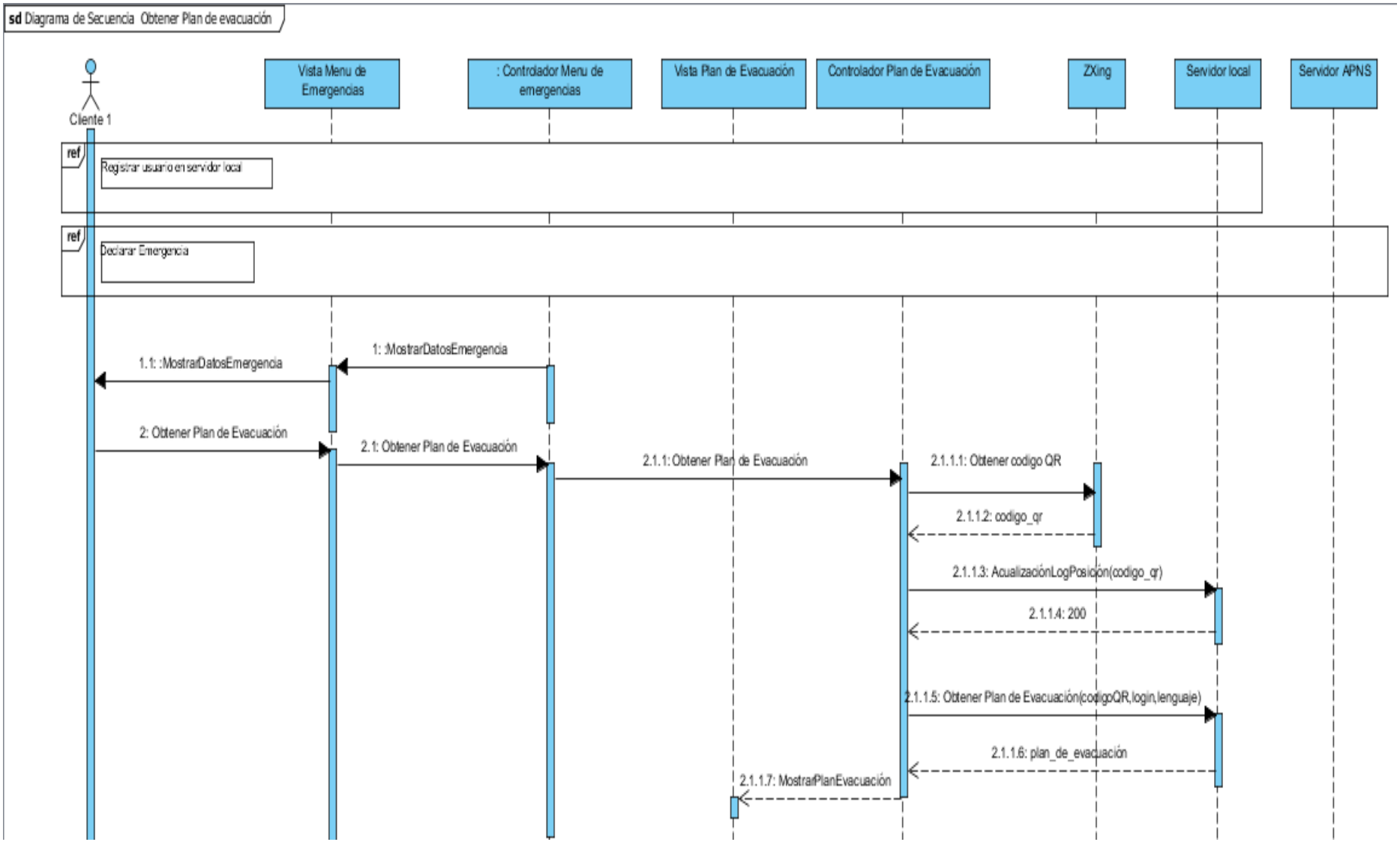
DIAGRAMA DE SECUENCIA - REGISTRAR USUARIO EN SERVIDOR LOCAL:



DIAGRAMAS DE SECUENCIA - DECLARAR EMERGENCIA



DIAGRAMAS DE SECUENCIA - OBTENER PLAN DE EVACUACIÓN



IX. Implementación del asistente personal de seguridad.

Al crear un proyecto en XCode para desarrollar una aplicación iOS se generan automáticamente una serie de ficheros. De ellos, hay dos que se encargan de iniciar los procesos necesarios para el funcionamiento de la aplicación.

Por un lado, los ficheros AppDelegate.h y AppDelegate.m, que gestionan el ciclo de vida de

la aplicación y que poseen los métodos que se ejecutan durante los diferentes estados de la aplicación como puede ser al inicio, al entrar en un estado de inactividad, al volver de él, etc. Por otro lado, se encuentra el fichero storyboard, cuya funcionalidad es nueva en iOS 5 y nos facilitara enormemente la gestión de múltiples pantallas en nuestra App. Anteriormente si teníamos una aplicación que tenía un total de 3 pantallas, en lo referente a la capa de vista, debíamos tener 3 archivos de Interface Builder, donde se elaboraba visualmente la estructura y elementos de cada pantalla. Ahora, con Storyboarding, esos múltiples ficheros de Interface Builder, se sustituyen por un único fichero .storyboard.

En este storyboard no solo podemos ver gráficamente cada una de las vistas de nuestra aplicación, sino que podemos conocer cual es la navegación que se establece entre cada una de ellas.

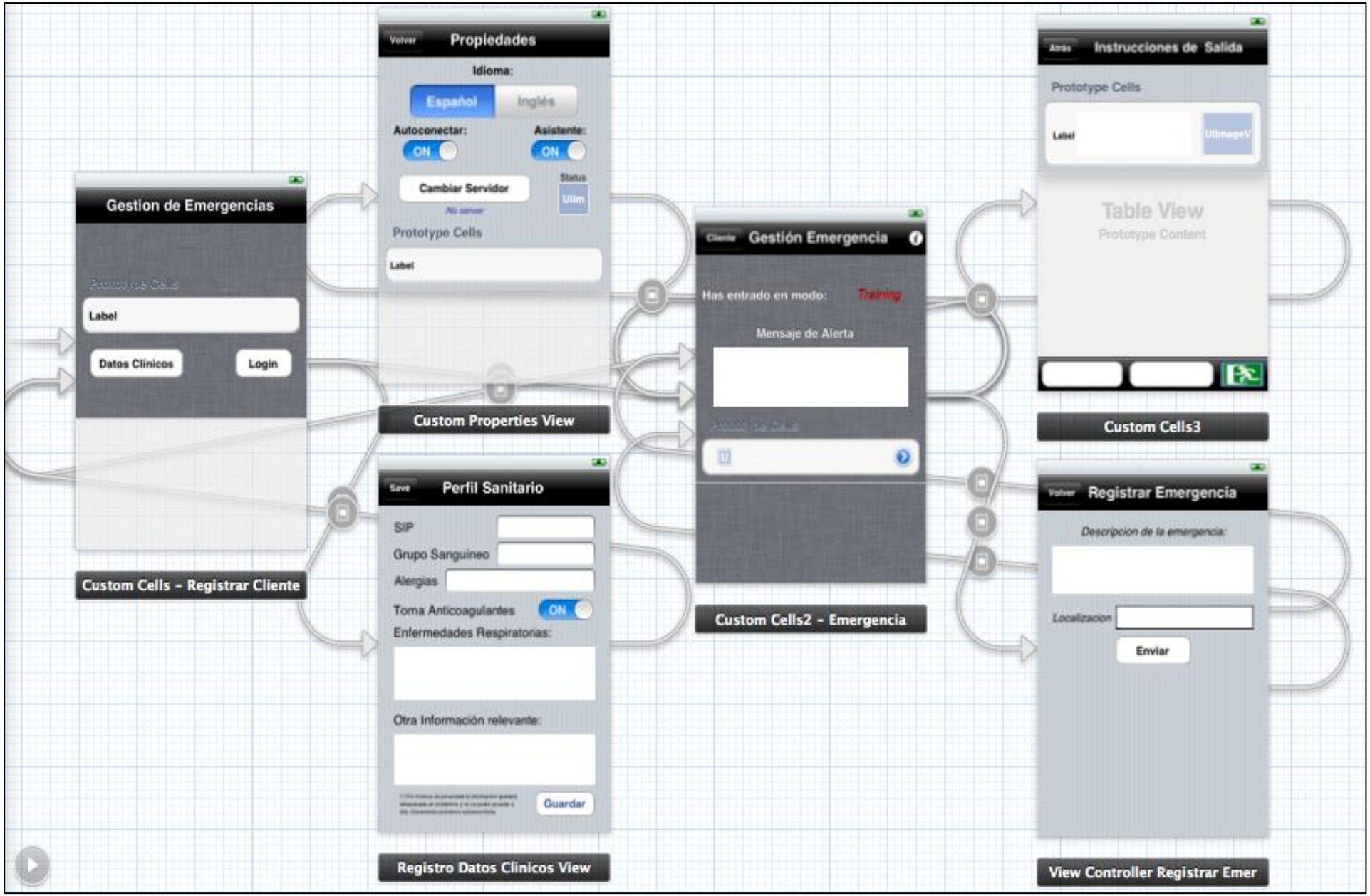
A partir de ahí se van agregando nuevos objetos en el storyboard que dan funcionalidad a la aplicación y que son invocados desde el AppDelegate.

IMPLEMENTACIÓN DE LA APLICACIÓN QUE OFRECE PLANES DE EVACUACIÓN:

En primer lugar, hemos añadido un ViewController o TableViewController en el caso oportuno por cada Vista que tenga la aplicación:

I. Vista para el menú de registro de usuario
II. Vista para el menú de registro del perfil sanitario
III. Vista para el menú principal
IV. Vista para gestionar las propiedades de la aplicación
V. Vista para gestionar los planes de evacuación.

De este modo el .storyBoard de nuestra primera aplicación viene representado por la figura de la siguiente página.


Gestion de Emergencias
Label
Datos Clinicos
Login
Custom Cells - Registrar Cliente
Volver
Propiedades
Idioma:
Español
Inglés
Autoconectar:
ON
Asistente:
ON
Cambiar Servidor
Status
Prototype Cells
Label
Custom Properties View
Save
Perfil Sanitario
SIP
Grupo Sanguíneo
Alergias
Toma Anticoagulantes
ON
Enfermedades Respiratorias:
Otra Información relevante:
Guardar
Registro Datos Clinicos View
Gestión Emergencia
Has entrado en modo:
Training
Mensaje de Alerta
Custom Cells2 - Emergencia
Atrás
Instrucciones de Salida
Prototype Cells
Label
Table View
Prototype Content
Custom Cells3
Volver
Registrar Emergencia
Descripcion de la emergencia:
Localizacion
Enviar
View Controller Registrar Emer

A continuación pasaremos a comentar cada uno de los controladores de nuestra aplicación:

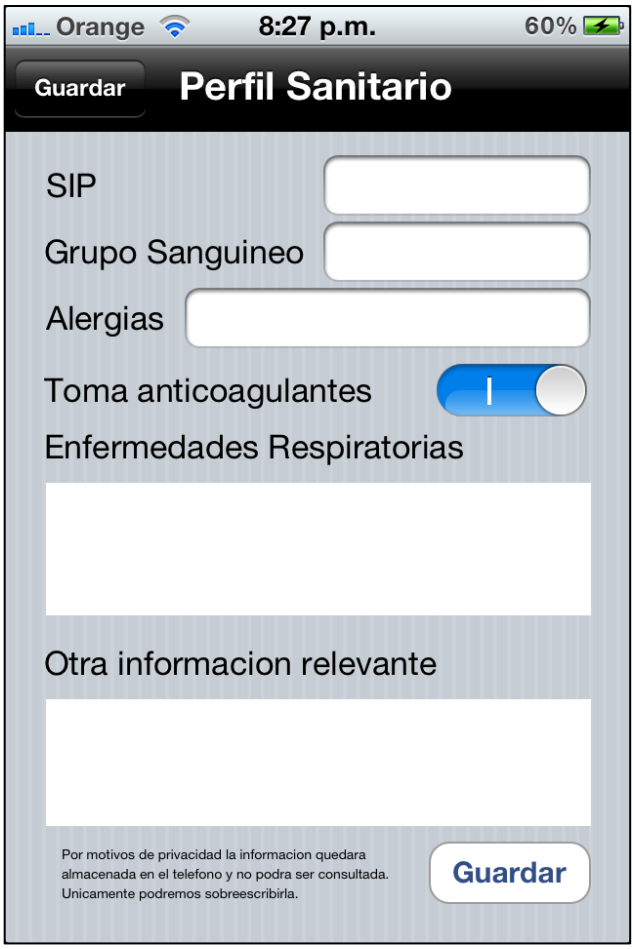
1. REGISTRAR CLIENTE:

Es una clase de tipo *UITableViewController* puesto que mostramos una tabla dinámica en la propia vista.
Dicha tabla contendrá varias celdas del tipo *MyCustomViewCell* (subclase de *UITableViewCell*) las cuales cada una de ellas contendrá un campo título de la etiqueta ('nombre', 'apellidos', 'teléfono' y 'teléfono familiar') que corresponderá a un objeto del tipo UILabel y otro campo valor que almacenará los datos introducidos por teclado. Este campo será del tipo *UITextField*. Además de ello, hemos introducido dos botones (*UIButton*) que permitirán navegar a distintas vistas.

El primero de ellos es Datos Clínicos, que como indica su nombre, permitirá navegar al controlador del perfil sanitario en caso de que el usuario haya introducido correctamente todos los datos por teclado. El segundo *UIButton* (Login) permitirá al usuario registrarse (mediante ASIHTTPRequest utilizando los parámetros introducidos) en caso de no existir sus datos en el servidor global, o modificarlos si el usuario ha cambiado alguno, y pasaremos a continuación al controlador de gestión de emergencias si los datos son validados correctamente. En su defecto, Se mostrarán alguna de las notificaciones de alerta mostradas a nuestra izquierda.

En las próximas páginas, mostraremos una opción que está ubicada en el menú propiedades que permitirá saltarnos esta vista en caso de habernos registrado previamente. De este modo cuando se declara una emergencia no tendremos que pasar por esta vista con lo que conllevaría un ahorro en tiempo al usuario a la hora de proporcionarle el plan de evacuación.

2. PERFIL SANITARIO:



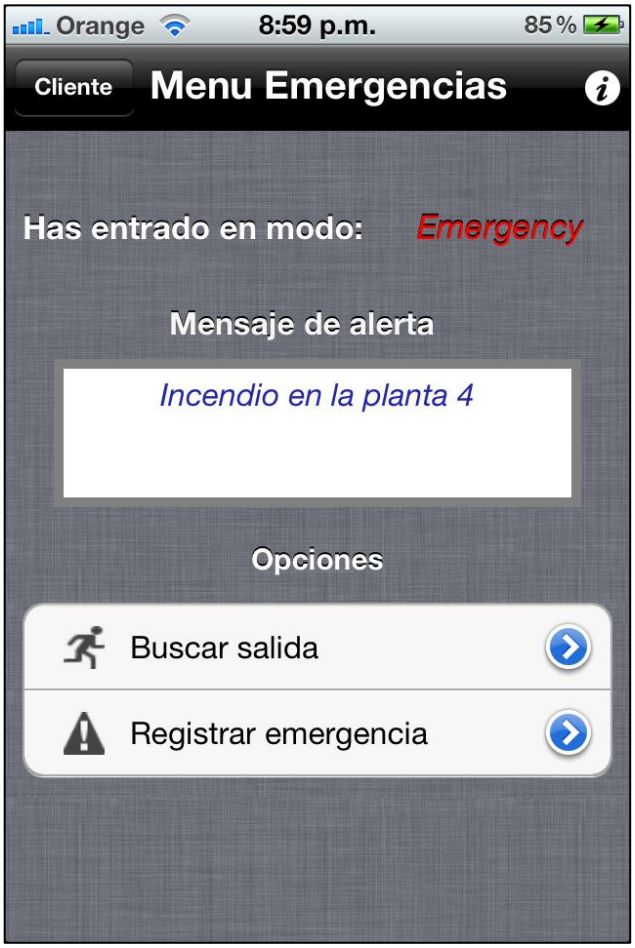
Es una clase de tipo *ViewController.*
Al igual que en la vista anterior cada etiqueta será mostrada mediante un UILabel:

```
UILabel *labelSIP;
UILabel *labelSanguineo;
UILabel *labelAlergias;
UILabel *labelEnf_resp;
UILabel *labelOtrainfo;
UILabel *labelAnticoagulantes;
```

Por otra parte para el almacenamiento de la información utilizaremos diferentes tipos de objetos que serán elegidos dependiendo tanto del tamaño como del tipo de información a introducir.
Para el caso de SIP, grupo sanguíneo y alergias se introducirán los datos en un *UITextField.*
En segundo lugar, para saber si toma anticoagulantes o no bastará con utilizar un *UISwitch* ya que la información que se va a almacenar es del tipo Boolean.
Por ultimo, tanto las enfermedades respiratorias como la información relevante se rellenaran en un *UITextView* ya que el tamaño de la información introducida en ambos campos puede resultar de gran envergadura.

Una vez introducida toda la información en el perfil sanitario presionaremos el botón Guardar(*UIButton*) el cual almacenará dichos datos en el servidor global en caso de que el proceso de validación declaré los datos introducidos como válidos. Si no es así, se mostrará la siguiente notificación al usuario:

3. MENU GESTION EMERGENCIAS:



Es una clase del tipo *UITableViewController* puesto que mostramos una tabla dinámica en la propia vista.

En la parte superior dispondremos de un objeto UINavigationBar que nos permitirá navegar a la ventana anterior (Registrar usuario) mediante el UIBarButtonItem "Cliente". Además, en la parte derecha dispondremos de un botón de ayuda (i) con el cual podremos navegar hasta el controlador que gestiona las propiedades.

Por otra parte, cuando la aplicación reciba una notificación de alerta de emergencia, dispondremos de varios objetos que notificaran al usuario de ello. Estos objetos notificarán tanto el modo en el cual nos hemos registrado (mediante un UILabel) así como el motivo de la emergencia(UITextView).

Como hemos comentado anteriormente, este controlador es del tipo UITableViewController y por tanto tendremos una tabla la cual contendrá dos celdas del tipo MyCustomCell2 (subclase de *UITableViewCell*) y a su vez cada una de ellas contendrá un campo titulo (UILabel) de la celda así como una imagen representativa de la misma (UIImageView). Ambas celdas representaran las operaciones "Buscar salida" y "registrar emergencia" que se activarán gracias a la función:

*- (void)tableView : (UITableView *tableViewdidSelectRowAtIndexPath: (NSIndexPath *)indexPath;*

Este controlador delegará de la clase <ZXing> que contiene los métodos necesarios para el escaneo y la lectura de códigos QR, los cuales serán utilizados al presionar sobre la celda "Buscar salida".

Por último comentar que para que la aplicación sea amigable, se intentará informar al usuario en todo momento de lo que debe de hacer , realizando notificaciones por medio de UIAlertViews . Ejemplos:

4. PROPIEDADES

Es una clase de tipo *ViewController.* Esta vista será utilizada para que el usuario tenga la posibilidad de configurar la aplicación a su gusto.

El idioma de la aplicación será configurable a partir de un UISegmentedControl. Nuestra aplicación deberá de disponer de dos ficheros del tipo Localizable.strings que se encarguen de almacenar los distintos idiomas de la aplicación.

A su vez dispondremos de dos UISwitches que nos aportaran las opciones de autoconectar aplicación (no mostrar el menú registrar usuario al iniciar la aplicación si ya estamos registrados) o Asistente que en caso de estar activado, utilizara la librería FliteTTL ya comentada anteriormente, la cual nos ofrecerá el plan de evacuación vía oral.

Una de las opciones más importantes de este ViewController es el de Cambiar Servidor cuyo procedimiento se llevará a cabo mediante la activación del UIButton "CambiarServidor" el cual, en primer lugar lanzara el lector de códigos QR y una vez decodificada la dirección del servidor, utilizara la librería Reachability.h para monitorizar la red y comprobar si dicho servidor se encuentra online.

El estado de dicho servidor vendrá representado por un UIImageView que podrá representar lo siguiente:

 Conectado al servidor vía WAN.

 Servidor offline.

 Conectado al servidor vía WIFI.

Debido a que nuestro ViewController utiliza los métodos de lectura/decodificación de códigos QR, nuestro controlador deberá delegar de la clase ZXing.

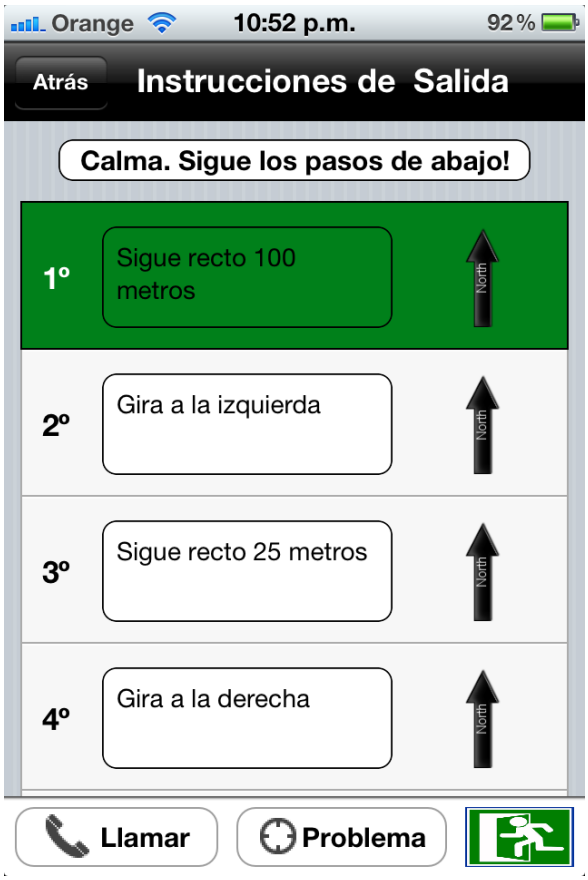
Por ultimo comentar que al final de la vista, se puede observar la tabla del controlador, la cual posee información sobre la tesis así como un video ilustrativo del funcionamiento de la misma.

5. DECLARAR EMERGENCIA

Es una clase del tipo ViewController. Esta vista será utilizada en caso de que el usuario quiera declarar una emergencia.
Para ello, deberá insertar dos campos. Un campo vendrá definido en un UITextView donde el usuario deberá de introducir la descripción de la emergencia. Por otra parte , el segundo campo indicará la localización de la emergencia por ejemplo: 2º planta habitación 4º. Esta localización estará almacenada en un UITextField.
Ahora bien, una vez introducidos los datos y pulsado el UIButton enviar puede que ocurran dos cosas:

- Si el usuario estaba registrado previamente en un servidor local, dicha emergencia será enviada con éxito utilizando el método POST de ASIHTTPRequest.
-
- Si el usuario no estaba previamente registrado en ningún servidor local, se le notificará al usuario por medio de una notificación local.

6. INSTRUCCIONES DE SALIDA



Es una clase de tipo UITableViewController, es decir una vista de tabla, y su función será la de mostrar cada una de las acciones que recibirá del servidor formando al final de ello el plan de evacuación.

Cada celda de la tabla es de la clase MyCustomViewCell3, subclase de UITableViewCell y se mostrará tanto la información como la dirección de dicha acción.

En la parte superior de la vista dispondremos de un UINavigationBar que nos permitirá volver atrás. Justo debajo de la barra de navegación, el usuario recibirá mensajes de apoyo y tranquilidad que facilitaran el transcurso del plan de evacuación al mismo.

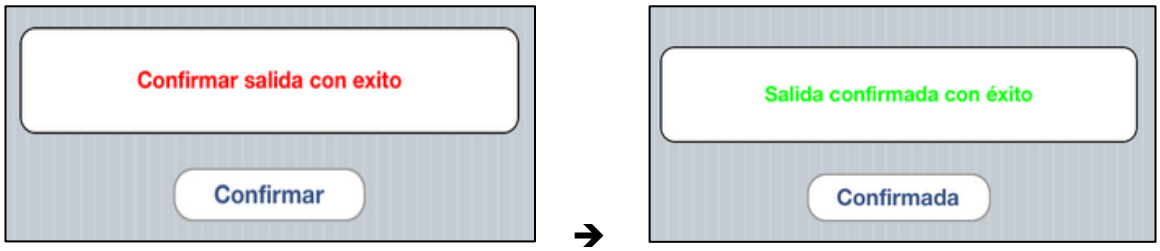
En la parte inferior de la vista podemos observar un UIToolBar donde dispondremos de diferentes acciones representadas por un UIBarButtonItem cada una :

- Llamar : en caso de querer comunicarse con el exterior se hará click en dicho botón y se le ofrecerá al usuario las opciones representadas a continuación:

- Problema: Si durante el seguimiento de un plan de evacuación el usuario se encuentra con algún problema deberá de pulsa dicho botón y elegir la opción que mas le corresponda en su caso.

- Salida Emergencia con éxito: En caso de que el usuario quiera confirmar que ha salido del edificio de forma exitosa podrá realizarlo de dos maneras distintas:

    - Siguiendo todos las acciones de evacuación en orden con lo cual le aparecerá la siguiente ventana la cual tendrá que confirmar:



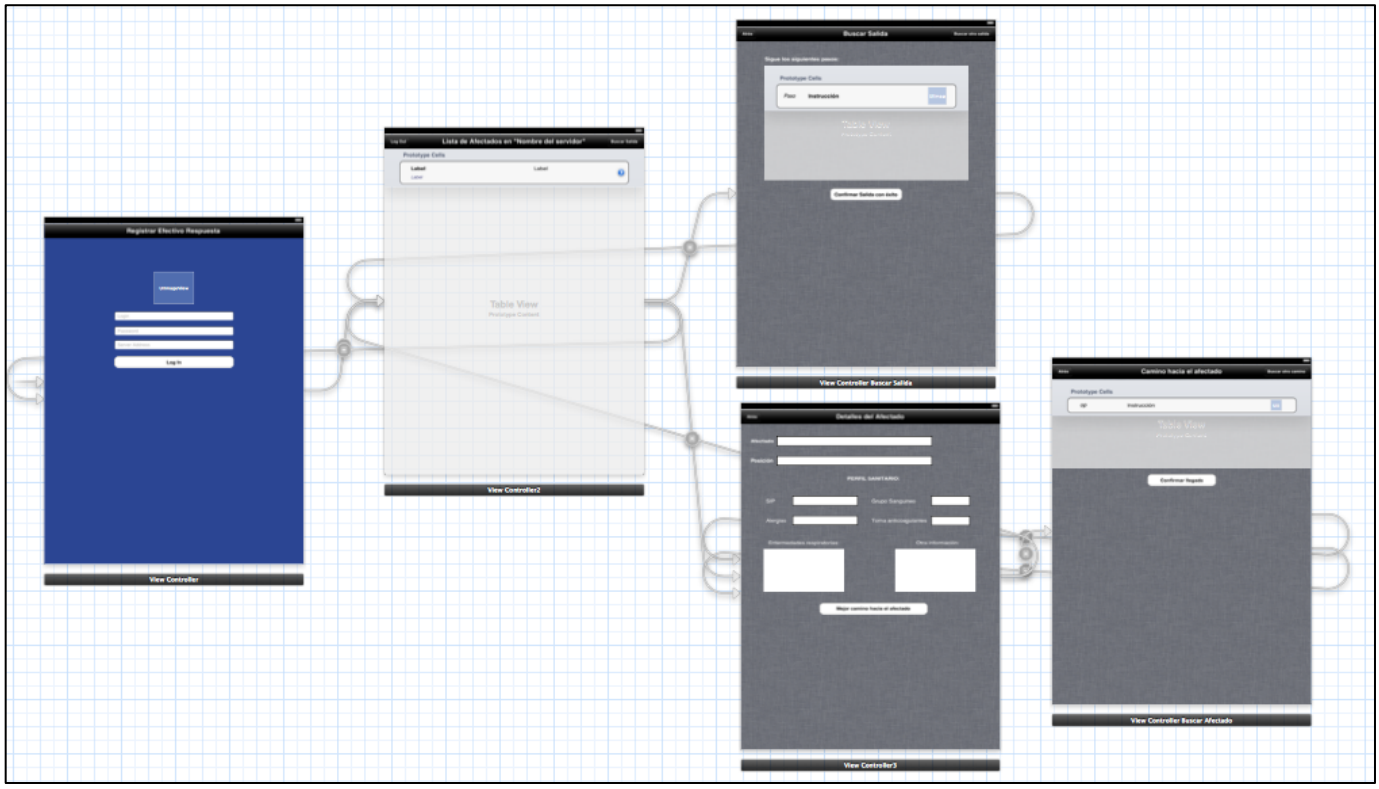
    - Si el usuario decide salir por su propio pie sin hacer caso al plan de evacuación deberá confirmar su salida mediante el botón verde ubicado en la parte inferior derecha de la aplicación. Dicho botón activara la siguiente notificación la cual deberemos de responder que SI en caso de salir con éxito.

IMPLEMENTACIÓN DE LA APLICACIÓN EN iPAD PARA LOS EQUIPOS DE RESCATE:

Al igual que en el la aplicación anterior, hemos añadido un ViewController o TableViewController en el caso oportuno por cada Vista que tenga la aplicación:

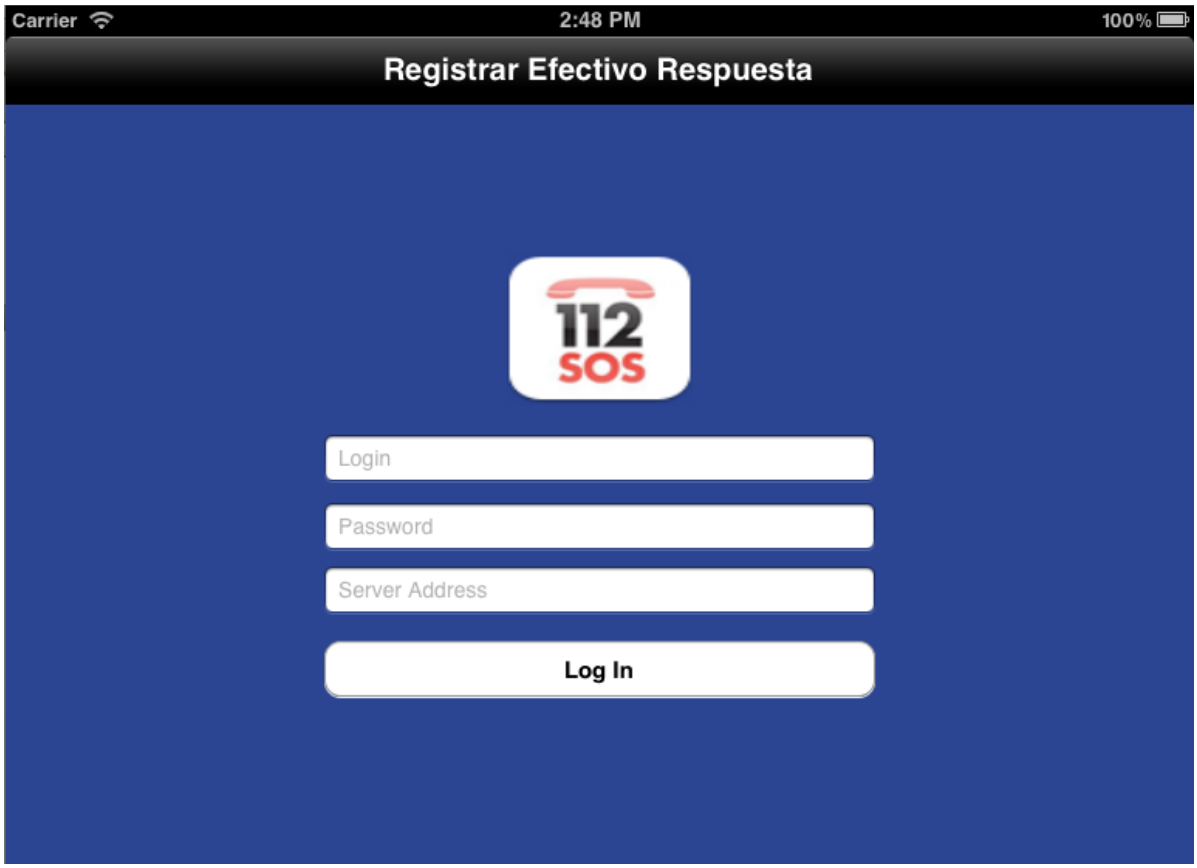
VI. Vista para el menú de registro en un servidor local por parte de un miembro de rescate
VII. Vista para el menú que se encarga de mostrar todos los afectados que se encuentran en el interior del edificio.
VIII. Vista para una ventana que se encargue de buscar una salida en caso de que el miembro de rescate se encuentre en peligro.
IX. Vista para una ventana que muestre los detalles del afectado.
X. Vista para proporcionar planes de rescate personalizados para cada afectado.

De este modo el .storyBoard de nuestra primera aplicación viene representado por la figura de la siguiente página.



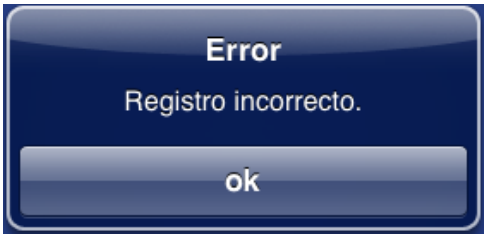
A continuación pasaremos a comentar cada uno de los controladores de nuestra aplicación:

1. REGISTRAR PERSONAL DE RESPUESTA EN SERVIDOR LOCAL:

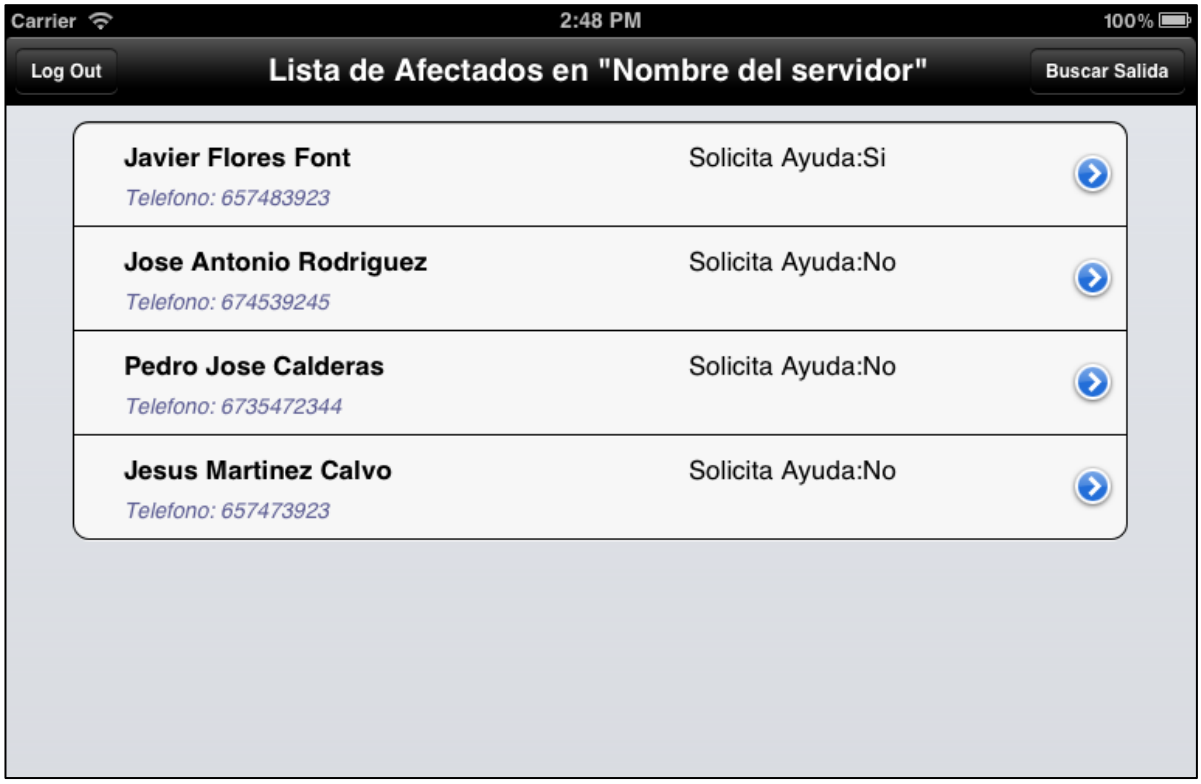


El controlador de esta vista es una clase del tipo *ViewController*. La idea de este controlador es conseguir registrar al personal de respuesta en un servidor local. Esto se consigue introduciendo de forma manual los datos en los *UITextField* (login, password y server address) que observamos en la figura y presionando el UIButton "Login".
Una vez pulsado el botón, a continuación se hara una llamada mediante ASIHTTPRequest al servidor global, y podremos encontrarnos frente a dos situaciones:

- En caso de que el usuario exista y sea parte del equipo de rescate quedará registrado en dicho servidor, quedando esta información grabada tanto en la base de datos del servidor local como en la del global. A continuación, el usuario tendrá acceso tanto al número de afectados que se encuentran en el edificio como a los datos personales de los mismos.
- En caso de que el usuario no exista se notificará mediante un *UIAlertView*:



2. LISTA DE USUARIOS AFECTADOS

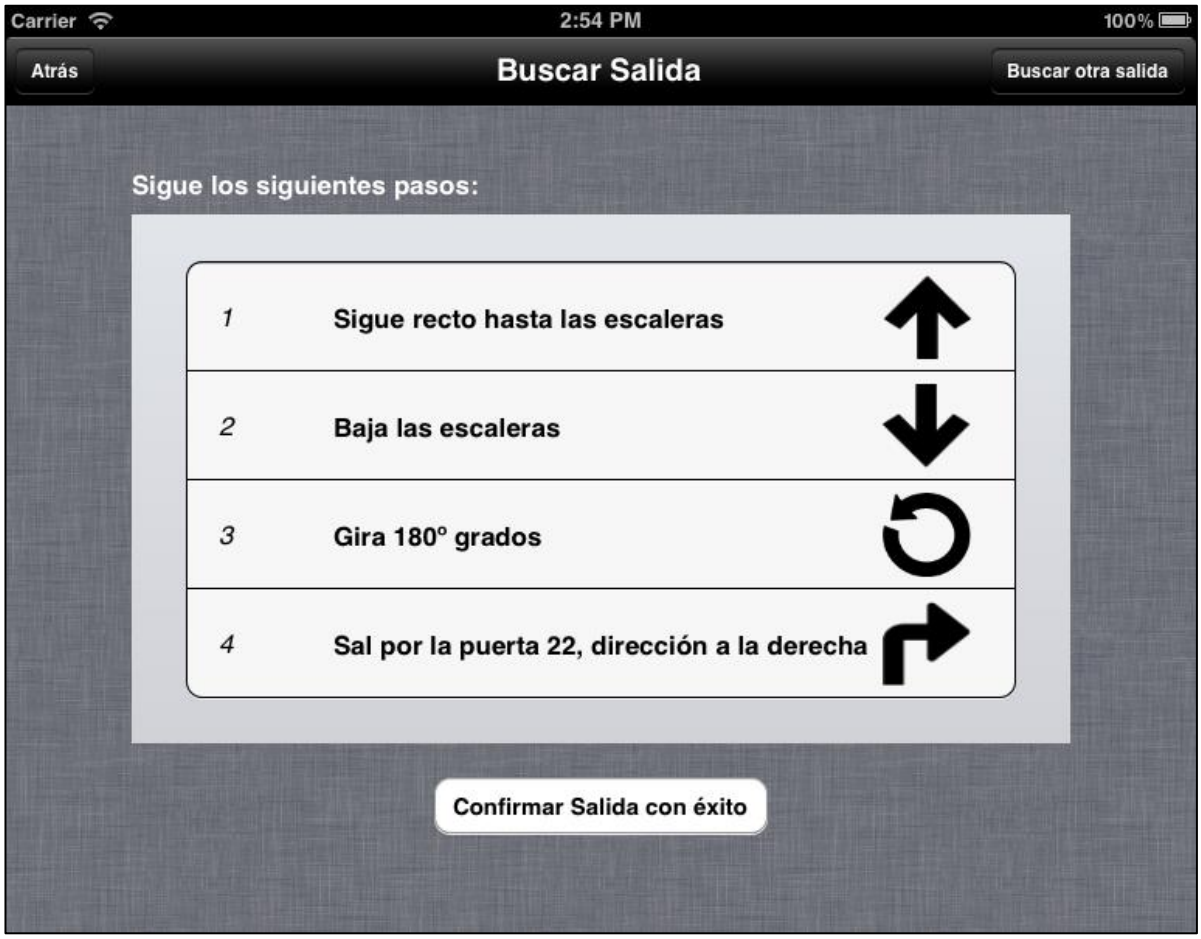


El controlador de esta vista es del tipo UITableViewController.
Una vez registrados los equipos de respuesta en servidor local, esta vista será utilizada para conocer de forma inmediata la situación de los afectados. La información se mostrará mediante una lista ordenada (utilizando un objeto de la clase UITableView y celdas del tipo MyCustomViewCell, subclase de *UITableViewCell*) que contendrá en primer lugar a los usuarios que han solicitado ayuda para que sean atendidos con mayor prioridad y a continuación los que no solicitan ayuda pero continuan también en el interior del edificio.
En caso remoto de que el usuario no se encuentre en su posición o suceda algún imprevisto, tendremos también acceso a su telefono personal (*UILabel*).
Una vez conocida la lista de afectados, podemos diferenciar dos situaciones:

- En caso de que el bombero se encuentre en peligro y decida salir del edificio, tendrá acceso a un objeto de la clase *UIBarButtonItem,* situado en la parte superior derecha ("Buscar Salida") , el cual le ofrecerá un plan de evacuación idéntico al ofrecido a los afectados.
- Por otra parte, si los bomberos hacen click sobre una celda de la lista se ejecutará la siguiente función , la cual se encargará de llamar al controlador de la vista que mostrará los datos del afectado seleccionado.

```
 - (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath
{
      [tableView deselectRowAtIndexPath:indexPath animated:NO];
   [self performSegueWithIdentifier:@"DetalleAfectado" sender:self];
}
```

3 BUSCAR SALIDA POR PARTE DE UN MIEMBRO DE RESCATE



Los bomberos puede que se encuentren en peligro o puede que esten junto a una persona que estan intentando rescatar y no encuentren el camino adecuado hacia la salida. Por este motivo, este controlador le ofrecerá al usuario un plan de evacuación como sucedia en el caso de los afectados aunque sin detalles como por ejemplo el guía por voz, puesto que estamos ante personas cualificadas que no necesitan de dichas opciones.

En primer lugar el framework ZXing nos ayudará a leer y codificar el codigo QR más cercano al bombero, pudiendo de esta forma localizarlo en alguna posición. A continuación, mediante el protocolo ASIHTTPRequest y utilizando el método GET, se realizará una petición al servidor local solicitando un plan de evacuación. El plan de evacuación devuelto por el servidor local y parseado se mostrará en una *UITableView* personalizada cuyas celdas contendran tanto las acciones a realizar como una imagen ilustrativa de ellas.

En caso de que siguiendo un plan de evacuación el bombero se equivocara o tuviera algún tipo de problema, hemos agregado un UIButton "Buscar otra salida" que volveria a utilizar

al framework ZXing y se realizaría la misma consulta al servidor pero utilizando la localización actualizada. De esta forma se le proporcionaría al bombero planes de evacuación alternativos .

4. DETALLES DEL AFECTADO

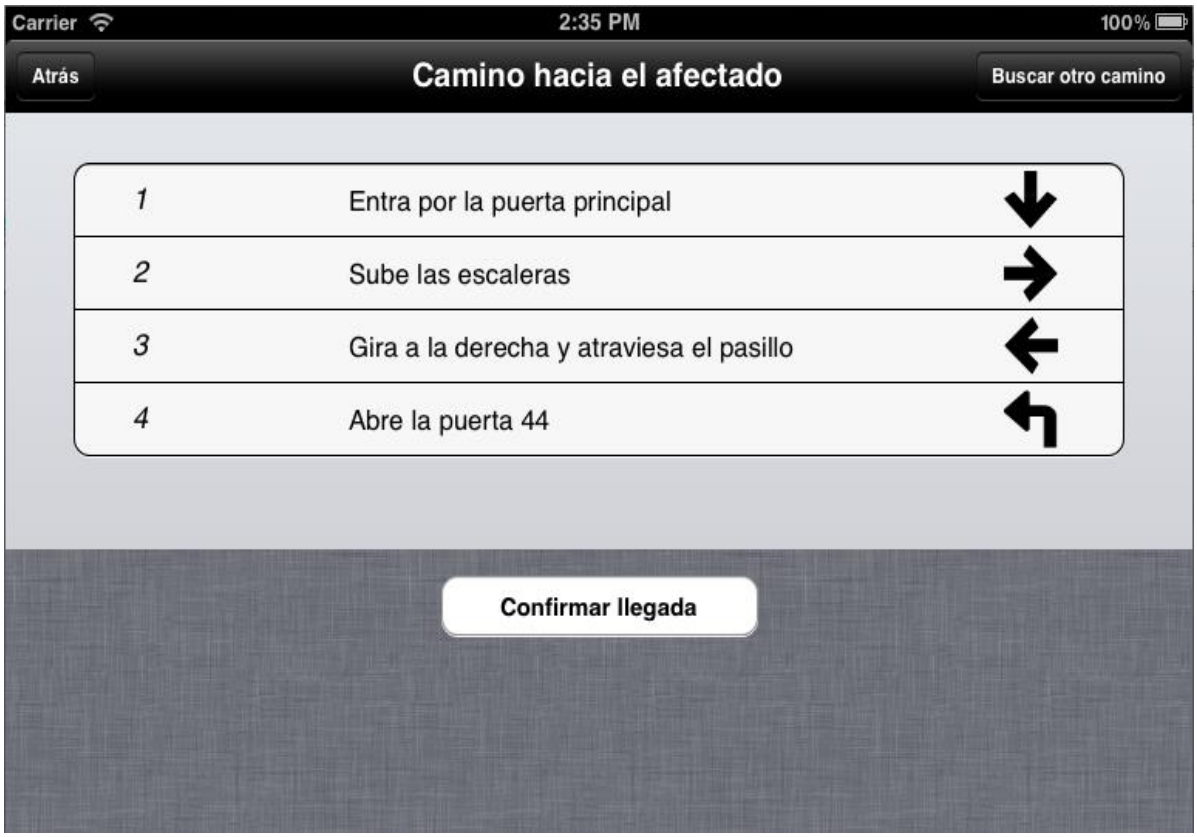
Es una clase del tipo *ViewController*. La idea de este controlador es mostrar información relevante respecto afectado seleccionado. Para ello, se han combinado varios objetos de la clase *UITextField* y varios *UITextView*.

El funcionamiento es el siguiente: El controlador obtiene el identificador del afectado seleccionado en la vista previa y realiza una consulta al servidor local preguntandole por su perfil sanitario. Estos datos se utilizarán para que los bomberos tomen unas medidas o otras durante el rescate, pudiendo llevar con ellos el material clínico necesario en su caso . Los datos obtenidos son parseados y mostrados en la vista. Además, se muestra la localización del afectado para que el bombero tenga conocimiento de la proximidad con respecto al mismo.

En caso de que el bombero decida intervenir en el rescate, pulsará el UIButton "Mejor camino hacia el afectado" en cuyo caso invocará al nuevo controlador indicado para ello.

Si el bombero observa que el afectado no esta a su alcance, pulsará el UIBarButtonItem "atrás" y regresara a la vista anterior eligiendo de esta forma otro candidato.

5. BUSCAR CAMINO HACIA EL AFECTADO



Es una clase del tipo *TableViewController*. La idea de este controlador es la de mostrar un plan de rescate personalizado que ayude al bombero a introducirse en el edificio y conseguir llegar lo más pronto posible al afectado.
El controlador solicitara previamente la localización del bombero utilizando para ello el framework ZXing que nos permite la lectura de los codigos QR. Una vez localizado el bombero y conocida la posicion del afectado, el controlador realiza una consulta al servidor local utilizando el metodo GET y preguntandole sobre el camino más corto entre ambos codigos QR. Por último, el respuesta del servidor es parseada y mostrada ordenadamente en la UITable de la figura.

Durante el seguimiento de un plan de rescate pueden tener lugar dos sucesos:

- Si el bombero consigue localizar al afectado, en ese caso se pulsará el UIButton "Confirmar llegada" el cual modificará la variable de estado del afectado "solicita_ayuda" con valor "NO".

- En caso de perderse o tener algun problema durante el seguimiento, puede obtener otro camino alternativo pulsando sobre el UIBarButtonItem "Buscar otro Camino".

Sección 4: Pruebas

I. Depuración con simulador y dispositivos

Para comprobar que la implementación de ambas aplicaciones cumple con nuestros requisitos hace falta una fase de testeo. Para ello,como hemos comntado en el estado del arte, el XCode dispone de un simulador de dispositivos que simula tanto dispositivos iPhone como iPad.

Durante la etapa de aprendizaje y el inicio de la implementación de la aplicación, el simulador ofrece los recursos necesarios para testearla, en cambio, a medida que aumenta el codigo y con ello la complejidad de la aplicación añadiendo nuevas funciones, puede suceder que el compilador nos de errores de código. Como hemos comentado en la seccion "Estado del arte" esto puede es debido a que el simulador se basa en ARM-7 o ARM-8.

El simulador ofrece prácticamente las mismas prestaciones que un dispositivo real, salvo algunas limitaciones:

- Gráfico 3D OpenGL
- Cámara UIImagePickerController
- Acelerómetro
- Interfaz multi-touch
- Localización
- Vibración

En nuestro caso, el simulador nos ha resultado útil a la hora de testear las vistas y comprobar la navegación entre las mismas. El problema surge cuando se requiere por ejemplo el uso de la camara integrada para leer codigos QR o incluso para testear el funcionamiento de las notificaciones PUSH.
Por tanto, comentar que el simulador no dispone de la capacidad para testear nuestra aplicación al completo y por ello, resulta imprescindible a la larga el uso de dispositivos reales para la fase de pruebas.

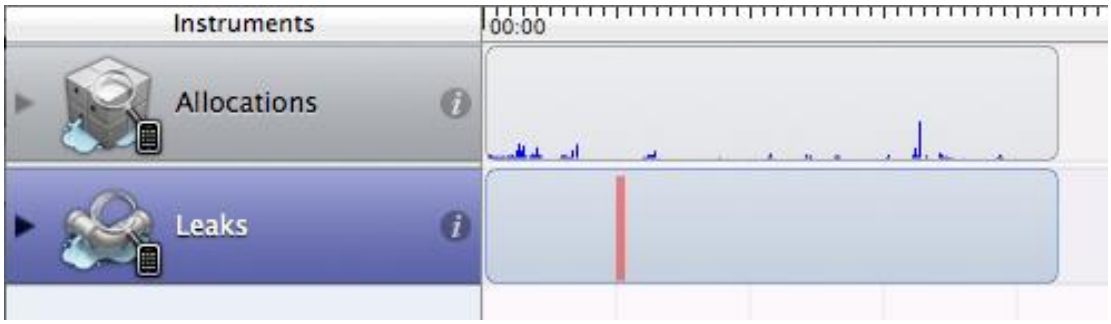
Hemos tenido a nuestra disposición un dispositivo movil iPhone 4S con el cual hemos podido testear la aplicación para los afectados. De este modo, nos hemos podido asegurar que dicha aplicación funciona correctamente y responde según nuestras expectativas.
En cambio, la aplicación para los equipos de respuesta no ha podido ser testeada en ningún dispositivo físico y hay segmentos de codigo comentado que a pesar de que funcionan en la aplicación para iPhone, no garantizamos su funcionamiento en iPad.

I. Monitorización con instruments

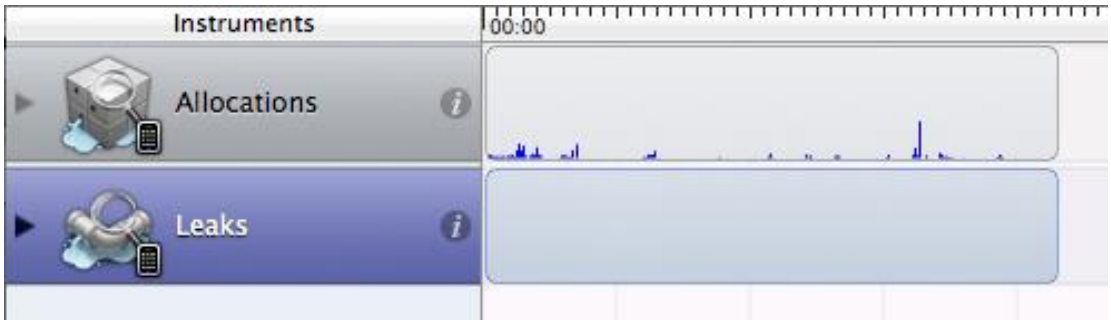
La siguiente herramienta que analizaremos se llama Instruments.
Instruments es una herramienta para comprobar el rendimiento de la aplicación y realizar análisis y pruebas de forma dinámica sobre el codigo iOS. Es una herramienta flexible y potente que permite rastrear uno o más procesos y analizar los datos recogidos. De esta manera, Instruments ayuda a entender el comportamiento de las aplicaciones de usuario y el sistema operativo.

El uso habitual que se hace de Instruments en iOS es la detección de fugas de memoria (Memory Leaks). Una fuga de memoria se da cuando la memoria es asignada por una aplicación pero nunca se libera. La forma en que Instruments sabe cuándo la aplicación tiene fugas de memoria, es mediante el control de todas las asignaciones de memoria que hace la aplicación y los punteros a dicha memoria. En el momento en que una aplicación no disponga de un puntero válido a la memoria que tenga asignada, Instruments sabe que dicha aplicación no puede liberar memoria y por consiguiente, ha tenido una fuga de memoria.

Ejecutando instruments sobre nuestra primera aplicación hemos observado la siguiente gráfica:

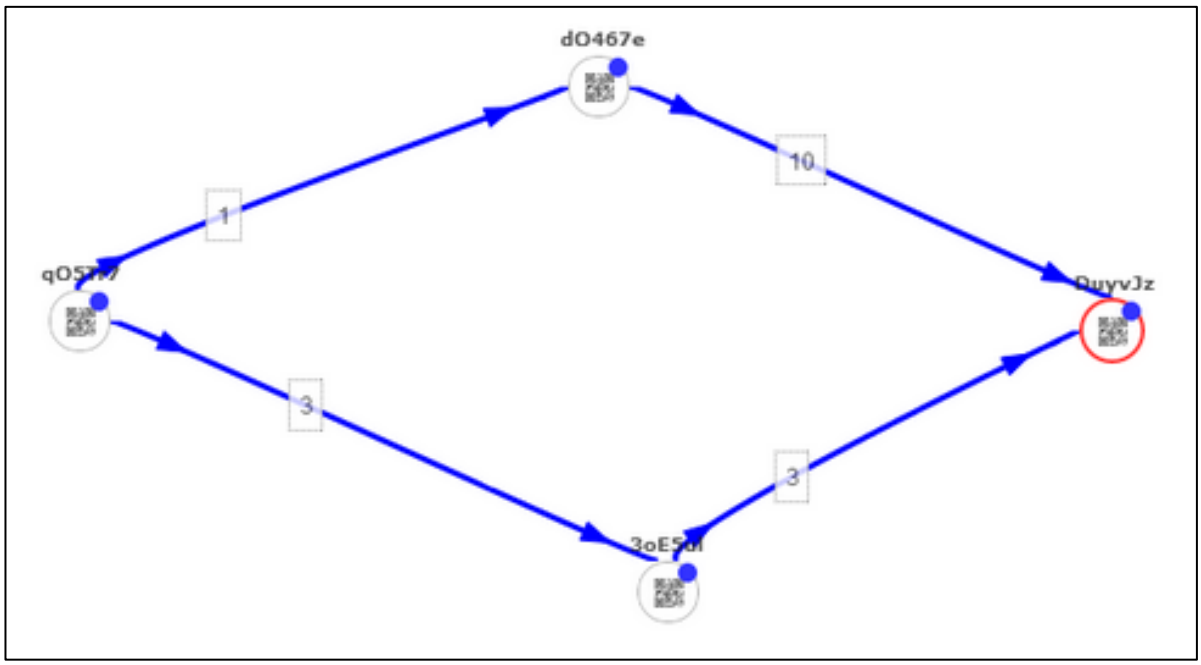


Como podremos comprobar, nuestra aplicación para los afectados contenía 1.47KBytes de fugas de memoria o "Leaks" y gracias a esta herramienta las hemos conseguido eliminar satisfactoriamente quedando de este modo:



II. Pruebas sobre un ejemplo

La finalidad de las pruebas en el dispositivo móvil es comprobar que nuestro asistente personal de seguridad responde adecuadamente frente a una emergencia.
Dado que disponemos únicamente de un dispositivo iPhone vamos a comprobar sobre el mismo que la aplicación responde correctamente con el ejemplo que vamos a proponer:
Inicialmente, hemos partido del siguiente plano modelado mediante un generador de planos desarrollado por la tesis complementaria , donde observamos los diferentes nodos incluido el nodo final (en rojo).



Además,por cada par de codigos QR hemos añadido las siguientes acciones

Codigo QR inicial	Codigo QR final	acciones
qO5Tr7	q077w7	• Sigue recto 100 metros • Gira a la izquierda
qO5Tr7	d0467e	• Baja las escaleras • Sigue recto hasta el extintor
d0467e	DuyvJz	• Abre la puerta 32 • Baja por las escaleras hasta la salida.
3oE5ul	DuyvJz	• Sigue recto 25 metros • Gira a la derecha

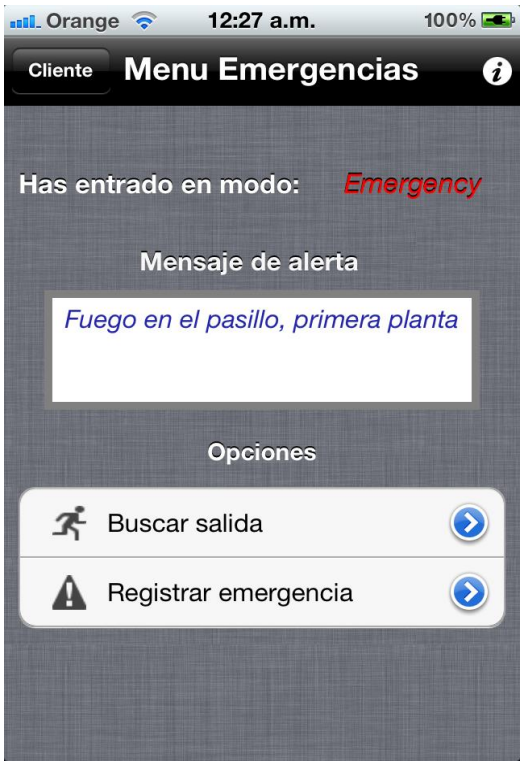
Vamos a declarar una emergencia y manualmente desde el servidor marcaremos el tramo correspondiente por los códigos QR " q077w7" y " d0467e " como impracticable. De esta forma, un usuario que se localiza en la posición "q077w7" deberá de obtener el plan de evacuación comprendido por los codigos QR: [qO5Tr7 ->3oE5ul -> DuyvJz].

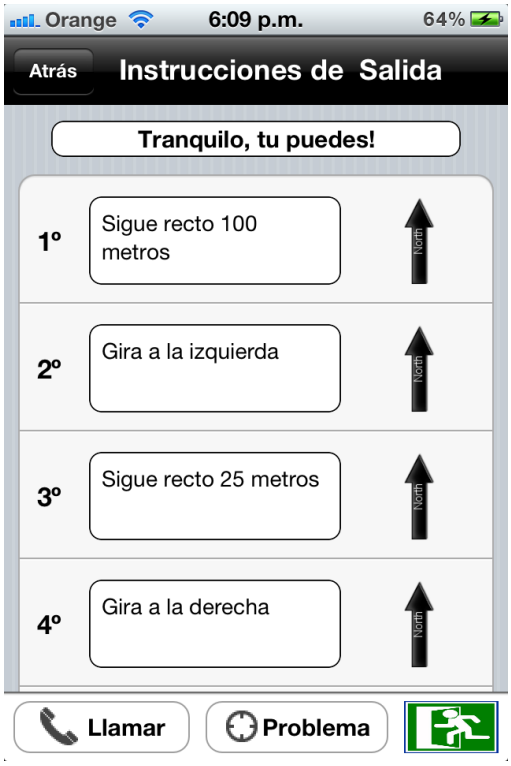
1.

2. Declaración de la emergencia:

Notificación de la emergencia

Al abrir la notificación nos encontraramos frente a la siguente ventana la cual nos muestra tanto el motivo de la alerta como su ubicación. En este momento el usuario hará click sobre "Buscar Salida" mostrando la figura siguiente.





Como podemos comprobar, la aplicación devuelve una lista con las acciones correctas a realizar, es decir, las acciones que corresponden con el plan de evacuación descrito anteriormente [qO5Tr7 ->3oE5ul -> DuyvJz].

Sección 5: Conclusiones

Respecto al trabajo realizado:

En el presente trabajo de investigación hemos realizado el análisis y diseño de un sistema de información para la gestión de emergencias, dando soporte al almacenamiento de la información y de gestión de toda la información referente al contexto de las emergencias.

Para hacer uso de esta información se han implementado dos aplicaciones que resultarían de gran ayuda en el ámbito de las emergencias.

El propósito de la primera aplicación es proporcionar al usuario toda la información que necesita de una forma rápida, sencilla e intuitiva ofreciéndole planes de evacuación personalizados dependiendo tanto de la posición como de su estado físico. Esta aplicación estaría enfocada a los afectados de un edificio.

En cambio, la segunda aplicación facilitaría la tarea a los equipos de respuesta proporcionándoles entre otras cosas, información sobre la localización de los afectados.

Una vez concluido el desarrollo de ambas aplicaciones podemos afirmar que los objetivos previstos inicialmente se han cumplido satisfactoriamente.

Por último comentar que se ha diseñado tambien todo el esquema de información y la base de datos del mismo utilizando mysql como lenguaje de base de datos.

Respeto al desarrollo de aplicaciones móviles para iPhone:

Comentar que mis conocimientos sobre desarollo de aplicaciones para iPhone cuando empezé la tesis eran prácticamente nulos y esta tesis me ha ayudado en parte a ampliar mis conocimientos utilizando gran cantidad del tiempo en aprender el lenguaje Objective-C.

Por últmo decir que el entorno que ofrece Apple de ayuda, junto con los foros de desarrolladores, etc... me ha facilitado en gran medida el desarrollo del asistente personal de seguridad.

SECCION 6: Trabajo Futuro

A continuación se muestra una lista con las posibles mejoras que se podrían incorporar a la larga en nuestro trabajo de tesis:

- Trabajar con información de contexto: Consistirá en manejar la información de un mejor modo, es decir, que se tengan en cuenta muchos conceptos a la hora de proporcionar planes de evacuación. Por ejemplo, si un usuario durante el seguimiento de un plan de evacuación alguien marca un tramo como afectado, en caso de verse afectado, la aplicación deberá actualizar automáticamente el plan de evacuación avisándole al mismo, sin que el usuario tenga que cerciorarse de que dicho tramo esta afectado. Otro punto a tener en cuenta será que los planes de evacuación suministrados dependan del tipo de perfil sanitario de cada afectado. Por ejemplo en caso de tratarse de un invalido, que éste mismo no tenga que indicar que está atrapado y que solicita ayuda sino que automáticamente se le declare de este modo.

- Informar de mas personas: Durante una emergencia, un usuario que utilice la aplicación para los afectados podrá avisar al equipo de emergencias que en interior se encuentran más personas, pudiendo indicar el estado de cada una de ellas. Bien es cierto que en nuestra aplicación hemos incorporado una opción de envio de mensajes con los equipos de respuesta, pero debería de haber algún modo más eficiente de realizar dicho informe.

- Ordenación de los afectados por proximidad: En la aplicación para los equipos de rescate se listan los usuarios basándose en el campo "solicita_ayuda". La idea es que podríamos incorporar en el propio sistema de emergencias dos variables por cada código QR que indiquen la posición en coordenadas (x,y) del mismo pudiendo, de esta forma, ordenar a los afectados por proximidad con respecto al bombero ,en vez de ser el propio bombero el que tenga que comprobar si un afectado se encuentra cerca de su posición.

- Mostrar un plano/mapa ilustrativo de la situación en el interior del edificio en directo: Incorporando las dos variables descritas en el último punto, podríamos de algún modo utilizando alguna de las librerías de dibujo que dispone iOS (como por ejemplo OpenGL) pintar un plano que muestre la posición de cada uno de los afectados en el interior del edificio.

- No introducir el teléfono de forma manual: La identificación de los usuarios la realizamos mediante su numero de teléfono, pero éste mismo, lo introducimos en la aplicación de forma manual. Esto podría dar lugar a que cualquier persona podría modificar

los datos de otra meramente introduciendo el teléfono de la misma. Con el fin de prevenir riesgos como por ejemplo que un usuario se haga pasar por otro o situaciones parecidas, habría que intentar capturar el numero de teléfono de forma automática. Este concepto no esta implementado en el SDK de iPhone luego habría que incorporar algún servicio especial como por ejemplo el que utiliza WhatsApp para la identificación de usuarios.

- Pruebas, simulacros: Con el fin de mejorar la usabilidad de la aplicación y ver posibles los posibles fallos cometidos, habría que hacer un estudio sobre un colectivo de personas y ver como se desenvuelven ante un simulacro de emergencia. Durante el transcurso de la tesis, se han mantenido contactos con el Observatorio Psicosocial de Recursos en Situaciones de Desastre (OPSIDE) perteneciente a la Universidad Jaume I de Castellón, los cuales nos podrían ayudar en dicha tarea.

- Privacidad de contenidos: Habría que realizar un estudio sobre que contenidos son de carácter privado y cuales no. Es decir, que información es la que se debe mostrar a cada usuario dependiendo de su rol y además que información debe estar protegida por contraseña.

- Estudio sobre los datos del perfil sanitario: Los datos introducidos en el perfil sanitario han sido verificados por un medico especialista. En cambio, hay otros datos que no aparecen que suponemos que también deberían de ser proporcionados por ejemplo:

- edad
- invalidez
- ceguera
- diabetes
- etc...

Por este motivo, habría que continuar investigando en este sentido.

- Portar el asistente personal de seguridad a otras plataformas: Aunque bien es cierto que iPhone y Android son dos de las plataformas más grandes y innovadoras actualmente, sería bueno también portar el asistente personal de seguridad a otras con el fin de ampliar el numero de usuarios del mismo.

Sección 6: Bibliografía

Enlaces relacionados en la sección "Tecnología empleada":

- **Economía de las plataformas móviles, revista Vision Mobile, June 2012**
  http://www.visionmobile.com/product/developer-economics-2012

- **Nokia, plataforma en "llamas", revista Engadged, Febrero 2011**
  http://www.engadget.com/2011/02/08/nokia-ceo-stephen-elop-rallies-troops-in-brutally-honest-burnin/ http://www.arturogoga.com/2012/06/20/windows-phone-8-la-siguiente-evolucion-de-la-plataforma-movil-de-microsoft/

- **Blog de móviles, categoría Android. Septiembre de 2012.**
  http://www.blogdemoviles.com.ar/category/sistema-operativo-android/

- **Página web "AndroidAyuda", artículo "Jelly Bean ya está aquí", 27 de Junio de 2012**
  http://androidayuda.com/2012/06/27/android-4-1-jelly-bean-ya-esta-aqui/

- **Introducción al IDE de XCode, Objective C e iOS, Mayo 2011.**
  http://www.programadorphp.org/blog/cursos/introduccion-al-ide-de-xcode-objective-c-ios/

- **Página oficial XCode.es, articulo "Guardando los ajustes con la clase NSUserDefaults"**
  http://xcode.es/saving-settings-using-nsuserdefaults-class

Enlaces relacionados en la sección "Arquitectura de la solución":

- **Página oficial de Apple para desarrolladores**
  https://developer.apple.com/

- **Primeros pasos como iOS developer, artículo escrito por Esteban Estayo, 18 de Febrero de 2012**
  http://www.estebanetayo.es/2012/02/18/primeros-pasos-como-ios-developer/

- **Depuración de errores:**
  http://www.stackoverflow.com/

- **Guía para leer un XML desde una aplicación iPhone. Daniel Navarro Murillo. 15 de Julio de 2011.**
  http://www.danielnavarroymas.com/aplicaciones-moviles/leer-un-xml-en-una-aplicacion-iphone/

- **Artículo "How to Enable and Use Text to Speech on iPhone & iPad", 30 de Mayo de 2012.**
  http://osxdaily.com/2012/05/30/text-to-speech-iphone-ipad/

- **Creación de iconos:**
  http://www.quirco.com/iPhoneIcon

- **Tutorial : UITableViewCell in Storyboard: Static, Dynamic & Custom cells (parte 1 y 2). Enero de 2012.**
  http://www.youtube.com/watch?v=fnzkcV_XUw8
  http://www.youtube.com/watch?v=0AsChJk422c

- **Lectura y decodigicación de códigos QR.**
  http://code.google.com/p/zxing/

- **Parte del servidor Apple Push Notifications en Java:**
  https://github.com/notnoop/java-apns

- **Parte del cliente Apple Push Notification en Objective-C**
  http://developer.apple.com/library/mac/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html

- **Comparación Rest VS Soap:**
  http://petra.euitio.uniovi.es/~i1893878/pmwiki/pmwiki.php?n=ServiciosWeb.RESTVsSOAP

- **Foros de ayuda**
  hhtp://www.iphonedevsdk.com

Enlaces relacionados en la sección "Pruebas":

- **Artículo " iPhone Simulator Limitation in Development", 24 de Marzo del 2009**
  http://webbuilders.wordpress.com/2009/03/24/iphone-simulator/

- **Guía sobre el uso de instruments:**
  https://developer.apple.com/library/mac/#documentation/DeveloperTools/Conceptual/InstrumentsUserGuide/Introduction/Introduction.html

- **Enlace a la página oficial del servidor:**
  https://www.cpanel.tfmemergency.com/

ANEXO:

SCRIPT DE LA BASE DE DATOS EN MYSQL:

- **BASE DE DATOS DEL SERVIDOR LOCAL:**

```
-- -----------------------------------------------------
-- Tabla CodigoQR
-- -----------------------------------------------------

CREATE  TABLE IF NOT EXISTS `mydb`.`CodigoQR` (
  `identificador` INT NOT NULL ,
  `descripcion` VARCHAR(45) NULL ,
  PRIMARY KEY (`identificador`) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Tabla UsuarioRegistrado
-- -----------------------------------------------------

CREATE  TABLE IF NOT EXISTS `mydb`.`UsuarioRegistrado` (
  `identificador` INT NOT NULL ,
  `CodigoQR_identificador` INT NULL ,
  `clave_movil` VARCHAR(200) NOT NULL ,
  PRIMARY KEY (`identificador`) ,
  INDEX `fk_Usuario_CodigoQR1` (`CodigoQR_identificador` ASC) ,
  CONSTRAINT `fk_Usuario_CodigoQR1`
    FOREIGN KEY (`CodigoQR_identificador` )
    REFERENCES `mydb`.`CodigoQR` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Tabla Log
-- -----------------------------------------------------

CREATE  TABLE IF NOT EXISTS `mydb`.`Log` (
  `identificador` INT NOT NULL AUTO_INCREMENT ,
  `codigo` VARCHAR(45) NULL ,
  `descripcion` VARCHAR(100) NULL ,
  `fecha` DATETIME NULL ,
  `id_efectivorespuesta` INT NULL ,
  `id_usuarioregistrado` INT NULL ,
  PRIMARY KEY (`identificador`) )
```

```
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Tabla EfectivoRespuestaRegistrado
-- -----------------------------------------------------

CREATE  TABLE IF NOT EXISTS `mydb`.`EfectivoRespuestaRegistrado` (
  `identificador` INT NOT NULL ,
  `CodigoQR_identificador` INT NULL ,
  `clave_movil` VARCHAR(200) NOT NULL ,
  PRIMARY KEY (`identificador`) ,
  INDEX `fk_EfectivoRespuesta_CodigoQR1` (`CodigoQR_identificador` ASC) ,
  CONSTRAINT `fk_EfectivoRespuesta_CodigoQR1`
    FOREIGN KEY (`CodigoQR_identificador` )
    REFERENCES `mydb`.`CodigoQR` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `mydb`.`Emergencia`
-- -----------------------------------------------------

CREATE  TABLE IF NOT EXISTS `mydb`.`Emergencia` (
  `identificador` INT NOT NULL ,
  `motivo` VARCHAR(45) NOT NULL ,
  `fecha` DATETIME NOT NULL ,
  PRIMARY KEY (`identificador`) )
ENGINE = InnoDB;
```

- **BASE DE DATOS DEL SERVIDOR GLOBAL:**

```
-- -----------------------------------------------------
-- Tabla Organización
-- -----------------------------------------------------

CREATE  TABLE IF NOT EXISTS `mydb`.`Organización` (
  `identificador` INT NOT NULL ,
  `Nombre` VARCHAR(45) NULL ,
  PRIMARY KEY (`identificador`) )
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Tabla  ServidorLocal
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`ServidorLocal` ;

CREATE  TABLE IF NOT EXISTS `mydb`.`ServidorLocal` (
  `identificador` INT NOT NULL ,
  `dirección` VARCHAR(45) NOT NULL ,
  `estado` INT NULL ,
  `Organización_identificador` INT NOT NULL ,
  PRIMARY KEY (`identificador`) ,
  INDEX `fk_ServidorLocal_Organización1` (`Organización_identificador` ASC) ,
  CONSTRAINT `fk_ServidorLocal_Organización1`
    FOREIGN KEY (`Organización_identificador` )
    REFERENCES `mydb`.`Organización` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Tabla CodigoQR
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`CodigoQR` ;

CREATE  TABLE IF NOT EXISTS `mydb`.`CodigoQR` (
  `identificador` INT NOT NULL ,
  `descripcion` VARCHAR(45) NULL ,
  `ServidorLocal_identificador` INT NOT NULL ,
  PRIMARY KEY (`identificador`) ,
  INDEX `fk_CodigoQR_ServidorLocal1` (`ServidorLocal_identificador` ASC) ,
  CONSTRAINT `fk_CodigoQR_ServidorLocal1`
    FOREIGN KEY (`ServidorLocal_identificador` )
    REFERENCES `mydb`.`ServidorLocal` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Tabla Usuario
-- -----------------------------------------------------
DROP TABLE IF EXISTS `mydb`.`Usuario` ;

CREATE  TABLE IF NOT EXISTS `mydb`.`Usuario` (
  `identificador` INT NOT NULL ,
  `nombre` VARCHAR(45) NOT NULL DEFAULT 'No Name' ,
  `apellidos` VARCHAR(45) NOT NULL DEFAULT 'No Surname' ,
  `telefono_familiar` VARCHAR(45) NOT NULL DEFAULT 'No Familiar Phone' ,
  `estado` INT NULL DEFAULT TRUE ,
  `clave_movil` VARCHAR(200) NOT NULL DEFAULT 'No clave' ,
  `operatingsystem` INT NOT NULL ,
  `CodigoQR_identificador` INT NULL ,
  `ServidorLocal_identificador` INT NULL ,
  PRIMARY KEY (`identificador`) ,
  INDEX `fk_Usuario_CodigoQR1` (`CodigoQR_identificador` ASC) ,
  INDEX `fk_Usuario_ServidorLocal1` (`ServidorLocal_identificador` ASC) ,
  CONSTRAINT `fk_Usuario_CodigoQR1`
    FOREIGN KEY (`CodigoQR_identificador` )
    REFERENCES `mydb`.`CodigoQR` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Usuario_ServidorLocal1`
    FOREIGN KEY (`ServidorLocal_identificador` )
    REFERENCES `mydb`.`ServidorLocal` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Tabla EfectivoRespuesta
-- -----------------------------------------------------

CREATE  TABLE IF NOT EXISTS `mydb`.`EfectivoRespuesta` (
  `identificador` INT NOT NULL ,
  `login` VARCHAR(45) NOT NULL ,
  `password` VARCHAR(45) NOT NULL ,
  `CodigoQR_identificador` INT NULL ,
  `ServidorLocal_identificador` INT NULL ,
  `clave_movil` VARCHAR(200) NOT NULL ,
  PRIMARY KEY (`identificador`) ,
  INDEX `fk_EfectivoRespuesta_CodigoQR1` (`CodigoQR_identificador` ASC) ,
  INDEX `fk_EfectivoRespuesta_ServidorLocal1` (`ServidorLocal_identificador`
ASC) ,
  CONSTRAINT `fk_EfectivoRespuesta_CodigoQR1`
    FOREIGN KEY (`CodigoQR_identificador` )
    REFERENCES `mydb`.`CodigoQR` (`identificador` )
```

```
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_EfectivoRespuesta_ServidorLocal1`
    FOREIGN KEY (`ServidorLocal_identificador` )
    REFERENCES `mydb`.`ServidorLocal` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Tabla Log
-- -----------------------------------------------------

CREATE  TABLE IF NOT EXISTS `mydb`.`Log` (
  `identificador` INT NOT NULL AUTO_INCREMENT ,
  `codigo` VARCHAR(45) NULL ,
  `descripcion` VARCHAR(100) NULL ,
  `fecha` DATETIME NULL ,
  `Usuario_identificador` INT NULL ,
  `EfectivoRespuesta_identificador` INT NULL ,
  `ServidorLocal_identificador` INT NOT NULL ,
  PRIMARY KEY (`identificador`) ,
  INDEX `fk_Log_Usuario1` (`Usuario_identificador` ASC) ,
  INDEX `fk_Log_EfectivoRespuesta1` (`EfectivoRespuesta_identificador` ASC) ,
  INDEX `fk_Log_ServidorLocal1` (`ServidorLocal_identificador` ASC) ,
  CONSTRAINT `fk_Log_Usuario1`
    FOREIGN KEY (`Usuario_identificador` )
    REFERENCES `mydb`.`Usuario` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Log_EfectivoRespuesta1`
    FOREIGN KEY (`EfectivoRespuesta_identificador` )
    REFERENCES `mydb`.`EfectivoRespuesta` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Log_ServidorLocal1`
    FOREIGN KEY (`ServidorLocal_identificador` )
    REFERENCES `mydb`.`ServidorLocal` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;

-- -----------------------------------------------------
-- Tabla PerfilSanitario
-- -----------------------------------------------------

CREATE  TABLE IF NOT EXISTS `mydb`.`PerfilSanitario` (
  `idPerfilSanitario` INT NOT NULL AUTO_INCREMENT ,
  `SIP` VARCHAR(45) NULL ,
```

```
  `GrupoSanguineo` VARCHAR(10) NULL ,
  `Alergias` VARCHAR(45) NULL ,
  `TomaAnticoagulantes` INT NULL ,
  `EnfermedadesResp` VARCHAR(100) NULL ,
  `OtraInfo` VARCHAR(100) NULL ,
  `Usuario_identificador` INT NOT NULL ,
  PRIMARY KEY (`idPerfilSanitario`) ,
  INDEX `fk_PerfilSanitario_Usuario1` (`Usuario_identificador` ASC) ,
  UNIQUE INDEX `Usuario_identificador_UNIQUE` (`Usuario_identificador` ASC) ,
  CONSTRAINT `fk_PerfilSanitario_Usuario1`
    FOREIGN KEY (`Usuario_identificador` )
    REFERENCES `mydb`.`Usuario` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Tabla Emergencia
-- -----------------------------------------------------


CREATE  TABLE IF NOT EXISTS `mydb`.`Emergencia` (
  `identificador` INT NOT NULL ,
  `Motivo` VARCHAR(45) NOT NULL ,
  `fecha` DATETIME NOT NULL ,
  `finalizada` TINYINT(1) NULL ,
  `ServidorLocal_identificador` INT NOT NULL ,
  PRIMARY KEY (`identificador`) ,
  INDEX `fk_Emergencia_ServidorLocal1` (`ServidorLocal_identificador` ASC) ,
  CONSTRAINT `fk_Emergencia_ServidorLocal1`
    FOREIGN KEY (`ServidorLocal_identificador` )
    REFERENCES `mydb`.`ServidorLocal` (`identificador` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```