UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Proyecto Final de Grado

Grado en Ingeniería Informática

Arquitectura para la gestión de notificaciones con dependencias temporales y geográficas

Escuela Técnica Superior de Ingeniería Informática

Universidad Politécnica de Valencia

Autor: María Pilar García Marchante

Profesor Responsable: Carlos Tavares Calafate

Curso 2014 / 2015

Resumen

Este Trabajo Fin de Grado trata del desarrollo, diseño e implementación, tanto del lado del servidor como de lado del cliente, de un sistema gestor de notificaciones con dependencias temporales y geográficas.

El sistema permite gestionar la entrega de notificaciones en base a la ubicación de los diferentes clientes.

Para ello, el sistema ofrece una interfaz al Administrador del Sistema para la gestión de notificaciones, y una interfaz a los terminales Android para la reproducción de las notificaciones recibidas. El Servidor recibirá las ubicaciones de los clientes y hará la entrega de las notificaciones correspondientes en base a las restricciones geográficas y temporales definidas, y llevando un seguimiento de las notificaciones entregadas a cada cliente para evitar duplicados.

Palabras Clave

Gestor de notificaciones; Android; servidor; dispositivos móviles; avisos basados en ubicación.

Abstract

This Project addresses the development, design and implementation, on both server and client sides, of a notification management system supporting temporal and geographical dependencies.

The system allows managing the delivery of notifications based on the location of the different clients.

With this purpose, the system offers an interface to the System Administrator to manage notifications, and an interface for Android terminals Android for reproducing the received notifications. The Server will receive the client locations and deliver the corresponding notifications based on the geographical and temporary restrictions defined, keeping track of the notifications delivered to each of the clients to avoid duplicates.

Keywords

Notification manager; Android; server; mobile devices; location-based notifications.

Índice de contenidos

1. Introducción

El tráfico, en la mayoría de ciudades, es un problema que perturba a los conductores que necesitan desplazarse para realizar sus actividades cotidianas. En muchas ocasiones, el tráfico aumenta el nivel de estrés a los ciudadanos, perjudicando a su salud física e incluso cambiando sus hábitos de vida. Ante esta situación, y teniendo en cuenta que en España hay un censo de 26.401.660 conductores (según el INE a fecha 31/12/2013), es más que viable estudiar distintas formas para prevenir los atascos en las carreteras españolas.

Por este motivo, se ha decidido implementar un sistema de notificaciones que ayude al conductor a conocer las incidencias que existen en la carretera. Para ello, se ha empleado la tecnología Android, ya que se ofrece como software libre y facilita la integración en distintos tipos de hardware. Además, según el *Instituto Nacional de Estadística [13*], el 67,2% de hogares españoles disponen de móviles de última generación. Considerando estas cifras, y que *Google Play* cuenta con más de 1,43 millones de aplicaciones en el mercado, la idea de crear una aplicación para Android cobra importancia.

1.1. Objetivo del proyecto

En este Trabajo Fin de Grado, se explica el diseño y la implementación, tanto del lado del servidor como de lado del cliente, de un sistema gestor de notificaciones con dependencias temporales y geográficas.

Para ello se ha desarrollado un servicio donde un servidor central gestiona la entrega de notificaciones en base a la ubicación de los diferentes clientes.

El sistema de notificaciones debe de ofrecer una interfaz al Administrador del sistema para gestionar las notificaciones. Además, éstas notificaciones deben de poder almacenarse en el servidor para posteriores envíos a los conductores.

Los terminales Android de los conductores deben de ser capaces de enviar al Servidor su posición actual. El Servidor recibirá las ubicaciones de los distintos conductores y hará la entrega de las notificaciones correspondientes en base a las restricciones geográficas y temporales definidas.

1.2. Estructura de la memoria

Esta memoria está estructurada en diez apartados. Cada apartado contiene información acerca de cada uno de los aspectos concretos que componen el proyecto. A continuación, se describe de forma breve de qué trata cada uno de estos apartados.

Introducción: el objetivo de este apartado es contextualizar el proyecto desarrollado, exponiendo la idea general y los objetivos que se persiguen.

Trabajos relacionados: en este apartado se verán diversas aplicaciones existentes en la actualidad y con similitud al proyecto desarrollado.

Aspectos técnicos: en este apartado se proporcionará información relativa a aquellos elementos tecnológicos involucrados en el proyecto.

Análisis del proyecto: en este apartado se van a estudiar los servicios que nuestro sistema de gestión de notificaciones debe de proporcionar, las restricciones sobre las que se va a construir y funcionar, y las formas en las que el sistema se va a usar para cumplir su propósito.

Diseño del sistema: en este apartado se describe con precisión la arquitectura del sistema y los distintos elementos que la componen.

Implementación del sistema: en este apartado se muestran los detalles de implementación de los principales elementos que componen el sistema.

Validaciones y pruebas: en este apartado se van a realizar las validaciones y las pruebas pertinentes, para comprobar que el Sistema funciona correctamente, y para estimar el rendimiento y la escalabilidad del Sistema.

Conclusiones: en este apartado se realiza una comparativa de los resultados obtenidos frente a los objetivos marcados al comienzo.

Ampliaciones: en este apartado se establecen futuras líneas de trabajo.

Referencias bibliográficas: en este último punto se enumeran los recursos bibliográficos consultados para la elaboración del proyecto.

2. Trabajos relacionados

En este punto se verán diversas aplicaciones existentes en la actualidad y con similitud al proyecto desarrollado. El estudio de otras aplicaciones con cierta similitud es útil en el desarrollo del proyecto ya que, gracias a ello, se pueden obtener ideas de interés y adaptar el sistema a los objetivos definidos.

2.1. WAZE

Waze [9] es una aplicación de navegación GPS gratuita que proporciona información del tránsito, además de varias funciones sociales y de geo-gaming. Los *Wazers* se pueden informar unos a otros sobre el tránsito, controles policiales, obras, radares y más. Al ser información generada por los propios usuarios, cuantos más lo usen, mejor.

Los conductores diarios cuentan con más de una ruta a sus destinos, por lo que Waze recolectará información sobre las rutas y aprenderá cuales son las rutas más convenientes.

Después de insertar la dirección del destino, los usuarios conducen con la aplicación abierta para contribuir pasivamente con información del tránsito. También pueden tomar un rol más activo al compartir alertas de incidentes en el tránsito como accidentes, controles policiales, o cualquier peligro que encuentren en la vía. Esto ayuda a otros usuarios que están en la zona porque reciben un aviso de lo que les espera más adelante en su ruta. Además de las comunidades locales que usan la aplicación, Waze también es parte de una comunidad muy activa de editores de mapa que aseguran que los mapas de Waze sean lo más actualizados posible.

2.2. RadarDroid

RadarDroid [10] es una aplicación para teléfonos Android, que alerta al usuario cuando se acerca a un radar o un control móvil. También se pueden activar preavisos antes de entrar en túneles donde hay radares instalados o cámaras en semáforos. Dispone de las siguientes funciones:

- Avisos por voz indicando el tipo de radar y el límite de la vía (personalizables por el usuario).
- Avisos visuales y sonoros.
- Avisos con vibrador (muy útil para motoristas).
- La pantalla se puede configurar para mantenerla siempre activa o apagarse en caso de inactividad. Si está apagada, se enciende automáticamente al recibir un aviso de radar.
- Modo diurno y nocturno.
- Selecciona pantalla en modo horizontal, vertical o automático.
- La pantalla principal permite ver los radares en el mapa y comprobar tu posición en todo momento, junto con tu velocidad y dirección.
- Descarga de la Base de Datos de radares más completa y actualizada de España desde Internet.
- Importación de la Base de Datos desde la tarjeta SD.

2.3. COYOTE

COYOTE es un avisador de alertas de tráfico y radares. [11]

Gracias a su tecnología y a su comunidad, COYOTE es capaz de ofrecer, en tiempo real, la situación de todos los radares fijos, de semáforo, radares móviles y radares de tramo**,** además de avisar del resto de incidencias del tráfico**:** Atascos, accidentes, emergencias, etc.

Para dar el mejor servicio a los usuarios, COYOTE actualiza diariamente la información sobre radares fijos y radares de semáforo, además de las actualizaciones en tiempo real de radares móviles, radares de tramo, nuevos radares y cambios de ubicación de los radares. Tanto los dispositivos COYOTE como la app iCOYOTE, se actualizan cada 3 minutos para que las informaciones reportadas por la comunidad lleguen en tiempo real a los demás conductores.

Dispone de las siguientes funcionalidades:

- Conocer cuántos conductores que usan el servicio COYOTE tienes delante de ti.
- Conocer el límite de velocidad en tiempo real del tramo de vía por el que circula.
- Conocer la situación de los radares fijos, de semáforo, de tramo y los radares móviles que notifique la comunidad COYOTE.
- Recibir todas las alertas e incidencias que reporte la comunidad: accidentes, atascos, emergencias, etc.

2.4. Dirección General de Tráfico

La aplicación para móviles, desarrollada por la Dirección General de Tráfico [17] constituye una herramienta útil y sencilla de ayuda al conductor. Tiene tres funcionalidades:

- Guardar las rutas más utilizadas para conocer las incidencias que existan en la carretera, como retenciones, obras, datos meteorológicos, eventos, etc. Indicar la mejor ruta hacía el punto destino, indicando la distancia y el tiempo de llegada

estimado. También permite configurar un aviso a los contactos seleccionados cuando el usuario llega al destino marcado.

- Acceder a las cámaras oficiales de tráfico, localización de radares y a la información de tráfico.
- Acceso directo al teléfono de emergencias y aviso sonoro de incidencias graves.

2.5. AppValencia

AppValencia [18] es una aplicación para dispositivos móviles desarrollada por el Ayuntamiento de Valencia que permite mantener un canal de comunicación con la administración municipal desde el que obtener información o acceder a numerosos servicios.

Dispone de nueve funcionalidades:

- **Mensajería**: permite una comunicación directa del Ayuntamiento hacia el ciudadano. AppValencia dispone de una serie de canales de información a los que el usuario puede suscribirse, como el de Tráfico a través del cual se reciben avisos de cortes de calle, tráfico denso, etc..
- **Información Municipal**: acceso a una serie de servicios de información de la web municipal como son: Noticias, Agenda de la Ciudad, Perfil de Contratante y Campañas municipales.
- **Mapas Municipales**: acceso a una serie de mapas interactivos, que además de información sobre los medios de transporte público EMT, Valenbisi, Metro, Taxis…., incluyen un callejero municipal, el estado del tráfico en tiempo real, una localización de los monumentos falleros e información sobre ellos, y una representación de la intensidad de tráfico en tiempo real.

- **Realidad Aumentada:** Integración de forma nativa de los servicios de realidad aumentada como equipamientos municipales, transporte público, fallas, edificios y lugares de interés.
- **Trámites y Gestiones:** si el usuario tiene instalado un certificado digital de la ACCV podrá acceder, en modo consulta, a la sede electrónica del Ayuntamiento de Valencia.
- **Servicios cercanos:** el apartado "*¿Qué hay cerca de mi?*" Permite consultar equipamientos cercanos a nuestra posición en un radio de 300 metros.
- **Notificaciones GPS. Aproximación vías de tráfico denso:** AppValencia puede detectar gracias al GPS y a la información que dispone el Ayuntamiento de Valencia sobre la densidad de tráfico en tiempo real, si el ciudadano se está aproximando a vías con problemas de densidad de tráfico o vías cortadas, enviando notificaciones.
- **Incidencias en Vía Pública**: el usuario podrá comunicar al Ayuntamiento incidencias en la vía pública (baldosas en mal estado, contenedores rotos, etc.), incluyendo fotografías georreferenciadas para que éste proceda a su reparación.
- **Lanzadera de aplicaciones publicadas por el Ayuntamiento**: AppValencia es un gestor de todas las aplicaciones móviles desarrolladas por y para el Ayuntamiento de Valencia. Este gestor informará al ciudadano de todas las aplicaciones móviles desarrolladas, indicando si la tiene instalada en el dispositivo, y en caso contrario dándole la oportunidad de instalarla.

2.6. Conclusiones del estudio de trabajos relacionados

Después de ver la gran acogida que han tenido las distintas aplicaciones pensadas en mantener al conductor informado en todo momento de lo que sucede a su alrededor, cobra importancia la idea de desarrollar un gestor de notificaciones genérico que permita a la administración enviar avisos a los conductores en distintas situaciones.

A diferencia de algunos de los ejemplos citados, la aplicación que se va a desarrollar en este proyecto no está pensada en que cualquier usuario dé de alta notificaciones, sino que sea un administrador central, con acceso a la interfaz de administrador, el que pueda crear las notificaciones. De esta manera se limita la creación de notificaciones con el objetivo de tener un mayor control de las notificaciones que van a llegar al usuario.

Por otro lado, se observa la necesidad de desarrollar una aplicación que no requiera de intervención manual del usuario, ya que está pensada para conductores, es decir, que la notificación pueda reproducirse automáticamente.

3. Aspectos técnicos

A continuación se va a proceder a detallar los distintos aspectos técnicos tenidos en cuenta durante el desarrollo del proyecto.

3.1. Análisis de requisitos a nivel de Software

3.1.1. Servidor

Para el desarrollo de la parte del Servidor se han empleado las tecnologías PHP, MySQL y APACHE.

PHP

PHP (*Hypertext Pre-processor*) es un lenguaje de programación que se emplea en la creación de sitios web. En este proyecto se ha empleado para crear el portal Web al que accede el administrador del gestor de notificaciones.

PHP es un lenguaje potente y robusto, embebido en documentos HTML, y que proporciona múltiples protocolos de comunicación en Internet, entre ellos el protocolo HTTP, que, como veremos más adelante, es el protocolo empleado para establecer la comunicación entre Cliente y Servidor. Es gratuito y su código fuente es abierto. Además, dispone de extensiones para la conexión con la gran mayoría de los sistemas de gestión de bases de datos (SGBD) para el almacenamiento de información permanente en el servidor, entre ellos: MySQL.

Por todo ello se ha decidido emplear esta tecnología en el desarrollo de la parte web del Servidor.

MySQL

MySQL es el sistema gestor de nuestra base de datos. Las bases de datos permiten almacenar, buscar, ordenar y recuperar datos de forma eficiente. El servidor de MySQL controla el acceso de los clientes a los datos y garantiza el uso simultáneo de varios usuarios. Utiliza SQL (*Structured Query Language*), que es el lenguaje estándar para la consulta de bases de datos. Al igual que PHP, MySQL se distribuye bajo una licencia de código abierto, aunque también existen licencias comerciales.

APACHE

Apache actúa como Servidor Web. También Apache es un proyecto de código abierto y uso gratuito.

Apache es el encargado de gestionar la entrega de las notificaciones a los clientes, y su labor principal es analizar y gestionar las peticiones recibidas por los clientes.

3.1.2. Cliente

Para el desarrollo de la parte del Cliente se ha utilizado la tecnología Android.

ANDROID

Android es un paquete de software basado en código abierto para teléfonos móviles, creado por Google y la Open Handset Alliance.

Android dispone de muchos servicios integrados, como por ejemplo el servicio de localización, que ha sido empleado en este proyecto para que el cliente mande la ubicación al servidor.

A continuación se procede a detallar los cuatro elementos básicos involucrados en la implementación de la aplicación Android de la parte del Cliente.

- Java, es el lenguaje de programación usado para escribir el código que genera la aplicación Android.
- Eclipse, es el IDE *(Integrated Development Environment*) empleado.
- Android SDK (*Software Development Kit*), ha sido necesario añadirlo a Java, y lo proporciona Google. Entre otras cosas, me ha permitido ejecutar un emulador del sistema Android en mi computadora.
- ADT, es el Pluging que permite integrar Android con Eclipse.

En este punto cabe destacar que Android dispone de distintas versiones, y hay que elegir la versión del sistema para la que deseamos realizar la aplicación. Existen clases y métodos que están disponibles a partir de una versión concreta, por ello es importante elegir la versión mínima necesaria. En este proyecto se ha utilizado la versión: Android 2.3 Nivel de API 9 de diciembre 2010, ya que facilita el seguimiento de la posición actual de un dispositivo móvil

entre otras muchas funciones. Para descargar las distintas versiones de Android se emplea *Android SDK Manager*.

Sitios Web para la descarga del software utilizado (último acceso en octubre de 2014):

- Instalación de MySQL: http://dev.mysql.com/downloads/
- Instalación de Apache: http://httpd.apache.org/download.cgi
- Instalación de JAVA: https://www.java.com/es/download/
- Instalación de Eclipse: https://eclipse.org/
- Instalación de Android SDK: https://developer.android.com/sdk/index.html
- Instalación de ADT Plugin: https://dl-ssl.google.com/android/eclipse/

3.2. Análisis de requisitos a nivel de Hardware

Los requisitos mínimos recomendados para un Servidor Web de un único ordenador son 2 GB de memoria RAM y 500 MB de almacenaje. También es necesario un Router con conexión a Internet. La conexión a Internet, junto con la memoria RAM utilizada, son los elementos que limitarán la carga de usuarios que pueden acceder simultáneamente a nuestro servidor.

El espacio de almacenamiento nos limitará el numero de notificaciones que podemos almacenar, pero no es una limitación preocupante en este proyecto, ya que las notificaciones no ocupan mucho espacio de disco para ser entregadas al cliente.

Por parte del cliente, el requisito imprescindible para poder ejecutar la aplicación desarrollada es un terminal móvil con Android 2.3 instalado, conexión a Internet, y el servicio de localización del terminal disponible y activado.

4. Análisis del proyecto

En este apartado se va a estudiar los servicios que nuestro sistema de gestión de notificaciones debe de proporcionar, las restricciones sobre las que se va a construir y funcionar, y las formas en las que el sistema se va a usar para cumplir su propósito [1].

4.1. Análisis de requerimientos del sistema

Para empezar se van a definir los requerimientos funcionales abstractos. En este punto se estudiarán las funciones básicas que el sistema debe proporcionar en un nivel abstracto.

- *Requerimientos funcionales abstractos*. Por un lado el gestor de notificaciones debe de proporcionar al administrador del sistema una interfaz desde la cual poder dar de alta diversos tipos de notificaciones. Para cada notificación el administrador seleccionará las restricciones temporales y geográficas deseadas. De ese modo podrá crear notificaciones destinadas a diversos objetivos. Por ejemplo, el administrador desea notificar a aquellos usuarios que se aproximen al centro de la ciudad, donde hay programada una actividad, que el tráfico permanecerá cortado, avisando así a los conductores que se aproximen a la ubicación señalada, para que puedan rectificar su trayecto y tomar un desvío a tiempo para llegar a su destino.

  Por otro lado el sistema permitirá al usuario tener la aplicación ejecutándose en su dispositivo móvil en segundo plano, y alertará al usuario con las notificaciones adecuadas según su ubicación. Como el sistema está pensado en dar esta funcionalidad a conductores que no pueden manipular su terminal móvil mientras conducen, ha de ser capaz de reproducir las notificaciones sin la intervención del usuario.

Otro punto importante a tener en cuenta en la definición de requerimientos es establecer un conjunto completo de objetivos que el sistema debe de cumplir.

- *Objetivos del sistema.* El sistema debe de ser capaz de cumplir explícitamente los siguientes puntos:
  - La interfaz a la que el administrador del sistema accede para gestionar las notificaciones debe de ser accesible desde cualquier terminal con acceso a Internet.
  - Las notificaciones deben de permanecer almacenadas en disco y estar accesibles por lo menos durante el periodo de tiempo que marque su restricción temporal.

- El sistema debe de ser capaz de conocer qué notificación ha sido entregada a qué cliente, para no entregar notificaciones duplicadas.
- El terminal Android debe de ser capaz de enviar al Servidor la ubicación en la que se encuentra.
- Cuando el Servidor recoge una ubicación de un cliente debe de ser capaz de comprobar si existe una notificación destinada a la ubicación recibida.
- El sistema debe de comprobar que se cumplan las restricciones temporales marcadas para cada notificación antes de su envío.
- El terminal Android debe de reproducir la notificación recibida sin necesidad de intervención manual del usuario.

4.2. Descripción de la parte Servidor

Una vez tenemos una visión global del servicio que va a prestar el sistema gestor de notificaciones, se procede a especificar con más detalle el papel que juega el Servidor.

Para ofrecer el servicio web deseado y atender a las peticiones de los clientes, el Servidor Web se ha montado sobre Apache. Para almacenar las notificaciones en disco de manera persistente se ha utilizado MySQL, y para desarrollar la plataforma web se ha empleado PHP. Con todo ello conseguimos cumplir los objetivos descritos en el punto anterior. Posteriormente, en el apartado *Implementación del Sistema,* profundizaremos en cómo se ha desarrollado el Servidor, pero ahora se procede a detallar la funcionalidad del Servidor Web de un modo descriptivo.

El Administrador de la aplicación accede al portal web mediante un navegador y se identifica. Existe un registro de administradores autorizados para utilizar la aplicación, debido a que es un portal que va a estar accesible desde Internet y podría verse dañado si el acceso fuera libre. Veremos más adelante, en sucesivos puntos, como se gestiona el acceso identificado al portal web. Como administrador de la aplicación, entendemos a aquél que desea emplear el gestor de notificaciones desarrollado y crear las notificaciones que el cliente recibirá en su terminal móvil.

Una vez identificado, el administrador puede crear notificaciones de audio, de video, con imágenes o de texto, y seleccionar en un mapa una o más ubicaciones de destino. También seleccionará la fecha y hora de inicio, y la fecha y hora de fin, así como fechas que restringen la durabilidad temporal de las notificaciones.

Una vez todo esté definido se creará la notificación, que se almacenará en la Base de Datos.

Por otro lado el Servidor recibe las ubicaciones de los clientes y, cada vez que recibe una ubicación, debe de comprobar si en la Base de Datos existe una notificación que cumpla las restricciones geográficas y temporales.

En caso de que exista una notificación que cumpla las restricciones debe de comprobar si el cliente ya ha recibido la notificación.

Tras efectuar las tres comprobaciones, si el Servidor encuentra una notificación válida debe enviarla al cliente.

4.3. Descripción de la parte Cliente

El cliente se instala la aplicación en un terminal Android. Una vez que el cliente se sube a su vehículo deberá ejecutar la aplicación.

Conforme el vehículo va desplazándose y la ubicación cambia, la aplicación Android manda la ubicación actualizada al Servidor. Debe de existir un control de cada cuánto tiempo se manda la ubicación al servidor para que, por una parte, no se colapse el sistema ni se gaste mucha batería del terminal Android, y, por otro lado, no se pierdan notificaciones destinadas a un punto geográfico determinado. Este punto lo veremos con más detalle en el apartado *Implementación del Sistema*.

El sistema del cliente está preparado para recibir las notificaciones que el Servidor le mande y para reproducirlas automáticamente. Reproducirá tanto vídeo como audio o imágenes y, en el caso de que le llegue una notificación de texto el sistema, Android será capaz de reproducirla en modo *Audio*, interpretando el texto, para que el conductor del vehículo no tenga que retirar la mirada de la carretera.

4.4. Descripción del protocolo de comunicación entre Cliente y Servidor

En este punto vamos a detallar la parte de la comunicación entre el Cliente y el Servidor. Se trata de un punto importante ya que, si no existiera esta comunicación, el sistema no tendría sentido.

El protocolo empleado es el *protocolo HTTP [3]*. HTTP es el protocolo de la capa de aplicación de la Web para transferencia de hipertexto (*HyperText Transfer Protocol*) que está definido en los documentos [RFG 1945] y [RFC 2616]. HTTP se implementa en dos programas, un programa cliente y otro programa servidor, que se ejecutan en sistemas terminales diferentes y se comunican entre sí intercambiando mensajes HTTP. El protocolo HTTP es el encargado de definir la estructura de estos mensajes, y de cómo el cliente y el servidor intercambian los mensajes. HTTP utiliza TCP como protocolo de transporte subyacente

El terminal móvil Android será el cliente que solicitará al Servidor las notificaciones. Para ello, en la solicitud, el terminal Android deberá de indicarle su ubicación y su identificador, para que el servidor pueda realizar las comprobaciones pertinentes y saber si tiene en su sistema alguna notificación que cumpla las dependencias temporales y geográficas, además de averiguar si no ha recibido la notificación en cuestión. Si no la ha recibido, en su respuesta le mandará la notificación, o en caso contrario le indicará que no tiene ninguna notificación para el cliente en ese momento.

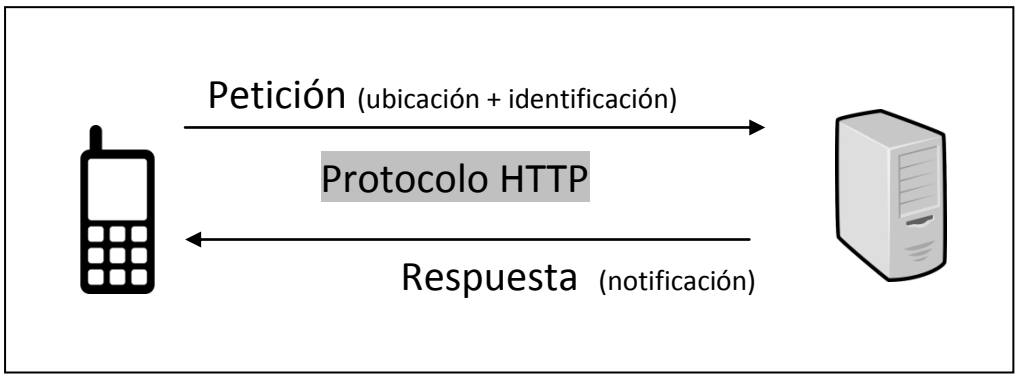

Figura 1. Gráfico descriptivo de una petición HTTP.

5. Diseño del sistema

Para garantizar que el sistema cumple con los objetivos marcados es necesario construir un modelo. Con este modelo podremos visualizar y controlar la arquitectura del sistema, y se podrá comprender mejor el sistema que se está desarrollando.

Según Booch et al. [4], un modelo es una simplificación de la realidad que nos proporciona una visión global del sistema. Para construir este modelo se va a emplear el Lenguaje Unificado de Modelado (*UML*). UML es un lenguaje gráfico para visualizar, especificar, construir y documentar el sistema desarrollado.

5.1. Diagrama de Casos de Uso

Los diagramas de caso de uso son uno de los tipos de diagramas de UML, y en concreto el que se va a emplear para el estudio del diseño del sistema.

Con UML, el diagrama de casos de uso se emplea para visualizar el comportamiento del sistema. Es un diagrama que muestra un conjunto de casos de uso, actores y sus relaciones.

Un actor representa un rol que es jugado por una persona o un dispositivo.

En la *Figura 2* se muestra el contexto del sistema de gestión de notificaciones, especificando los límites del sistema y destacando a los actores. Se observan dos actores, con dos roles diferenciados. Por una parte el Administrador del sistema, encargado de acceder a la interfaz Web y dar de alta las notificaciones pertinentes. Y por otro lado el Cliente, quien ejecuta una aplicación Android y es quien envía su ubicación al Servidor y a continuación recibe y reproduce las notificaciones.

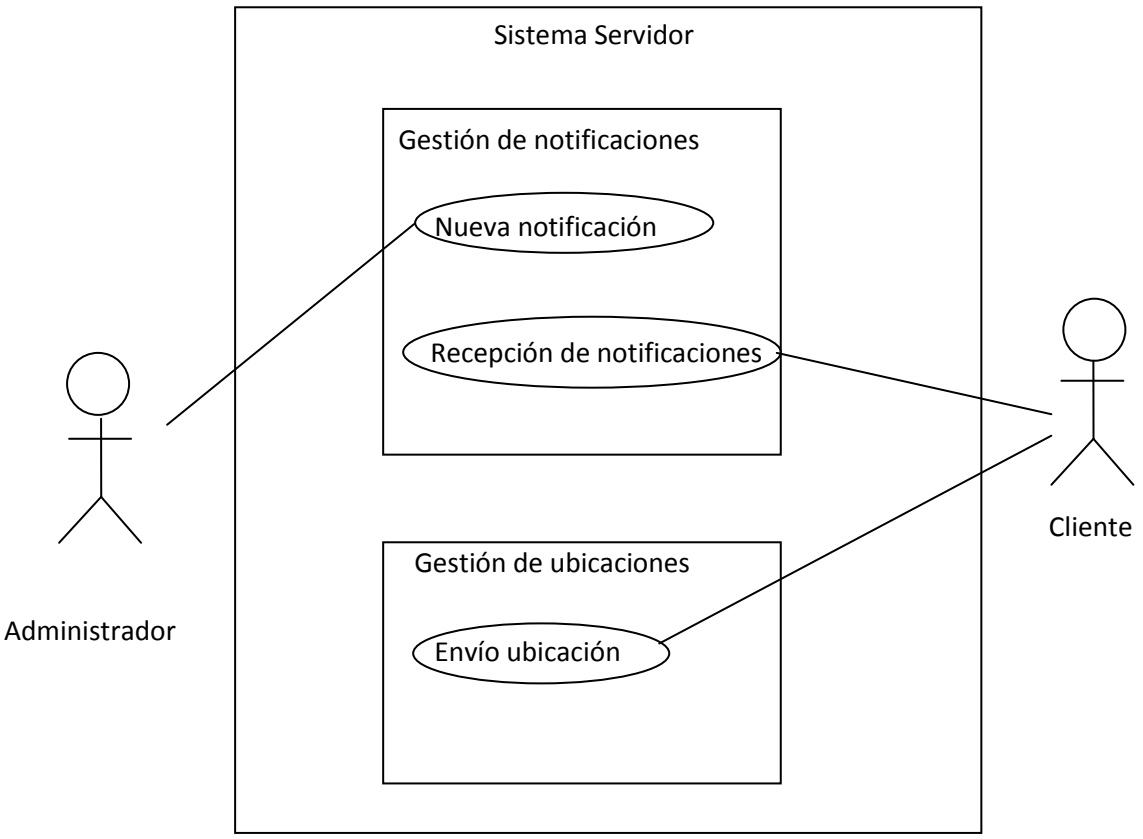

Figura 2. Modelado del contexto del sistema.

En este gráfico existen dos subsistemas: gestión de notificaciones y gestión de ubicaciones, que a su vez puede contener otras agrupaciones y casos de uso. La descomposición jerárquica termina cuando en un nivel tenemos únicamente actores y casos de uso. Se procede a descomponer cada una de dichas agrupaciones.

En la *Figura 3* y el la *Figura 4* se muestra los actores y casos de uso asociados a la gestión de notificaciones y a la gestión de ubicaciones respectivamente.

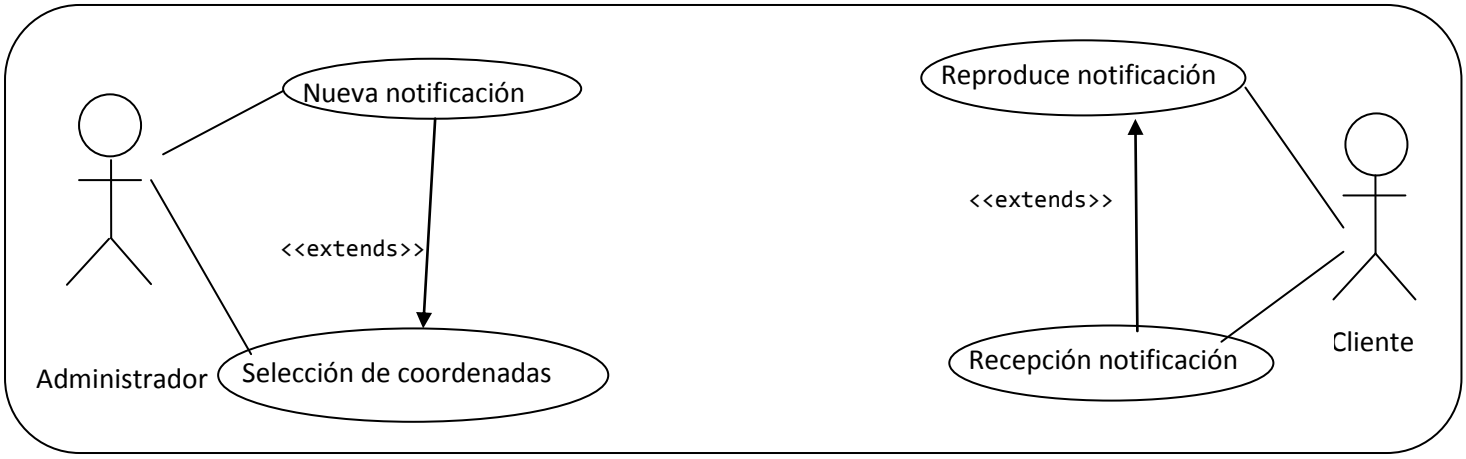

Figura 3. Modelado del subsistema gestión de notificaciones.

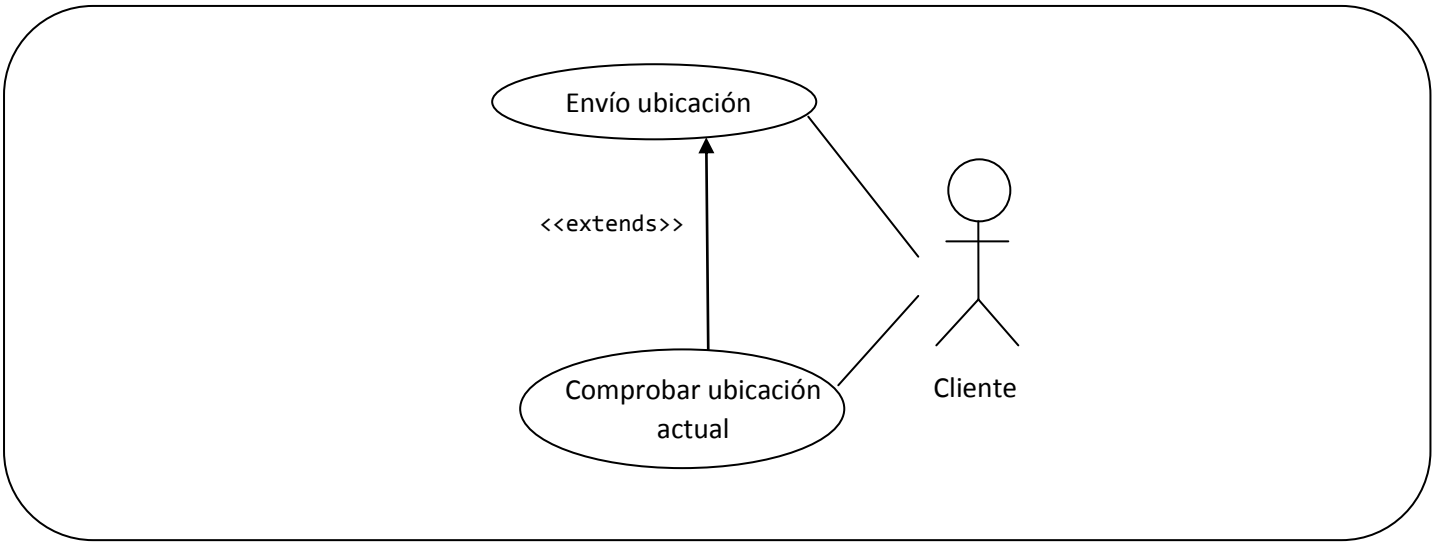

Figura 4. Modelado del subsistema gestión de ubicaciones.

<<extends>>: Un caso de uso B extiende a un caso de uso A, si en la descripción de A figura una condición cuyo cumplimiento origina la ejecución de todos los eventos que aparecen descritos en B. [5]

En el contexto del sistema visto, se observa la forma en la que interactúan los actores con el Sistema. Por otro lado, el Sistema de Gestión de Notificaciones, realiza las actividades que podemos observar en el diagrama de flujo que se muestra en la *Figura 5*.

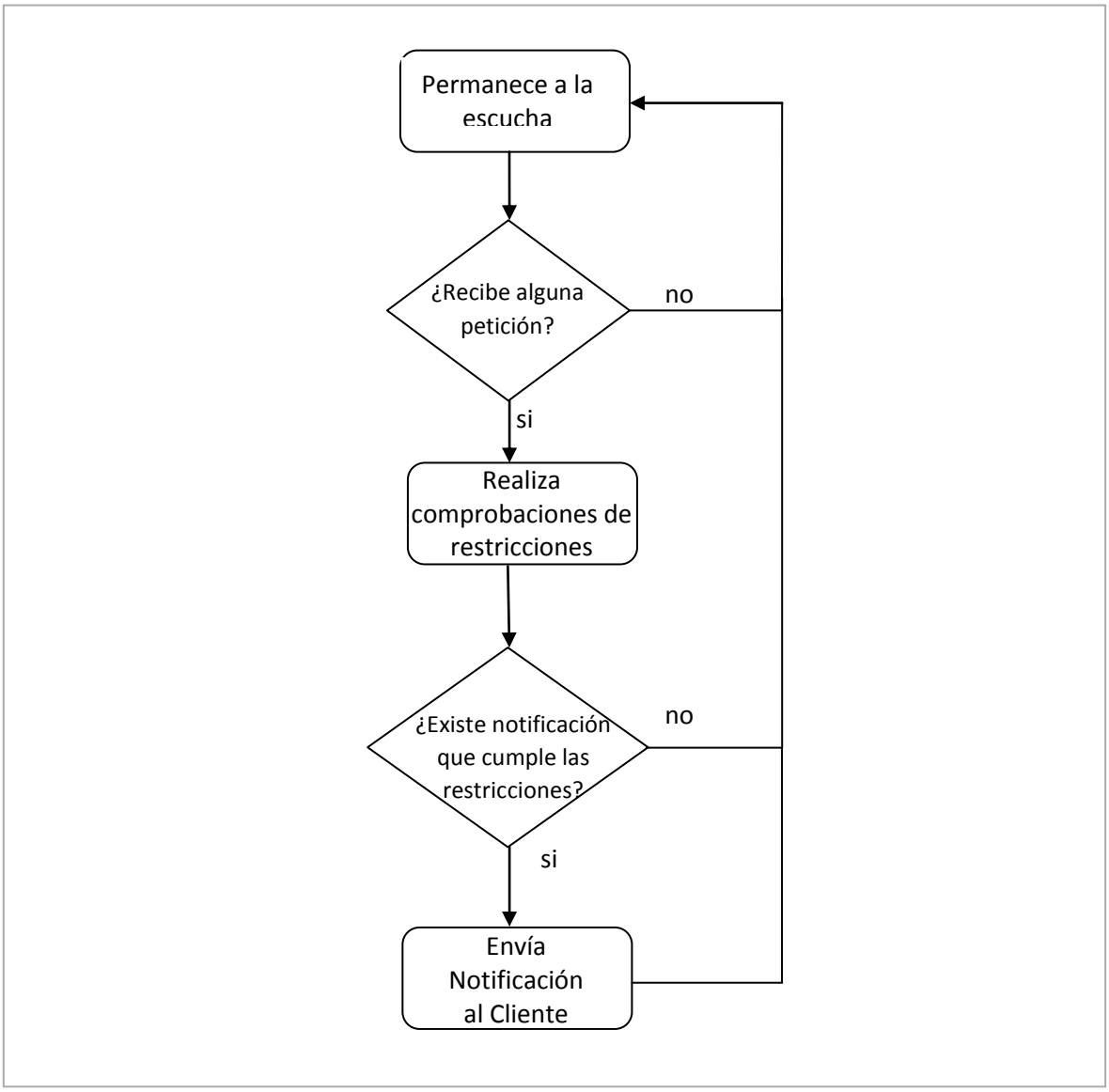

Figura 5. Diagrama de Flujo del comportamiento del Sistema Servidor Gestor de Notificaciones.

5.2. Diagrama de clases

El diagrama de clases se utiliza para modelar la vista estática del sistema, mostrando las clases y las relaciones entre ellas. La vista estática del sistema soporta principalmente los requisitos funcionales del sistema y los servicios que proporcionará al cliente.

Una **clase** es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica. En la representación gráfica de una clase se muestra el nombre de la clase y sus atributos, y se representa en un rectángulo.

Una **relación** es una conexión entre clases. En el modelado orientado a objetos, las tres relaciones más importantes son las dependencias, las generalizaciones y las asociaciones. Gráficamente, una relación se representa como una línea, usando diferentes tipos de línea para diferenciar los tipos de relaciones.

En nuestro caso, tratamos con un diagrama de clases muy sencillo, y por ese motivo solo existe un tipo de relación, la **asociación**, que es una relación estructural que especifica que los objetos de un elemento están conectados con los objetos de otro. Dada una asociación entre dos clases, se puede navegar desde un objeto de una clase hasta un objeto de la otra clase, y viceversa. Gráficamente, una asociación se representa como una línea continua que conecta con la misma diferentes clases. Una asociación puede tener un nombre, que se utiliza para describir la naturaleza de la relación.

Es importante señalar cuántos objetos pueden conectarse a través de una instancia de una asociación; a esto se le denomina **multiplicidad** de la asociación, y se escribe como una expresión que se evalúa a un rango de valores o a un valor explícito. Cuando se indica una multiplicidad en un extremo de una asociación, se está especificando que, para cada objeto de la clase en el extremo opuesto, debe haber tantos objetos en este extremo.

Una vez introducido el significado de un diagrama de clases, se procede a mostrar el diagrama de clases, en la *Figura 6,* que concierne al sistema gestor de notificaciones, objeto de este proyecto.

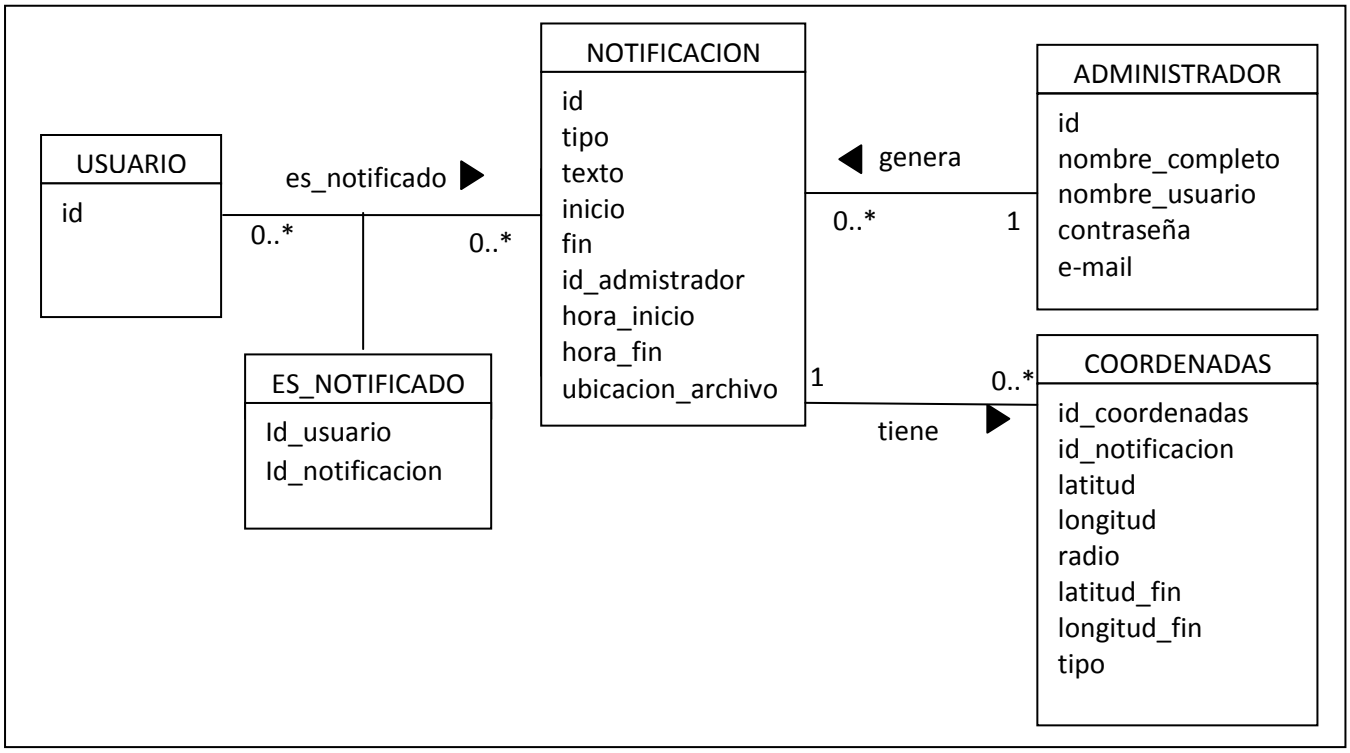

Figura 6. Diagrama de clases.

6. Implementación del sistema

Hasta ahora se ha especificado el análisis y el diseño del sistema, lo que ha permitido definir su funcionamiento y su estructura. A continuación se procede a exponer los detalles de implementación de los principales elementos que componen el sistema.

6.1. Base de datos

El sistema gestor de notificaciones emplea una base de datos relacional, para garantizar la persistencia de los datos. La importancia de que la base de datos se estructure y se diseñe correctamente es vital para el correcto funcionamiento del sistema.

Para facilitar el trabajo con la base de datos se ha empleado la herramienta *phpMyAdmin,* que es un administrador Web de bases de datos MySQL y que permite realizar tareas generales de administración de bases de datos.

En base al estudio del diseño del sistema, se procede a crear las tablas necesarias y sus relaciones. A continuación, en la *Figura 7*, se puede observar el esquema obtenido tras haber procedido a crear la base de datos a través de phpMyAdmin.

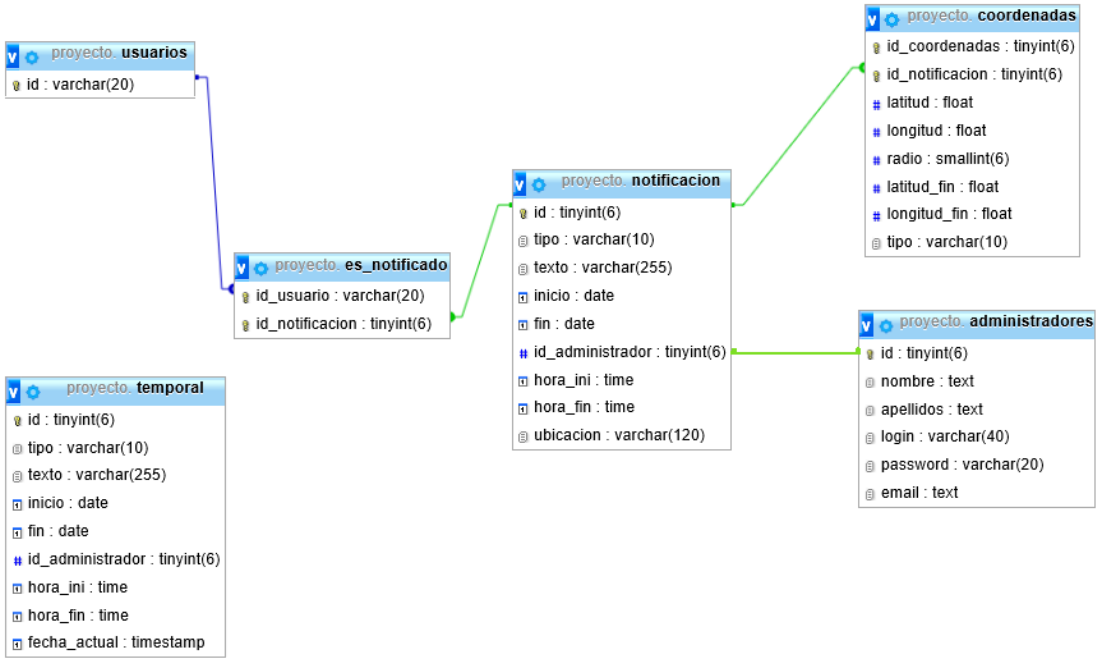

Figura 7. Esquema de la base de datos extraído de phpMyAdmin.

En el esquema de la base de datos se observa una tabla que no había sido contemplada en el diseño del sistema. Se trata de la tabla *temporal*, se ha considerado necesario crearla tras observar la necesidad de almacenar la notificación dada de alta por el Administrador, en un lugar temporal, mientras se seleccionan las ubicaciones de destino.

6.2. Interfaz Servidor

Para la instalación e integración de las distintas tecnologías utilizadas, se ha empleado el paquete de software XAMPP[14], que contiene el servidor Apache, el sistema de gestión de bases de datos MySQL, y el intérprete del lenguaje de programación PHP.

La interfaz del servidor está disponible a través de un portal Web. Para explicar el comportamiento de la Interfaz Servidor de una forma clara y concisa, se va a emplear un diagrama de flujo, donde se podrán observar las acciones que el Administrador del sistema podrá realizar accediendo al portal Web.

Más adelante se detallará cada una de las páginas *php* que componen el portal Web y su funcionalidad.

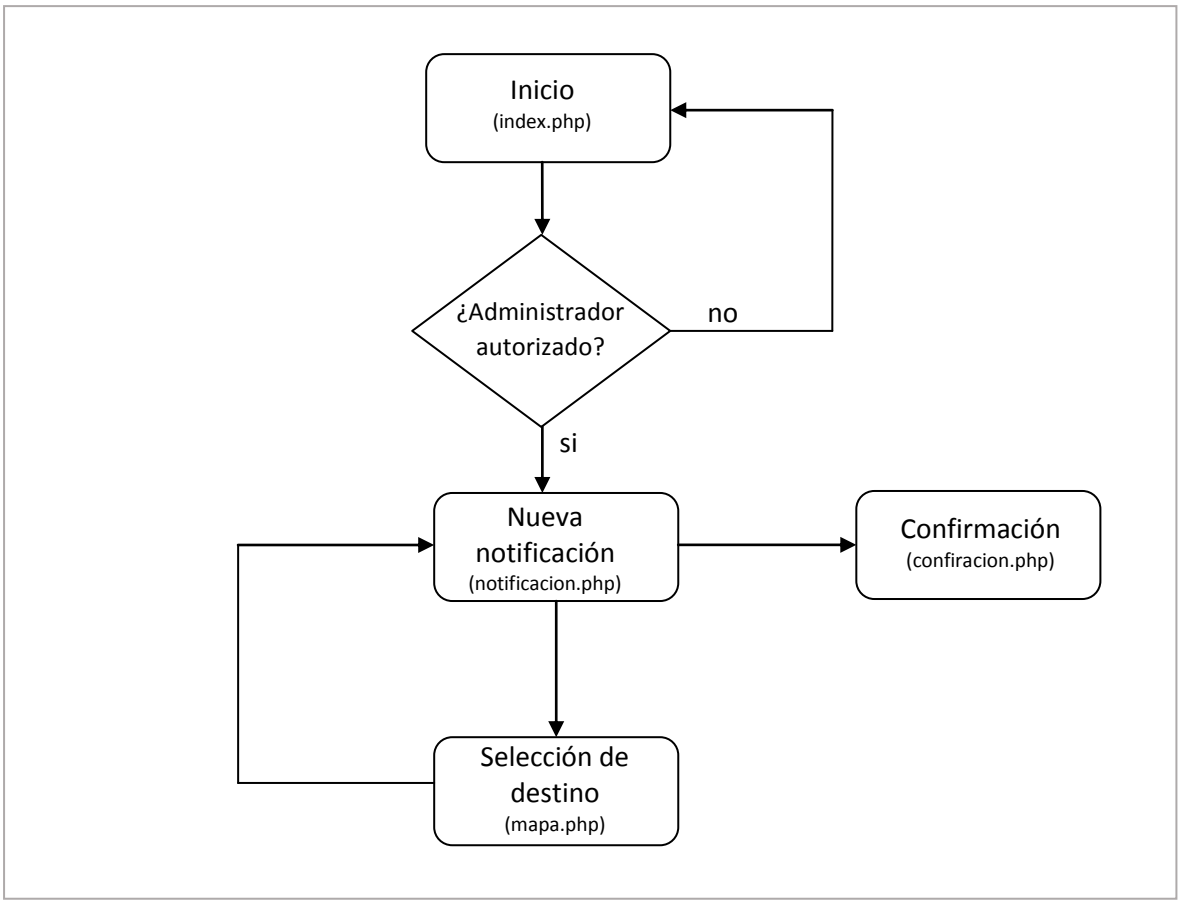

Figura 8. Diagrama de flujo portal Web.

index.php

En primer lugar el Administrador de la aplicación accede al portal mediante un navegador y se identifica. Para ello se crea *index.php,* que contiene un formulario de entrada donde el Administrador del sistema inserta sus credenciales.

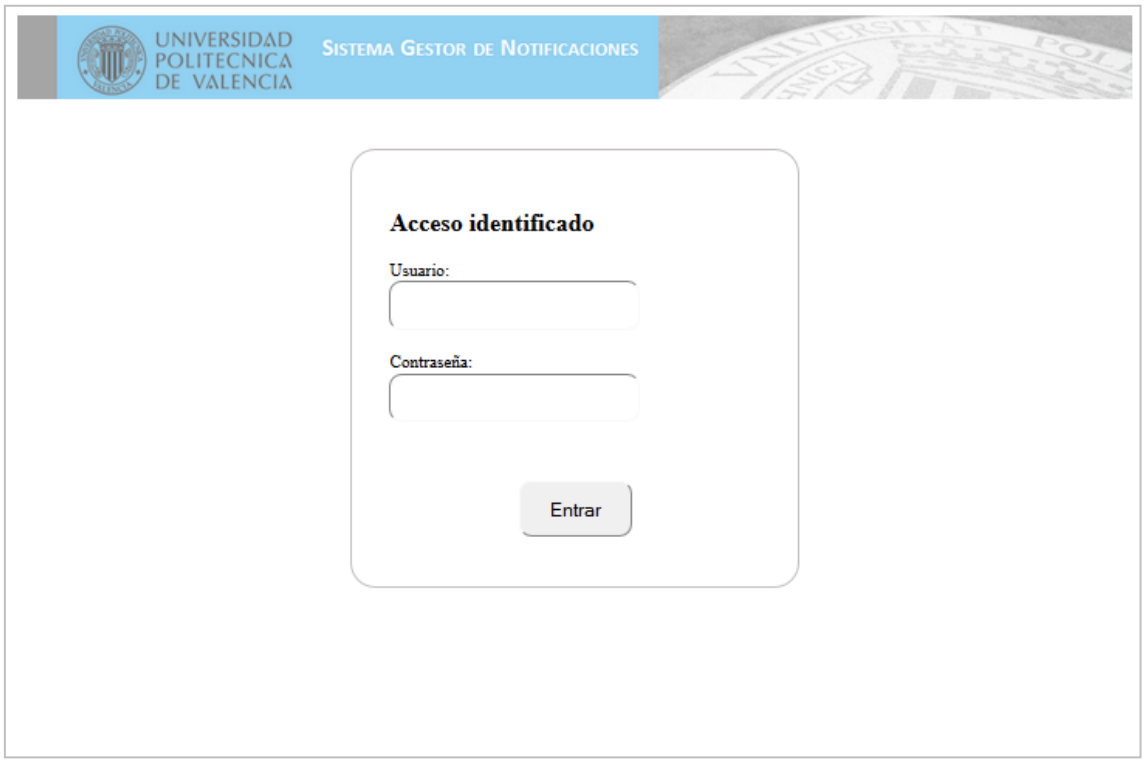

Figura 9. Página de inicio del portal web *(index.php).*

Index.php redirecciona el Administrador del portal, a la página *notificacion.php.* Para gestionar el acceso identificado al portal web, en *notificación.php,* se procede a establecer conexión con la base de datos para comprobar que, en la tabla *administradores,* existe el usuario y la contraseña especificados. Caso contrario se carga la página index2.php, la cual indica al Administrador que ha ocurrido un error en su identificación y que tiene que volver a introducir sus credenciales.

notificacion.php

Una vez correctamente identificado, el administrador puede crear notificaciones de audio, de video o de texto, y especificar las restricciones temporales y geográficas correspondientes.

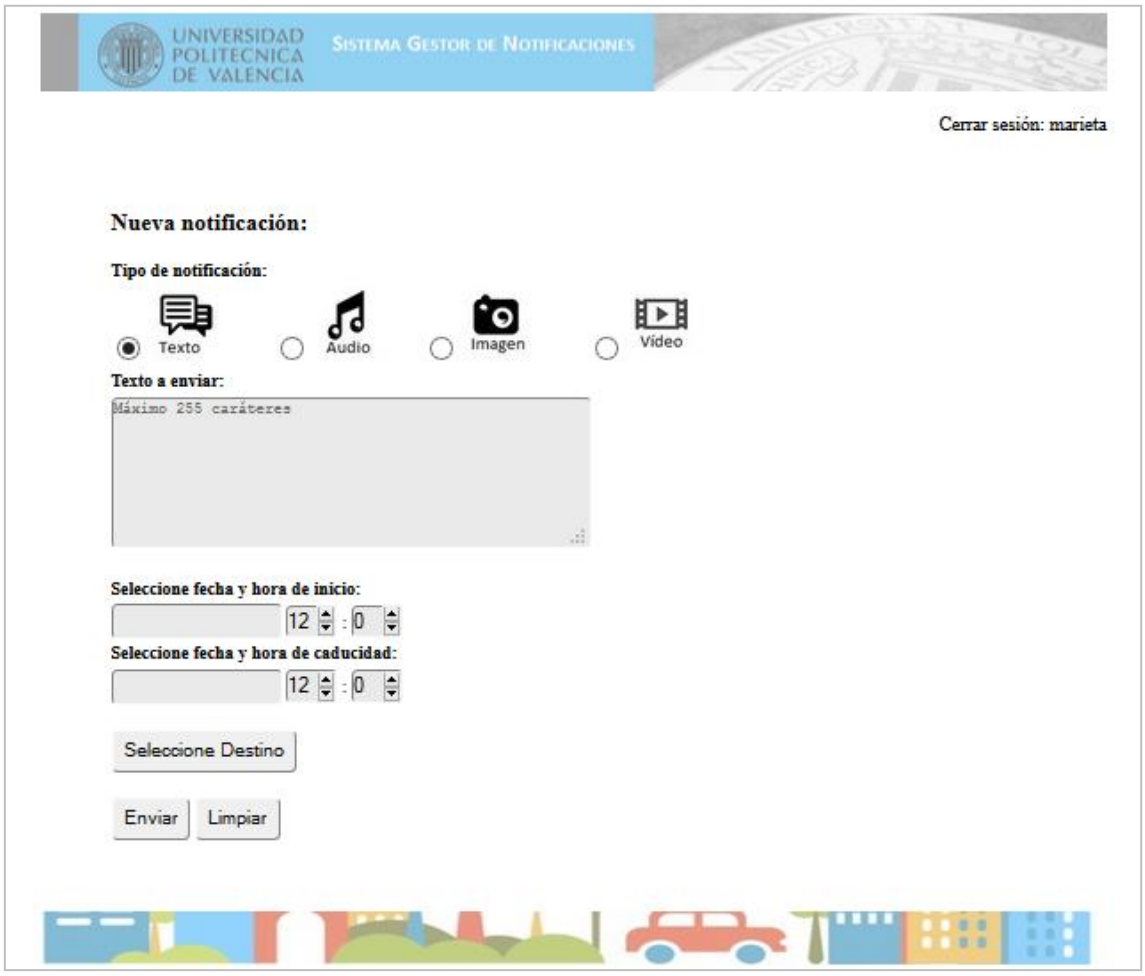

Figura 10. Página para la creación de notificaciones *(notificacion.php).*

En función del tipo de notificación seleccionada se mostrarán unos datos de entrada u otros. Más concretamente, si seleccionamos texto como tipo de notificación, se mostrará un cuadro de texto de 255 caracteres de longitud máxima, tal y como se puede observar en la *Figura 10*. Y si seleccionamos audio, imagen o video se mostrará un botón para seleccionar la ruta del archivo que se desea adjuntar.

Para seleccionar el destino o los destinos de la nueva notificación se accede a "*Seleccionar Destino*", que cargará la página *mapa.php*. Para que los datos introducidos no se pierdan, se ha implementado una función en *javaScript,* que almacena en la tabla *temporal* de la Base de Datos, los datos recogidos a través del formulario *Nueva notificación*. Se emplea el uso de sesiones. De esta manera, una vez seleccionado el destino, los datos de la notificación se cargarán en el formulario de *Nueva Notificación* para evitar al Administrador tener que

volverlos a introducir. Tras 5 minutos de inactividad, estos datos serán eliminados de la tabla *temporal.*

mapa.php

Para la implementación de un mapa en el portal Web que permita al Administrador seleccionar las ubicaciones de destino de las notificaciones creadas, se ha empleado la API proporcionada por MapBox [15].

MapBox es un software de código abierto que permite controlar cómo se ven los mapas y qué datos aparecen en ellos. El mapa global, ofrecido por MapBox, recolecta sus datos de OpenStreetMap *[16]*, pero a partir de esos datos se elaboran una serie de opciones que permiten un alto nivel de personalización en la creación de mapas a medida. Por este motivo se ha empleado MapBox.

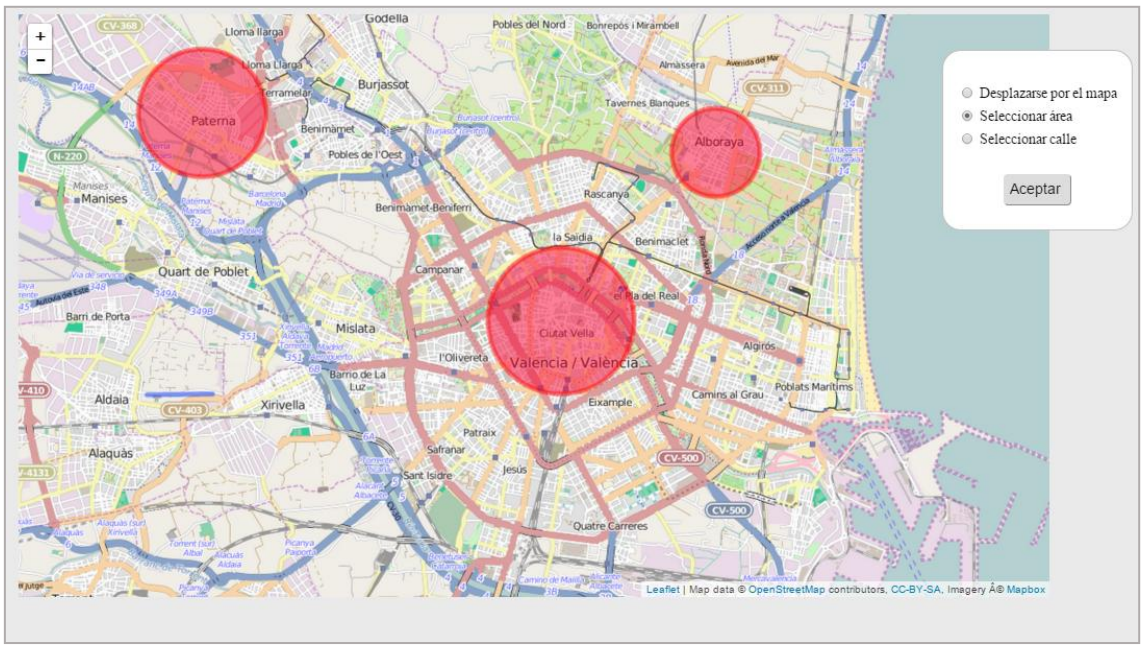

Figura 11. Página para la selección de ubicaciones de destino *(mapa.php).*

En el sistema gestor de notificaciones se permite la selección de varias áreas circulares de destino y la selección de varias áreas rectangulares, que pretenden representar la selección de calles. Las áreas circulares seleccionadas se representan en el mapa mediante un círculo rojo, y en caso de seleccionar una calle se representa con una línea azul sobre el mapa. En la *Figura 11,* se puede observar cómo se han seleccionado tres zonas (tres áreas circulares). Esto implica

que los terminales Android que se encuentren en alguna de esas tres zonas, recibirán la notificación para la que se han especificado estas ubicaciones de destino.

El sistema implementado permite hacer zoom sobre el mapa. Por defecto, se ha configurado el mapa para que se visualice la zona de la comunidad valenciana. Sin embargo, se pueden efectuar desplazamientos sobre el mismo para seleccionar cualquier parte del mundo.

La implementación de las funciones que nos permiten seleccionar y almacenar las diferentes ubicaciones de destino se han realizado en *javaScript*. A continuación, se procede a mostrar las funciones implementadas más relevantes para este punto explicando su funcionalidad.

Chequear_estado() es la función que permite realizar desplazamientos por el mapa, seleccionar un área o seleccionar una calle, en función de la opción marcada en el cuadro de *checkbox* que aparece en la esquina superior izquierda del mapa.

```
function chequear_estado(){
	var seleccionado_desplazarse= document.getElementById("desplazarse").checked;
	var seleccionado_circulo= document.getElementById("circulo").checked;
	var seleccionado_rectangulo= document.getElementById("rectangulo").checked;

	if (seleccionado_circulo) {
		map_select.dragging.disable();//quitar arrastre mapa
		map_select.on('mousedown', set_ini_circulo);
		map_select.on('mouseup', set_fin_circulo);
	} else{
		if (seleccionado_rectangulo) {
			map_select.dragging.disable();//quitar arrastre mapa
			map_select.on('mousedown', set_ini_cuadrado);
			map_select.on('mouseup', set_fin_cuadrado);

		}else{
			if (seleccionado_desplazarse){
				map_select.dragging.enable();
			}
		}
	}
}
```

Figura 12. Función chequear_estado *(mapa.php).*

Otra de las funcionalidades del portal web a destacar, es la función de dibujar manualmente, el área que se desea seleccionar sobre el mapa. Esto permite al Administrador pueda observar gráficamente el/los destino/s de la notificación que está dando de alta. Como se ha comentado anteriormente, se puede seleccionar un área circular o un área rectangular para representar una calle seleccionada. Para esto se han implementado las funciones pintar_circulo() y pintar_rectangulo(), respectivamente.

```
function set_ini_circulo(e){ //Se define el punto de inicio del área seleccionada por el Administrador
        INI_C=e.latlng;
}
function set_fin_circulo(e){ //Se define el punto de fin del área seleccionada por el Administrador
        FIN_C=e.latlng;
        pintar_circulo();
}

function pintar_circulo() {
        if(document.getElementById("circulo").checked){  //Opción Seleccionar área
                MI_OBJETO={}; //Inicialización MI_OBJETO
                var distancia=INI_C.distanceTo(FIN_C); //Calculo distancia entre inicio_circulo y fin_circulo

                var longitud=((INI_C.lng - FIN_C.lng)/2);
                var latitud=((INI_C.lat - FIN_C.lat)/2);
                var radio=distancia/2;

                longitud=INI_C.lng - longitud; //Calculo coordenada longitud
                latitud=INI_C.lat - latitud; //Calculo coordenada latitud

                MI_OBJETO = { tipo:"circulo", longitud:longitud, latitud:latitud, radio:distancia/2 };
                coordenadas=[latitud, longitud];
                circulo_dibujado=L.circle( coordenadas ,distancia, { //Pinar área seleccionada
                                color: 'red',
                                fillColor: '#f03',
                                fillOpacity: 0.5
                                }).addTo(map_select);

                confirmar(); //Llamo al método confirmar

        }
}
```

Figura 13. Función pintar_circulo *(mapa.php).*

```
function set_ini_cuadrado(e){ //Se define el punto de inicio del área seleccionada por el Administrador
        MI_OBJETO={};
        MI_OBJETO = {tipo:"rectangulo", longitud_ini:e.latlng.lng, latitud_ini:e.latlng.lat };

}
function set_fin_cuadrado(e){ //Se define el punto de fin del área seleccionada por el Administrador
        MI_OBJETO.longitud_fin=e.latlng.lng;
        MI_OBJETO.latitud_fin=e.latlng.lat;
        pintar_rectangulo();
}

function pintar_rectangulo() {
        if (document.getElementById("rectangulo").checked) {
                rectangulo_dibujado = L.polygon([
                        [MI_OBJETO.latitud_ini,MI_OBJETO.longitud_ini],
                        [MI_OBJETO.latitud_fin,MI_OBJETO.longitud_fin]
                        ]).addTo(map_select);
                confirmar();
        }
}
```

Figura 14. Función pintar_rectangulo *(mapa.php).*

A continuación se detalla el método confirmar(). Tras haber seleccionado el área y haberse mostrado en el mapa, el sistema muestra un cuadro de diálogo que pregunta al Administrador si desea guardar el área seleccionada. En caso afirmativo se guardará, y en caso negativo se eliminará del mapa.

```
function confirmar() {
        if((document.getElementById("circulo").checked) ||document.getElementById("rectangulo").checked){
                var r = confirm("Desea guardar el area seleccionada");
                if (r == true) {
                        LISTA_DE_OBJETOS.push(MI_OBJETO);
                        document.getElementById("variable").value=JSON.stringify(LISTA_DE_OBJETOS);
                } else {
                        var tipo=MI_OBJETO.tipo;
                        switch(tipo){
                                case "circulo":         borrar_circulo(circulo_dibujado); break;
                                case "rectangulo":   borrar_rectangulo(rectangulo_dibujado); break;
                                }
                }
        }
}
function borrar_circulo(circulo_dibujado){
        map_select.removeLayer(circulo_dibujado);
}
function borrar_rectangulo(){
        map_select.removeLayer(rectangulo_dibujado);
}
```

Figura 15. Función confirmar (*mapa.php).*

Por último, una vez seleccionado el/los destino/s de la nueva notificación en *mapa.php,* el sistema se redirige a *notificacion.php,* como se ha visto anteriormente en el diagrama de flujo, y le envia las coordenadas seleccionadas. Los parámetros que se envían son: *latitud, longitud, radio, latitud_fin, longitud_fin* y *tipo*. Estos parámetros son los que utilizaremos más adelante para calcular si las posiciones de los clientes recibidas están dentro de alguna de las áreas seleccionadas.

Una vez en *notificacion.php,* el Administrador comprobará que todo es correcto y aceptará la creación de la notificación, lo que redirigirá al sistema a la página *confirmacion.php.*

confirmacion.php

En confirmacion.php el sistema recoge todos los datos de la notificación y los inserta en la Base de Datos. Si todo es correcto muestra un mensaje de confirmación, en caso contrario notifica al Administrador que ha ocurrido un problema.

Si la notificación contiene un archivo adjunto, de audio, vídeo o imagen, se guarda en una carpeta llamada *subidas*, dentro del proyecto, con el nombre del archivo adjunto, y en la Base de Datos, se almacena esta ruta y el nombre del archivo, dentro de la tabla *notificación,* en el campo *ubicación*. De este modo, el Cliente accederá a esta ruta para descargarse el archivo adjunto.

Con todo esto el interfaz Web para la creación de notificaciones queda implementado y las notificaciones son dadas de alta en la Base de Datos.

Por otro lado, se ha implementado un Servicio Web para permitir al Servidor recibir las ubicaciones de los clientes.

android.php

El Servicio Web que se ofrece a los clientes, se ha implementado en el fichero *android.php*. Los clientes realizan una petición GET a este servicio, a través de la cual pasan como parámetros las coordenadas de su ubicación y su identificador. Más adelante, en el apartado de *implementación del Cliente,* se verá como se obtienen estos parámetros y como se realiza la petición *GET*. Ahora se procede a detallar cómo el servicio web trata estas peticiones.

El objetivo que tiene que cumplir el servicio Web, es procesar las peticiones *GET* recibidas y comprobar si existe alguna notificación que cumpla las restricciones temporales y geográficas. En caso afirmativo se comprobará si el cliente ya ha recibido esta notificación. Si el cliente no la ha recibido previamente, el Servidor le enviará la notificación en respuesta a la petición GET del cliente. Para ello se procede a detallar la implementación del servicio.

En primer lugar se accede a la Base de Datos y se realiza una consulta *SQL* para recoger las notificaciones dadas de alta. Tras este paso, se verifica si el cliente está registrado en la Base de Datos. Esta comprobación se realiza a través del parámetro: *identificador*, que envía el cliente en la petición GET, ya que es único. Si no existe en la Base de Datos, se inserta. De este modo se podrá tener un registro de todas las notificaciones que han sido enviadas a los clientes, consultando la tabla *es_notificado*, con el objetivo de evitar envíos duplicados.

Se procede a comprobar si existe alguna notificación que cumpla las restricciones temporales y geográficas, y si el cliente ha recibido dichas notificaciones.

Para comprobar las restricciones geográficas de un área circular contamos con los parámetros: lat (latitud de la ubicación del cliente), long (longitud de la ubicación del cliente), lat_not (latitud de la notificación), long_not (longitud de la notificación), radio (radio de la notificación) y tipo = circulo. Con estos datos, y aplicando el teorema de Pitágoras, se obtiene el cálculo de la ecuación de la circunferencia. El código *php* desarrollado se muestra en la *Figura 16* .

```
if ($tipo=="circulo"){ // si el tipo de notificación es un área circular
   if($lat_not!=0 && $long_not != 0 && $radio !=0){//Comprobar restricción por ubicación
        $calculo=sqrt(pow(($lat_not - $lat),2) + pow(($long_not - $long),2));
       //comprobación para saber si las coordenadas del cliente están dentro del área de la notificación
       if( $calculo<= $radio){    //si se cumplen las restricciones geográficas se comprueban las restricciones temporales
            comprobar_fecha()
       }
}
```

Figura 16. Comprobaciones restricciones geográficas para el área seleccionada en caso de círculo (android.php).

Para comprobar las restricciones geográficas de una calle contamos con los parámetros: lat_not (indica la latitud del punto de inicio de la calle), long_not (indica la longitud del punto de inicio de la calle), latitud_fin (indica la latitud del punto de fin de la calle), longitud_fin (indica la longitud del punto de fin de la calle), lat (latitud de la ubicación del cliente), long (longitud de la ubicación del cliente) y tipo = rectángulo. El código *php* desarrollado se muestra en la *Figura 17* .

```
if ($tipo=="rectangulo"){ //Si el tipo de notificación es una calle
    $esta_lat=0;
    $esta_long=0;
    if($lat_not<=$lat_fin){
            if($lat >= $lat_not && $lat <= $lat_fin){
                $esta_lat=1;
            }
    }
else{
     if($lat<=$lat_not && $lat >= $lat_fin){
        $esta_lat=1;
     }
}
if($long_not<=$long_fin){
      if($long >= $long_not && $long <= $long_fin){
            $esta_long=1;
      }
}else{
       if($long<=$long_not && $long >= $long_fin){
            $esta_long=1;
      }
}
if ($esta_lat==1 && $esta_long==1){ //La ubicación del cliente cumple restricción geográfica para la calle
//Comprobar restriccion temporal para id_not
    $comprobar_fecha()
}
```

Figura 17. Comprobaciones restricciones geográficas para el área seleccionada en caso de rectángulo (android.php).

En caso de que exista una notificación que cumpla las restricciones geográficas se debe de comprobar si se cumplen las restricciones temporales y si el cliente ya ha recibido la notificación. Para ello se accede la Base de Datos y, mediante consultas SQL, se realizan dichas comprobaciones.

Tras efectuar las tres comprobaciones, si el Servidor encuentra una notificación válida, debe de enviarla al cliente; para ello se encapsula la respuesta en *JSON.*

JSON es un formato ligero para el intercambio de datos. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, se va a emplear para el envío de las notificaciones del Servidor al Cliente.

6.3. Protocolo de comunicación

El Cliente solicita al Servidor las notificaciones a través de una petición GET, donde le pasa como parámetros su ubicación y su identificador. En la *Figura 18,* se puede observar la definición de los parámetros de la petición, así como la propia petición GET, que hay implementado en el código Android del Cliente. En el próximo apartado se profundizará más sobre el código implementado en la aplicación Android.

```
double latitud=loc.getLatitude(); //Obtenemos la latitud de la ubicación del Cliente
double longitud=loc.getLongitude(); // Obtenemos la longitud de la ubicación del Cliente
// Se obtiene el identificador del Terminal Android del cliente:
String myId = Secure.getString(getBaseContext().getContentResolver(), Secure.ANDROID_ID);

URL url =
new URL("http://10.0.2.2:8080/proyecto/android.php?lat="+latitud+"&long="+longitud+"&myid="+myId );
HttpURLConnection conn2 = (HttpURLConnection)url.openConnection();
conn2.setRequestProperty("User-Agent", "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)");
```

Figura 18. Petición GET que realiza el Cliente. (MainActivity.java).

El Servicio Web, que hay implementado en el Servidor, está atendiendo a todas las peticiones que le llegan y las procesa, tal y como se ha visto en el anterior apartado.

6.4. Interfaz Cliente

Android define un esquema de permisos para proteger ciertos recursos. En caso de que una aplicación intente acceder a un recurso del que no se ha solicitado permiso, se generará una excepción de permiso y la aplicación será interrumpida inmediatamente. Por este motivo, hay que dar permiso de acceso a Internet y permiso de acceso a la localización. Para ello, se han incluido en el archivo: *AndroidManifest.xml,* las lineas de codigo que se observan en la *Figura 19*.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.INTERNET" />
```

Figura 19. Concesión de permisos a Android. (AndroidManifest.xml).

La implementación del cliente, se ha desarrollado para que realice la petición GET al Servidor cada 10 segundos, o cada 100 metros recorridos, pasándole las coordenadas en dicha petición GET. De este modo, nos aseguramos de no colapsar al servidor con peticiones, ni de perder notificaciones destinadas a un punto geográfico determinado.

Para esta implementación, y para poder obtener las coordenadas del Clientee, se ha empleado un servicio de sistema, que se obtiene haciendo una llamada a la función *getSystemService()*. En concreto se ha utilizado un objeto *LocationManager*. También se ha empleado un L*istener* que escucha los cambios de estado del *GPS*. La F*igura 20* muestra la definición de ambos objetos.

```
LocationManager milocManager = (LocationManager)getSystemService(Context.LOCATION_SERVICE);
LocationListener milocListener = new MiLocationListener();
long minTime = 10000; // Intervalo minimo de actualizacion, 10 segundos.
long minDistance = 100; // Distancia minima de actualizacion, 100 metros.
milocManager.requestLocationUpdates( LocationManager.GPS_PROVIDER, minTime, minDistance,
milocListener);
```

Figura 20. Creación del objeto milocManager y del Listener milocListener. (MainActivity.java).

Se ha creado la clase *MiLocationListener* que implementa a la clase *LocationListener*, tal y cómo se puede observar en la *Figura 21.*

```
public class MiLocationListener implements LocationListener{
  public void onLocationChanged(Location loc) {
    double latitud=loc.getLatitude();
    double longitud=loc.getLongitude();
    String myId = Secure.getString(getBaseContext().getContentResolver(), Secure.ANDROID_ID);
    JsonTarea tarea = new JsonTarea(latitud, longitud, myId);
    tarea.execute();
 }
... // métodos; onProviderDisabled, onProviderEnabled, onStatusChanged
}
```

Figura 21. Clase MiLocationListener. (MainActivity.java).

En esta clase, observamos el método: *onLocationChanged(), a* este método se le llama cuando la posición *GPS* cambia. Por lo tanto, cada vez que la posición *GPS* cambie se llamará a la clase Json*Tarea*, que se detalla a continuación, pasándole como parámetros las coordenadas de la ubicación del cliente (*latitud* y *longitud*) y el identificador único del cliente (*myId*).

A continuación se procede a detallar los métodos implementados más relevantes de la clase *JsonTarea*. En primer lugar, se establece la conexión con el Servidor, realizando la petición GET y procesando la respuesta obtenida, en el método *doInBackground()* de la clase JsonTarea, tal y como se puede observar en la *Figura 22.*

```
protected String doInBackground(URL... urls) {
 String notificacion = null;
 URL url = new
URL("http://10.0.2.2:8080/proyecto/android.php?lat="+latitud+"&long="+longitud+"&myid="+myId);
HttpURLConnection conn2 = (HttpURLConnection)url.openConnection();
 conn2.setRequestProperty("User-Agent", "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)");
 if (conn2.getResponseCode()==HttpURLConnection.HTTP_OK) { //Leo la respuesta obtenida
  InputStream in = new BufferedInputStream(conn2.getInputStream());
  BufferedReader buffer = new BufferedReader(new InputStreamReader (in));
  StringBuilder builder = new StringBuilder();
  String cadena;
  while ((cadena=buffer.readLine())!= null){
     builder.append(cadena+" ");
   }
 buffer.close();
 notificacion= builder.toString(); //la variable notificación contiene la respuesta obtenida
}
```

Figura 22. Método *doInBackground* de la clase *JsonTarea*, establecimiento de conexión con el Servidor. (MainActivity.java).

En este punto ya tenemos la notificación recibida del Servidor, a continuación, se procede a pasear el *JSON* obtenido. Para ello, se crea un objeto *JSONObjetc*, y de este objeto se extraen las variables: *tipo, texto y ubicación.* Con estas variables se gestiona la reproducción automática de la notificación, tal y como se puede observar en la *Figura 23*.

```
@Override
protected void onPostExecute(String notificacion) {
   JSONObject objJson = new JSONObject (notificacion);
   String tipo = objJson.getString("tipo");
   String texto = objJson.getString("texto");
   String ubicacion = objJson.getString("ubicacion_fichero");
   if (!TextUtils.isEmpty(texto)&& !TextUtils.equals(texto, "null")) {
     try {
         Intent myIntent = new Intent(Intent.ACTION_VIEW,
         Uri.parse("http://translate.google.com/translate_tts?tl=es&q="+texto));
         startActivity(myIntent);
     }catch (ActivityNotFoundException e) {
              Toast.makeText(getApplicationContext(),
            "No se ha podido reproducir el texto de a notificación",
              Toast.LENGTH_LONG).show();
              e.printStackTrace();
        }

    }else{
        if (!TextUtils.isEmpty(ubicacion)&&!TextUtils.equals(ubicacion, "null")) {
          try { Intent myIntent = new Intent(Intent.ACTION_VIEW,
           Uri.parse(ubicacion));
            startActivity(myIntent);
          }catch (ActivityNotFoundException e) {
              Toast.makeText(getApplicationContext(),
             "No se ha podido reproducir el archivo de la notificación",
              Toast.LENGTH_LONG).show();
              e.printStackTrace();
          }
}
```

Figura 23. Método *onPostExecute* de la clase *JsonTarea*, tratamieno de la Notificación. (MainActivity.java).

Para transformar una notificación de tipo texto a audio se ha empleado un servicio que ofrece Google para este propósito, se trata de*: Text-to-Speech*, que es un sistema que permite la conversión de textos a voz sintética, esta característica únicamente funciona a través de un navegador, por ello, tras comprobar que la notificación recibida es de tipo texto, se crea myIntent; Un Intent es un mecanismo que tiene Android para invocar aplicaciones externas a la nuestra. En myIntent detallamos la acción a realizar, llamar a la aplicación *Text-to-Speech*, a través de la URL: "http://translate.google.com/translate_tts?tl=es&q=" y concatenando la cadena de texto recibida.

Si la notificación es de otro tipo (audio, imagen o vídeo), se crea un *Intent* para conectarse a la URL especificada en el campo *ubicación* de la notificación,

De este modo conseguiremos nuestro objetivo de reproducir una notificación sin intervención manual del Cliente.

7. Validaciones y pruebas

En este apartado se van a realizar las validaciones y las pruebas pertinentes, para comprobar que el Sistema funciona correctamente, y para estimar el rendimiento y la escalabilidad del Sistema.

7.1. Validaciones.

Para mostrar el correcto funcionamiento del Sistema, se va a proceder a mostrar un ejemplo realizado con un Dispositivo Nexus S, emulado por el *AVD Manager*, y con la versión de *Android 5.0, API Level 21*.

En primer lugar, se procede a crear varias notificaciones, de distintos tipos y destinadas a diferentes puntos geográficos, lo cual se realiza a través del interfaz del Administrador. Y como resultado, las notificaciones creadas quedan almacenadas en la Base de Datos, más concretamente, en la tabla "*notificación" y en la tabla "coordenadas*".

id 1	tipo	texto	inicio	fin	id_administrador	hora_ini	hora_fin	ubicacion
116	texto	RETENCIÓN / CONGESTIÓN por CIRCULACIÓN	2015-06-30	2015-06-30	1	12:00:00	12:00:00	
115	video		2015-06-01	2015-06-30	1	12:00:00	12:00:00	http://10.0.2.2:8080/proyecto/subidas/A Magi...
114	imagen		2015-06-01	2015-06-30	1	12:00:00	12:00:00	http://10.0.2.2:8080/proyecto/subidas/io
113	audio		2015-06-01	2015-06-30	1	12:00:00	12:00:00	http://10.0.2.2:8080/proyecto/subidas/a
111	texto	OBSTÁCULO FIJO por INUNDACIONES con circulación interrumpida en: - La CARRETERA PROVINCIAL CO-5202 desde el km 0 en NUEVA CARTEYA (CÓRDOBA) al km 6.5 en CASTRO DEL RIO (CÓRDOBA) sentido AMBOS SENTIDOS Advertencia: CORTE TOTAL	2015-06-01	2015-06-30	1	12:00:00	12:00:00	

Figura 24. Tabla Notificación.

id_coordenadas	id_notificacion 1	latitud	longitud	radio	latitud_fin	longitud_fin	tipo
125	115	32.4855	-0.383415	3871	0	0	circulo
124	114	39.4812	-0.381012	3172	0	0	circulo
123	113	39.4741	-0.385475	3630	0	0	circulo
120	111	42.4746	-0.374146	2833	0	0	circulo
119	110	38.4685	-0.381355	2100	0	0	circulo

Figura 25. Tabla coordenadas.

Tras este paso, se procede a mostrar la reproducción, en el Cliente, de aquellas notificaciones que cumplan las restricciones temporales y geográficas. Para ello, a través del emulador del *AVD Manager,* se emula un cambio de posición del dispositivo Android.

A través de la herramienta *DDMS (Dalvik Debug Monitor Server*)**,** que actúa como un intermediario para conectar el entorno de desarrollo con las aplicaciones que se ejecutan en el dispositivo, establecemos las nuevas coordenadas del dispositivo Android, tal y como se puede observar en la *Figura 26*.

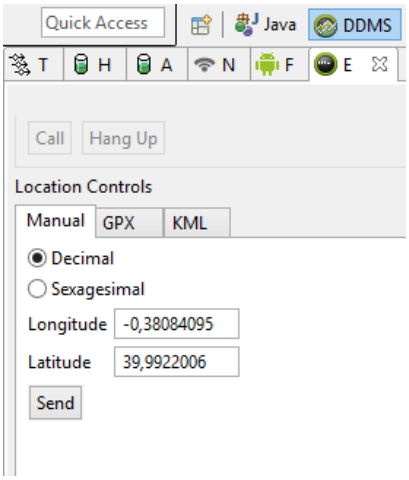

Figura 26. Emular cambio de posición del terminal Android.

Como se ha visto en el apartado: Implementación del Sistema, cuando la ubicación del Cliente cambia, se realiza una petición GET al Servidor; y el Servidor, al procesar la petición y tras realizar las comprobaciones necesarias, si encuentra alguna notificación la enviará al Cliente.

En este punto, el Sistema ha realizado las comprobaciones necesarias, y ha detectado dos notificaciones destinadas al Cliente; una notificación es un audio *(identificador 113*), y la otra notificación es una imagen (*identificador 114*). Si nos fijamos en el identificador de las notificaciones, en la tabla "*coordenadas*" (*Figura 25)*, podemos estimar que ambas notificaciones cumplen las restricciones geográficas. Si nos fijamos en la tabla "*notificación*" (*Figura 24),* podemos observar que cumplen las restricciones temporales.

A continuación, se va a mostrar una captura de pantalla del terminal Andorid, con la reproducción automática de las notificaciones recibidas por el Cliente.

Figura 26. Reproducción en el terminal Android de una notificación de Audio.

Figura 27. Reproducción en el terminal Android de una notificación de Imagen.

A continuación el Sistema registra, en la tabla "es_notificado", que el Cliente ya ha recibido ambas notificaciones, para no realizar envíos duplicados.

id_usuario	id_notificacion
962b19e09f007a69	114
962b19e09f007a69	113

Figura 28. Tabla *es_notificado*

Con todo esto, se ha validado el correcto funcionamiento del Sistema.

7.2. Pruebas de rendimiento y escalabilidad.

En el ámbito de nuestra aplicación servidora, el rendimiento se define como la velocidad a la que el Sistema es capaz de procesar las peticiones de los Clientes.

Para medir el nivel de rendimiento alcanzado, se ha procedido a simular un gran número de peticiones simultáneas, tratando de emular a un gran número de Clientes accediendo simultáneamente al Sistema.

Características hardware del servidor en que se han realizado las pruebas:

Modelo Servidor :	*MacBook Pro*
Procesador:	*Intel Core i5*
Velocidad del procesador:	*2,5 GHz*
Cantidad de procesadores:	*1*
Cantidad total de núcleos:	*2*
Caché de nivel 2 (por núcleo):	*256 KB*
Caché de nivel 3:	*3 MB*
Memoria:	*8 GB*

En concreto se han realizado 1000 peticiones, y el Sistema ha tardado en procesar todas las peticiones 110 segundos, lo que significa que el sistema tarda alrededor de 110 milisegundos en tratar una petición. Este dato es un indicativo de que el rendimiento del Sistema es aceptable para un número bajo de clientes. Además, el hecho de que el sistema tenga un buen rendimiento no implica que sea un Sistema escalable.

Para que un sistema sea escalable tiene que poder adaptarse a un incremento en el número de usuarios y a un incremento en el tamaño de los datos almacenados.

Existen dos tipos de escalabilidad: [19]

Escalabilidad en vertical. El escalado en vertical se realiza utilizando un hardware mejor. El escalado incluye agregar más memoria, más procesadores o procesadores más rápidos o, simplemente, migrar la aplicación a un único equipo más potente. El uso de un único equipo disminuye la tolerancia de errores del sistema. Además se trata de una solución cara.

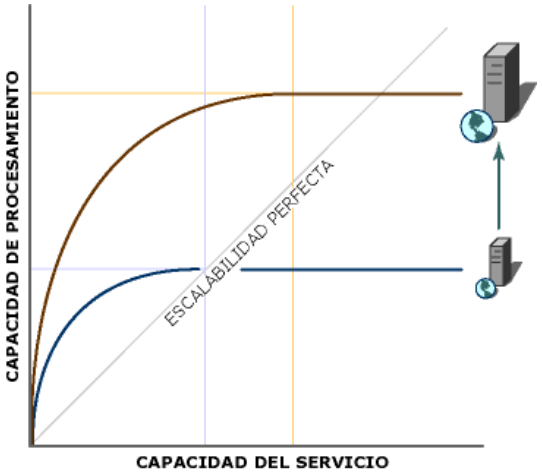

Figura 29. Escalabilidad en vertical.

Escalabilidad horizontal. En una escalabilidad horizontal se añaden equipos para dar más potencia a la red de trabajo. Aunque el escalado en horizontal se logra utilizando muchos equipos, la colección funciona esencialmente como un único equipo. Al dedicar varios equipos a una tarea común, mejora la tolerancia de errores de la aplicación. Existe una gran variedad de técnicas de equilibrio de carga para escalar en horizontal. El equilibrio de carga permite escalar un sitio en horizontal a través de un clúster de servidores, facilitando la adición de capacidad, agregando más servidores duplicados. También proporciona redundancia, de manera que el servicio permanece disponible para los usuarios incluso si uno o más servidores fallan .

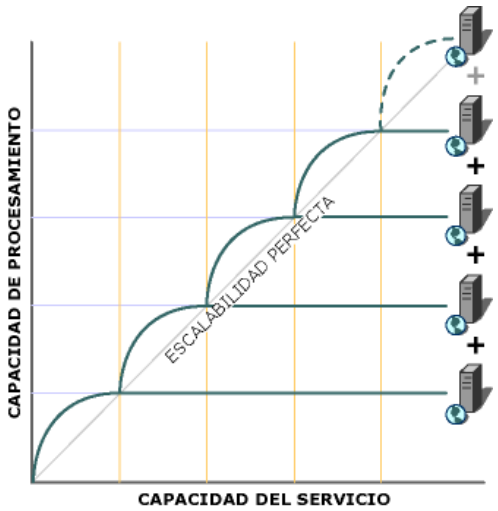

Figura 30. Escalabilidad en horizontal.

Existen muchos estudios que tratan el tema para la mejora de la escalabilidad de un sistema, sin embargo, dichos aspectos son ajenos a los objetivos de este TFG.

8. Conclusiones

Una vez superadas todas las fases del desarrollo, se procede a evaluar si se han conseguido los objetivos marcados al inicio del proyecto.

Un primer objetivo buscaba conseguir una interfaz accesible desde cualquier terminal con acceso a Internet para el Administrador del Sistema, y esto se ha conseguido tras la configuración del servidor Apache y el desarrollo de la interfaz de Administrador.

Por otro lado, se pretendía dar persistencia a las notificaciones creadas por el Administrador, y a través del sistema de gestión de bases de datos MySQL se ha alcanzado este segundo objetivo.

Igualmente, se ha alcanzado el objetivo de tener un control de las notificaciones enviadas a los Clientes para evitar envíos duplicados, así como de realizar las comprobaciones oportunas para enviar las notificaciones que cumplan las restricciones temporales y geográficas.

De la parte del cliente, se comprueba que el terminal Android envía correctamente al Servidor la ubicación en la que se encuentra, y que es capaz de reproducir las notificaciones recibidas.

Globalmente, se puede afirmar que se ha obtenido un Sistema de Gestión de Notificaciones que cumple los objetivos inicialmente propuestos.

Para finalizar, confío en haber transmitido los conocimientos adquiridos durante la realización de este proyecto y en haber plasmado la motivación que me ha llevado a realizarlo.

Es una satisfacción haber concluido el proyecto, ya que desde el lado personal me ha permitido enfrentarme a nuevos retos, y desde el lado profesional me ha dotado de conocimientos y experiencia valorada muy positivamente en cualquier ámbito laboral.

9. Ampliaciones

A pesar de que en este proyecto se han cubierto las necesidades básicas funcionales de la aplicación, existen aspectos que son susceptibles de ser considerados de cara a un desarrollo futuro.

Por un lado, el diseño, tanto de la interfaz del Administrador, como de la interfaz del Cliente, se puede mejorar, para darle un aspecto estético más atractivo.

Otro punto a considerar, es la mejora del desarrollo de la aplicación del Cliente. Se puede gestionar la reproducción automática de las notificaciones de una forma más autónoma, sin la necesidad de utilizar servicios externos.

El cálculo que realiza el Sistema, para comprobar que las coordenadas del cliente están dentro de un área se puede implementar de una forma más precisa, empleando funciones trigonométricas.

En el terminal Android, se podría implementar una funcionalidad que almacene las notificaciones recibidas; de ese modo, el Cliente podría tener acceso a un historial de notificaciones.

En el Sistema Servidor, se podría mantener un histórico de notificaciones. De esta forma, cuando una notificación no tenga validez temporal, sería eliminada automáticamente de la tabla "notificación" y de la tabla “coordenadas”, y sería almacenada en otras tablas de histórico. De ese modo, el Sistema no las tendrá en cuenta en las comprobaciones de las restricciones, y por otro lado, el Administrador del Sistema tendrá un registro de las notificaciones que ha dado de alta.

También se podrían implementar estadísticas para conocer datos sobre las notificaciones entregadas, como por ejemplo, la cantidad de veces que se ha enviado una notificación.

Otro punto interesante que se podría tener en cuenta para integrarlo a esta aplicación, es desarrollar un mapa interactivo para conocer, en tiempo real, la cantidad de usuarios que hay en las diferentes ubicaciones, mostrando algún tipo de señal en el mapa que se muestra en la interfaz del Administrador.

10. Referencias bibliográficas

[1] **Ingeniería del Software**. Editorial: Pearson Addison Wesley. Autor: Ian Sommervi. 7ª edición. Año de Publicación: 2005. Identificador: ISBN 8478290745.

[2] **Creación de sitios Web con PHP5**. Editorial: Mc Graw Hill. Autores: F. Javier Gil Rubio. Santiago Alonso Villaverde. Jorge A. Tejedor Cerbel. Agustín Yagüe Panadero. Año de Publicación: 2006. Identificador: ISBN 844819814X.

[3] **Redes de computadoras. Un enfoque descendente**. Editorial: Pearson. Autores: James F. Kurose. Keith W. Ross. 5ª Edición. Año de Publicación: 2011. Identificador: ISBN 9788478291199; ISBN 8478291199.

[4] **El lenguaje unificado de modelado**. Editorial: Booch Jacobson. Autores: Grady Booch, James Rumbaugh, iVar Jacobson. 2ª Edición. Año de Publicación: 2007. ISBN: 8478290761.

[5] **Apuntes asignatura: *Ingeniería del Software***. DSIC-UPV, curso 2013-2014.

[6] **Desarrollo Web con PHP y MySQL**. Editorial: ANAYA. Autores: Luke Welling. Laura Thomson. Año de Publicación: 2003. Identificador: ISBN 8441515697

[7] **Programación Android**. Editorial: ANAYA. Autor: Ed Burnette. Año de Publicación: 2011. Identificador: ISBN 9788441528765

[8] **Fundamentos Desarrollo Web con PHP, Apache y MySQL**. Editorial: ANAYA. Autores: Michael K. Glass, Yann Le Scouarnec, Elizbeth Naramoe, Gary Mailes, Jeremy Stolz, Jason Gerner. Año de Publicación: 2010. Identificador: ISBN 9788441526228

[9] **Waze**. URL: www.waze.com/

[10] **RadarDroid, avisador GPS de radares móviles**. URL: http://www.radardroid.com/

[11] **Mi Coyote, People Inside**. URL: www.mycoyote.es

[12] **Android Developers**. URL: http://developer.android.com/about/dashboards/index.html

[13] **Instituto Nacional de Estadística.** URL: http://www.ine.es/prensa/np864.pdf

[14] **XAMPP, Apache + MySQL + PHP + Perl**. URL: https://www.apachefriends.org/index.html

[15] **Web Services Mapbox.** URL: https://www.mapbox.com/developers/api/

[16] **OpenStreetMap**. URL: http://www.openstreetmap.org

[17] **Dirección General de Tráfico.** URL: http://www.dgt.es/es/app-movil.shtml

[18] **Ayuntamiento de Valencia**. URL: http://www.valencia.es/ayuntamiento/atencion_ciudadano.nsf/vDocumentosTituloAux/Aplicaciones%20m%C3%B3viles?opendocument

[19] **Microsoft Application Center 2000 reduce la complejidad y el coste del escalado en horizontal**. URL: http://www.microsoft.com/applicationcenter/default.htm