UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Aplicación para dispositivos Android para la señalización de alertas en la ciudad

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Rubén Panadero Navarrete

Tutor: Pietro Manzoni

Curso académico 2013-2014

Resumen

En el siguiente documento se presenta un nuevo software y el proceso de desarrollo que se ha seguido.

Además, cabe destacar que el presente documento se ha elaborado "siguiendo" una metodología de desarrollo ágil. Teniendo en cuenta que son muy usadas en grupos pequeños.

En la introducción se explicara el ciclo de vida que se ha seguido para desarrollar el producto software.

Por otro lado, dentro del documento podemos encontrar todos los detalles de la aplicación así como las decisiones que se han ido tomando en cada una de las partes que forman la aplicación.

También, podemos encontrar los detalles más importantes para el mantenimiento de la aplicación así como para el análisis de rendimiento de la aplicación.

Por último, se incluyen manuales muy útiles para el desarrollador así como para el usuario como anexos. El objetivo de estos manuales es facilitar el uso de la aplicación así como el desarrollo de futuras versiones de la misma.

Palabras clave:

notificacion, alerta, geolocalización, posición, suscripción,señalización,ciudad,mapa,recibir,android,drawer,crashlytics,php,phpmyadmin,googlemaps,marker,responsivo,iu-fluida

Abstract

A new software product and development process, which has been followed, is presented in the following document

Moreover it is important to show what the present document has been elaborated following an agile development methodology. Given that these are very useful in little groups.

Life cycle, which has been followed to develop the new software, will be explained document introduction

Moreover, within the document we can find all application details and the decisions what have been taken in each of the parts that made up the application.

Furthermore, we can find the most important details for maintenance of the application and to analyze performance application

Finally, very useful manual for the developer and user are included as annexes. The purpose of these manuals is to facilitate the use of the application and the development of future versions of the same.

Keywords:

notification,alert,geolocation,city,map,marker,receive,android,drawer,crashlytics,php,phpmyadmin,googlemaps,responsive,fluid-ui

Tabla de contenidos

1. Prólogo

En el presente documento se presentará un problema actual que afecta en cierta manera a la población y que podría subsanarse.

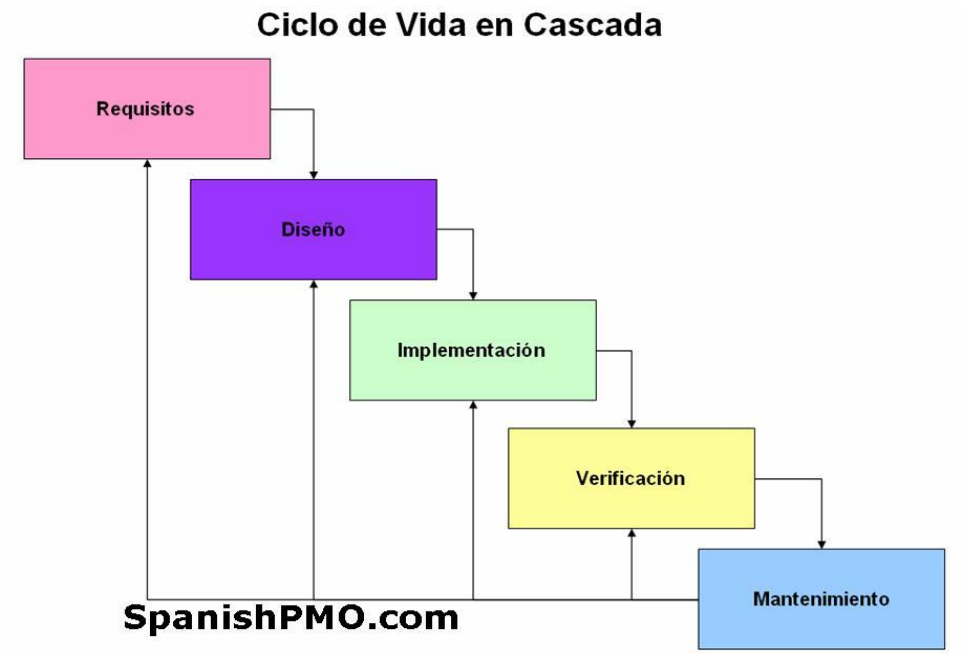
Para solucionar este problema, en las siguientes secciones se presentará una solución software para dispositivos móviles y tabletas con el sistema operativo de Android.

Una vez analizado el problema y presentada la solución que se pretende implantar, se llevará a cabo un análisis de los sistemas antecesores respecto al nuevo producto. Estos sistemas se analizarán con el fin de mejorar lo ya existente y poder adoptar funcionalidades de interés para nuestra aplicación.

Aunque exista un problema y se presente un proyecto software para solucionar este problema, resulta muy importante realizar un estudio de viabilidad del proyecto para comprobar si resulta de interés para el resto de la población. Por lo tanto, también se llevará a cabo un análisis de viabilidad del proyecto.

Tras el análisis del sistema, abordaremos la fase de análisis/requisitos del ciclo de vida del software.

Figura 1. Ciclo de vida en cascada del software



FUENTE: http://www.galeon.com/zuloaga/conceptgen.html

En la fase de análisis se detallarán todos los requisitos de nuestra aplicación, tanto los funcionales como los no funcionales y se procederá a presentar un prototipo de interfaz para nuestra aplicación.

Una vez abordada la fase de análisis del sistema, procederemos a presentar el diseño de la aplicación. El diseño de la aplicación se encargará de la parte del modelo de datos, el diseño de la base de datos, tecnología utilizada, así como la arquitectura de nuestro sistema y la estructura de nuestra aplicación.

Y para finalizar, realizaremos una evaluación del software desarrollado, analizando los diferentes consumos y realizando un estudio de usabilidad.

En los anexos del documento, podemos encontrar una guía para el usuario y otra para el programador. Resulta de interés leer la guía del usuario para entender cómo funciona la aplicación.

2. Introducción

En la actualidad nos encontramos ante muchas situaciones en las cuales es necesario avisar/notificar a otras personas de ciertos acontecimientos. Si conseguimos avisar al resto de población de un acontecimiento podemos evitar males mayores y que la situación específica mejore.

Por ejemplo, si un usuario se encuentra con un accidente en la carretera podría avisar tan pronto como fuese posible, es decir cuando llegase a su destino, de que en ese punto hay un accidente. Por lo tanto, todos los usuarios podrían ver esa alerta y decidir si tomar una ruta alternativa para evitar atascos o cualquier otra cosa que le genere un problema.

Avisando de una situación específica tenemos la posibilidad de que la gente pueda tomar una decisión y no tener que encontrarse con el problema de manera inesperada.

En la actualidad, tenemos redes sociales que podrían ser de ayuda para notificar de ciertos sucesos pero no existe una herramienta como tal para alertar a los usuarios.

Ante esta situación se ha planteado el desarrollo de la aplicación “AlertMap”.

2.1 Solución software

Se ha planteado el desarrollo de una aplicación móvil que se encargue de alertar a otros usuarios de aquello que está ocurriendo en un punto específico del mundo.

Los usuarios de esta aplicación podrán especificar qué tipo de alertas quieren recibir, gestionar sus avisos, es decir, configurar su perfil y/o aplicación.

Con la solución propuesta se pretende dar cobertura a los accidentes, desperfectos, tráfico y eventos sociales que ocurren en todo el mundo. Aunque a priori se muestren únicamente estas opciones se pretende dar flexibilidad para incluir nuevos tipos de alertas a la aplicación. Para ello, si se desea añadir un nuevo tipo de alerta, la aplicación recibiría una actualización descargable desde Play Store.

Otro punto a destacar de la aplicación y uno de los más importantes es que la aplicación correrá en tiempo real, ya que los usuarios tienen la necesidad de recibir las alertas lo más pronto posible.

Por lo tanto, se podrá configurar el intervalo de refresco de las alertas recibidas para que la aplicación no consuma batería en exceso y siempre esté actualizada al gusto

del usuario. Debido a que se implementará en mecanismo de notificaciones pull, es necesario asignar un intervalo de tiempo entre diferentes cargas de datos.

En cuanto a la administración de la suscripción, el usuario podrá también indicar que sólo quiere recibir alertas locales, es decir aquellas que se encuentran a menos de 10 km de distancia del punto en el que se encuentra el usuario en todo momento. Por lo tanto, la posición del usuario debe de ser recalculada mientras esté en movimiento.

Además, se pretende que la interacción usuario-app sea mínima y rápida, sin necesidad de incluir texto para alertar a otros usuarios, etc. Toda interacción usuario-app será limitada a botones, si así lo desea el usuario.

También es importante que el consumo de batería sea el menor posible, debido a que el uso de GPS por parte de la aplicación es alto.

Por otro lado, aunque la primera versión del producto este en un único idioma (Español), la aplicación puede ser usada en cualquier parte del mundo, la cual cosa supone que en posterior versiones se realice una internalización de la aplicación.

En conclusión, se pretende obtener un producto manejable, fácil de usar y atractivo para los usuarios, de manera que se pueda alertar a la sociedad de hechos relevantes y de importancia para otras personas.

Por lo tanto, el objetivo a corte alcance sería crear una "comunidad" auto-gestionada, de manera que se comparta información relevante para todas las personas.

La solución software propuesta pretende ser de alta calidad y además para ello se utilizarán herramientas como Sonar.

2.2 Objetivos del proyecto

En esta sección abordaremos los objetivos del proyecto software.

En primer lugar, se diseñara e implementará un **Modelo de Datos** que dé soporte a la solución software propuesta, permitiendo la persistencia de la información de los usuarios, así como de la aplicación.

En segundo lugar, se diseñarán las interfaces de usuario para Android y se implementarán los controladores correspondientes a cada una de las vistas.

En cuanto a la parte del servidor, por una parte se creará la base de datos relacional partiendo del modelo de datos y por otro parte se crearán los servicios para persistir la información de la aplicación.

Cabe destacar que todos los servicios creados en el servidor serán expuestos bajo un dominio específico y serán accesibles desde Internet y por tanto podrán ser accedidos mediante HTTP.

Como último objetivo del proyecto, sería la elaboración del presente documento, junto con una serie de manuales tanto para el usuario como para el desarrollador/programador. Todo manual, será incluido como anexo del presente documento y serán de vital importancia para el mantenimiento de la aplicación.

3. Antecedentes

A continuación detallaré los sistemas ya existentes en el área de la aplicación y que pueden resultar de ayuda a la hora del desarrollo de la aplicación.

La primera aplicación a analizar, podría ser Twitter.

3.1 Twitter

Twitter se trata de una red social, mundialmente conocida, que permite compartir información con el resto de la comunidad de twitter de manera que nos permitiría avisar de un hecho puntual.

Sin embargo, Twitter no nos permite añadir ningún tipo de mapa o similar a nuestro tweet, por lo tanto únicamente se puede añadir texto descriptivo del hecho, indicando lugar, etc.

Este sistema se limita como mucho a indicar en qué lugar del mundo se ha enviado el tweet, zona (Por ejemplo: Valencia (España)), pero no en qué punto exacto.

Ante este sistema el nuestro tiene ventajas, ya que en nuestro sistema podremos definir un punto exacto y que los usuarios vean en qué punto está ocurriendo cierto suceso.

En general, toda red social existente, como Facebook, Tuenti o similares, se podrían analizar de la misma manera.

Otro sistema que podríamos analizar sería Waze.

3.2 Waze

Se trata de un sistema similar al nuestro.

Waze es una app de mapeo, tráfico y navegación basados en la comunidad. En esta aplicación millones de conductores notifican a otros para evitar el tránsito, ahorrar tiempo y dinero de combustible, y mejorar los trayectos diarios para todos.

Además, Waze permite una navegación completamente guiada por voz . Re-enrutamiento automático conforme cambian las condiciones viales.

Por lo tanto, Waze sería más bien un GPS o se podría considerar como tal.

Sin embargo, esta aplicación está totalmente orientada a conductores, olvidándose del resto de gente que la quiera utilizar para notificar de otras cosas.

Por último analizaremos una red social, diferente a las ya citadas.

3.3 Pinterest

Pinterest es una red social para compartir imágenes que permite a los usuarios crear y administrar, en tableros personales temáticos, colecciones de imágenes como eventos, intereses, hobbies y mucho más.

Los usuarios pueden buscar otros tableros a los que seguir para ver sus actualizaciones.

Por lo tanto, el objetivo de esta red social es notificar a todo el mundo de hechos puntuales que parecen interesantes.

Esta red social se diferencia en que es únicamente basada en imágenes, cosa que podría ser interesante, ya que no hay mejor manera de notificar de algo que con una imagen en la cual veamos lo que ocurre.

Los inconvenientes de esta aplicación son los ya citados en Twitter.

3.4 Conclusiones

Después de analizar todos estos sistemas se han extraído una serie de conclusiones que son importantes para el desarrollo de este nuevo producto software.

La primera cosa a tener en cuenta es que es necesario un mapa donde podamos indicar en qué punto exacto ocurre un hecho, ya que ninguna aplicación proporciona el punto exacto de una notificación.

Por otro lado, puede ser muy interesante y/o satisfactorio de cara al usuario final, la posibilidad de asociar una imagen a una alerta de manera que con un simple vistazo podamos ver lo que está ocurriendo sin necesidad de añadir una descripción.

Con esto, conseguiríamos que el usuario no tenga la necesidad de teclear para añadir una alerta.

Y como última conclusión, parece muy importante preparar a la aplicación para adoptar cambios de manera sencilla y rápida, ya que se debería de poder añadir un tipo de alerta nueva de manera rápida y sin necesidad de dejar al usuario sin servicio.

4. Análisis de mercado

Para llevar a cabo un análisis de mercado, es recomendable realizar análisis DAFO para estudiar la situación del nuevo producto dentro del mercado, analizando así sus características internas (Debilidades y Fortalezas) y su situación externa (Amenazas y Oportunidades). Con este análisis podemos obtener una idea del impacto de nuestro nuevo producto en el mercado.

Figura 2. DAFO

Fortalezas	Peso
Innovación de Productos/Servicios	2
Originalidad de Productos/Servicios	1
Presencia Online	4
Calidad de Productos/Servicios	2
Costos	2
Precio Competitivo	6
Producto personalizable	4
Total	**21**

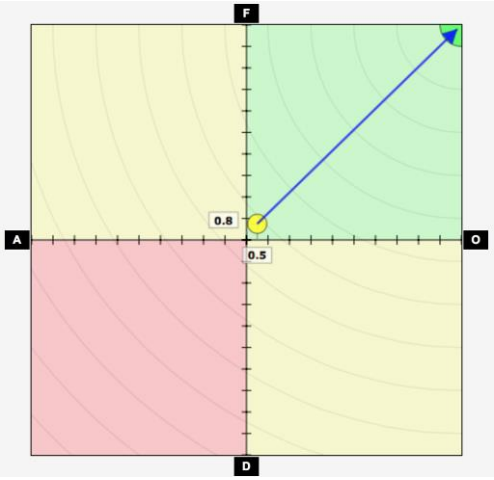
Debilidades	Peso
Marcas	4
Reputación	2
Competencia en el Mismo Rubro	2
Posicionamiento en el Mercado	3
Necesidad de Internet	1
Total	**12**

Oportunidades	Peso
Mayor utilización de Internet por los Consumidores	7
Uso de tablets y smartphones	8
Total	**15**

Amenazas	Peso
Prestamos	3
Posición en el mercado de competidores	3
Situación económica del país	5
Alternativas en el mismo ámbito	1
Total	**12**

Fuente: Elaboración propia

Figura 3. Gráfica de situación del producto



Fuente: Elaboración propia

Como podemos ver en el análisis DAFO (Figura 3), nuestro producto se encuentra en una posición más o menos ideal, puesto que tiende hacia las oportunidades y fortalezas.

Esto se debe principalmente a las fortalezas indicadas en el análisis, en especial al precio competitivo, ya que la primera versión del producto no va a tener ningún precio para el cliente, será gratuita.

Por otro lado, en cuanto a las oportunidades, cabe destacar que el hecho de que el uso de Internet está de moda en la sociedad, favorece el uso de aplicaciones móviles como podría ser la nuestra. Asimismo, otro punto muy importante para este producto es que el uso de tabletas y dispositivos ligeros como los móviles se ha acentuado en los últimos años.

5. Análisis

A continuación describiremos las diferentes fases que se han llevado a cabo dentro de la etapa de análisis.

En primer lugar, procederemos con la fase de "Identificación de requisitos" (Elicitación) en la cual se han descubierto y extraído los diferentes requisitos del sistema.

5.1 Identificación de requisitos

Como primera tarea, se han listado las diferentes fuentes de requisitos, con el objetivo de averiguar qué es lo que se pretende que haga la aplicación en cuestión.

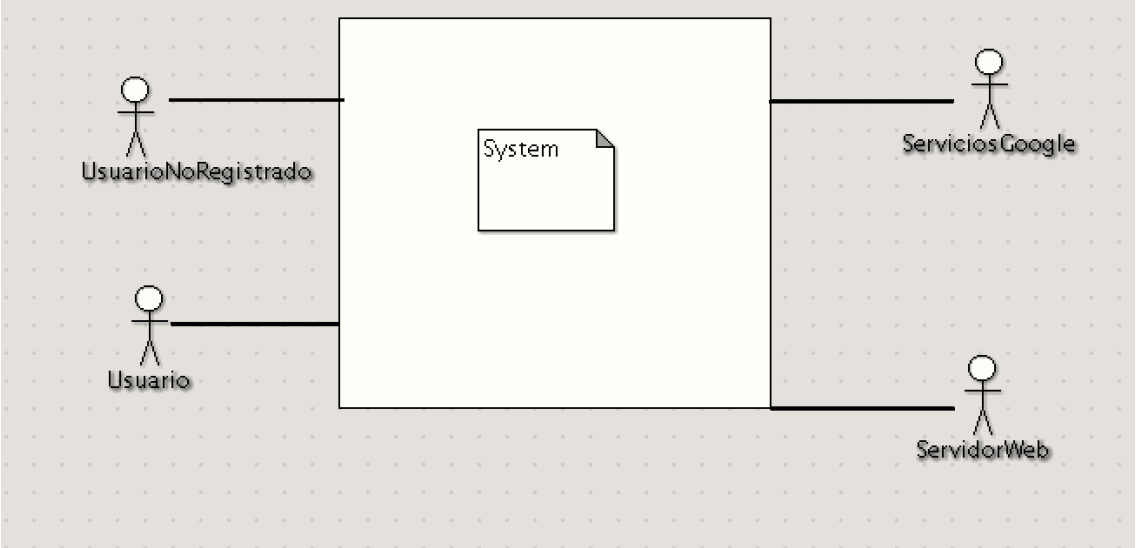
En primer lugar, se identifican los stakeholders relevantes del sistema, es decir todas las personas que se verán afectadas/relacionadas con el proyecto. Los stakeholders que se han obtenido son los clientes de la aplicación y como stakeholder a destacar, que entraría dentro de los clientes, sería el tutor del trabajo de fin de grado (TFG).

Por otro lado, identificaremos los usuarios que interactuarán con el sistema y los roles con los que estos puedan actuar (tipos de usuarios). Para ello, se ha elaborado un modelo de actores y un diagrama de contexto.

Figura 4. Modelo de actores

FUENTE: Elaboración propia

Figura 5. Diagrama de contexto



FUENTE: Elaboración propia

Como se puede observar tanto en la Figura 4 como en la Figura 5, se distinguen 4 actores diferentes. Los dos de la derecha serían actores externos al sistema mientras que los otros dos serían actores internos. Todos estos se consideran actores puesto que interactúan con el sistema para compartir información.

En primer lugar, el actor "UsuarioNoRegistrado" es aquel que interactúa con el sistema hasta que éste inicia sesión, es decir se autentica y pasa a actuar como "Usuario". Y por otro lado, el usuario con rol "Usuario" es aquel que puede hacer uso de todas las funcionalidades principales que la aplicación proporciona.

Es importante destacar que como no se tiene previsto una versión Premium (versión de pago), no se distinguen otros tipos de actores que hereden funcionalidad y además proporcionen otra funcionalidad Premium.

Por otra parte, en cuanto a los actores externos, tenemos los Servicios de Google, ya que la aplicación se comunica con éstos para enviar y recibir datos de los Google Maps; en concreto para hacer uso de la "API de GoogleMaps V2 para Android" que Google proporciona para los desarrolladores Android.

Tal y como se destacó en apartados anteriores, cabe recordar que uno de los puntos fuertes de esta aplicación es hacer uso de los Google Maps y así poder cumplir con el cometido principal de la aplicación que es alertar a los usuarios de los diferentes sucesos que tienen lugar en la carretera, sociedad, etc… en un punto exacto.

Y por último, tenemos el actor "ServidorWeb" que representa al servidor con el que nuestra aplicación se comunicará para enviar y recibir información de la base de datos remota. Este actor, por lo tanto se limitará a representar la comunicación con el servidor donde se alojarán los diferentes servicios web.

Para concluir, es importante remarcar de ambas figuras(figura 4 y 5), que no se puede establecer ningún tipo de relación de herencia entre los actores, ya que son totalmente independientes uno de los otros y no se hereda ninguna funcionalidad proporcionada por otro actor.

Mientras se ha elaborado el diagrama de actores y el diagrama de contexto, se han llevado a cabo varias entrevistas con el cliente para identificar requisitos, es decir capturarlos.

- SE HAN REALIZADO PREGUNTAS CENTRADAS AL CLIENTE:

i. ¿Quién utilizará la aplicación?

Todo aquel que desee alertar a otras usuarios de la aplicación de diferentes situaciones que se ocurren durante el día a día de las personas y pueden ser interesantes para otras, como pueden ser accidentes, etc.

- Y TAMBIÉN SE HAN REALIZADO PREGUNTAS ORIENTADAS AL PROBLEMA Y LA SOLUCIÓN:

i. ¿Qué beneficio obtendrán los usuarios de la aplicación?

Poder recibir alertas en tiempo real y alertar de manera rápida a otros usuarios.

ii. ¿Qué tipo de situaciones se consideran importantes y dará soporte a la aplicación?

En primer lugar, se dará soporte a los accidentes de tráfico, a los desperfectos del mobiliario urbano y a los diferentes eventos que tienen lugar (estreno de cine, entrega de premios, etc.).

iii. ¿Es imprescindible que el usuario realice pocas acciones para alertar a los usuarios?

Sí, los usuarios deben de alertar de manera muy rápida al resto, introduciendo pocas cosas por teclado.

iv. Pero, todos los usuarios no tienen los mismos intereses ¿todos los usuarios recibirán alertas de la misma manera?

No, los usuarios podrán realizar una selección de las alertas de las que se quieren enterar, por ejemplo, sólo me quiero enterar de los accidentes de tráfico.

v. ¿Este software tiene algún rival?

Sí, Twitter se podría considerar un rival, ya que se puede avisar de las diferentes situaciones. Así como Pinterest, que es una red social, poco conocida pero que tiene un propósito parecido.

vi. ¿Es interesante que los usuarios al alertar envíen una imagen?

Podría ser interesante.

La entrevista es una técnica muy útil para descubrir las metas y objetivos del nuevo sistema, para identificar ideas del nuevo sistema y para identificar requisitos.

Como segunda fase dentro del análisis, tenemos la Definición de requisitos. En esta fase, obtendremos descripciones orientadas al cliente y que serán la base para la posterior especificación de requisitos.

5.2 Definición de requisitos

En esta fase de análisis, se utiliza el lenguaje natural por la facilidad para ser entendido, sin embargo puede llegar a generar muchos problemas por sus ambigüedades.

Es importante destacar que se incluyen, en modo lista, todos los requisitos funcionales del sistema, así como los requisitos no funcionales para poder analizarlos más rápidamente durante la fase de especificación.

Como **requisitos no funcionales**:

1. Tiempo de respuesta corto, inferior a 10 segundos.
2. Consumo de batería mínimo.

Y como **requisitos funcionales**:

1. Un usuario de la aplicación puede iniciar sesión.
2. Es posible dar de alta un nuevo usuario en el sistema si no has iniciado sesión.
3. El usuario que ha iniciado sesión puede cerrarla pudiendo acceder con otro usuario.
4. Se puede visualizar la suscripción actual del usuario que ha iniciado sesión.
5. Un usuario puede cambiar su suscripción.
6. Se tiene la posibilidad de ver las alertas que han sido insertadas en el sistema con los datos básicos de cada una de las alertas.
7. El usuario puede recibir alertas en tiempo de ejecución.
8. Un usuario puede añadir alertas nuevas en el sistema, visualizándolas si cumple con la suscripción actual del mismo.
9. Se podrán visualizar las alertas introducidas por el usuario en un listado.
10. Asimismo, se permitirá eliminar las alertas introducidas por el usuario que ha iniciado sesión.

11. Se permite la edición de perfil.

12. El usuario podrá configurar el intervalo de refresco entre alertas, es decir el tiempo en consultar si hay novedades.

13. El servidor web se puede comunicar con la aplicación para enviar información almacenada en la base de datos.

14. Los servicios de Google pueden enviar a la aplicación datos geográficos y de geolocalización.

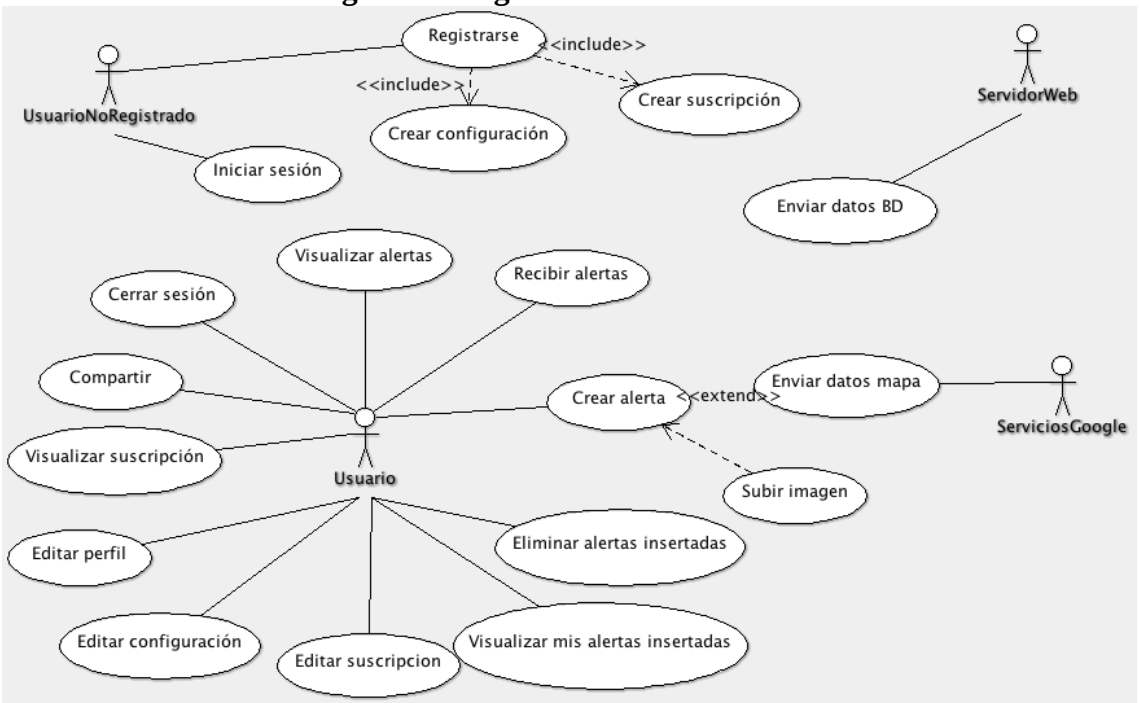
15. El usuario de la aplicación podrá compartir información de la aplicación para publicitarla.

5.3 Especificación de requisitos

A continuación se detallarán los requisitos definidos en el punto anterior.

En este caso se ha desarrollado un diagrama de casos de uso, cada uno de los casos de uso ha sido complementado con lenguaje natural para que se pueda entender mejor el funcionamiento de cada uno de ellos.

Figura 6. Diagrama de casos de uso



FUENTE: Elaboración propia

Haciendo referencia a la Figura 6, podemos explicar los diferentes casos de uso:

Como ya se ha comentado en la fase anterior, el actor "UsuarioNoRegistrado" representa a quien inicia sesión en la aplicación y además se le da la posibilidad de llevar a cabo un registro desde la misma aplicación sin tener que llevarla a cabo desde la web. Para llevar a cabo el registro el usuario tendrá que rellenar un formulario típico.

En segundo lugar, cabe destacar que cuando el "UsuarioNoRegistrado" procede a realizar el registro se le asigna una configuración y una suscripción por defecto, de ahí el <<include>> hacia los casos de uso de "Crear configuración" y "Crear suscripción".

Por otro lado, tenemos al "Usuario" que podrá cerrar sesión en el dispositivo y pasaría a ser otra vez "UsuarioNoRegistrado", también podrá visualizar el mapa de alertas, viendo únicamente las alertas de interés definidas en su suscripción.

Como es de esperar, el "Usuario" puede también "visualizar y editar sus suscripciones" en tiempo de ejecución y así solo recibir aquello que es de su interés.

Además, el "Usuario" se mantendrá a la espera de CU: "Recibir alertas", para ello se solicita al sistema cada cierto tiempo si hay una alerta nueva de interés para el usuario que ha iniciado sesión en la aplicación.

Asimismo, el "Usuario" podrá tanto editar su perfil, pudiendo restablecer los valores definidos durante el registro, como editar la configuración de la aplicación. En cuanto a los valores de la configuración, solo puede ser modificado el intervalo de refresco de alertas, es decir, el tiempo que el sistema espera para realizar una consulta a base de datos y comprobar si hay alertas nuevas de interés.

Por ejemplo, si el usuario indica en su configuración, un intervalo de refresco de 1 minuto, el sistema cada vez que pase 1 minuto consultará a la base de datos si se ha insertado una nueva alerta y si es así, el usuario recibirá una notificación.

También como caso de uso asociado al "Usuario" tenemos el caso de uso de "Crear alerta". Este caso de uso es el más importante de nuestra aplicación, puesto que es la principal motivación de la aplicación. El usuario alertará a los demás que ha ocurrido un hecho importante en un punto del mapa, teniendo la posibilidad de adjuntar una imagen a una alerta si así lo desea y poder añadir una pequeña descripción. Es importante destacar que esta alerta introducida puede no ser visualiza por el "Usuario" que la ha introducido, ya que cuando ésta sea introducida se comprobará si cumple con la suscripción actual del propio "Usuario".

Por otro lado, para publicitar la aplicación el "Usuario" podrá compartir tanto por Twitter como por Whatsapp datos de la aplicación, tales como el nombre de la aplicación. Con esta funcionalidad, lo que se pretende conseguir es generar publicidad y clientes nuevos para la aplicación y así aumentar el número de usuarios.

Es cierto que la base de esta aplicación es crear una gran comunidad de usuarios para estar al día de todo lo que ocurre y que sea de interés.

Por último, tenemos los casos de uso de "Visualizar mis alertas insertadas" y de "Eliminar alerta insertada". Una vez, una alerta haya sido introducida podemos visualizarla en un listado para poder eliminarla en un futuro si ésta ha finalizado o ha sido resuelta por algún motivo.

En cuanto, al actor "ServidorWeb" tenemos únicamente el caso de uso, "Enviar datos BD" que representa el envío de datos del servidor web hacía la aplicación.

Y para concluir, tenemos el actor "ServiciosGoogle" relacionado con el caso de uso "Enviar datos mapas" con el fin de representar el envío de datos geográficos a

nuestra aplicación. Este caso de uso, también se considera de vital importancia, puesto que el poder visualizar un mapa con alertas es una funcionalidad que no es proporcionada por muchas aplicaciones.

Para englobar las diferentes funcionalidades que ofrece la aplicación se han definido una serie de **características**:

1. **Gestión de usuarios:** se encargará de gestionar todo aquello relacionado con las cuentas de usuario.

2. **Gestión de alertas:** esta característica representa a toda funcionalidad relacionada con las alertas, como puede ser la notificación y/o la recepción.

3. **Administrar suscripción:** representa la gestión de una suscripción, englobando tanto la creación como la edición.

4. **Administrar configuración:** esta característica representa toda operación relacionada con la configuración de un usuario, como puede ser la creación y la edición.

5. **Envío de datos externos:** representa el transporte de datos a la aplicación por parte de actores externos.

6. **Publicidad:** se encargar de gestionar todo lo relacionado con la publicidad.

A continuación, se muestra un guion de tipo narrativo de los diferentes requisitos funcionales de la aplicación (Casos de uso (CU)) a modo de resumen aclarativo.

Como los casos de uso se han categorizado se mostrarán detalladamente dentro de las características correspondientes.

5.3.1 Gestión de Usuarios

En la aplicación se lleva a cabo una gestión de los usuarios, ya que los usuarios tienen la posibilidad de iniciar sesión, cerrar sesión, registrarse y editar su perfil.

CU 1: Inicio de sesión

Descripción	Permite a un "UsuarioNoRegistrado" iniciar sesión en la aplicación, indicando su nombre de usuario y su contraseña. A continuación, pasará a actuar como "Usuario".
Precondición	No haber iniciado sesión previamente en la aplicación o cerrar sesión.

CU 2: Registrarse

Descripción	Posibilita registrar nuevos usuarios en el sistema, para que puedan disfrutar de la aplicación. Se rellenará un formulario, indicando nombre de usuario, nombre, apellidos y contraseña.
Precondición	No haber iniciado sesión previamente en la aplicación o cerrar sesión.

CU 4: Cerrar sesión

Descripción	Este caso de uso permite finalizar la sesión del usuario que ha iniciado sesión y así poder iniciar nuevamente sesión con otro usuario.
Precondición	Haber iniciado sesión.

CU 5: Editar perfil

Descripción	Este caso de uso autoriza al usuario editar cualquiera de los datos definidos durante el registro (nombre, apellidos, nombre de usuario o contraseña).
Precondición	Haber iniciado sesión.

5.3.2 Gestión de alertas

CU 6: Visualizar alertas

Descripción	Se permite al “Usuario” visualizar en el mapa o un listado, las alertas a las que se encuentra suscrito. En el listado se podrán ver rápidamente las alertas con su detalle, por orden de llegada. Y en el mapa, si pulsamos sobre una alerta se visualizará el detalle de esa alerta.
Precondición	Haber iniciado sesión.

CU 7: Crear alerta

Descripción	Se concede al "Usuario" la posibilidad de añadir una alerta en la posición indicada en el mapa, pudiendo definir el tipo de alerta (accidente, desperfecto, evento o tráfico), además de opcionalmente añadir una descripción e imagen del hecho.
Precondición	Haber iniciado sesión.

CU 8: Subir imagen

Descripción	Este caso de uso posibilita al "Usuario" adjuntar una imagen a una alerta, de manera que una alerta esté representada con una imagen.
Precondición	Haber iniciado sesión y haber seleccionado que desea subir imagen.

CU 9: Recibir alertas

Descripción	Este caso de uso otorga al "Usuario" la capacidad recibir notificaciones ante nuevas alertas de interés. El "Usuario" busca periódicamente nuevas alertas a mostrar y visualiza una notificación. Es lo que se denominan, notificaciones pull. El sistema cada cierto tiempo (intervalo de refresco), lanza un servicio que hace una consulta a BD para comprobar si hay nuevas alertas y si las hay, el usuario en cuestión las recibirá.
Precondición	Haber iniciado sesión.

CU 10: Visualizar mis alertas insertadas

Descripción	Este caso de uso permite al "Usuario" visualizar en un listado las alertas que ha creado, indicando de manera descriptiva donde ha situado esa alerta (Por ejemplo, Calle Colón - Valencia (España))
Precondición	Haber iniciado sesión.

CU 11: Eliminar alertas insertadas

Descripción	Permite al "Usuario" poder eliminar cualquiera de las alertas que ha creado y está visualizando en un listado, simplemente pulsando sobre ella.
Precondición	Haber iniciado sesión.

5.3.3 Administrar suscripción

CU 12: Crear suscripción

Descripción	Crea una suscripción por defecto para el "UsuarioNoRegistrado" al realizar el registro, estableciendo una suscripción global a todo tipo de alertas (accidentes, tráfico, desperfectos y eventos).
Precondición	No haber iniciado sesión.

CU 13: Visualizar suscripción

Descripción	Este caso de uso permite ver la suscripción actual del "Usuario". Pudiendo ver el tipo de alerta a la que se encuentra suscrito y si está suscrito a alertas globales (todo el mundo) o locales (dentro de un radio de 10km).
Precondición	Haber iniciado sesión.

CU 14: Editar suscripción

Descripción	Permite la modificación de la suscripción del usuario que inicia la aplicación cambiando la zona de suscripción (local o global) y el tipo de alertas que desea recibir (accidentes, desperfectos, eventos y tráfico)
Precondición	Haber iniciado sesión.

5.3.4 Administración configuración

CU 15: Crear configuración

Descripción	Este caso de uso permite crear una configuración por defecto para el "UsuarioNoRegistrado" al realizar el registro, estableciendo el intervalo de refresco a 1 minuto.
Precondición	No haber iniciado sesión.

CU 16: Editar configuración

Descripción	Se da al "Usuario" la posibilidad de editar la configuración que tiene definida en ese momento, pudiendo cambiar el intervalo de refresco de alertas.
Precondición	Haber iniciado sesión.

5.3.5 Envío de datos externos

CU 17: Enviar datos mapa

Descripción	Este caso de uso posibilita que el sistema reciba datos geográficos y de geolocalización mediante el envío por parte de los "ServiciosGoogle" (API Google Maps v2 Android).
Precondición	Haber iniciado sesión.

CU 18: Enviar datos BD

Descripción	Concede al "ServidorWeb" el permiso de enviar datos que son consultados por el sistema .
Precondición	Haber realizado una conexión http vía POST con el servidor.

5.3.6 Publicidad

CU 19: Compartir

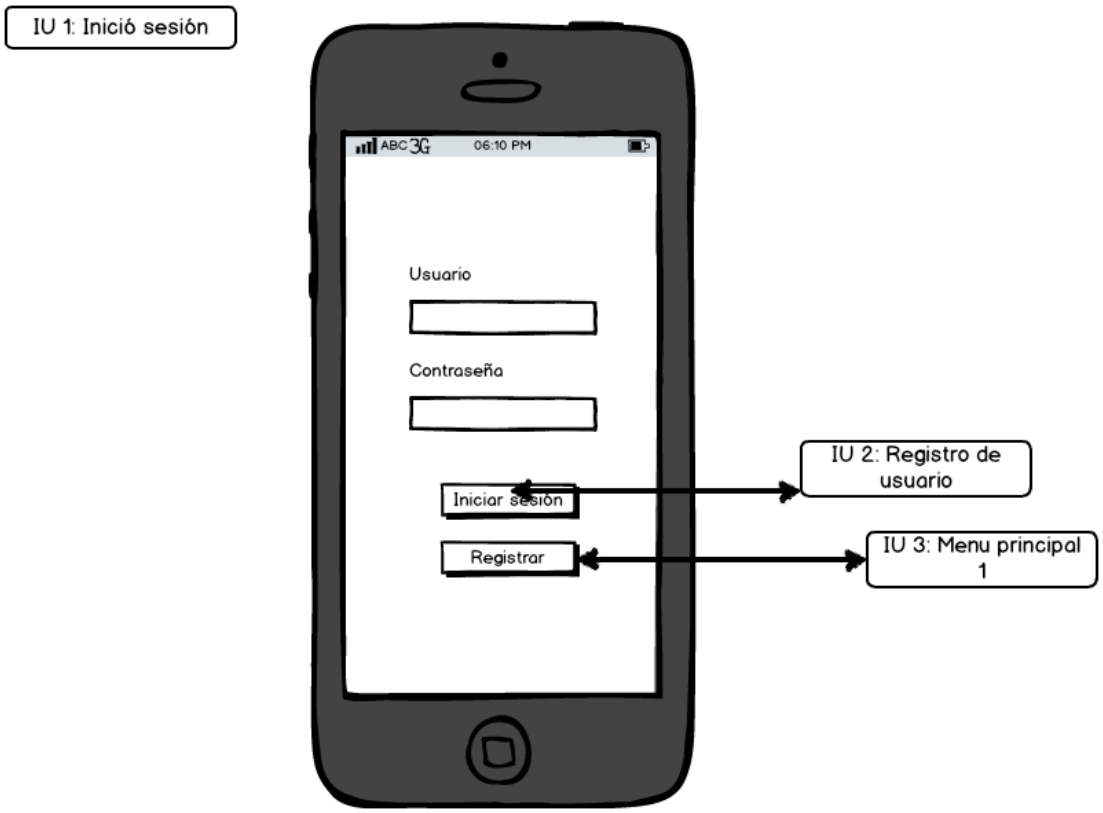
Descripción	Se permite al "Usuario" compartir el nombre de la aplicación por Whatsapp y/o Twitter, según quiera el usuario, para generar publicidad de la aplicación.
Precondición	Haber iniciado sesión.

5.5 Validación de requisitos

Para realizar la validación de requisitos y demostrar que estos son los que el cliente realmente desea, se ha optado por la técnica del prototipado.

A continuación, se muestra un prototipo de cada interfaz de usuario, cada una de las cuales va a ser desarrollada y diseñada en el sistema.

5.5.1 Prototipo IU 1: Inicio sesión



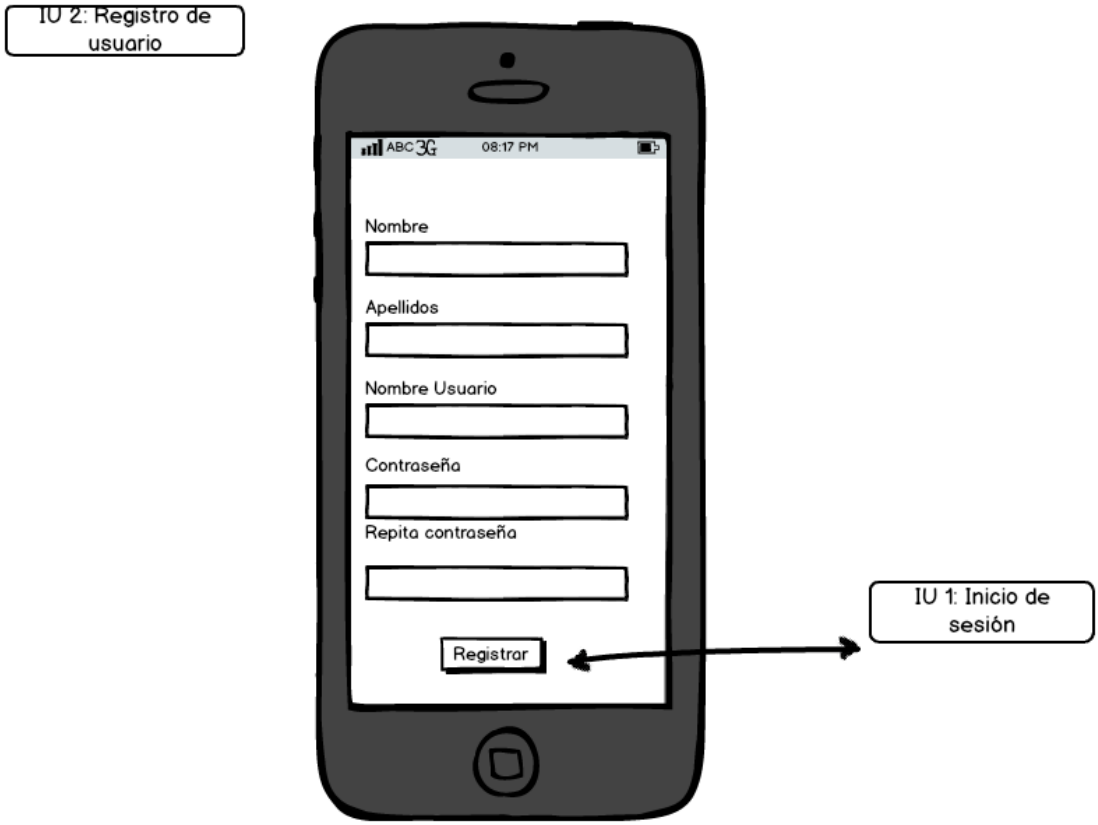
Descripción:

En esta interfaz el usuario de la aplicación (actor "UsuarioNoRegistrado") llevará a cabo la autenticación o podrá pasar a la interfaz de registro.

Casos de Uso incluidos:

Esta interfaz incluye el caso de uso de "Iniciar sesión" (CU 1).

5.5.2 Prototipo IU 2: Registro de usuario



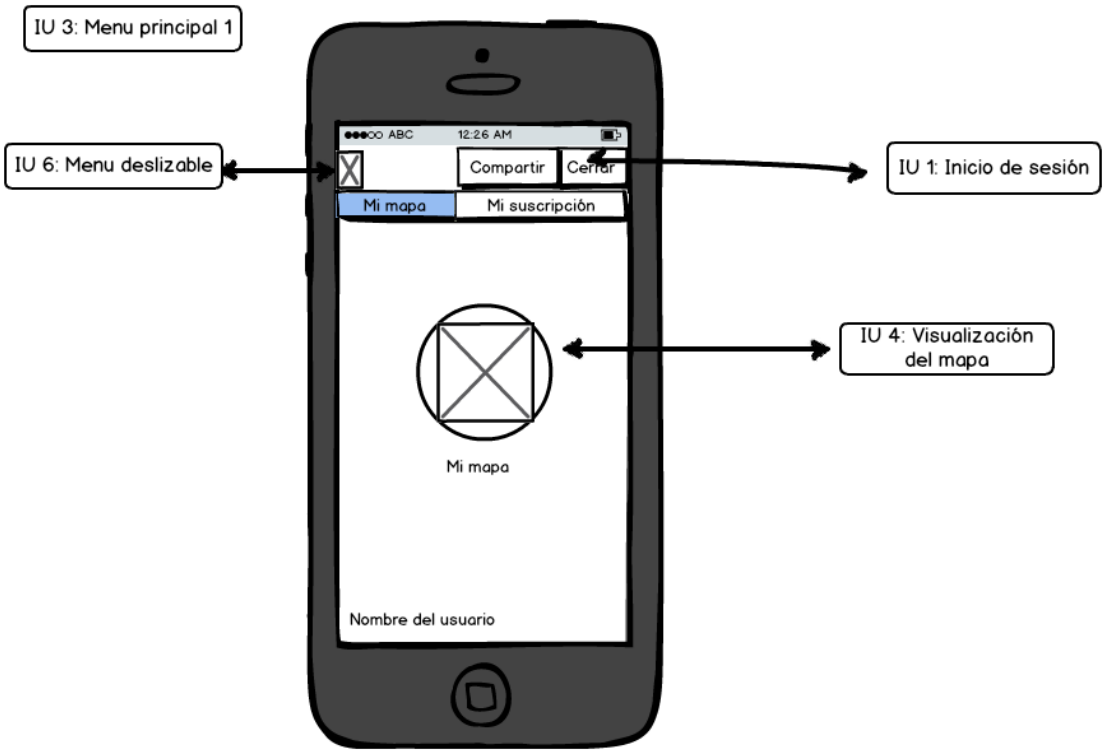
Descripción:

En esta interfaz el usuario de la aplicación (actor "UsuarioNoRegistrado") podrá registrar un nuevo usuario en el sistema.

Casos de Uso incluidos:

Esta interfaz incluye los caso de uso de "Registrarse" (CU 2), "Crear suscripción" (CU 12) y "Crear configuración" (CU 15).

5.5.3 Prototipo IU 3: Menú principal 1



Descripción:

En esta interfaz el usuario de la aplicación (actor "Usuario") podrá iniciar el mapa de alertas, cerrar sesión y regresar a la IU 1 o bien abrir el menú deslizable para seleccionar una serie de opciones disponibles. Además, el usuario tener feedback propio viendo su nombre en la parte inferior del menú.

Por último, se debe remarcar el acceso a la IU X para poder compartir datos de la aplicación de manera que se genera publicidad.

Casos de Uso incluidos:

Esta interfaz incluye el caso de uso de "Cerrar sesión" (CU 4) y "Compartir" (CU 19).

5.5.4 Prototipo IU 4: Mi muro

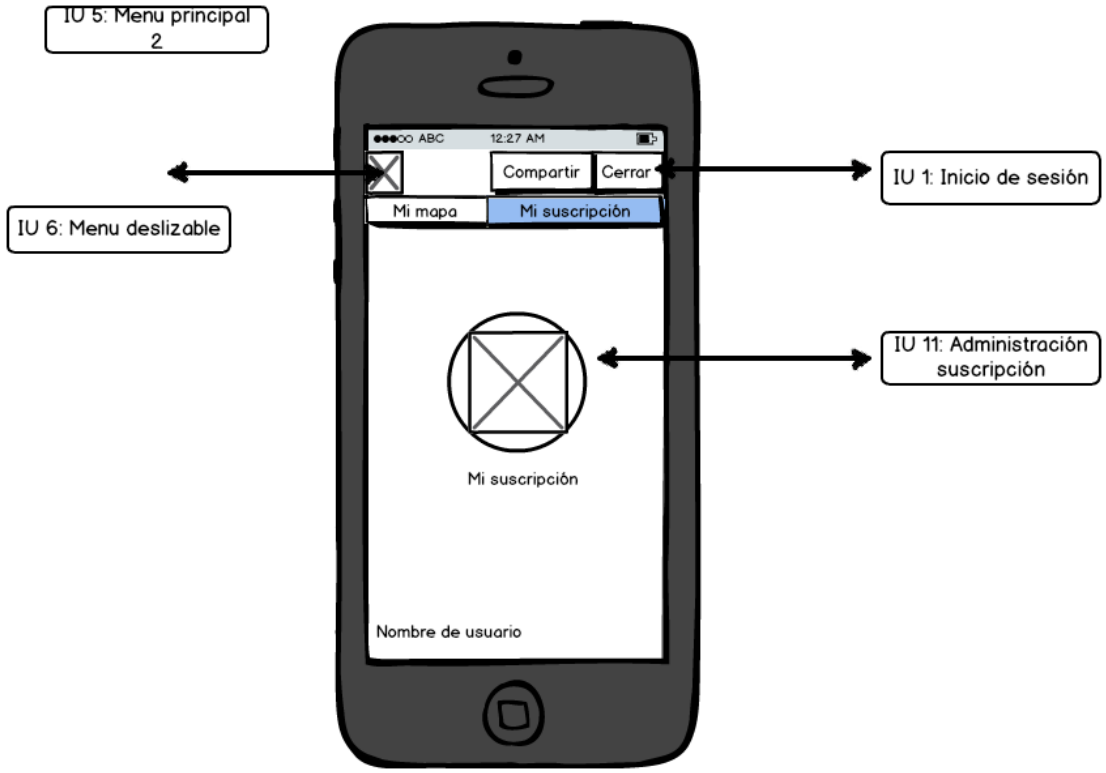
Descripción:

En esta interfaz el usuario de la aplicación (actor "Usuario") podrá ver las alertas de las que se encuentra suscrito y además podrá acceder a la IU 6 realizando una pulsación larga sobre el punto en el que desea añadir una alerta.

Casos de Uso incluidos:

Esta interfaz incluye el caso de uso de "Visualizar alertas" (CU 3).

5.5.5 Prototipo IU 5: Menú principal 2



Descripción:

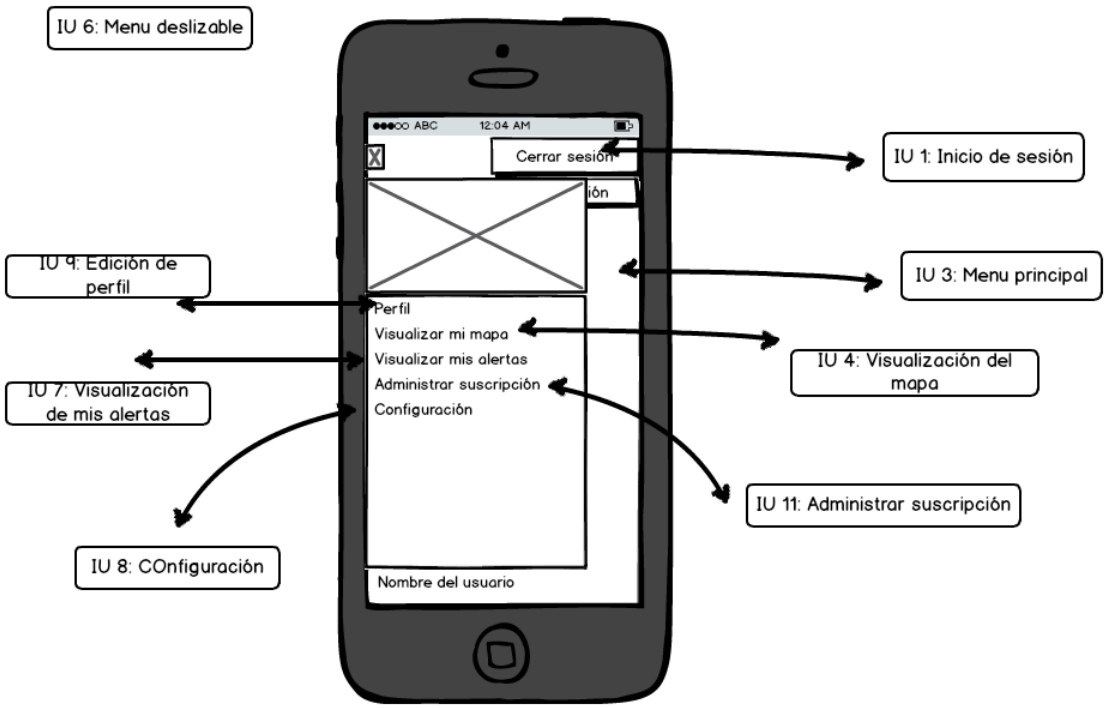
En esta interfaz el usuario de la aplicación (actor "Usuario") podrá acceder a ver su suscripción de alertas, cerrar sesión y regresar a la IU 1 o bien abrir el menú deslizable de la parte superior izquierda. Además, el usuario tener feedback propio viendo su nombre en la parte inferior del menú.

Por último, se debe remarcar el acceso a la IU X para poder compartir datos de la aplicación de manera que se genera publicidad.

Casos de Uso incluidos:

Esta interfaz incluye el caso de uso de "Cerrar sesión" (CU 4) y "Compartir" (CU 19).

5.5.6 Prototipo IU 6: Menú deslizable



Descripción:

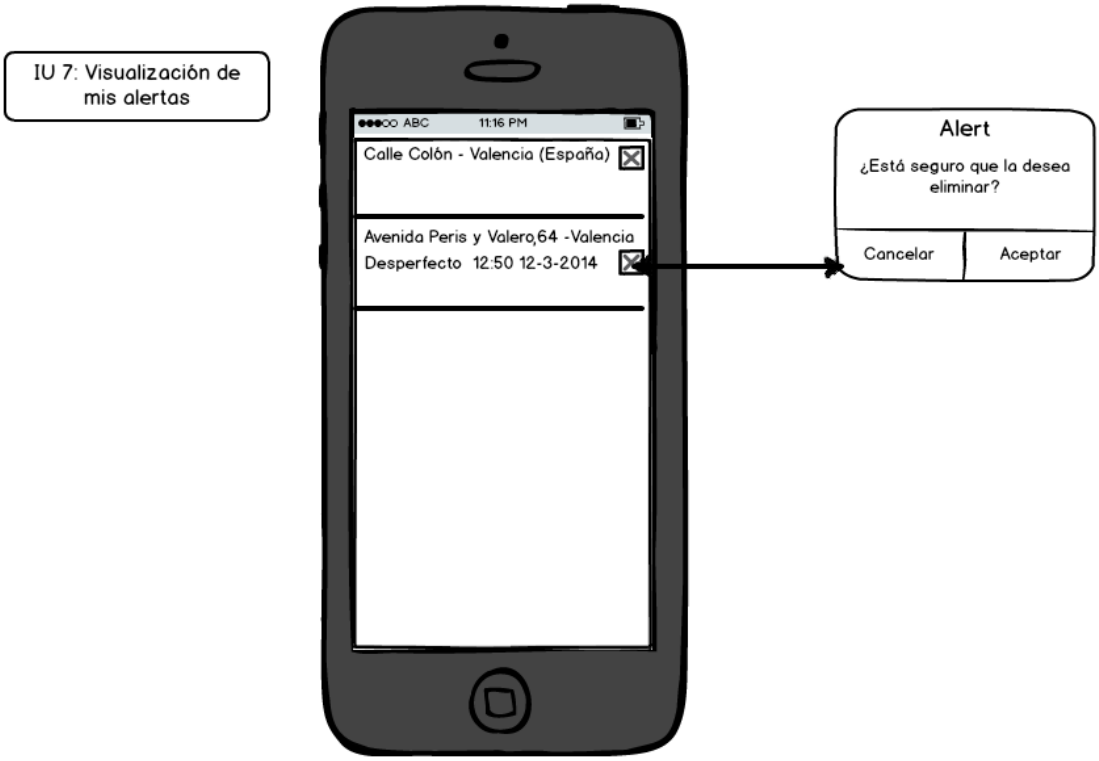
En esta interfaz el usuario de la aplicación (actor "Usuario") podrá acceder al menú principal de la aplicación, teniendo acceso a todas las opciones.

Este menú, saldrá por la parte izquierda de la pantalla pulsando sobre el botón superior izquierda o haciendo swipe izquierda-derecha.

Casos de Uso incluidos:

Esta interfaz únicamente incluye el caso de uso de "Cerrar sesión" (CU 4).

5.5.7 Prototipo IU7: Visualización de mis alertas



Descripción:

En esta interfaz el usuario de la aplicación (actor "Usuario") verá una lista con todas las alertas que ha introducido en el sistema y que todavía no han sido eliminadas. El usuario podrá decidir en cualquier momento si la desea eliminar, pulsando sobre la imagen de la derecha (cruz).

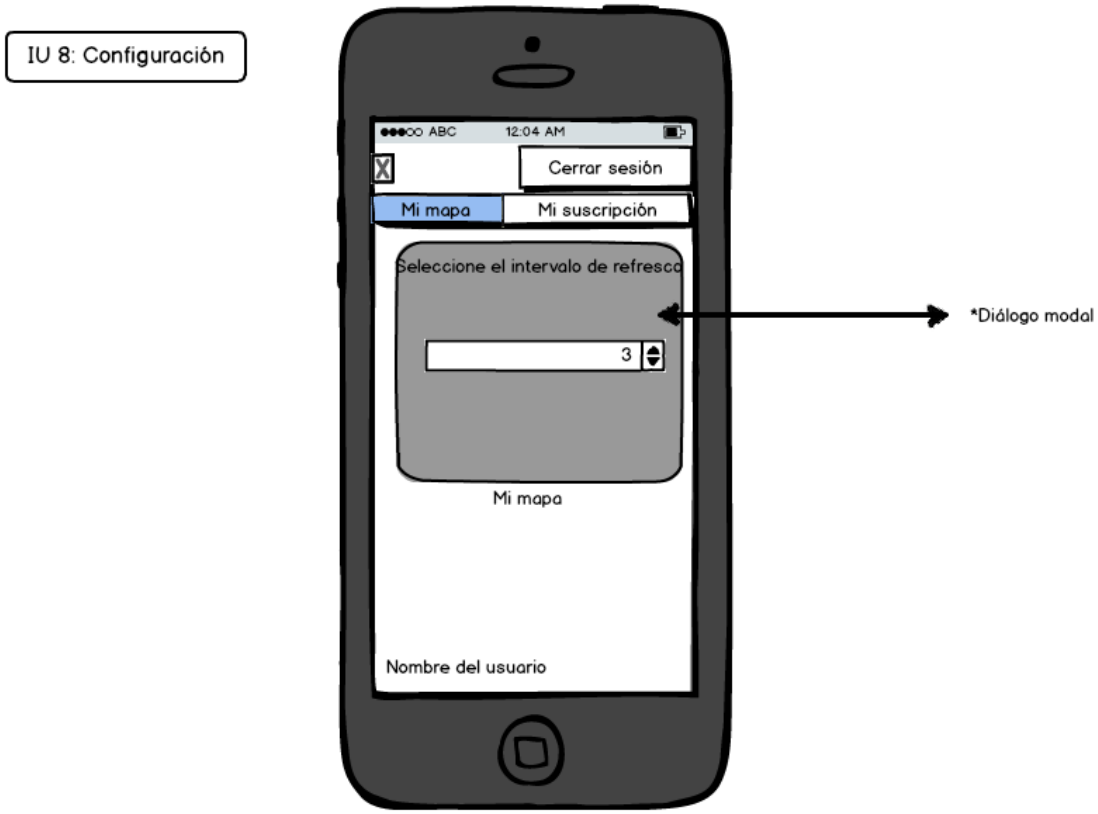
Cabe destacar que el usuario verá la calle en la que ha ocurrido la alerta, la descripción enviada y la hora en la que la notificó.

Si el botón de eliminar es pulsado aparecerá un diálogo de confirmación.

Casos de Uso incluidos:

Esta interfaz incluye los casos de uso de "Eliminar alertas insertadas"(CU 11) y "Visualizar mis alertas insertadas" (CU 10)

5.5.8 Prototipo IU 8: Configuración



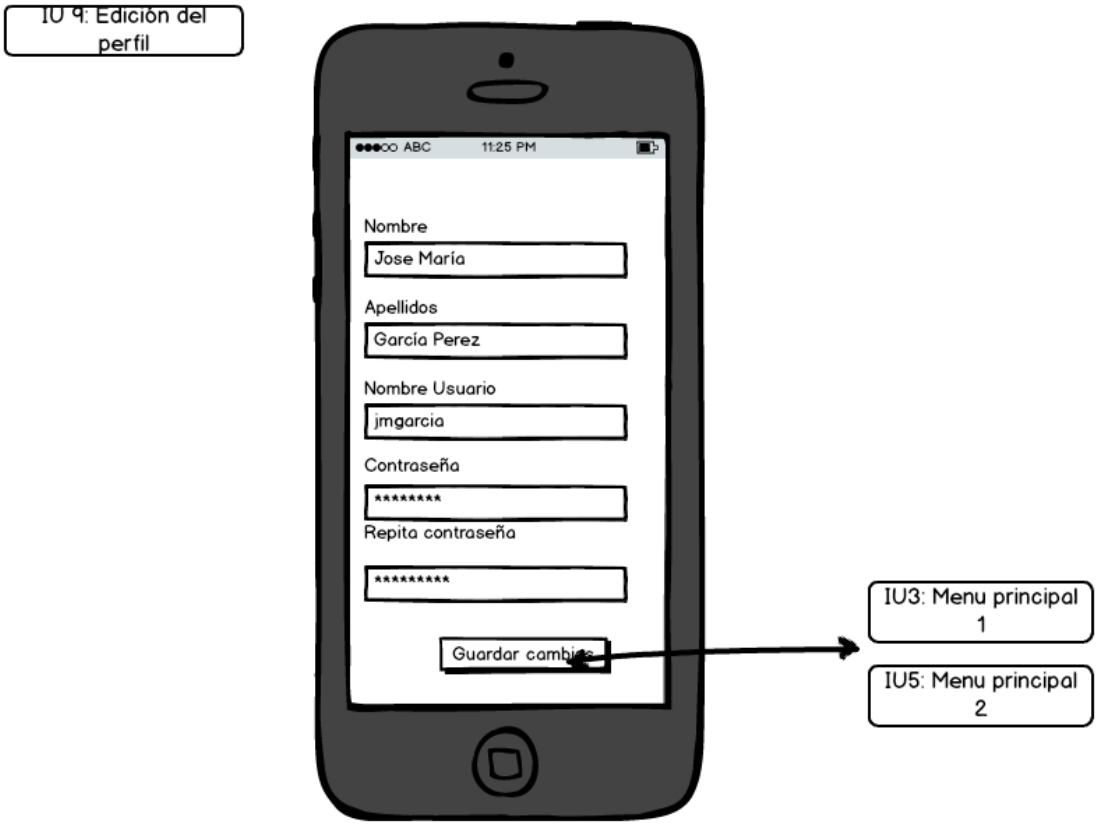
Descripción:

En esta interfaz el usuario de la aplicación (actor "Usuario") podrá seleccionar el intervalo de refresco de la aplicación, es decir el tiempo de espera a comprobar si hay nuevas alertas que sean de mi interés de acuerdo a mi suscripción.

Casos de Uso incluidos:

Esta interfaz únicamente incluye el caso de uso de "Editar configuración" (CU 15).

5.5.9 Prototipo IU 9: Edición del perfil



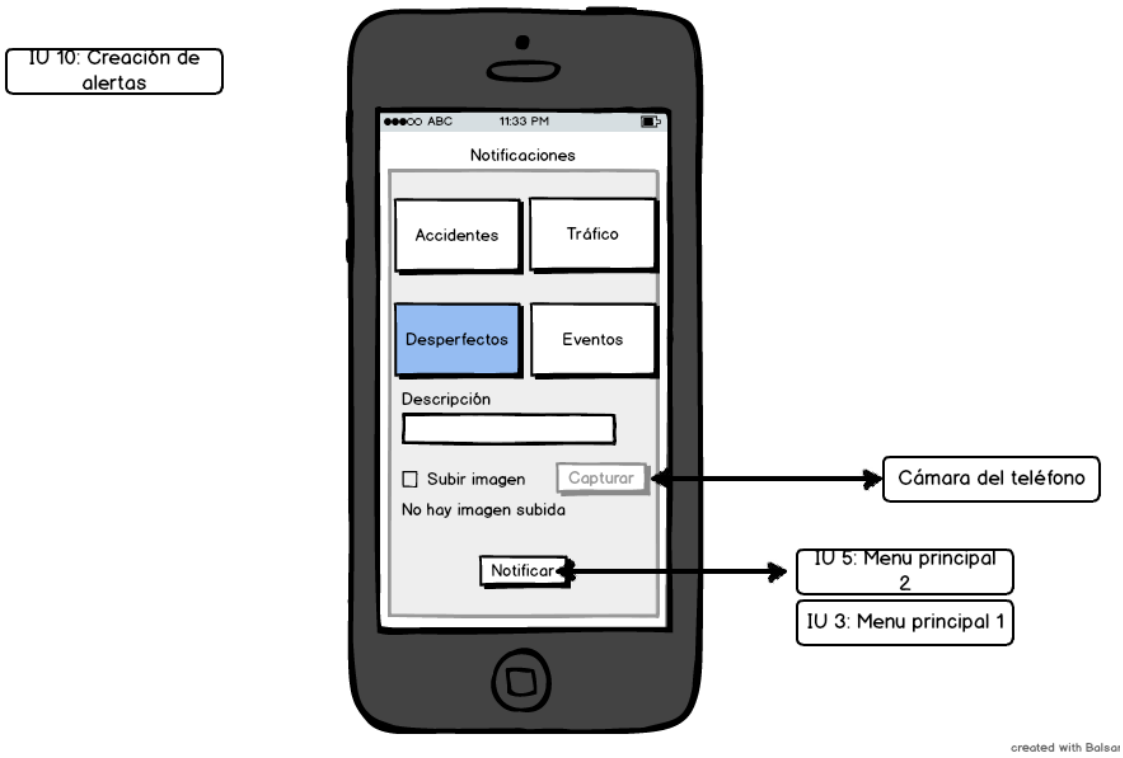
Descripción:

En esta interfaz el usuario de la aplicación (actor "Usuario") podrá modificar cualquiera de los datos introducidos durante el registro.

Casos de Uso incluidos:

Esta interfaz únicamente incluye el caso de uso de "Editar perfil" (CU 5).

5.5.10 Prototipo IU 10: Creación de alertas



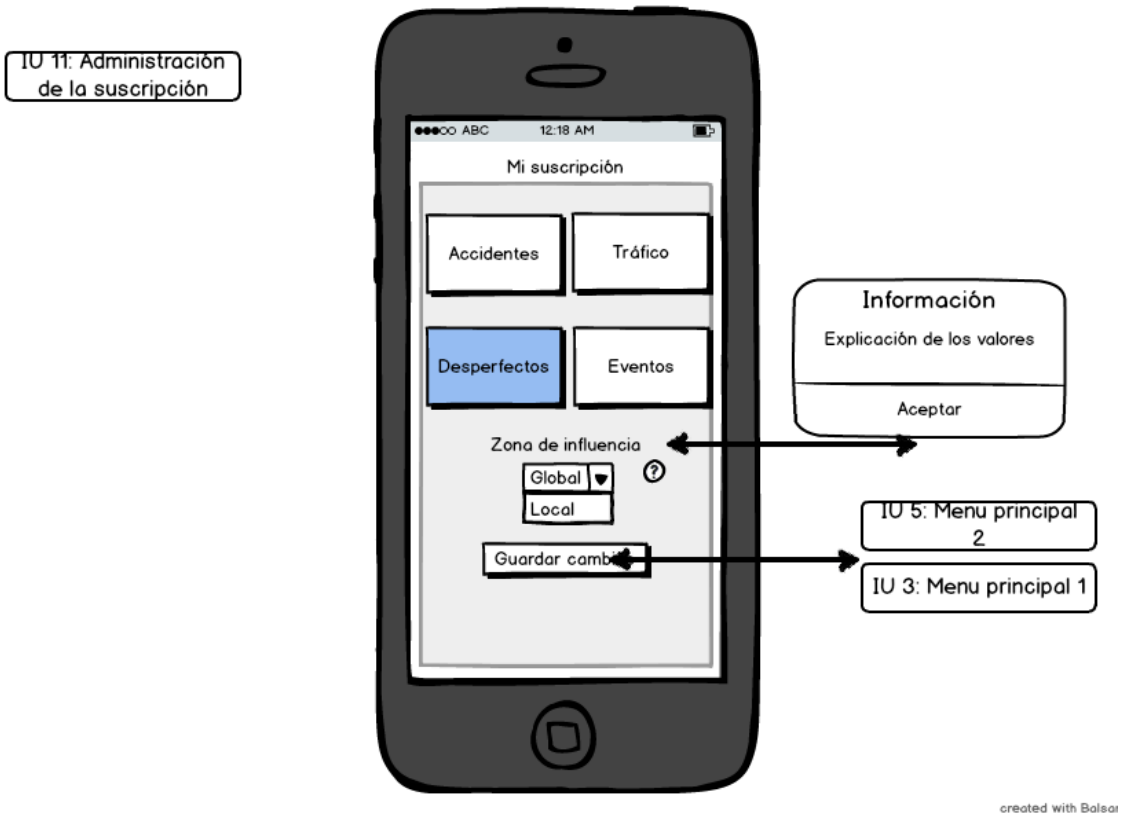
Descripción:

En esta interfaz el usuario de la aplicación (actor "Usuario") podrá crear una alerta, para notificar al resto de personas de un suceso. El usuario podrá seleccionar únicamente un tipo de alerta, añadir si lo desea una descripción y/o una imagen que será capturada por la cámara del teléfono.

Casos de Uso incluidos:

Esta interfaz incluye los casos de uso de "Crear alerta" (CU 7) y "Subir imagen" (CU 8), si se selecciona una imagen para subir.

5.5.11 Prototipo IU 11: Administración de la suscripción



Descripción:

En esta interfaz el usuario de la aplicación (actor "Usuario") podrá seleccionar los tipos de alertas de los cuales desea recibir alertas. Se permite, como es lógico, la selección múltiple. Además, el usuario podrá seleccionar la zona de influencia (global o local).

Casos de Uso incluidos:

Esta interfaz únicamente incluye el caso de uso de "Editar suscripción" (CU 14).

6. Diseño

6.1 Entorno de desarrollo

A continuación se detallan las herramientas más importantes que se utilizan en el desarrollo de este proyecto.

Para este proyecto se ha trabajado tanto en con el sistema operativo de Windows como con Mac OS X.

Herramientas de organización y seguimiento

Subversion

Sistema de control de versiones que permite el versionado del código software a desarrollar. Este sistema almacena todos los cambios de código, de manera que automatiza la subida y descarga de versiones de un archivo. Cabe destacar que permite ver en un historial todos los cambios realizados.

Cloudforge

Es un software que proporciona servicios para otras herramientas de desarrollo como pueden ser Jit o Subversion. Proporciona servicios de "hospedaje" para estas herramientas, es decir actúa de repositorio web para que nuestro código pueda ser recuperado.

Herramientas de desarrollo

EclipseADT

Entorno de desarrollo (IDE) para el desarrollo de aplicaciones Android, ya que proporciona una suite de plugins para facilitar el desarrollo de aplicaciones, es decir proporciona diferentes características que hacen que el desarrollo sea más rápido.

TextWrangler

Es una herramienta, editor de texto, para el desarrollo de aplicaciones web. Ha sido usada para el desarrollo de scripts en php.

Herramientas para la gestión de la base de datos

Phpmyadmin

Es una herramienta escrita en php con la intención de manejar la administración de MySQL.

Servidores web

Hostinguer

Es un sistema que proporciona servicios de almacenamiento web y hosting gratis.

Servidor de publicación UPV

Servidor web donde podemos desplegar nuestra parte del "servidor" de la aplicación. En este entorno se desplegará la aplicación en producción.

Herramientas para la creación y procesamiento de imágenes

Adobe Photoshop Illustrator CS6

"Software de diseño gráfico que crea ilustraciones vectoriales con las herramientas avanzadas y precisas de dibujo y tipografía". (www.adobe.com)

6.2 Detalles de implementación

El proyecto esta implementado utilizando Java, lenguaje orientado a objetos muy popular. Está desarrollado para dispositivos Android, ya que no es necesario ningún tipo de licencia para el desarrollo en Android y la API es de acceso libre. http://developer.android.com/develop/index.html).

Para realizar la interfaz de usuario (capa de presentación de la aplicación), se ha hecho uso de su API de Android, además de Apis externas como son la API Google Maps v2 para Android y Apis de soporte para versiones anteriores de Android como son support-v4 y appcompat v7, que hace que la interfaz de usuario sea visible en dispositivos con una versión de Android antigua y nos permite hacer uso de widgets más nuevos. En este caso, de la action bar, fragments y navigation drawer.

En cuanto a la persistencia de datos, se ha optado por el uso de una base de datos relacional como MySQL, administrada mediante phpmyadmin.

Por otro lado, cabe destacar que para hacer uso de los mapas de Google es recomendable hacer pruebas sobre un dispositivo físico externo y no hacer uso de la máquina virtual de Android proporcionada por Eclipse, ya que no tiene los GoogleServices instalados y ralentiza mucho el desarrollo.

Por último, cabe destacar el uso del sdk de Crashlytics. Crashlytics es un kit de desarrollo que nos permite registrar los fallos que la aplicación sufre (excepciones recuperables y no recuperables) con todo detalle. Todos estos detalles quedan registrados en la cuenta asociada a la aplicación en Crashlytics.

En primer lugar, para usar esta librería es necesario solicitar una clave de autenticación, es decir no sirve con un simple registro, requiere ponerse en contacto con el staff de Crashlytics.

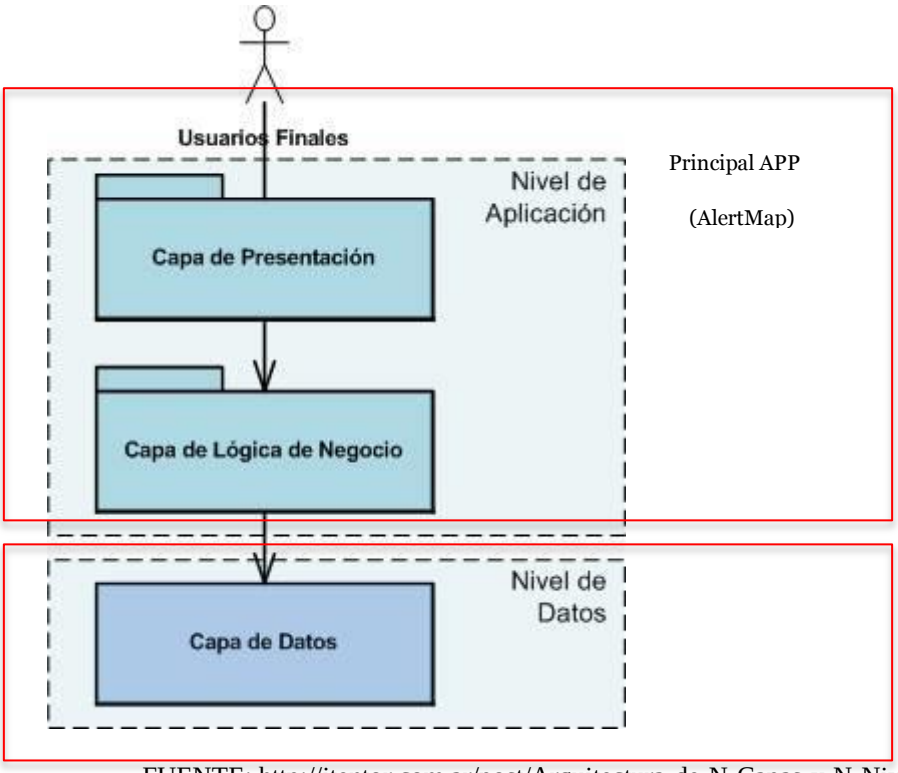
Y a continuación, una vez recibida la clave de autenticación, se deben de seguir los pasos guiados por Crashlytics.

Para finalizar, cabe destacar que entre los detalles de un error registrado por Crashlytics, podemos encontrar: dispositivo usado, estado del dispositivo, características del dispositivo, línea del error, excepción lanzada, etc. Para más información visitar “http://try.crashlytics.com”.

6.3 Arquitectura del sistema

A continuación abordaremos la arquitectura que poseerá nuestra aplicación de manera gráfica.

Figura 7. Arquitectura AlertMap.



FUENTE: http://jtentor.com.ar/post/Arquitectura-de-N-Capas-y-N-Niveles.aspx

Como se puede observar en la figura, la aplicación consta de una estructura de 3 capas:

- **Capa de presentación**
  - Android XML
- **Capa de negocio.**
  - Java 7
- **Capa de persistencia.**
  - MySQL

Esta arquitectura se **caracteriza** por los siguientes puntos:

- Desacoplar los componentes de forma vertical.
- Los componentes también requerirán algún tipo de estructuración horizontal.
- En cada nivel todos los componentes que lo constituyen trabajan al mismo nivel de abstracción y pueden interactuar a través de conectores (interfaces).
- En la forma más pura del patrón, los niveles no deben ser by-paseados: los niveles más altos acceden a los niveles más bajos sólo a través del nivel inmediatamente inferior.
- Dos niveles adyacentes pueden considerarse como un par cliente-servidor, el nivel más alto será el cliente y el nivel más bajo el servidor.

Capa de presentación:

Esta capa se encarga de resolver toda la interacción que tiene el usuario con el sistema, proporcionando las diferentes interfaces de usuario. Por lo tanto, esta capa es la que el usuario ve. Cabe resaltar que esta capa también gestiona los diferentes cambios de interfaces para que el usuario pueda avanzar por el sistema. En resumen, lleva a cabo la navegación entre pantallas.

Además, se encarga de mostrar los datos y de formatearlos para que el usuario los vea tal y como desea. Por ejemplo, visualizar una fecha con el formato dd/mm/yyyy. Asimismo, puede ocultar o deshabilitar determinado dato/control si se da una cierta circunstancia controlada por la lógica de presentación.

Por otro lado, la capa de presentación también solicita los datos al usuario e incluye una pequeña lógica para validar los datos introducidos. Asimismo, comunica esta información introducida al sistema, en un mínimo proceso.

Otra característica a destacar de esta capa es que tiene el objetivo de informar de los errores lógicos y de ejecución. Para ello, se puede tomar la decisión de mostrar un diálogo modal con el error o gestionar ese error de otra manera.

Por último, cabe destacar que esta capa puede estar dividida en subcapas (servidor y cliente), aunque normalmente suele estar centralizada. En este caso, la capa de presentación no se encuentra dividida y se encuentra alojada en el dispositivo.

API Google Maps

Dentro de la capa de presentación cabe destacar de una manera especial los mapas de Google, ya que se ha hecho hincapié en hacer uso de estos mapas (Google Maps).

Google proporciona una API para poder geolocalizarnos así como poder formatear y ver en un mapa los datos que se deseen y de la manera que el desarrollador crea conveniente.

En la aplicación en cuestión, se va a hacer uso de Google Maps para mostrar al usuario las diferentes alertas que se encuentran activas en un momento determinado. Por lo tanto, el objetivo es que el usuario pueda ver estas alertas de manera rápida y sin distracciones, es decir visualmente y en un punto exacto del mapa. Además, se deberá de ofrecer la información más básica de la alerta.

Por lo tanto, la API de Google Maps v2 para Android nos proporciona muchas facilidades para conseguir el objetivo propuesto.

En primer lugar, nos facilita métodos para añadir “markers” (chinchetas) en un punto exacto del mapa y ofreciendo la posibilidad de cambiar el color de los mismos si así se desea. Por lo tanto, el usuario simplemente con el color podrá distinguir los tipos de alertas de un simple vistazo.

Por otro lado, la API nos ayuda a obtener la posición del usuario y a poder hacer zoom en esta posición. Con esta funcionalidad podemos conseguir que el usuario vea rápidamente las alertas que están a su alrededor.

Y por último, en la aplicación, se hará uso de la ventana de información de un “marker”, es decir, cuando se pulse sobre un “marker” podremos ver el detalle de esa notificación/alerta en una pequeña ventana de información.

Capa de Negocio o lógica:

Esta capa tiene la responsabilidad de: implementar procesos de negocio identificados durante el análisis funcional, control de acceso a los servicios de negocio desde otras capas, publicación de los servicios de negocio.

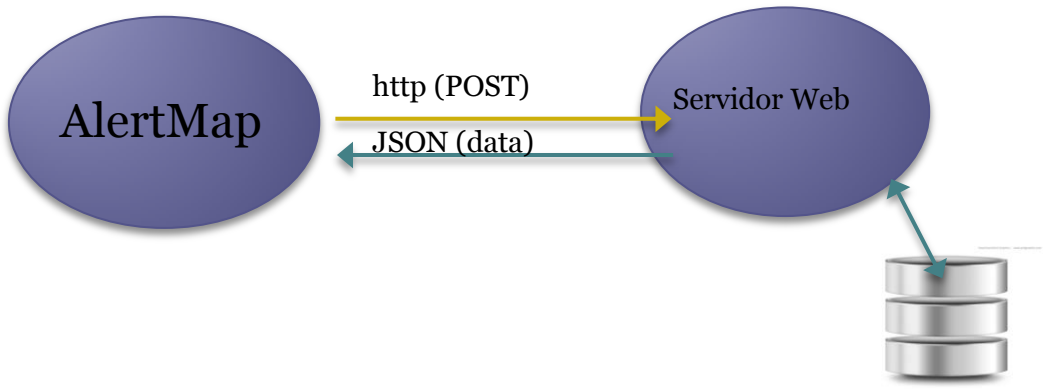
Cabe destacar que el cliente solo conoce la interfaz, sin embargo hay una implementación detrás.

Capa de persistencia

Esta capa reside en mayor medida en el servidor web y es implementada mediante PHP. Sin embargo las comunicaciones HTTP con el servidor vienen definidas dentro de la aplicación cliente como es de esperar.

Esta capa es la que se encarga de acceder a los datos almacenados en la base de datos de MySQL y devolverlos a la aplicación.

La aplicación se comunica con el servidor mediante el protocolo HTTP y envía los datos al servidor mediante POST. Asimismo, el servidor web envía los datos a la aplicación en formato JSON y esta los trata como tal.



Conclusiones

Como es fácil deducir, nuestra aplicación está dividida en dos niveles totalmente diferenciados.

El primer nivel, incluirá toda la capa de presentación y la capa de negocio. Mientras que el segundo nivel, contendrá la capa de persistencia.

Entre las ventajas de desarrollar aplicaciones, separando el nivel de aplicación (Nivel 1) del nivel de datos (nivel 2), encontramos que no se desvelan ningún tipo de información de acceso a datos, como puede ser nombre y contraseña de la base de datos. Además, tampoco se da información de la estructura que tiene nuestra base de datos.

Por lo tanto, el desarrollo de la aplicación se puede considerar seguro ya que el servidor web es el único que contiene esta información y ante una posible descompilación de nuestra aplicación ningún dato que se encuentre almacenado en la base de datos, podría resultar en peligro.

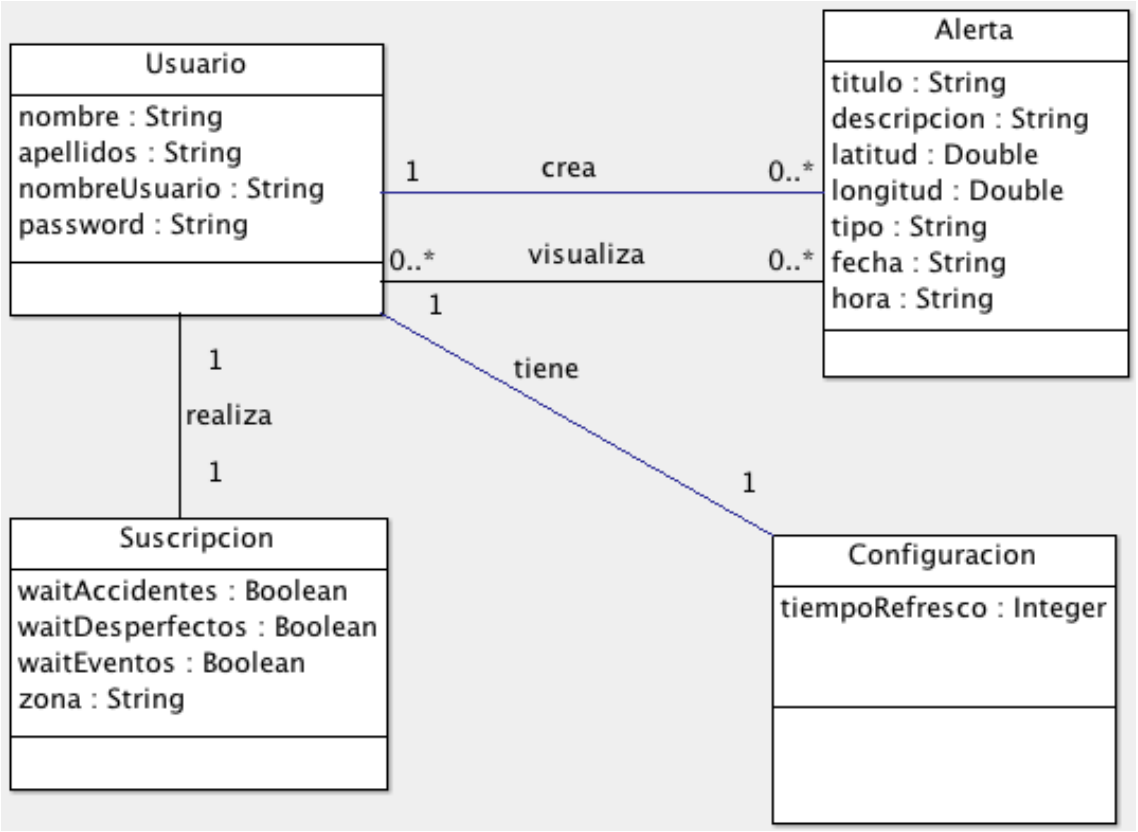
Asimismo, se realiza un cifrado de contraseña en md5 para que el autenticado sea seguro y la contraseña del usuario de la aplicación no pueda ser desvelada de ninguna de las maneras.

Una vez definida la arquitectura del sistema, elaboraremos un modelo de diseño o también denominado diagrama de clases.

6.4 Modelo de datos

El diagrama de clases nos describe la estructura del sistema de "AlertMap" mostrando sus clases y, además, podemos visualizar las relaciones entre las clases.

Figura 8: Diagrama de clases



FUENTE: Elaboración propia

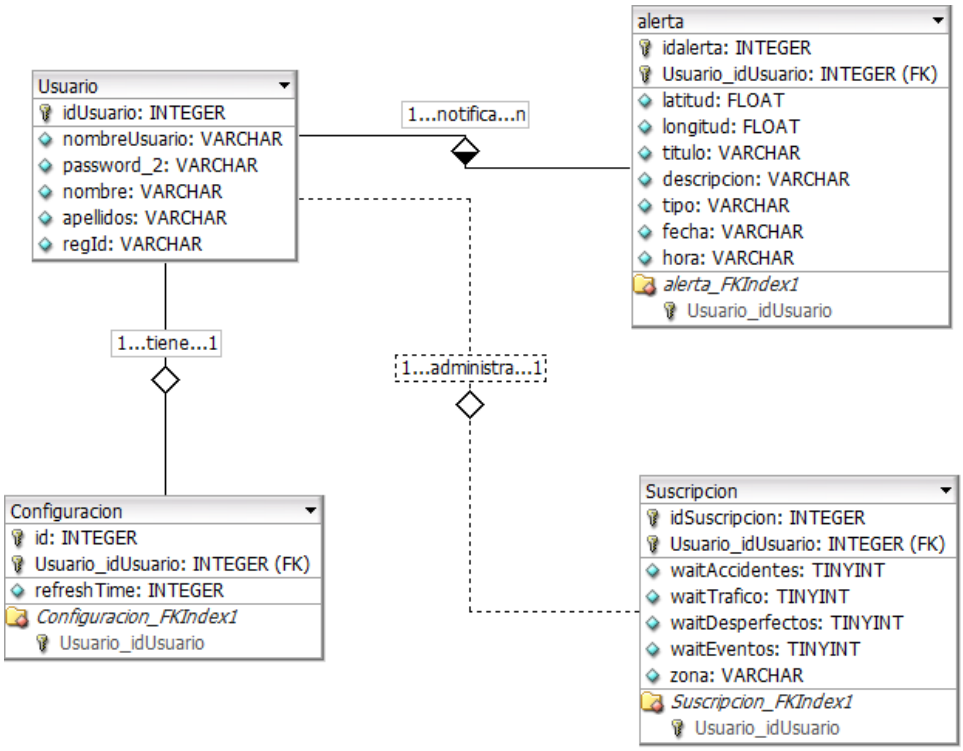
Como se puede observar se diferencian 4 clases, Usuario, Suscripción, Configuración y Alerta, son las clases necesarias para llevar a cabo la lógica del negocio.

Un usuario tiene dos relaciones con "Alerta" puesto que hay que diferenciar entre alertas que son insertadas y alertas que son visualizadas. Cabe destacar que una misma alerta puede ser visualizada por muchos usuarios.

Por otro lado, un usuario sólo puede tener una suscripción asociada y una configuración.

6.5 Diseño de la base de datos

Figura 9. Diseño de la base de datos relacional



FUENTE: Elaboración propia

Nuestro diseño de la base de datos, nos permite almacenar a todos los usuarios registrados en nuestra aplicación. Como se puede observar en la figura, la tabla de usuario tiene un atributo, denominado regid para una futura versión de la aplicación, por si fuera necesario utilizar un mecanismo de notificaciones push.

En cuanto a configuración, el usuario únicamente gestionará el intervalo o tiempo de refresco, ya mencionado y explicado en el caso de uso de "Recibir alertas" (CU 9). Como es obvio, un usuario sólo podrá tener activa una configuración.

Por otro lado, el usuario podrá almacenar su suscripción de manera que cada vez que entre en la aplicación esta se actualice sin necesidad de volver a cambiarla.

Y por último, tenemos las alertas. El sistema almacenará toda alerta que esté activa, es decir todas aquellas que han sido notificadas y aún no han sido eliminadas. Como es de esperar, un usuario podrá notificar de *n* alertas.

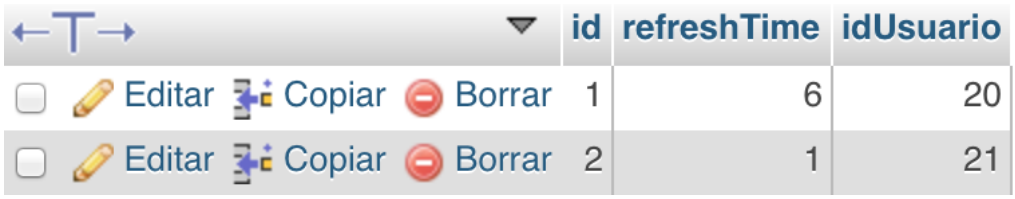
Ahora se observará con detalle de la base de datos creada y gestionada mediante la plataforma web de phpmyadmin.

Figura 10. Tabla de usuario

	idUsuario	nombreUsuario	password	nombre	apellidos	regId
Editar Copiar Borrar	20	admin	admin	José Mnú Manuel	García de la barca	0
Editar Copiar Borrar	21	admina	admin	ppe	panadero	0

FUENTE: Elaboración propia

Figura 10. Tabla de usuario



	id	refreshTime	idUsuario
Editar Copiar Borrar	1	6	20
Editar Copiar Borrar	2	1	21

FUENTE: Elaboración propia

Figura 12. Tabla de suscripción

	idSuscripcion	waitAccidentes	waitTrafico	waitDesperfectos	waitEventos	zona	idUsuario
Editar Copiar Borrar	8	1	0	0	0	Local	20
Editar Copiar Borrar	9	1	0	0	0	Global	21

FUENTE: Elaboración propia

Figura 13. Tabla de alerta

	idAlerta	latitud	longitud	titulo	descripcion	tipo	fecha	hora	idUsuario
Editar Copiar Borrar	92	39.4154588070036	-0.39823729544878	Accidente		Accidente	23/04/2014	23:58:09	20
Editar Copiar Borrar	95	46.8802365758129	-2.57813528180122	Accidente	Colisión	Accidente	28/04/2014	19:57:49	20
Editar Copiar Borrar	96	39.4192505224404	-0.397755168378353	Accidente	Choque frontal	Accidente	28/04/2014	20:02:58	20
Editar Copiar Borrar	97	32.3225450526688	11.2415973097086	Tráfico		Tráfico	28/04/2014	20:09:46	20
Editar Copiar Borrar	98	39.4254827437815	-0.385947413742542	Accidente		Accidente	02/05/2014	01:30:16	20
Editar Copiar Borrar	99	26.8806872477113	2.544573135674	Accidente		Accidente	02/05/2014	01:30:31	20
Editar Copiar Borrar	93	48.4583732277469	6.09374284744263	Tráfico		Tráfico	23/04/2014	23:59:29	21
Editar Copiar Borrar	94	42.2068416072408	13.5937444120646	Accidente		Accidente	24/04/2014	00:05:29	21

FUENTE: Elaboración propia

Como se puede observar en las diferentes figuras, se muestra un estado de la base de datos, con dos usuarios, con su configuración y suscripción correspondiente y una serie de alertas introducidas por ellos.

6.6 Servidor web

En esta sección abordaremos con más detalle la implementación del servidor y los diferentes servicios que ofrece.

Como ya se ha comentado en el punto anterior y en el diagrama de clases, es necesario almacenar información de los usuarios que usan la aplicación, de las alertas que están introducidas en el sistema, así como la configuración y la suscripción actual de un usuario en concreto.

Para todo esto, se ha implementado un API pública mediante PHP, la cual será descrita con más detalle. Los diferentes servicios se encontrarán alojados en dos entornos diferentes.

- Entorno de testing/desarrollo
- Entorno de producción

Se han creado dos entornos para evitar problemas mientras la aplicación evoluciona y así siempre el cliente, en uno de los dos entornos, puede estar usando la aplicación sin verse afectado por un bug menor u otra circunstancia.

En primer lugar, el entorno de testing/desarrollo, está pensado para ser usado en fase de desarrollo y poder realizar pruebas y comprobar que todo está funcionando como es debido. Este entorno se encontrará alojado en un servidor público gratuito, como es “Hostinguer”, debido a que no es necesario un gran ancho de banda ya que no van a haber muchos usuarios usando la aplicación simultáneamente.

Por lo tanto, al entorno de test/desarrollo, únicamente tendrán acceso aquellos usuarios que hagan de testers y el desarrollador, en este caso yo.

Por otro lado, el entorno de producción ha sido creado para el propio cliente y así éste puede estar usando la aplicación sin verse afectado por el desarrollo, es decir, el cliente se encuentra totalmente independiente y separado del desarrollo y del testing.

Para este entorno se ha usado un servidor privado ofrecido por la Universidad Politécnica de Valencia para alumnos (“Servicio de Publicación Web Avanzado”). Para más detalles de cómo obtener este servicito, etc…visita la siguiente URL.

http://www.upv.es/entidades/ASIC/catalogo/433436normalc.html

Ambos entornos están configurados con una versión actualizada de PHP y con MySQL. Además, en ambos, tenemos phpmyadmin para gestionar la base de datos con más facilidad.

6.6.1 API Pública

El API proporciona todos los servicios que la aplicación necesita para almacenar toda la información importante de un usuario y del propio sistema, como son las alertas.

Login

Este servicio se invoca mediante POST para poder llevar a cabo la autenticación de un usuario.

Como parámetros de entrada, el servicio espera los siguientes:

- $_POST[‘user’]. Nombre del usuario que está iniciando sesión.
- $_POST[‘pass’]. Contraseña del usuario que está iniciando sesión.

En cuanto a la respuesta, el servicio devuelve el usuario en JSON.

Crear alerta

Lleva a cabo la inserción de una alerta en el sistema. Como requiere de una serie de parámetros de entrada se invoca por POST. A continuación, se listarán los parámetros de entrada requeridos.

- $_POST[‘titulo’]. Título que se desea que tenga la alerta.
- $_POST[‘descripcion’]. Descripción de la alerta, aunque será un campo opcional (en la aplicación) se debe de enviar.
- $_POST[‘tipo’]. Tipo de alerta (Accidente, Desperfecto, Tráfico, Evento)
- $_POST[‘hora’]. Hora en la que se inserta la alerta. Ejemplo: 12:02:01
- $_POST[‘fecha’] . Fecha en la que se crea la alerta. Ejemplo: 12/06/2014
- $_POST[‘latitud’]. Dato necesario para la geolocalización de la alerta.
- $_POST[‘longitud’]. Dato necesario para la geolocalización de la alerta.
- $_POST[‘idUsuario’]. Id del usuario que ha creado/notificado la alerta.

Y como respuesta, el servicio devuelve un JSON con el “ID” de la alerta insertada.

Eliminar alertas

Ofrece la posibilidad de eliminar una alerta del sistema, de manera que los usuarios la dejan de visualizar si están conectados a Internet y/o, además, permite marcar como "resuelta" la alerta notificada.

Para llevar a cabo la operación debemos de invocar el servicio mediante POST y pasar como parámetro el identificador de la alerta que deseamos borrar.

- $_POST['id']. Id de la alerta que se desea borrar.

Si la operación tiene éxito, el servicio devolverá un 200 como código HTTP y si hay error devolverá "error" en el body del mensaje.

Obtener alertas

Este servicio se invoca mediante GET, ya que no requiere ningún parámetro de entrada, únicamente nos devolverá las alertas introducidas en la base de datos. El posterior filtrado se lleva a cabo en la aplicación, ya que el usuario puede cambiar su suscripción mientras está esperando nuevas alertas (notificaciones pull).

Por lo tanto, como respuesta, nos devuelve un JSON de alertas.

Existe usuario

Permite comprobar si un nombre de usuario ya está en uso. Se trata de un servicio invocado por POST.

Por lo tanto, como parámetros de entrada, espera:

- $_POST['nombreUsuario']. El nombre de usuario que se desea comprobar si existe.

Y como respuesta, devuelve *true* si existe y *false*, si no.

Crear configuración

La finalidad de este servicio es crear una configuración para un usuario en concreto.

Como se puede ver en el modelo de datos, la configuración lleva asociado un tiempo de refresco entre recepción de alertas. Este atributo es imprescindible para poder implementar el mecanismo de notificaciones pull.

Este servicio se invoca mediante POST y recibe los siguientes parámetros de entrada:

- $_POST[‘time’]. Nuevo tiempo de refresco entre alertas.
- $_POST[‘id’]. Id del usuario al cual se le va a crear una configuración.

En cuanto a la respuesta, devuelve “error” si hay fallo o un código HTTP 200 si ha tenido éxito.

Actualizar configuración

Este servicio permite actualizar la configuración actual de un usuario, mediante el envío por POST de los siguientes parámetros:

- $_POST[‘id’]. Identificador del usuario que desea editar su configuración.
- $_POST[‘time’]. Nuevo tiempo de refresco entre alertas.

En cuanto a la respuesta, devuelve “error” si hay fallo o un código HTTP 200 si ha tenido éxito.

Crear suscripción

Este servicio genera una suscripción para un usuario, es decir marca que tipos de alertas quiere recibir un usuario. Para ello, la aplicación debe de invocar este servicio mediante POST, enviando los siguientes parámetros:

- $_POST[‘idUsuario’]. Identificar del usuario que desea crear una nueva suscripción.
- $_POST[‘accidentes’]. Valor “true”, si desea recibir alertas de tipo accidente, “false” si no.
- $_POST[‘desperfectos’]. Valor “true”, si desea recibir alertas de tipo desperfecto, “false” si no.
- $_POST[‘eventos’]. Valor “true”, si desea recibir alertas de tipo evento, “false” si no.
- $_POST[‘trafico’]. Valor “true”, si desea recibir alertas de tipo tráfico, “false” si no.
- $_POST[‘zona’]. Parámetro con valor “Local” o “Global” para especificar el tipo de suscripción. En anteriores secciones se ha comentado el significado de estos parámetros.

Es importante recordar que un usuario no puede tener nos suscripciones activas.

Por último, este servicio devuelve “error” si hay fallo o un código HTTP 200 si ha tenido éxito la creación de la suscripción.

Editar suscripción

Este servicio permite la edición de la suscripción de un usuario. Por lo tanto, para invocar este servicio se hace mediante POST, enviando los siguientes parámetros en el body:

- $_POST$_POST[‘idUsuario’]. Identificar del usuario que desea crear una nueva suscripción.
- $_POST[‘accidentes’]. Valor “true”, si desea recibir alertas de tipo accidente, “false” si no.
- $_POST[‘desperfectos’]. Valor “true”, si desea recibir alertas de tipo desperfecto, “false” si no.
- $_POST[‘eventos’]. Valor “true”, si desea recibir alertas de tipo evento, “false” si no.
- $_POST[‘trafico’]. Valor “true”, si desea recibir alertas de tipo tráfico, “false” si no.
- $_POST[‘zona’]. Parámetro con valor “Local” o “Global” para especificar el tipo de suscripción. En anteriores secciones se ha comentado el significado de estos parámetros.

En cuanto a la respuesta, devuelve “error” si hay fallo o un código HTTP 200 si ha tenido éxito la creación de la suscripción.

Obtener suscripción

Permite obtener un JSON con la suscripción actual de un usuario. Para ello, únicamente se envía el identificador del usuario del cual se desea obtener la suscripción por POST.

- $_POST[‘idUsuario’]. Identificador de usuario

Por lo tanto, como respuesta, devolverá un JSON con los datos de la suscripción del usuario.

Editar usuario

Este servicio ofrece la posibilidad de editar los datos de un usuario. Para ello, se invoca mediante POST

Lo único que no se puede modificar de un usuario son su nombre y sus apellidos, ya que no es muy normal cambiar el nombre o el apellido dentro de la aplicación.

Como parámetros de entrada, se deben de enviar los siguientes:

- $_POST[‘id’]. Identificar del usuario que se desea editar.
- $_POST[‘nombreUsuario’]. Nuevo nombre de usuario para el usuario que se desea editar.
- $_POST[‘password’]. Nueva contraseña para el usuario.

Como respuesta, devuelve un código HTTP 200 si ha tenido éxito la consulta o “error” si no.

Subir imagen

Este servicio se encarga de subir actualizar los datos de una alerta guardando una imagen si así lo desea el usuario.

Únicamente, se debe de enviar por POST el identificar de la alerta a la cual se le desea añadir una imagen y la imagen codificada en base 64.

- $_POST[‘image’]. Imagen codificada en BASE 64.
- $_POST[‘id’]. Identificador de la alerta a la cual se le desea añadir la imagen.

Como respuesta, devuelve un código HTTP 200 si ha tenido éxito la consulta o “error” si no.

Registro

Permite llevar a cabo el registro de un usuario en la aplicación. Para ello, se envían mediante POST los siguientes parámetros.

- $_POST[‘nombre’]. Nombre de la persona que desea registrarse.
- $_POST[‘nombreUsuario’]. Nombre de usuario de la persona que desea registrarse, es decir, el nombre con el que un usuario desea iniciar sesión en la aplicación.
- $_POST[‘apellidos’]. Apellidos de la persona que desea registrarse, teniendo en cuenta que puede poner uno, dos, etc.
- $_POST[‘password’]. Contraseña de usuario, con la cual el usuario llevará a cabo el login.

Como respuesta, el servicio nos devuelve “error” si hay fallo o un código HTTP 200 si ha tenido éxito.

6.7 Estructura de la aplicación Android

En este punto, pasaremos a analizar la estructura de la aplicación. Cabe destacar que la aplicación se encuentra estructurada en cinco módulos que permiten que la aplicación tenga poco acoplamiento.

Una aplicación con poco acoplamiento tiene el beneficio de que sus módulos se pueden reutilizar con más facilidad en otras aplicaciones.

En primer lugar, tenemos el módulo de ***model,*** en el cual se alojaran cada una de las clases del modelo de la aplicación.

En segundo lugar, tenemos el módulo de ***controllers***, módulo en el cual residirán los diferentes controladores de las vistas, en este caso las Activity.

En tercer lugar, tenemos un módulo de ***services***, donde implementaremos los servicios que la aplicación requiera, como por ejemplo el mecanismo de notificaciones pull.

Por otro lado, de una manera más diferenciada, tenemos el módulo de ***utils***. En este módulo alojaremos cualquiera de los adaptadores que se requieran en la aplicación, tales como adaptadores para las listas.

Y por último, tenemos el módulo de ***res o (resources)*** en el cual se alojarán los recursos del sistema y en especial los layouts (interfaces de usuario).

Con esta estructura podemos implementar el Modelo-Vista-Controlador (MVC), con el cual conseguimos los siguientes beneficios:

- Toda vista tiene un controlador asociado que se encarga de gestionar todo aquello que realiza el usuario en esa vista, es decir la interactuación.
- Todo aquello que sea modificado en el modelo será directamente visualizado en la vista.

Estos beneficios, hacen que se trate de un patrón de diseño muy utilizado por los desarrolladores software.

En el anexo se especificarán una serie de buenas prácticas que se han llevado a cabo en el desarrollo de la aplicación.

7. Detalles de mantenimiento

En esta sección abordaremos como mantener los dos módulos principales de la aplicación y analizaremos su funcionamiento para una mejor comprensión.

7.1 Introducción

Llevando a cabo una pequeña introducción, cabe destacar que la mantenibilidad, se puede definir como el esfuerzo que tiene que hacer una persona, a veces ajena a la organización, para entender el funcionamiento del sistema y poder realizar cambios si ocurre algún fallo o algo similar. Como es obvio, un sistema es poco mantenible si no sigue buenas prácticas de desarrollo, si el código es ilegible, etc.

Por lo tanto, la mantenibilidad de un sistema software se puede relacionar directamente con el esfuerzo de requerido para que este sea "entendido".

En el desarrollo software, la mantenibilidad es un aspecto muy importante, puesto que los sistemas, a lo largo de su vida, sufren o necesitan cambios para adaptarse al entorno o a las necesidades del cliente.

En muchos casos, la mantenibilidad está relacionada con el acoplamiento, ya que un sistema con un acoplamiento muy alto suele ser mantenible. Esto se debe a que si los diferentes módulos que forman un sistema dependen demasiado unos de los otros se está desarrollando algo que no va a poder ser reutilizado ni fácilmente modificable.

En conclusión, es muy importante un sistema con una mantenibilidad baja, con poco acoplamiento y modularizado.

En los siguiente subapartados, analizaremos el módulo de recepción de alertas y el de suministrar la posición (listener de posición).

7.2 Receptor de alertas

Se trata de un servicio que es ejecutado mediante el AlarmManager de Android para que éste se ejecute cada cierto tiempo a modo de alarma.

El intervalo de refresco, ya mencionado anteriormente, hace referencia al tiempo que debe de pasar entre ejecución y ejecución de este servicio. Este intervalo debe de ser especificado de manera programática a la hora de lanzar a ejecución el servicio.

public void **setRepeating** (int type, long triggerAtMillis, long intervalMillis, PendingIntent operation) Added in API level 1

Parameters

type	One of `ELAPSED_REALTIME`, `ELAPSED_REALTIME_WAKEUP`, `RTC`, or `RTC_WAKEUP`.
triggerAtMillis	time in milliseconds that the alarm should first go off, using the appropriate clock (depending on the alarm type).
intervalMillis	interval in milliseconds between subsequent repeats of the alarm.
operation	Action to perform when the alarm goes off; typically comes from `IntentSender.getBroadcast()`.

En la imagen superior, vemos el método que debe de ser invocado con una instancia del AlarmManager.

A continuación, mostramos la instanciación de este método por parte de la aplicación.

```
    alarmManager.setRepeating(AlarmManager.RTC_WAKEUP, Calendar
            .getInstance().getTimeInMillis(), user.getContiguration()
            .getRefreshTime() * 60000, pendingIntent);
```

Como se puede ver en la figura, definimos el intervalo a partir del tiempo de refresco especificado en la configuración del usuario.

Una vez lanzado el servicio por el AlarmManager se pasa a ejecución el servicio de recepción de alertas.

```
class BucarAlertasEjecucion extends AsyncTask<Void, Alert, Boolean> {

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        if (MainMenuActivity.getUser() == null) {
            this.cancel(true);
            stopSelf();
        }


    }

    @Override
    protected Boolean doInBackground(Void... params) {
        http = new Httppostaux();
        ArrayList<NameValuePair> postParameters = new ArrayList<NameValuePair>();

        JSONArray jsonData;
        try {
            jsonData = http
                    .getserverdata(postParameters,
                            com.alertmap.utils.Constants.URL_SERVER+"alertasBySuscripcion.php");
        } catch (IOException e1) {
            Log.e("SERVER-ERROR", e1.getMessage());
            Utils.showProblemComunication(getApplicationContext());
            return false;
        }
        int i = 0;
        MainMenuActivity.getUser().deleteAlertsNoRelatedToSubscription();
        ArrayList<Alert> alertas = new ArrayList<Alert>();
        while (jsonData != null && i < jsonData.length()) {
            JSONObject objetoJSON = null;
            double latitud;
            double longitud;
            String titulo, descripcion, tipo, fecha, hora;
            Alert alerta = null;
            int idAlerta;
            Bitmap bitmap = null;

            try {
                objetoJSON = jsonData.getJSONObject(i);
                idAlerta = objetoJSON.getInt("idAlerta");
                latitud = objetoJSON.getDouble("latitud");
                longitud = objetoJSON.getDouble("longitud");
                titulo = objetoJSON.getString("titulo");
```

```
                descripcion = objetoJSON.getString("descripcion");
                tipo = objetoJSON.getString("tipo");
                fecha = objetoJSON.getString("fecha");
                hora = objetoJSON.getString("hora");
                if (!objetoJSON.get("image").toString().equals("null")) {
                    String encodingImage = (String) objetoJSON.get("image");
                    bitmap = Utils.decodeBase64(encodingImage);
                }

                alerta = new Alert(idAlerta, titulo, descripcion, latitud,
                        longitud, tipo, fecha, hora);
                if (bitmap != null) {
                    alerta.setBitmap(bitmap);
                }
                alertas.add(alerta);
                if (!MainMenuActivity.getUser().isDisplayed(alerta)) {
                    if (MainMenuActivity.getUser()
                            .addAlertBySubscription(alerta))
                        publishProgress(alerta);
                }
                i++;

            } catch (JSONException e) {
                e.printStackTrace();
                return null;
            }

        }
        return true;

    }

    @Override
    protected void onCancelled() {
        super.onCancelled();
        stopSelf();
    }

    @Override
    protected void onProgressUpdate(Alert... values) {

        // Envio de notificación
        Intent intent = new Intent();
        intent.setAction("com.alertmap.receivingalerts");
        sendBroadcast(intent);




     }

     @Override
     protected void onPostExecute(Boolean result) {
         if (!result) {
             this.cancel(true);
             stopSelf();
         }
     }

}// asyncTask
```

El servicio de recepción de alertas ejecuta interiormente la AsyncTask definida en la figura superior.

Esta tarea asíncrona se ejecuta en segundo plano, mientras el usuario lleva a cabo otras acciones.

El objetivo de esta tarea, como se puede ver en el "doInBackground()", es obtener las alertas introducidas en la base de datos, comprobar que cumple con la suscripción actual del usuario y sí cumple y no está visualizado, entonces se envía un "broadcast" con el fin de crear una notificación para el usuario de la aplicación.

Tal y como se puede visualizar, la tarea asíncrona se encuentra protegida por si ésta es cancelada, y por lo tanto así evitar algún problema si Android decide finalizar este servicio por falta de memoria RAM en el dispositivo.

7.3 Listener de posición

Otro módulo muy importante de la aplicación es el “listener” de posición.

Este servicio se encarga de obtener la posición actual del usuario, comprobando constantemente si este se ha movido y si la nueva posición obtenida es más precisa respecto a la obtenida con anterioridad.

Para ello, este servicio se suscribe a los suministradores de posición de Android: “Network provider” y “GPS provider”. El primero, requiere de tener conexión a Internet.

Cabe destacar que ambos proporcionan posiciones independientemente del otro proveedor, por ese motivo es necesario comprobar que posición es más precisa.

En primer lugar, para que el servicio obtenga posiciones tenemos que solicitar actualizaciones a los proveedores ya citados anteriormente.

```
locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
listener = new LocationListener();
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 4000, 0, listener);
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 4000, 0, listener);
```

Asimismo, como se puede observar en la figura superior, se asocia a estas peticiones un listener. Este listener recibirás las posiciones suministradas y será el encargado de decidir si una posición es más precisa que la otra y más nueva o más vieja que la otra.

```
protected boolean isBetterLocation(Location location, Location currentBestLocation) {
    if (currentBestLocation == null) {
        return true;
    }

    // Check if the new location is newer.
    long tiempo = location.getTime() - currentBestLocation.getTime();
    boolean isSignificantlyNewer = tiempo > TWO_MINUTES;
    boolean isSignificantlyOlder = tiempo < -TWO_MINUTES;
    boolean isNewer = tiempo > 0;

    if (isSignificantlyNewer) {
        return true;
    } else if (isSignificantlyOlder) {
        return false;
    }

    int precisionPosicion = (int) (location.getAccuracy() - currentBestLocation.getAccuracy());
    boolean isLessAccurate = precisionPosicion > 0;
    boolean isMoreAccurate = precisionPosicion < 0;
    boolean isSignificantlyLessAccurate = precisionPosicion > 200;

    boolean isFromSameProvider = isSameProvider(location.getProvider(), currentBestLocation.getProvider());
    if (isMoreAccurate) {
        return true;
    } else if (isNewer && !isLessAccurate) {
        return true;
    } else if (isNewer && !isSignificantlyLessAccurate && isFromSameProvider) {
        return true;
    }
    return false;
}
```

Para ello, se implementa el método “isBetterLocation ()”. En este método, en primer lugar, se comprueba si la nueva posición es más nueva o más vieja que la ya obtenida y si la diferencia no es relativamente importante, comprueba si la nueva posición es más o menos precisa.

Una vez obtenidos todos los datos, se puede tomar una decisión coherente y que hace que la aplicación funcione perfectamente.

Este snippet ha sido extraído y adaptado a nuestra aplicación desde “http://stackoverflow.com/questions/14478179/background-service-with-location-listener-in-android”.

8. Caso práctico de la aplicación (Demo)

En esta sección procederemos a mostrar una demo de la aplicación para poder visualizar de una manera directa y diferente las diferentes funcionalidades que la aplicación nos proporciona.

Asimismo, se presentarán los diseños finales de las vistas. Es importante destacar que estos diseños están basados principalmente en los prototipos, sin embargo hay que tener en cuenta que se pueden llevar a cabo variaciones que adapten la aplicación más a la plataforma.

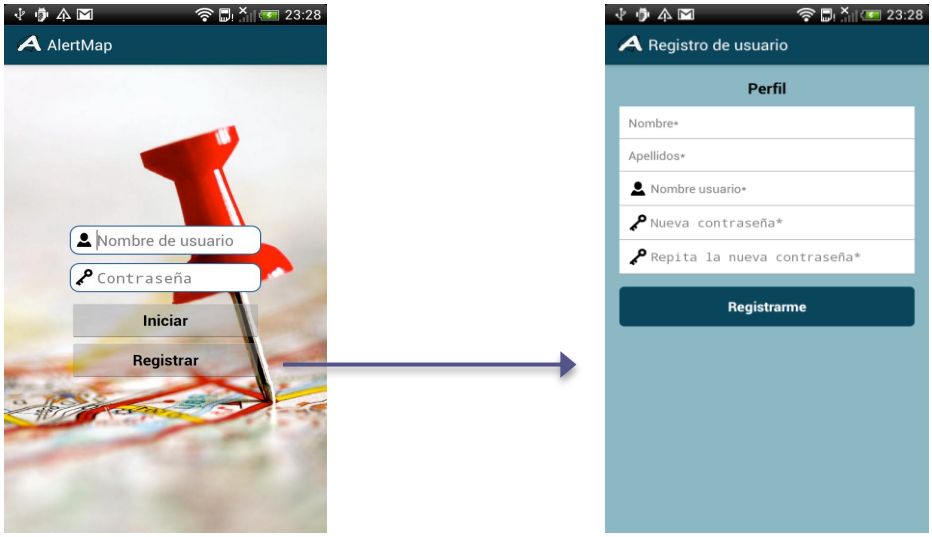
En muchos casos, una misma aplicación no puede ser visualizada de la misma manera en Android e iOS, debido a las peculiaridades técnicas de cada una de las plataformas. Sin embargo, es importante no desviarse del prototipo, puesto que cabe recordar que la validación del software se realiza a partir del prototipado.

"A groso modo", el objetivo principal de esta sección es presentar las funcionalidades de la aplicación de manera que sirva también de guía para el usuario.

8.1 Usuario no registrado

Un usuario para registrarse accederá a la aplicación y podrá acceder a la vista de registro, simplemente pulsando en el botón de "Registrarse".

Figura 15. Inicio de sesión, usuario no registrado.



FUENTE: Aplicación Alertmap

Una vez el usuario se encuentre en el registro deberá de rellenar de una manera obligatoria todo campo marcado con * en la figura 15, ya que como es habitual así se indica que el campo es obligatorio.

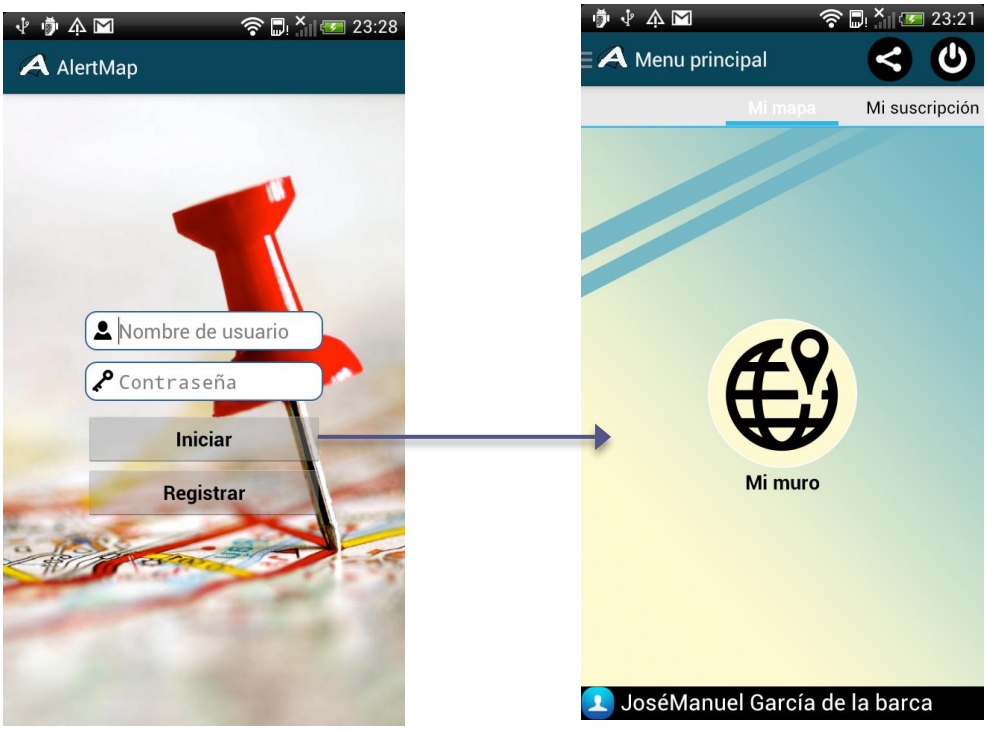
Por lo tanto, se deberá de rellenar el formulario al completo. Hay que tener en cuenta que la contraseña debe de tener al menos 6 caracteres, si no esta se marcará como incorrecta. Además, es muy importante que el nombre de usuario sea fácil de recordar puesto que será la puerta para acceder a la aplicación.

Durante el registro, como ya se comentó en partes anteriores del documento, se establece una suscripción por defecto suscribiendo al usuario a todo tipo de alertas de todo el mundo.

8.2 Usuario registrado

En este subapartado, se abordará la funcionalidad que se le proporciona a un usuario registrado de nuestra aplicación AlertMap.

Figura 16. Menú principal, usuario registrado.



FUENTE: Aplicación alertmap

En la imagen superior, figura 16, podemos ver el menú principal de nuestra aplicación, el cual pasará a primer plano cuando el usuario haya introducido correctamente los valores de nombre de usuario y contraseña y estos hayan sido validados posteriormente en la parte del servidor.

El usuario, en la vista del menú principal, podrá visualizar su nombre en la parte inferior (footer) de manera que se proporciona feedback. El usuario sabe en todo momento quien es.

Por otro lado, en la parte superior (action bar), el usuario podrá llevar a cabo dos funcionalidades la de compartir y la de cerrar la sesión activa para poder iniciar la aplicación con otra cuenta de usuario.

En cuanto a la función de compartir, ésta está diseñada para proporcionar publicidad a la aplicación, ya que como la aplicación es totalmente gratuita hay que generar algo de marketing/publicidad y poder obtener beneficios de una manera indirecta.

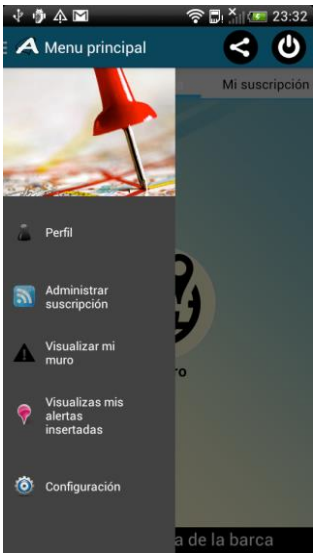
Cuando un usuario pulse sobre "Compartir" [1], visualizará un diálogo para seleccionar el método mediante el cual desea compartir. Los métodos disponibles serán Whatsapp y Twitter. Sin embargo, si un dispositivo sólo dispone de una de las dos opciones no se mostrará el diálogo para compartir y directamente se seleccionará esa aplicación.

Por otro lado, tenemos también en la parte superior izquierda un menú deslizable con una serie de opciones, éste nos permitirá navegar a otras partes de la aplicación que son menos importantes para el funcionamiento de la aplicación pero que nos proporcionan un extra de configuración para el usuario y la app en cuestión.

Cabe destacar, que el menú se puede visualizar tanto haciendo swipe desde el extremo izquierdo hacia la derecha como pulsando en el botón de la parte superior izquierda.

[1] De esta funcionalidad no se inserta captura para no comprometer de ninguna manera los datos personales de una cuenta personal de Whatsapp o Twitter.

Figura 17. Menú deslizable



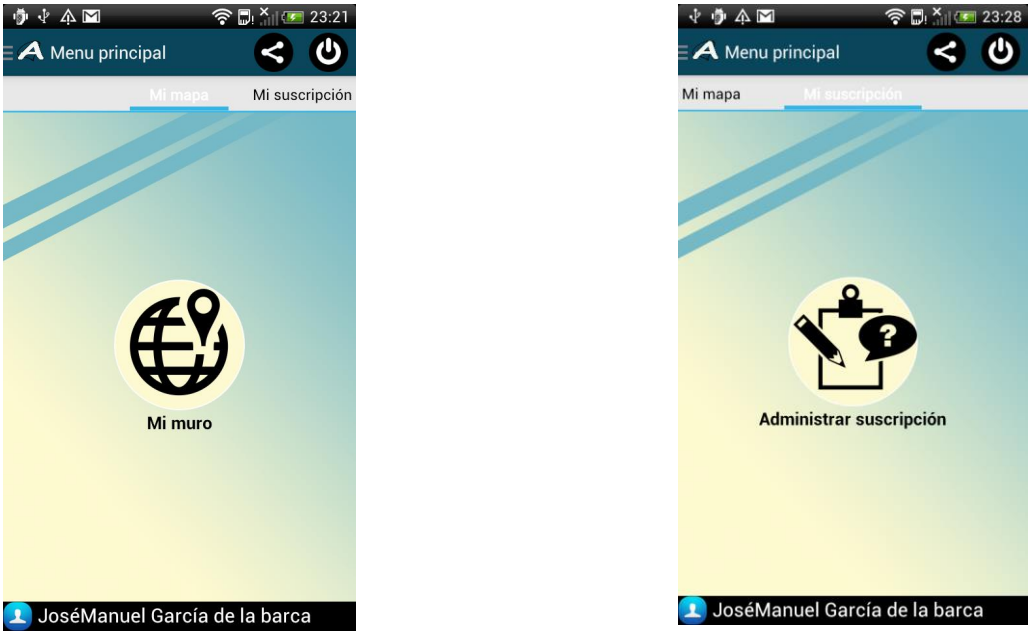
FUENTE: Aplicación Alertmap

Como podemos ver en la figura 17, si se hace swipe desde el centro de la pantalla hacía la derecha o pulsando sobre las tabs superiores podemos pasar de un menú a otro, estos menús están diseñados para un acceso rápido para acceder a "Mi muro" y a "Administrar suscripción".

En "Mi muro", el usuario pasará a visualizar las alertas a las que se encuentra suscrito y que aún no han sido eliminadas/resueltas.

Y en "Administrar suscripción", se puede acceder al detalle de la suscripción actual del usuario que ha iniciado sesión.

Figura 18. Menús principales



FUENTE: Aplicación Alertmap

A continuación, abordaremos las opciones que son presentados en el menú deslizable de la figura 17.

En primer lugar, tenemos el "Perfil", esta opción permite al usuario acceder a una vista donde podrá modificar sus datos de usuario.

En segundo lugar, tenemos la opción de "Administrar suscripción" que será explicada más adelante con mayor detalle.

Como tercera opción, tenemos "Visualizar mi muro" para abrir el mapa de las alertas a las que te encuentras suscrito.

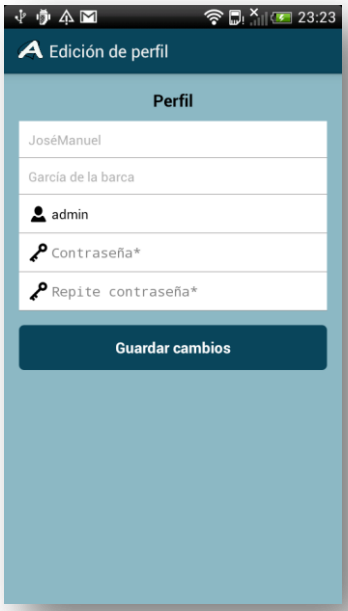
En cuarto lugar, se presenta la opción de "Visualizar mis alertas insertadas" en la que pulsando, accederemos a un listado de nuestras alertas insertadas.

Y por último, tenemos la opción "Configuración" en la cual podremos modificar el intervalo de refresco de la aplicación.

Cabe destacar que las opciones 2 y 3 de la figura 17, han sido incluidas para dar consistencia al diseño, ya que el menú principal se utiliza de acceso rápido a las funcionalidades principales de la aplicación.

Perfil

Figura 19. Perfil



El usuario podrá modificar sus datos, sin posibilidad de modificar su nombre y apellido, ya que no tiene cabida que un usuario cambie su nombre. Sin embargo, si puede modificar su nombre de usuario y su contraseña.

Para finalizar la edición y confirmar que desea llevar a cabo esos cambios, debe de pulsar en el botón de "Guardar cambios".

FUENTE: Aplicación Alertmap

Administrar suscripción

Figura 20. Suscripción

El usuario podrá visualizar de una manera rápida su suscripción actual, además podrá cambiarla sin necesidad de introducir ningún dato desde teclado.

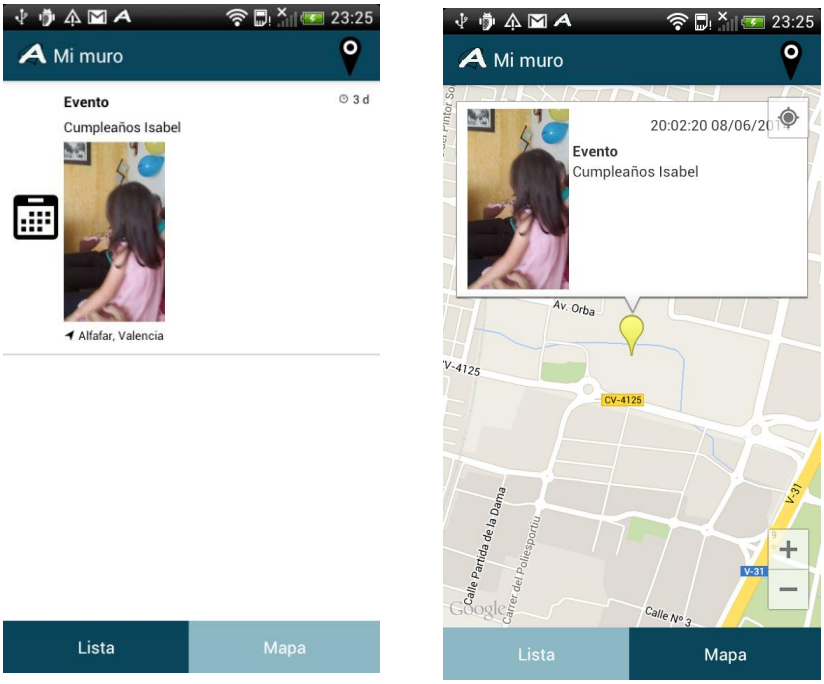
Únicamente, se debe de pulsar en los tipos de alertas a los que te quieres suscribir, es decir de los que deseas obtener alertas. También se puede seleccionar el tipo de zona de tu suscripción, local o global.

Por último, para confirmar el cambio en la suscripción pulsamos en "Guardar cambios".

FUENTE: Aplicación alertmap

Visualizar mi muro

Figura 21. Visualizar mi muro



FUENTE: Aplicación Alertmap

Como ya se ha mencionado anteriormente, esta opción está accesible tanto desde el menú deslizable como por el menú principal de acceso rápido.

En esta vista se pueden visualizar las diferentes alertas a las que se encuentra suscrito el usuario.

Esta vista tiene dos posibilidades: vista de lista o vista de mapa.

En la vista de lista, se ven las alertas por orden descendente en el tiempo, es decir, la última alerta será la primera de la lista. En esta vista, se pueden visualizar todos los detalles de una alerta de un simple vistazo, reduciendo el tiempo a la hora de consultar las alertas nuevas.

Y por otro lado, tenemos la vista de mapa que permite visualizar de manera rápida dónde está la alerta, ver una imagen de la alerta, si el usuario ha decidido introducir una, una descripción y la fecha en la que ha sido introducido, es decir recibida por el sistema.

Para ver estos detalles, el usuario deberá de hacer click sobre la alerta (marker) para visualizar la ventana de detalle.

Por otro lado, el usuario podrá añadir/crear nuevas alertas si así lo desea. Para ello, se debe de realizar una pulsación larga sobre el punto del mapa dónde deseamos introducir ese alerta.

Ante la pulsación larga, visualizaremos una vista para crear una alerta.

Notificación de alertas

En esta vista, muy parecida a la vista de suscripción, podemos señalar un tipo único de alerta, introducir de manera opcional una descripción de máximo 100 caracteres y adjuntar una imagen, la cual tiene que ser capturada en tiempo real.

Es importante destacar que una alerta introducida en el sistema puede no ser visualizada por la persona que la crea, ya que un usuario únicamente visualizará aquellas alertas que cumplen con la suscripción actual. Sin embargo, esta será insertada en el sistema y será recibida por todas aquellas personas que según su suscripción la deseen recibir.

Como se puede ver en la imagen, estaríamos creando una alerta de tipo “Accidente”, con una descripción de colisión múltiple y sin una imagen adjuntada.

Una imagen puede ser cargada y posteriormente remplazada si el usuario decide cambiar su imagen.

Si el usuario carga una imagen y luego decide no quererla enviar, deberá de pulsar sobre capturar imagen y volver hacia detrás.

Figura 22. Crear alerta

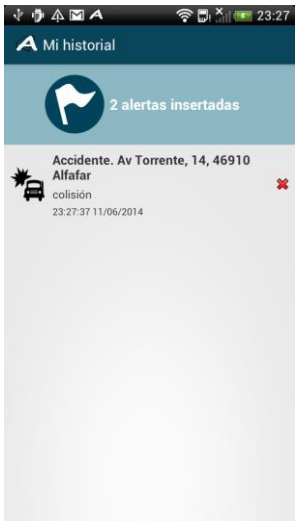
FUENTE: Aplicación Alertmap

Visualizar mis alertas insertadas

Es está vista visualizaremos, los datos de las alertas insertadas, en nuestro caso, visualizamos únicamente la alerta que acabamos de insertar.

En la lista, podemos ver el tipo de alerta introducida, cuándo, qué ha pasado y en qué dirección.

Figura 23. Mis alertas insertadas



Observando la figura, vemos que la alerta ha sido introducida en la "Av.Torrente, 14", con código postal 46910 (Alfafar) y que ha sido generada a las 23:27:37 del 11/6/2014.

Asimismo, podemos ver que a la derecha de cada fila tenemos una cruz para poder borrar una alerta cuando se desee, es decir pa ra marcarla como resulta y así que los usuarios no la reciban.

FUENTE: Aplicación Alertmap

Configuración

Como última parte del software, podemos realizar un cambio de configuración que afectará al mecanismo de notificaciones pull, tal y como se ha explicado anteriormente en la especificación de requisitos.

Figura 24. Configuración

FUENTE: Aplicación Alertmap

Aumentando y disminuyendo el intervalo de refresco en minutos, una vez se ha tomado la decisión presionamos en “Aceptar”. Sin embargo, si el usuario no quiere llevar a cabo los cambios, es decir confirmarlos, puede volver atrás mediante el botón nativo del dispositivo Android que esté siendo utilizado.

9. Evaluación del sistema

La aplicación presentada, AlertMap, ha estado instalada en un dispositivo móvil físico con una versión de Android 4.0.3 (4.0+).

9.1 Características del dispositivo físico

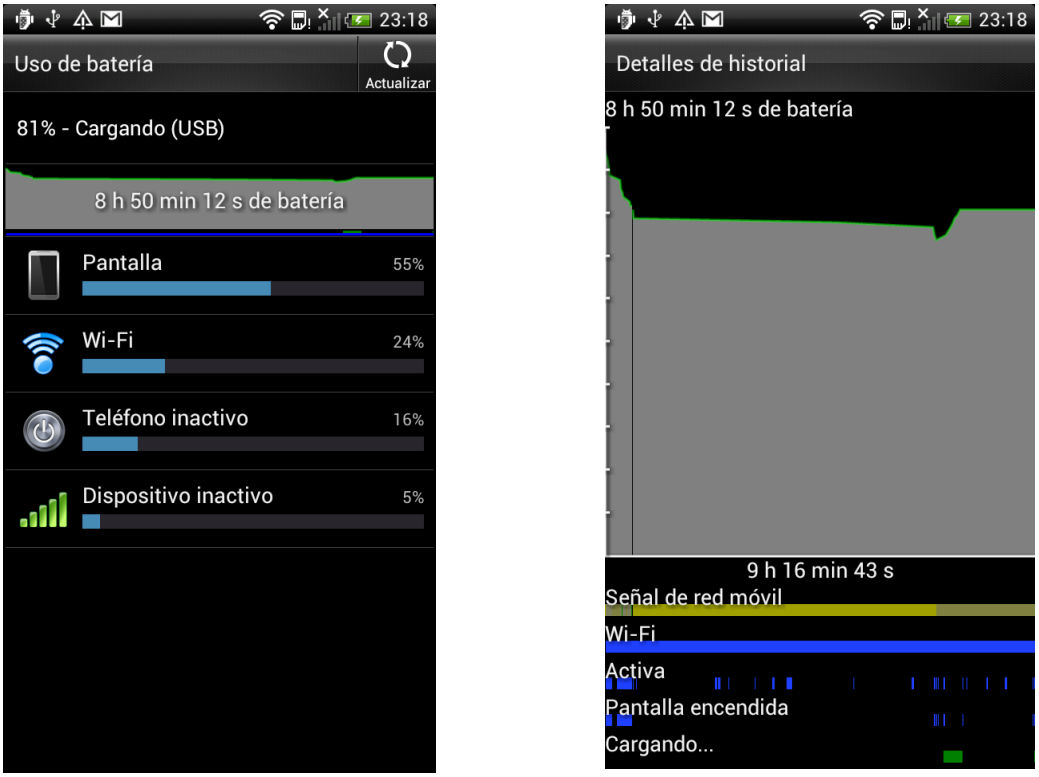
FUENTE: infomoviles.com

Característica	Detalle
Procesador	1.2 GHz dualcore
Memoria	768MB
Pantalla	4.3” QHD resolution
Cámara principal	8MB
Wi-Fi	802.11 b/g/n
Bluetooth	Sí
Batería	Standard, Li-Ion 1520 mAh

9.2 Consumo de batería

Tras casi 10 horas, con la aplicación instalada, el consumo de energía por parte de la aplicación ha sido insignificante comparado con el de otras Apps.

Figura 25. Consumo de batería 1



FUENTE: Dispositivo móvil personal

Como se puede ver en la figura 25, en el detalle de consumo de la batería, la aplicación no aparece entre las que más consumen. Además, en el gráfico podemos observar que el consumo de batería por parte de la aplicación ha sido lineal mientras este no se ha encontrado conectado a la red eléctrica cargando.

Hay que tener en cuenta, que es muy importante que el tipo de suscripción del usuario sea de zona “Global”, ya que el GPS no está en constante funcionamiento para obtener la posición del usuario y comprobar si este se mueve.

Esto es muy importante, debido a que el funcionamiento del GPS en versiones de Android inferiores a la 4.0 consume una cantidad importante de batería.

A continuación, analizaremos el rendimiento de nuestra aplicación indicando el tipo de zona “Local” y comprobando el consumo de batería.

Figura 26. Consumo de batería 2

FUENTE: Dispositivo móvil personal

Tal y como se puede visualizar en la imagen superior, el consumo de batería por parte de la aplicación no se puede considerar elevado, ya que no aparece entre las aplicaciones que más consumen.

Este análisis se ha llevado a cabo dejando el dispositivo con la aplicación instalada y con una sesión iniciada con suscripción de tipo "Local". Como ya hemos mencionado anteriormente, indicando este tipo de suscripción hacemos que la aplicación intente obtener la posición actual del dispositivo en todo momento.

Este tipo de configuración de la aplicación, podría generar unos consumos elevados de la aplicación, sin embargo, parece que el consumo de batería por parte de este dispositivo no se ve afectado por la configuración.

9.3 Fluidez de la IU

Otro punto a destacar en la evaluación, es la fluidez de la interfaz de usuario. El intercambio entre las diferentes vistas es bastante fluido y si alguna vez es más lento de lo debido se debe a respuestas por parte del servidor, ya que se muestra un dialogo de progreso indicando la operación en el lado del servidor. Con un servidor más potente el sistema funcionaría realmente bien.

En cuanto al diseño responsivo, la aplicación se adapta perfectamente al dispositivo que sea, siempre y cuando la API en uso sea 11 o superior, ya que en dispositivos que hagan uso de un API inferior esta aplicación no puede ser instalada.

La aplicación gracias a su diseño responsivo puede ser utilizada en móviles y tabletas, puesto que las diferentes vistas se redimensionan teniendo en cuenta el tamaño de la pantalla y de la densidad de pixeles. Esto se ha conseguido realizando un diseño independiente de la densidad de pixeles y usando posiciones relativas.

9.4 Detalles relevantes de la IU

En cuanto a algunos detalles a destacar de la IU, se podría resaltar el uso de un "Navigation Drawer", ya que la mayoría de aplicaciones ya hacen uso de este nuevo elemento que hace que las diferentes funcionalidades de la aplicación puedan ser accedidas rápidamente y se puedan visualizar todas las opciones de un simple vistazo.

El "Navigation Drawer" es el elemento descrito en este documento como menú deslizable.

Otro punto a destacar, sería el uso de un ViewPager que también es usado en muchas aplicaciones y da un aspecto más moderno a la aplicación así como sofisticado. Además, el "ViewPager" en nuestra aplicación es usado para acceder a las principales funcionalidades de la aplicación y proporcionar un entorno de acceso rápido.

10. Conclusiones

En conclusión, el sistema software desarrollado, AlertMap, se trata de un sistema notablemente rápido, atractivo para el usuario y con un módulo diseñado para publicitar el mismo.

Debido a que es gratuito el sistema no generará ningún beneficio por venta pero se espera obtener beneficios por publicidad.

Además, en un futuro se puede analizar si la situación del producto aceptaría una nueva ampliación del mismo para implementar nuevas funcionalidades útiles para el usuario y promover así una versión Premium o de pago con estas nuevas funcionalidades.

A modo de resumen, AlertMap, nos proporcionará la capacidad de notificar/alertar a otras personas de situaciones cotidianas o no de una manera rápida.

Entre los puntos a destacar y que ya han sido analizados en apartados anteriores, la rapidez del sistema se debe primordialmente a que ningún usuario se ve obligado a insertar texto para llevar a cabo cualquiera de estas dos acciones principales, consultar alertas y/o crear alertas.

Además, esta rapidez también se debe al diseño de la interfaz de usuario ya evaluado en la evaluación del sistema y que proporciona una visión moderna de la aplicación.

Por lo tanto y para finalizar, este sistema, actual, moderno y con un rendimiento notable, nos proporciona funcionalidades muy útiles para el día a día y sobre todo para aquellas personas que cogen el coche a diario.

Teniendo en cuenta el problema planteado y la propuesta “idealmente” planteada el resultado es más que satisfactorio.

11. Anexo

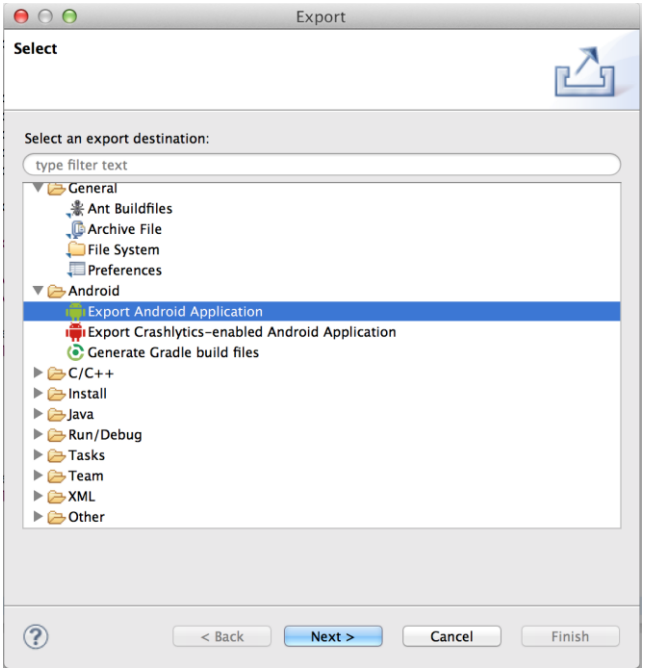
11.1 Manual de generación APK e instalación

Generación APK

En primer lugar, para generar el APK de la aplicación hacemos click derecho sobre el proyecto y pulsamos sobre “Export…”.

A continuación, seleccionamos la opción de “Export Android Application…”

Figura 27. Generación APK



FUENTE: Eclipse ADT Bundle

Durante el proceso de exportación, debemos de seleccionar el proyecto a exportar y seleccionar que vamos a usar una “keystore” existente o a crear una específica para el proyecto.

Hay que tener en cuenta que el firmado de la aplicación es muy importante y que los Google Maps dependen de la “keystore” con la que se está firmando la aplicación. Esto significa que se tiene que dar de alta en “Google developers console” toda “keystore” con la que se va a firmar la aplicación.

Para más detalle sobre el firmado de aplicaciones es interesante visitar este enlace "http://developer.android.com/tools/publishing/app-signing.html".

Una vez finalizado el proceso de exportación, habremos generado un APK, el cual debemos de ejecutar cuando deseemos instalar la aplicación.

Instalación

Para instalar la aplicación, simplemente deberemos de acceder a los ajustes a "Opciones de desarrollador" y habilitar la opción de permitir "Fuentes desconocidas".

Una vez habilitada la opción, únicamente debemos de pulsar sobre el APK que nos hayan enviado y aceptar los permisos que la aplicación solicita para ser instalada. ¡Ya podemos disfrutar de AlertMap"!

Es importante recordar, que si tenemos "AlerMap" ya instalado, para hacer una instalación limpia debemos de borrar los datos de la aplicación ya existente y posteriormente desintalarla.

Por último, cabe destacar que este proceso es útil cuando la aplicación no se encuentra en el "market" de Android, ya que si está en el "market" solo debemos de pulsar sobre la aplicación e instalarla de manera sencilla.

11.2 Guía de uso

En este anexo, analizaremos paso por paso como poder disfrutar de las diferentes funcionalidades que proporciona el sistema software, AlertMap.

Inicio de sesión

1. Ejecutar la aplicación AlertMap desde el menú de aplicaciones.

2. Insertar un nombre de usuario y contraseña válidos.

3. Pulsar en Iniciar.

Registrarse

1. Ejecutar la aplicación AlertMap desde el menú de aplicaciones.

2. Pulsar en Registrarse.

3. Rellenar el formulario con todos los campos obligatorios (marcados con *).

4. Pulsar en Registrarse.

Cerrar sesión

1. Desde el menú principal, pulsar sobre el botón de la parte superior derecha. 

Compartir

1. Desde el menú principal, pulsar sobre el botón de la parte superior derecha. 

2. Opciones (Sólo Whatsapp y/o Twitter):

    a. Pulsar sobre una aplicación mostrada para compartir.
    b. Si sólo hay una aplicación para compartir se selecciona esa.

3. Enviar mensaje definido si el usuario lo desea.

Editar perfil

1. Una vez el usuario ha iniciado sesión.

2. Desde el menú principal pulsar sobre el navigation drawer de la izquierda.

3. Seleccionar la opción de Perfil.

4. Modificar los campos que se deseen.

5. Pulsar en guardar

Visualizar mi muro

1. Desde el menú principal,

    a. Pulsar sobre el acceso rápido, “Mi muro”.
    b. Pulsar sobre el navigation drawer y seleccionando la opción de “Visualizar mis alertas”

2. Si se escoge la vista de mapa, pulsando sobre una alerta podemos ver su detalle.

Visualizar mis alertas insertadas

1. Desde el menú principal pulsamos sobre el navigation drawer.
2. Seleccionar la opción “Visualizar mis alertas insertadas”.
3. Se visualiza un listado con las alertas.

Borrar alertas insertadas

1. Desde la ventana de visualización de alertas insertadas (Ver punto anterior).
2. Pulsar sobre la  de la alerta que deseamos eliminar.
3. La alerta ha sido eliminada.

Crear alerta

1. Desde la vista de mapa de “Mi muro” (Ver “Visualizar mi muro”)
2. Mantener pulsado sobre el punto que deseemos añadir una alerta
3. Introducimos el tipo de alerta.
4. Introducir no la descripción
5. Añadir o no una foto
6. Seleccionar si la deseamos subir o no.
7. Pulsar en “Notificar alerta” para publicar la alerta.

Administrar suscripción

1. Desde el menú principal,

    a. Pulsar sobre el acceso rápido, “Administrar suscripción”.
    b. Pulsar sobre el navigation drawer y seleccionando la opción de “Administrar suscripción”

2. Seleccionar el tipo de alertas a la que te deseas suscribir

3. Seleccionar la zona, global o local.

4. Pulsar en "Guardar cambios”.

Cambiar configuración

1. Desde el menú principal, pulsar en el navigation drawer, seleccionar “Configuración”.

2. Cambiar el intervalo de refresco por el valor deseado, entre 1 y 100.

3. Pulsar “Aceptar” para guardar los cambios.

11.3 Manual de buenas prácticas

11.3.1 Introducción

Este manual pretende ser una guía para el desarrollo de aplicaciones móviles en Android.

Siguiendo estas reglas o buenas prácticas conseguiremos un código más mantenible, es decir de mayor calidad.

A continuación se detallarán las diferentes técnicas para lograr el objetivo propuesto.

11.3.2 Nomenclatura de código

En esta sección se detallan las guías de estilo de codificación de aplicaciones Android.

- **Carpetas**
  - Estándar comunmente utilizado (src/main/java).
- **Paquetes**
  - No se debe de usar el guion bajo y todo debe de escribirse en minúsculas.
  - Toda clase, objeto, así como Enums deberán ser contenidas dentro del paquete correspondiente al modelo. Ejemplo: com.alertmap.model.
  - Habrá un paquete para los modelos, otro para los controladores y otro para las vistas, estas últimas deberán de ir dentro de la carpeta res, ya que es necesario ubicarlas ahí para auto-generar los identificadores de las vistas.
- **Clases**
  - Se debe de evitar el uso de clases obsoletas.
  - **Prefijo**: Todas las clases que describan una acción sobre alguna entidad específica, deberán tener como sufijo la entidad, por ejemplo: ServiceManager.

- **Sufijo**
  - Helper: Es conveniente crear una clase de utilidad (Utils.java) donde implementar métodos estáticos que sean reutilizables desde cualquier parte del código, como por ejemplo los diálogos.
  - Listener: Toda clase que reciba invocaciones ante eventos específicos será considerada como listener y será recomendable adherir el sufijo "Listener" al nombre de la clase. Por ejemplo, LocationListener.

- **Formateo de código**
  - Se utilizará la codificación UTF-8.

- **Variables y Ficheros**
  - Los ficheros relacionados con los recursos gráficos llevarán como prefijo, la especificación del mismo, por ejemplo si se trata de un layout, "layout_*.xml", una animación, "anim_*.xml" o un icono "ic_*.png".
  - Se debe de evitar el uso de variables sin inicializar.
  - Variables que son de tipo constante, deben de ser declaradas en mayúsculas y con modificador static y final. Ejemplo:

```
public static final String URL="http://www.upv.es
```

11.3.3 Código limpio

En general deberán aplicarse las siguientes reglas en cuanto a nomenclatura :

- **Idioma**: Todo código de la aplicación debe de escribirse en inglés, sin embargo los comentarios y la documentación pueden estar escritos en español si así se desea.

- **Nombres que revelen intención**: El nombre de una función, argumento o variable debe responder a las grandes preguntas: ¿por qué existe? ¿qué hace?. Por ejemplo, setBackground.

- **Evitar desinformación**: Evitar el uso de métodos que proporcionen poca información como "execute" o similares. Una solución podría ser encapsularlos dentro de métodos con nombre diferentes.

- **Nombres pronunciables**: no recortar vocales o hacer abreviaciones e intentar favorecer siempre la pronunciación. Aunque esto esté en contra de las guías de estilo de código java. La herramienta autocompletar ayuda a escribir nombre largos y es de mucha utilidad.

- **Nombres localizables**: siempre que sea posible intentar nombrar siguiendo el mismo patrón utilizado en otros componentes similares al que se está desarrollando, ejemplo: *ServiceManager, ServiceListener, etc.*

- **Evitar codificación**: por ejemplo la notación húngara o poner prefijos a las variables miembro.

- **Evitar mapeos mentales**: evitar el uso de las variables i, j o k, para recorrer colecciones de elementos y usar índices descriptivos, ejemplo "userIndex".

Funciones / Métodos

A continuación, se van a enumerar una serie de recomendaciones dirigidas principalmente a mejorar la legibilidad del código y su mantenibilidad.

- **Deben ser pequeñas**, no más de 10 líneas.

- Si incluimos **comentarios**, posiblemente es necesario dividir la función ya que puede que esté realizando varias cosas.

- **Una función solo debe de hacer una cosa.**

- **La regla Stepdown:** La idea es que el método principal o inicial sea una enumeración de métodos con las acciones que están comprendidas en la resolución de esa funcionalidad, y nada más.

```
public void savePersonalData(...) {
    checkSignature(...);
    checkValidations(...);
    invokeSavePersonalDataService (...);
    updateUsers(...);
}
```

- **Switch**: se recomienda encapsular los switch en una función específica que sólo contenga esta lógica.

- **Más de tres argumentos** síntoma de acoplamiento.

- **Evitar efectos colaterales**. Una función sólo debería modificar idealmente el resultado que retorna o los parámetros de entrada cuando son listas u objetos.

- **No repetir código, promover la reutilización.**

11.3.4 Comentarios

Los comentarios de código en muchos casos indican que el código es complejo y difícil de leer.

- **Explícate con código**, funciones con nombres descriptivos.

- **Buenos comentarios**:
  - Informativos: por ejemplo para explicar expresiones regulares.
  - Advertencias: por ejemplo para explicar las decisiones técnicas que se tomaron para utilizar una implementación sincronizad.
  - TODO: dejar indicado puntos que faltan por implementar o que hay que mejorar.
  - Jabado en Apis públicas, ya que estas pueden ser usadas por terceros y requieren saber cómo se debe de utilizar cada método y que realiza.
- **Malos comentarios**:
  - Murmuraciones: comentarios que únicamente sabe explicar quién lo escribió.
  - Redundantes: cuando explican algo evidente por el nombre de la función o variable.
  - Históricos: los comentarios que indican fecha, descripción etc. de una modificación ya que para eso tenemos el gestor de versiones de código, como Subversión o Jit.

- Ruido: por ejemplo, comentarios a los gentes y setter de atributos que no realizan ninguna tarea particular adicional.
- Marcadores: comentarios que ponemos para indicar donde terminan los IF. A veces, significa que se debe de particionar el método.
- Atribuciones: comentarios para indicar el autor de un código. Innecesario porque para eso tenemos el gestor de versiones de código.

11.4 Manual de patrones implementados

11.4.1 Introducción

En esta sección se explicarán los patrones de diseño que se han empleado para el desarrollo de esta aplicación.

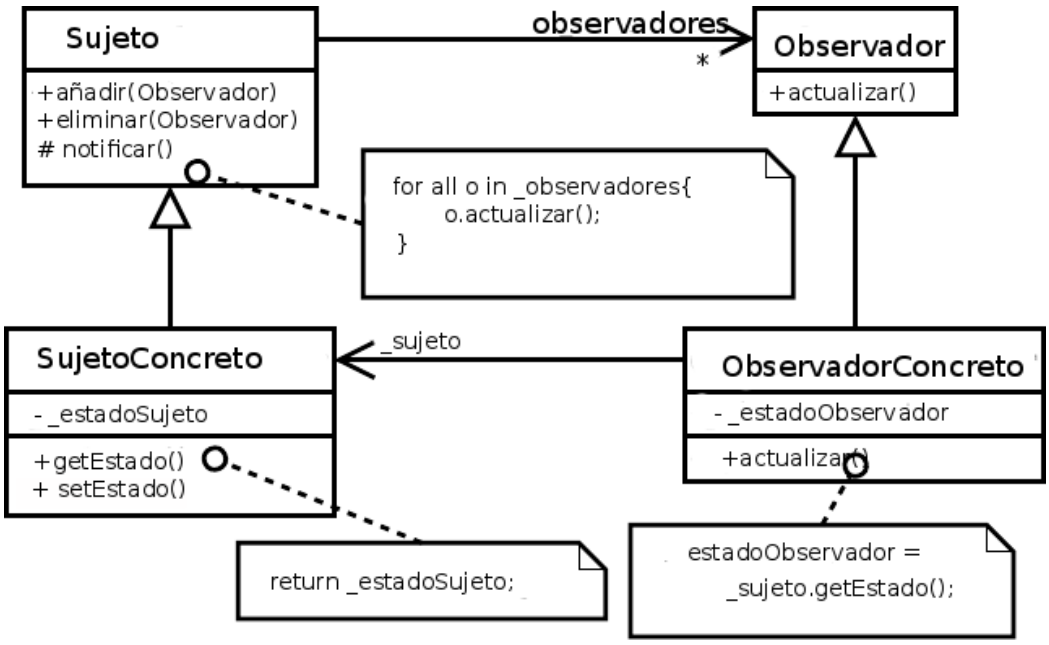
En primer lugar, analizaremos el patrón observador y en segundo lugar el patrón modelo-vista-controlador.

11.4.2 Observador

El patrón observador se basa en la notificación por parte de un objeto a otro, debido a un cambio en su estructura interna. Esta notificación se envía a todos los objetos dependientes del objeto que ha cambiado.

Cabe destacar que este patrón también es conocido como el famoso publish-suscriber. Como ya hemos dicho, se basa en el hecho de que un objeto "publica" un cambio y todos los suscritos se actualizan, tal y como ellos deseen dependiendo de su implementación interna.

Figura 28. Patrón Observador



FUENTE: http://es.wikipedia.org/wiki/Observer_(patrón_de_diseño)

La estructura, ya comentada del patrón puede resumirse visualizando la figura superior, figura 28.

Este patrón está implementado por defecto en los frameworks de interfaces gráficas orientados a objetos para capturar los eventos lanzados por la propia interfaz. Los eventos son capturados en los listeners.

En el caso de AlertMap, se ha utilizado este patrón para notificar a una interfaz que la suscripción asociada a un usuario ha sido modificada y posiblemente deba de lanzar un listener para obtener la posición actual del usuario. Este listener es lanzado cuando la suscripción cambia a zona local, de manera que el usuario recibe solo alertas cercanas a él a un radio de 10 km. Por lo tanto, es necesario refrescar la posición del usuario en todo momento.

Para la implementación del patrón observador, se ha optado por hacer uso de la implementación interna que proporciona java de este patrón.

Detalle de la implementación

Para hacer uso del patrón observador en Java, la clase que se encarga de notificar a otra, debe de extender de la clase Observable.

Figura 29. Extendiendo Observable

```
public class Usuario extends Observable implements Serializable {

    /**
```

FUENTE: Elaboración propia

Figura 30. Notificación de observadores

```
public void setSuscripcion(Suscripcion suscripcion) {
    this.suscripcion = suscripcion;
    if (initFlag) {
        setChanged();
        notifyObservers();
        return;
    }
    initFlag = true;
}
```

FUENTE: Elaboración propia

Como podemos ver en la figura 30, la clase/objeto "Usuario" sería el notificador y para notificar cuando su suscripción cambia se invoca al método notifyObservers(). Este método lo que hace es avisar a todos los observadores que la suscripción ha cambiado.

Todos los objetos que desean ser observadores en primer lugar deben de implementar la interfaz de Observer y como segundo requisito deben de registrarse/suscribirse a un notificador.

Figura 31. Implementando Observer

```
public class MenuPrincipal extends ActionBarActivity implements Observer {

    user.addObserver(this);
```

FUENTE: Elaboración propia

Para suscribirse a un evento, notificador, el objeto usuario debe de añadir el observador.

Por último, el observador al implementar la interfaz de Observer, debe de implementar el método update.

Figura 32. Implementando observador

```
@Override
public void update(Observable observable, Object data) {
    user.removeAlertasVisualizadas();
    if (observable instanceof Usuario) {
        Usuario changeUser = (Usuario) observable;
        goToService(changeUser);

    }

}
```

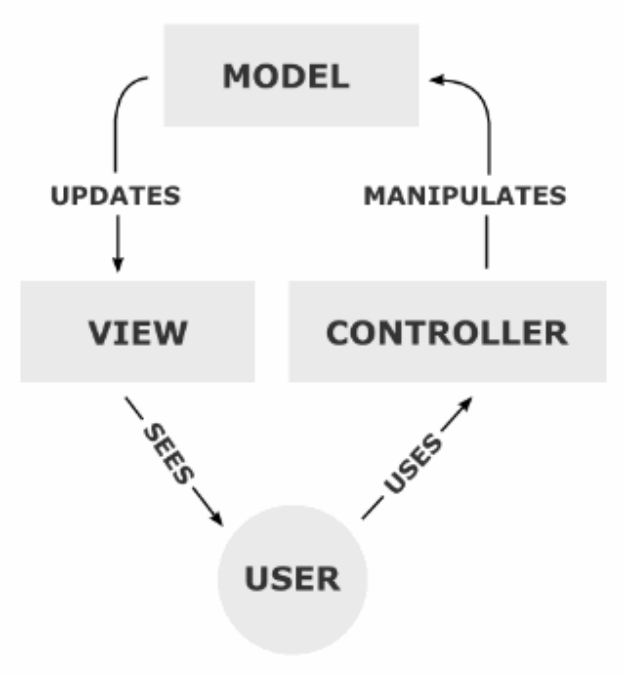
FUENTE: Elaboración propia

En este método, se recibe el objeto que ha cambiado para que el observador pueda llevar a cabo los cambios oportunos.

11.4.3 Patrón modelo-vista-controlador.

El patrón modelo-vista-controlador(MVC), se trata de un patrón muy utilizado durante el desarrollo software y muy útil para organizar nuestra aplicación.

Figura 33. Patrón Modelo-Vista-Controlador



FUENTE: Elaboración propia

Como se puede ver en la figura 33, el patrón MVC se divide principalmente en tres componentes claramente diferenciadas y las diferentes interacciones que suceden entre componentes.

En primer lugar, nos encontramos con el modelo que representa a la información que va a ser representada en la vista, es decir envía a la vista los elementos necesarios para representar la información que se desea mostrar al usuario. Cuando el modelo es actualizado, la vista cambia, es decir se actualiza.

Como segunda componente, tenemos la vista. En la vista se representa la información mantenida en el modelo en un formato correcto. En la mayoría de casos, se trata de la interfaz de usuario. Como es obvio, el usuario verá la vista (interfaz de usuario).

Por último, tenemos la componente del controlador. El controlador responde a eventos que lleva a cabo el usuario, como puede ser un click sobre un botón o similar. Además, se encarga de manipular el modelo, es decir modificarlo, para que la vista cambie.

Aunque hemos dicho que el usuario lo que ve es la vista, cabe destacar que toda operación que el usuario realiza contra la interfaz supone que el usuario esté utilizando el controlador asociado a la vista.

Como ya hemos mencionado durante el documento, nuestra aplicación, está dividida en estas tres componentes, modelo, vista y controlador.

1. El paquete “model” situado dentro de la carpeta src representa al modelo del sistema.
2. Los “layouts”, interfaces de usuario, son las vistas y se encuentran dentro de la carpeta res.
3. Los “controllers” son los controladores de las vistas y permiten al usuario interactuar con la aplicación tal y como hemos comentado en párrafos anteriores.

Por último, cabe resaltar que en nuestra aplicación el patrón se ha implementado de la siguiente manera:

- Toda vista tiene un controlador asociado, cada controlador asociado se trata de un Activity.

- Todo aquello que modificamos en el modelo, como es el caso cuando en la aplicación se eliminan alertas, la vista se actualiza.

12. Bibliografía

Cursos de desarrollo en Android

- Google. Android Developers [Online]. Disponible: http://developer.android.com/index.html
- J. Tomás, V. Carbonell, C. Vogt, M.G. Pineda, J.B. Mascarell, D. Ferri, "El gran libro de Android avanzado" , 1 ed. Barcelona, España
- S.G. Oliver. sgoliver.net.blog [Online]. Disponible: http://www.sgoliver.net/blog/?page_id=2935
- Vogella. Android Development Starter Tutorials [Online]. Disponible: http://www.vogella.com/tutorials/android.html
- Soluciones informáticas para dispositivos móviles. UPV. 2013-2014.
- Codigofacilito. Youtube [Online]. Disponible: https://www.youtube.com/watch?v=sS3oDIcHNFo&list=PLAB8000C00E2814CB

API Google Maps Android

- Google. Android Developers, API de Google Maps Android [Online]. Disponible: https://developers.google.com/maps/documentation/android/
- Android Curso [Online]. Disponible: http://www.androidcurso.com/index.php/tutoriales-android/41-unidad-7-seguridad-y-posicionamiento/223-google-maps-api-v2
- R. Tamada. Android Hive, Android working with Google Maps [Online]. http://www.androidhive.info/2013/08/android-working-with-google-maps-v2/
- S.G. Oliver. sgoliver.net.blog, Mapas en Android [Online]. Disponible: http://www.sgoliver.net/blog/?p=3271

Problemas de desarrollo software

- Lockwood. Fragment Transactions & Activity State Loss [Online]. Android Design Patterns. Rep. Aug 20, 2013. Disponible: http://www.androiddesignpatterns.com/2013/08/fragment-transaction-commit-state-loss.html
- Stackoverflow [Online] . Disponible: http://stackoverflow.com/
- Google Groups. Android Developers [Online]. Disponible: https://groups.google.com/forum/#!forum/Android-developers
- Stackoverflow, Listenner Snippet. Disponible: http://stackoverflow.com/questions/14478179/background-service-withlocation-listener-in-android

Herramientas software

- Balsamiq. Balsamiq Mockups [Online]. Disponible: http://webdemo.balsamiq.com/
- ArgoUML. Modelado UML (MacOSX)
- Inghenia swot. ISOKEY [Online] http://www.inghenia.com/gadgets/swot/swot.php

Recursos gráficos

- IconArchive [Online]. Disponible: http://www.iconarchive.com/
- FlatIcon [Online]. Disponible: http://www.flaticon.com/

Otros

- Wikipedia. Twitter [Online]. Disponible: http://es.wikipedia.org/wiki/Twitter
- Wikipedia. Pinterest [Online]. Disponible: http://es.wikipedia.org/wiki/Pinterest
- Googleplay.Waze[Online].Disponible: https://play.google.com/store/apps/details?id=com.waze&hl=es
- EcuRed. Arquitectura de tres niveles [Online]. Disponible: http://www.ecured.cu/index.php/Arquitectura_de_tres_niveles
- Fani Calle. Arquitectura 3 capas [Online]. Disponible: http://es.slideshare.net/Decimo/arquitectura-3-capas
- Wikipedia. Programación por capas [Online]. Disponible: http://es.wikipedia.org/wiki/Programaci%C3%B3n_por_capas