UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCOLA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

"Programación de una plataforma RTCU IDE para controlar una flota de vehículos a través de GSM, Bluetooth, WiFi y GPS."

TRABAJO FINAL DE GRADO

Autor/a:
César Sánchez López

Tutor/a:
María Consuelo Part Escriva

GANDIA, 2018

Agradecimientos:

Juan Fernández de Ybarra del Rey, técnico informático del Instituto Cartográfico Valenciano.

Cayetano Candela Montes, técnico informático

Índice

1. Índice de Figuras

2. Resumen (abstract)

Este trabajo consiste en programar sobre la plataforma RTCU (Unidad de Control Telemática y Remota) IDE (Entorno de Desarrollo Integrado), de la empresa danesa Logic IO ApS, una aplicación que permita visualizar los vehículos dotados con este sistema, a través de GSM (Sistema Global para las comunicaciones móviles), Bluetooth, WiFi y GPS (Sistema de Posicionamiento Global), y representarlos en un mapa digital del ICV (Instituto Cartográfico Valenciano), mostrando la posición, el sentido de movimiento, la velocidad actual y la velocidad máxima alcanzada. Se utilizarán los dispositivos "RTCU MX2i pro/pro+"

Se revisará documentación de la página del fabricante y estudiaran catálogos y manuales para programar la aplicación, programando ejemplos sencillos de la aplicación para conseguir programar la aplicación definitiva.

A continuación, comprobaremos el correcto funcionamiento de la aplicación instalando equipos RTCU IDE en vehículos y realizando algún recorrido para validar el adecuado funcionamiento de la aplicación solucionando posibles problemas.

This work consists in programming on the platform RTCU (Remote Telemetry and Control Unit) IDE (Integrated Development Environment), of the Danish company Logic IO ApS, an application that allows to visualize the vehicles endowed with this system, through GSM (Global System for mobile communications), Bluetooth, WiFi and GPS (Global Positioning System), and represent them on a digital map of the ICV (Valencian Cartographic Institute), showing the position, the direction of movement, the current speed and the maximum speed reached. The devices "RTCU MX2i pro / pro +" will be used

Documentation of the manufacturer's page will be reviewed and catalogs and manuals will be studied to program the application, programming simple examples of the application to get the definitive application programmed.

Next, we will check the proper functioning of the application by installing RTCU IDE equipment in vehicles and making a tour to validate the proper functioning of the application solving possible problems.

3. Introducción

Vamos a presentar un centro de gestión de emergencias que propone una mejora en cuanto a las prestaciones que proporcionan los centros de emergencias actuales. Así pues nuestro gestor pretende dar un mejor servicio que sea rápido, eficaz, eficiente y adaptable a cada uno de los municipios en donde sea implantado para la gestión de alarmas y emergencias.

El centro de gestión de emergencias de este TFG se desarrolla en colaboración con el instituto cartográfico valenciano y en el proyecto se encuentra involucrada otra institución oficial como el cuerpo de policía de Onteniente, por consiguiente se han dispuesto todos los medios pertinentes por ambas partes para el desarrollo de esta aplicación.

Tal y como se puede observar en nuestra sociedad, los servicios de gestiones de emergencias están presentes en el día a día de todo ciudadano, desde la mujer que es maltratada y no sabe a quien acudir, pasando por el trabajador que tiene un percance, incluso el conductor que sufre o es testigo de un accidente.

Todos los casos tienen algo en común y es que, en el momento de padecer una emergencia se está confuso y desorientado, muchas veces el propio afectado no dispone de la información necesaria para poder ser ayudado. Esto junto con la falta de medios y/o recursos para poder enviar al servicio adecuado (policía, bomberos, etc...) al lugar y tiempo adecuado hace que el solucionar una emergencia en un momento dado sea una tarea dificultosa y muchas veces ineficiente.

Así pues, ¿qué podemos hacer si tenemos una emergencia? Voy a motivar la problemática que vamos a abordar en este proyecto a través del siguiente relato.

3.1. Historia de una emergencia

La siguiente historia tiene lugar en el término municipal de Onteniente.
Nuestro amigo Juan decide celebrar su cumpleaños con todos sus familiares y amigos. Para ello los invita a una barbacoa en su casa rural situada cerca del monte, pese a ser una época de intenso calor la gente decide aceptar de buen gusto la invitación de Juan y acudir al evento.

Domingo 13.00 horas: como la mayoría de la gente desconoce la localización de la casa rural y para que nadie se pierda en el trayecto de ida o vuelta, Juan alquila un vehículo que él mismo conduce, quedan todos en la entrada del pueblo y de allí todos suben al vehículo para dirigirse al lugar de la celebración.

A las 14.00 horas todos se congregan en el jardín para la celebración, junto con todos los invitados han venidos los niños pequeños de los familiares, a las 15.00 horas la comida ha terminado y todo ha sido un éxito. Tras darse por finalizada la barbacoa, Juan acomoda a sus invitados dentro de la casa rural. Sin embargo surge un problema conocido de sobra, los niños no quieren descansar dentro, los padres con tal de poder descansar les dejan jugar en el jardín y la arboleda cercana a la casa rural, siempre y cuando estos vuelvan al cabo de un rato.

Juan para hacer la celebración aún más completa decide ir al pueblo a por unas pastas para tomar con el café con lo que se marcha con el único vehículo disponible en la casa rural.

Al poco tiempo los niños se aburren de estar jugando en el jardín, entonces Miguel el inquieto del grupo decide que podrían jugar a hacer señales de humo con las brasas sobrantes de la barbacoa.

Al poco tiempo de estar jugando se aburren y se vuelven a la casa, con la imprudencia de haberse dejado las brasas medio encendidas en el bosque, al cabo de diez minutos de estar los niños en casa, se observa una humareda intensa proveniente de la arboleda cercana a la urbanización.

Los adultos se acercan y se percatan que se ha originado un fuego y que se está avivando por momentos. En ese instante cunde el pánico entre los asistentes de la celebración, empiezan a surgir los problemas: el propietario de la casa rural esta ausente y junto con él el único vehículo que había, los asistentes desconocen la posición de la casa rural.

Uno de los adultos se pone en contacto con el centro de emergencias 112 pero entre la confusión del momento y la imposibilidad de ubicarse este le mantiene en espera intentando dar con la localización exacta de la casa rural.

De repente otro asistente de la celebración recuerda que hace poco en Onteniente se había implantado un sistema de gestión de emergencias. Rápidamente se ponen en contacto con este centro de emergencias y son atendidos por un operador, el cual resulta ser un policía local perteneciente al término municipal de Onteniente, tras explicarle la emergencia nos pide la ubicación de la casa rural, la cual seguimos sin saber, pero aún cuando todo parecía perdido, el agente nos dice que no nos preocupemos que tal y como tenían previsto para estos casos, también les servia el nombre del propietario de la parcela o en su defecto su referencia asignada, incluso su teléfono móvil. Al buscar por el teléfono se pudo ubicar al instante la casa rural de Juan.

En un tiempo récord los bomberos y sanitarios aparecieron en la casa rural, y gracias a la rápida y eficaz coordinación del centro de emergencias y los efectivos se consiguió sofocar el fuego que de haber seguido más tiempo se habría vuelto incontrolable.

El relato que he introducido para motivar la problemática a tratar muestra que como en el momento de surgir una emergencia, el tiempo es un factor crítico, y que por tanto cuanta más rápida y precisa sea la intervención por parte del cuerpo de seguridad pertinente mayor será la probabilidad de éxito ante la emergencia. Por todas estas razones ha surgido la necesidad de la creación de un centro de emergencias que utilizando los mapas propios del instituto cartográfico valenciano pretende dar un servicio rápido, eficaz y eficiente, así como adaptable a las necesidades de cada uno de los municipios en donde decida ser implantado.

3.2. Objetivos

De sobra son conocidos los objetivos que se pretenden abarcar con un sistema de gestión de emergencias.

Observando las carencias de la mayoría de los sistemas de emergencias en el mercado actual se ha estimado que los siguientes objetivos son de necesidad para un rápido y eficiente centro de emergencias.

• Seguimiento en tiempo real desde la central de emergencias de los efectivos que atienden las emergencias (vehículos de policía, personal sanitario, bomberos, etc.).
• Gestión de las alarmas/emergencias (creación, asignación y finalización).
• Gestión de la mensajería entre el centro de emergencias y los efectivos (envío y recepción de mensajes).
• Gestión de efectivos destinados a atender las emergencias (asignación de la alarma y consiguiente enrutamiento al lugar, y finalización de la alarma).
• Disponer de un sistema de cartografía capaz de guiar los efectivos a edificaciones o lugares que no permiten los sistemas convencionales.
• Disponer de información adicional del lugar en donde se provoca la emergencia.

4. Especificación de requisitos

En este capítulo detallamos los requisitos que hacen posible alcanzar los objetivos presentados en el punto anterior. Se pretende dar una visión del entorno donde está implantada la aplicación y de los actores que forman parte del sistema. Esto junto con las acciones que pueden llevar a cabo en el mismo nos dará una visión global del sistema y sus funcionalidades.

Un centro de emergencias es un entorno complejo y amplio en el que intervienen diversos actores por lo que se debe distinguir bien los mismos así como los entornos en los que se encuentran ubicados.

A continuación, pasamos a identificar los actores que intervienen en el sistema, para posteriormente ubicar a cada uno de ellos en el entorno al que pertenece y ver las acciones que pueden llevar a cabo sobre el mismo.

4.1. Identificación de Actores

Como una imagen vale más que mil palabras, se muestra la siguiente figura, en la cual se aprecian cada uno de los actores que hay en el sistema, y algunas de las tecnologías de las que hacen uso.

En la parte izquierda de la imagen, y de abajo hacia arriba, tenemos al programador que es el encargado de implementar y diseñar la aplicación partiendo de los requisitos establecidos previamente por el cliente; inmediatamente después observamos al ciudadano que sería el que se pondría en contacto con el operador del centro de emergencias para comunicarle la alarma u emergencia.

El operador del centro de emergencias sería el encargado de llevar a cabo todo el proceso de asignación de la alarma y los efectivos pertinentes.

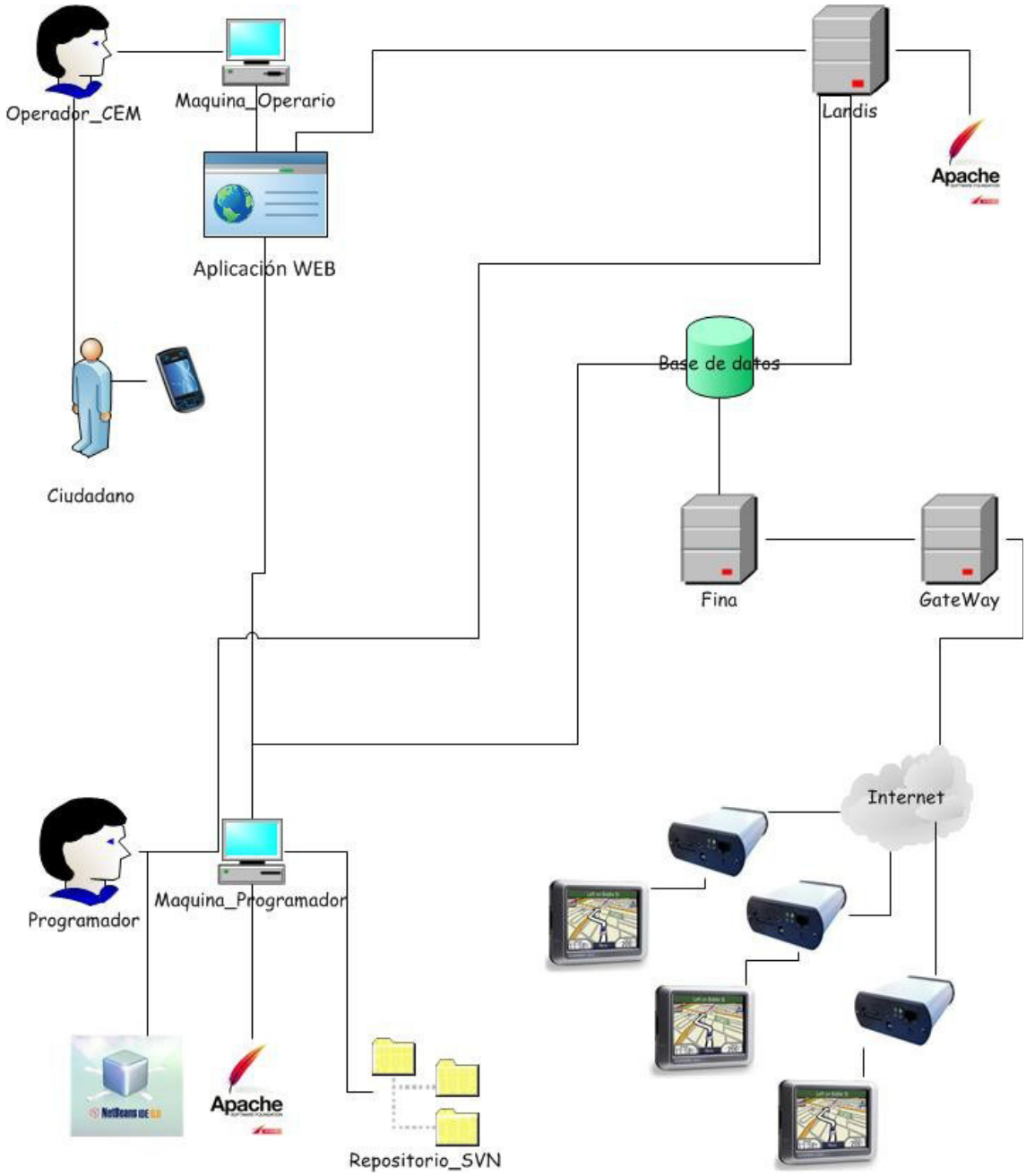


Figura 1: Identificación de los actores

Otros actores de tipo hardware que intervienen en el sistema son los servidores como Landis, en donde se encuentran los servicios del servidor Apache[1] y PHP[2] junto con los de cartografía WMS[3] para el correcto funcionamiento de la aplicación.

En el servidor Fina disponemos del software específico que sirve para almacenar y extraer la información referente a alarmas, vehículos, emergencias, y transmitirla al servidor Gateway.

Gateway es un servidor que dispone de conexión a Internet y se encarga de comunicar los dispositivos RTCU[4] con el servidor Fina y así garantizar que tanto los datos que se envían como los que se reciben quedan debidamente almacenados en la base de datos.

4.2. Identificación de entornos

Al identificar los distintos entornos existentes en la aplicación observamos una clara división en cuanto al entorno de **producción** propiamente dicho y el entorno de **desarrollo.** En la siguiente imagen pasamos a ver con mayor detalle cada uno de ellos y analizar las peculiaridades de los mismos.

4.2.1. Entorno de producción

En el entorno de producción se observan diversos actores los cuales intervienen en el sistema, algunos de forma directa y otros de forma indirecta.

En primer lugar, como el ciudadano que es quien origina todo el flujo de la información, al ponerse en contacto con el operador del centro de emergencia y alertar sobre una emergencia. Este hecho hace que se ponga en marcha todo el plan de actuación de emergencia.

Al recibir la alerta del ciudadano el operador del CEM[5] consulta la aplicación web en busca de la posición en donde se originó la emergencia y de información que pueda ser de utilidad para solucionar el problema. En ese momento se transmite información al servidor Landis que procesa las peticiones del operario y envía información a través del Gateway a los dispositivos de GPS[6] implantados en los efectivos encargados de atender las emergencias.

---

1 **Apache**: Es un servidor web HTTP de código abierto que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.(http://www.apache.org/)
2 **PHP**: Es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas (http://www.php.net/)
3 **WMS:** Es un servicio que produce mapas de datos referenciados espacialmente, de forma dinámica a partir de información geográfica.
4 **RTCU**: Unidad Remota de telemetría y control permite el rápido desarrollo de las aplicaciones especificas del cliente combinando localización GPS y recogida de datos con avanzadas técnicas de comunicación tales como envío de mensajes a la unidad y desde ella.(http://www.logicio.com/)
5 **CEM:** Centro de Emergencias.
6 **GPS**: Sistema de posicionamiento Global, es un sistema global de navegación por satélite que permite determinar en todo el mundo la posición de un objeto, una persona, un vehículo o una nave, con una precisión hasta de centímetros (si se utiliza GPS diferencial), aunque lo habitual son unos pocos metros de precisión, en nuestra aplicación es utilizado para establecer el punto exacto de la emergencia y así guiar por el camino más corto a los efectivos hasta el lugar.

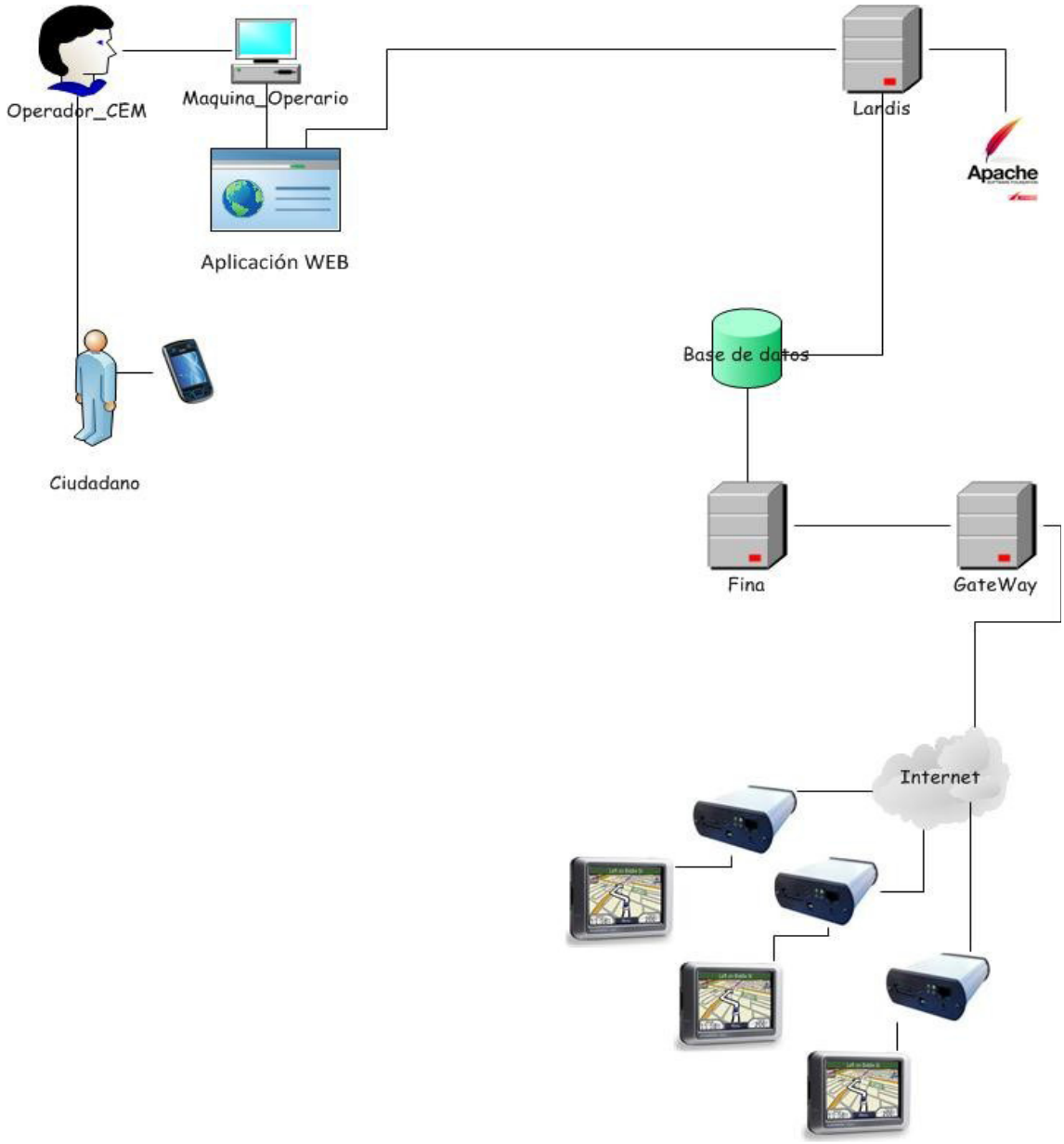


Figura 2: Entorno de producción

No hay que olvidar que tanto los dispositivos GPS, como los RTCU, están constantemente enviando información acerca de la posición, tiempo estimado de llegada, ruta seguida, etc... Información la cual se almacena en la base de datos y se refresca en la aplicación WEB para tener un seguimiento en tiempo real de los vehículos y de las alarmas.

4.2.2. Entorno de Desarrollo

El entorno de desarrollo es muy parecido al entorno de producción, con la peculiaridad de que en este el programador es el responsable de las interacciones que tienen lugar en el sistema.

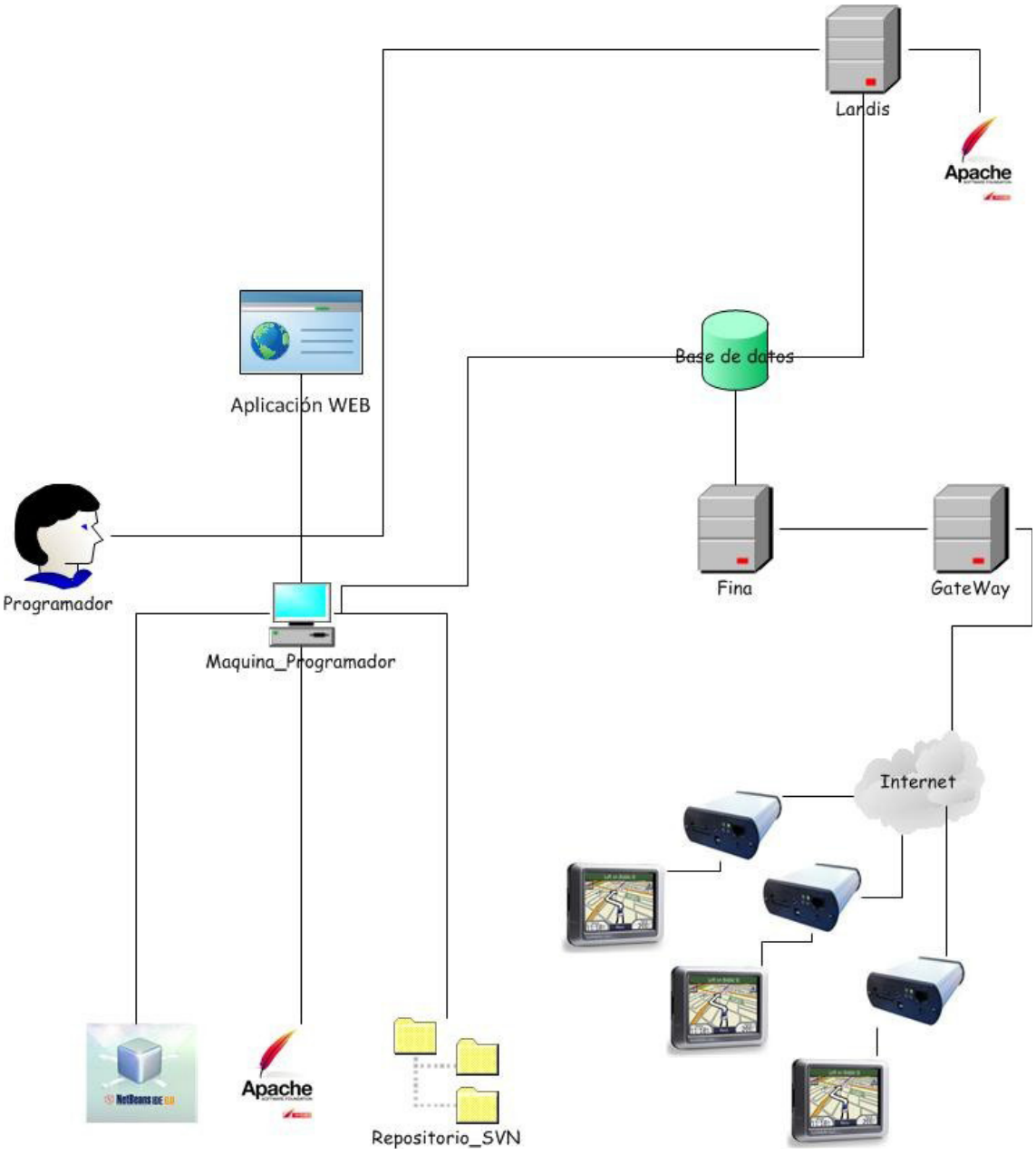


Figura 3: Entorno de desarrollo

En la misma máquina se dispone de un servidor Apache ejecutándose en local y de un entorno de desarrollo NetBeans[7]. Además los cambios que se efectúan en el proyecto se almacenan en un repositorio de SVN[8], y una vez comprobado su correcto funcionamiento en la maquina se sincroniza con el servidor Landis en producción.

SUBVERSION

Es un sistema de control de versiones diseñado específicamente para reemplazar al popular CVS[9]. Es un software libre bajo una licencia de tipo Apache y se le conoce también por SVN por ser el nombre de la herramienta utilizada en la línea de órdenes.

Una característica importante de SubVersion es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en un instante determinado.

SubVersion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar más rápidamente sin un único conducto por el cual deban pasar todas las modificaciones. Y puesto que el trabajo se encuentra bajo el control de versiones, no hay razón para temer porque la calidad del mismo vaya a verse afectada si se ha hecho un cambio incorrecto a los datos, simplemente deshaga ese cambio.

Ventajas de un sistema SVN:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Puede ser servido mediante Apache.
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).

---

7 **NetBeans:** Es una plataforma de código abierto que permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes software llamados *módulos*. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. (http://netbeans.org/)

8 **SVN**: **SUBVERSION** es un sistema de control de versiones. Es un software libre bajo una licencia de tipo Apache/BSD y se le conoce también por SVN por ser el nombre de la herramienta utilizada en la línea de órdenes. (http://subversion.tigris.org/)

9 **CVS**: El **Concurrent Versions System** (CVS), es una aplicación informática que implementa un sistema de control de versiones lo cual mantiene el registro de todo el trabajo y los cambios en los ficheros. (http://www.cvshome.org/)

• Permite seleccionar el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.
• Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PALM, etc.)

Si observamos la estructura de árbol de SubVersion podemos distinguir varios directorios de entre los que cabe destacar:

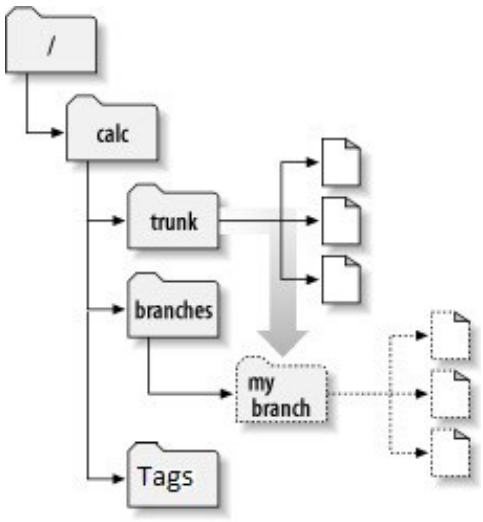


Figura 4: SubVersion

• El repositorio en sí mismo, raíz de la estructura que se va creando.

• Conforme bajamos nos encontramos con los Branches, Tags, y Trunk.

• Un Branch es una rama de desarrollo normalmente utilizada para seguir una línea de desarrollo distinta a la que se sigue en Trunk. Un mismo repositorio de SVN puede tener varios Branches, incluso se puede crear un Branch en un momento dado para tener como una copia hasta un punto dado, aunque esto no suele hacerse y que precisamente la filosofía del SVN está pensada para que en caso de que surja algún error o dificultad se pueda volver a una versión anterior de la aplicación.

• El directorio de Tags sirve para poder etiquetar las distintas versiones de la aplicación que vayamos desarrollando, y así ser capaz de poder volver a cualquiera de estas versiones que hemos etiquetado.

• El directorio principal es el Trunk, en el que tendremos todos los ficheros y las correspondientes revisiones de cada uno de ellos ordenados por revisiones que se han ido guardando.

4.3. Acciones

En este punto nos centramos en las acciones que se pueden llevar a cabo en el sistema, así pues pasamos a listar cada una de las acciones, de entre las cuales cabe destacar:

- El **Operador desde la aplicación** debe ser capaz de:

  - Dar de alta y baja alarmas o emergencias.
  - Asignación/Des-asignación los vehículos destinados a emergencias.
  - Seguimiento del estado de las alarmas en tiempo real.
  - Seguimiento en tiempo real de los vehículos destinados a emergencias.
  - Envío/recepción de mensajes entre el vehículo y el centro de emergencias.
  - Navegación por el municipio en vista callejero/Ortofoto[10].
  - Búsqueda de edificaciones y lugares de interés.
  - Posicionamiento directo a raíz de una búsqueda (ir a mapa).

Por otra parte, el **sistema** permite de forma directa o indirecta las siguientes acciones:

- Envío de información entre el vehículo de emergencia y la aplicación.
- Almacenamiento y restauración de los datos en un repositorio SVN.
- Almacenamiento y tratamiento de la información de la Base de Datos en Postgres[11].
- Sincronizar los datos entre la máquina de desarrollo y producción con Rsync[12].

4.4. Interacción entre Actores y sistema

En este apartado pasamos a detallar mediante diagramas de interacción, las consecuencias que tiene la interacción de los distintos actores que conforman el sistema por ello analizamos tanto actores humanos como Hardware y Software.

En el siguiente diagrama se observa la interacción del operario del sistema de gestión de emergencias al enviar y recibir mensajes.

---

10 **Ortofoto**: Es una presentación fotográfica de una zona de la superficie terrestre, en la que todos los elementos presentan la misma escala, libre de errores y deformaciones, con la misma validez de un plano cartográfico.

11 **Postgre**: Es un sistema de gestión de base de datos relacional orientada a objetos y libre. (http://www.postgresql.org/)

12 **Rsync**: Es una aplicación de software para sistemas de tipo Unix que ofrece transmisión eficiente de datos incrementales Compresión de datos comprimidos y cifrados. Mediante una técnica de delta encoding, permite sincronizar archivos y directorios entre dos máquinas de una red o entre dos ubicaciones en una misma máquina, minimizando el volumen de datos transferidos. (http://samba.anu.edu.au/rsync/)

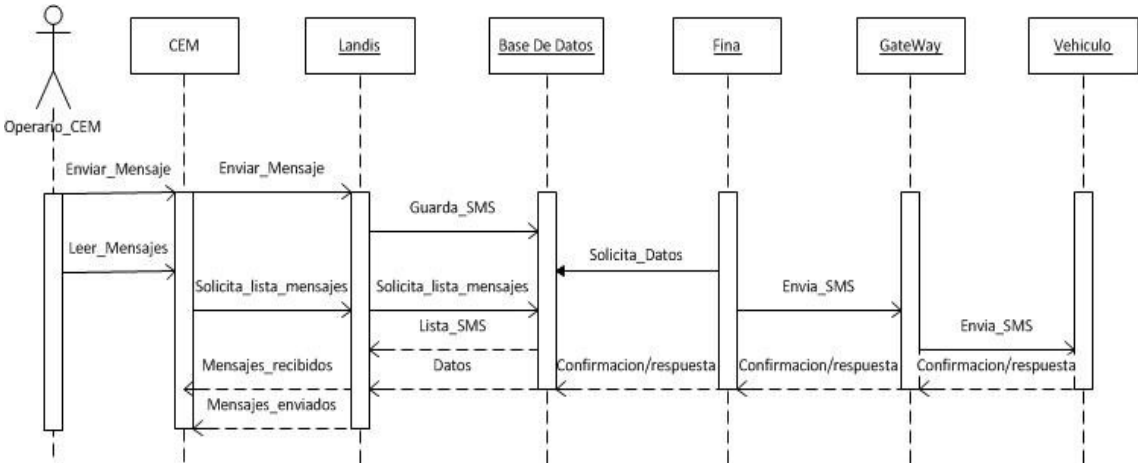


Figura 5: Diagrama interacción; creación y asignación de una alarma

En este otro diagrama se observa la actuación por parte del operario del CEM de la búsqueda de edificaciones y lugares de interés, así como posicionamiento directo en el mapa.

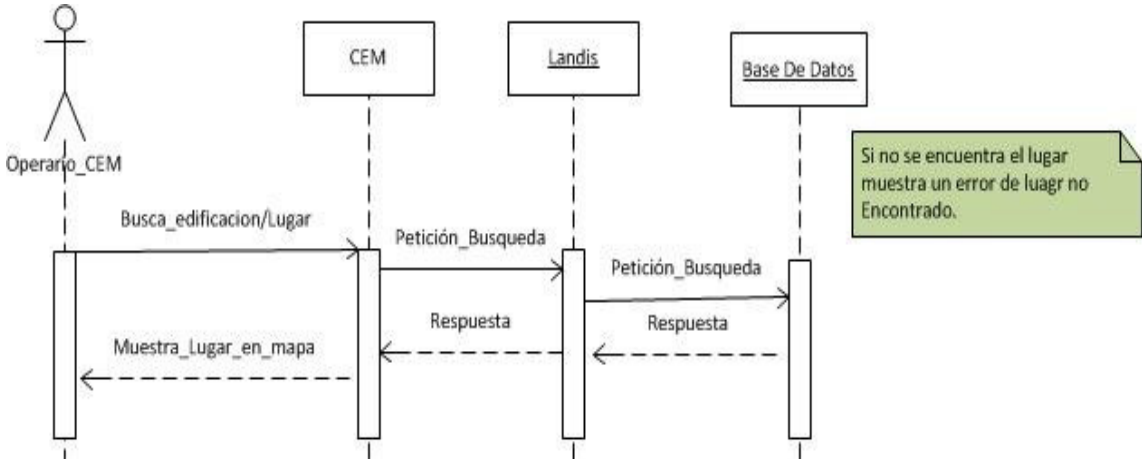


Figura 6: Diagrama de interacción por parte del operador

En siguiente diagrama de interacción podemos observar el proceso de comunicación de una alarma por parte de un ciudadano, y el tratamiento que le da a la alarma el operario, además de todas las posibles actuaciones que se pueden llevar a cabo por el operario del CEM, tales como la creación de la alarma, finalización, asignación, selección de vehículo, etc...

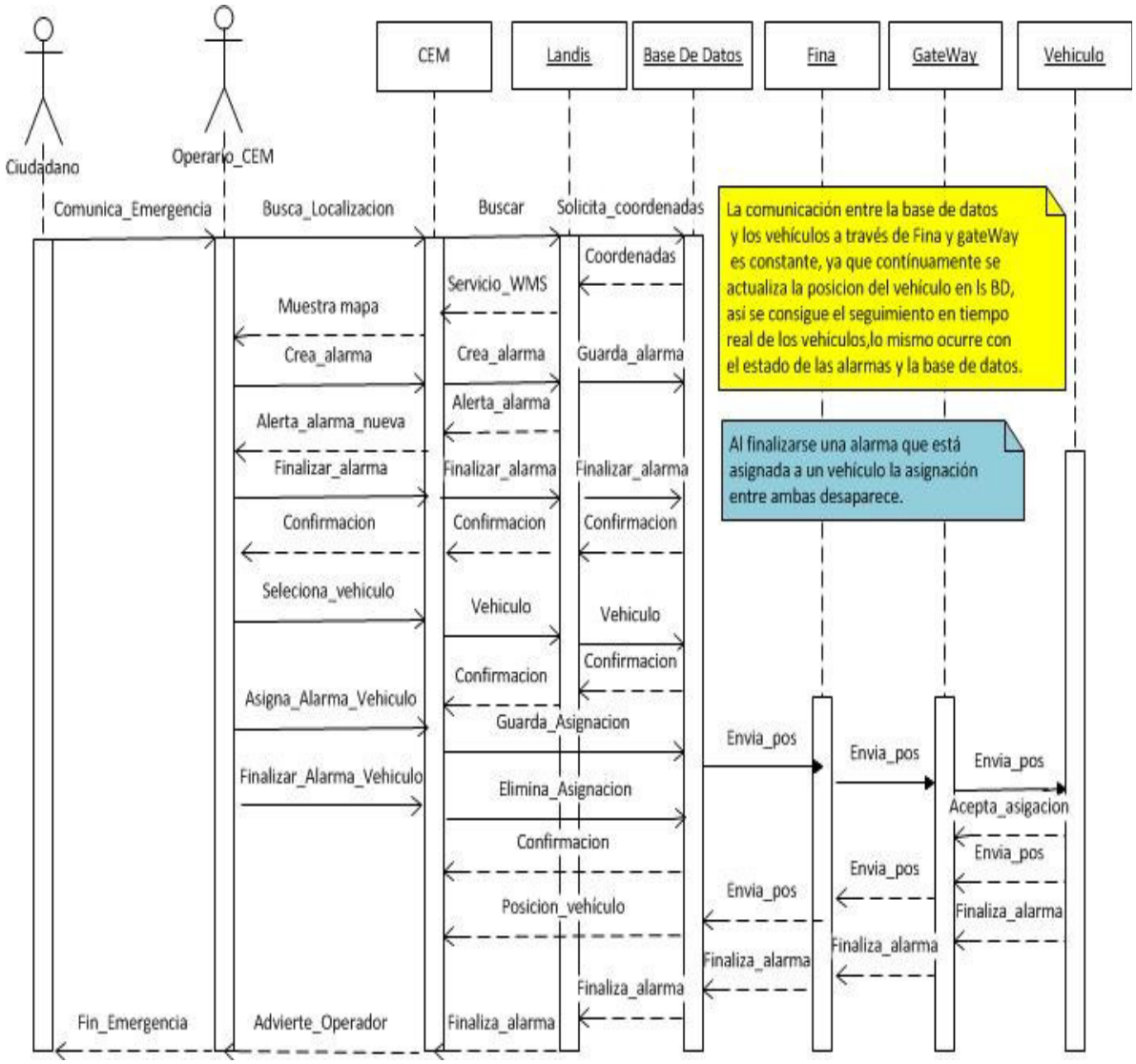


Figura 7: Diagrama de interacción emergencia

En estos tres diagramas de interacción se detallan las principales acciones que pueden llevar a cabo los actores para con el sistema, aunque entre los servidores haya más interacciones que no quedan totalmente reflejadas en estos diagramas.

5. Arquitectura

Tras definir los objetivos principales de la aplicación y los requisitos para que estos se lleven a cabo, en este apartado pasamos a la presentación de la arquitectura elegida para dar solución al entorno elegido, lo cual nos dará una mayor visión del Software y del Hardware empleado para disponer de un sistema con las mejores prestaciones.

5.1. Visión Global

En primer lugar, vamos dar una visión global, especificando con mayor detalle, el Hardware y Software empleados para la solución del sistema, y con ello hacer posible que se cumplan los requisitos demandados.

Se pueden observar dos entornos bien diferenciados en el sistema, por una parte está el entorno de **desarrollo** y por otra el entorno de **producción**, así mismo dentro de cada uno de ellos se puede distinguir una arquitectura a tres capas en donde quedan separadas, la capa de presentación, la capa de lógica, y la capa de persistencia o datos.

Una de las grandes ventajas de tener una arquitectura a tres capas bien diferenciada es que el desarrollo puede llevarse a cabo en varios niveles, y en caso de que sobrevenga algún cambio, solo hay que atacar el nivel requerido sin tener que revisar todo el código.

Además, permite distribuir el trabajo de creación de una aplicación por niveles; de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API[13] que existe entre niveles para pasar de uno a otro.

A continuación, pasamos a ver con mayor detalle el porqué de esta elección, y conocer, las partes especificadas anteriormente.

13 **API:** Una interfaz de programación de aplicaciones o API es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Usados generalmente en las bibliotecas.

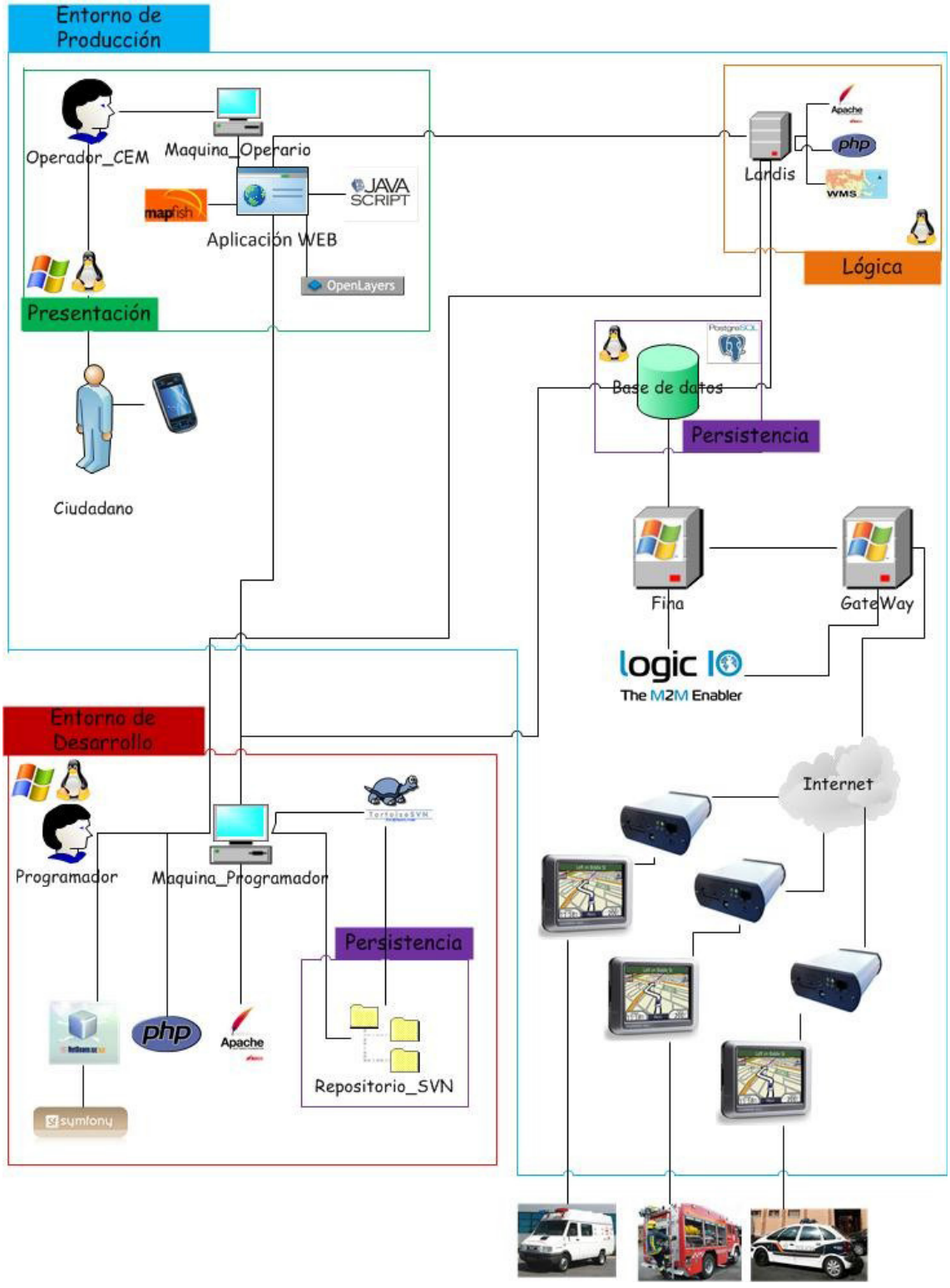


Figura 8: Visión global de la arquitectura

LEYENDA

Mapfish, es un framework para manejo de mapas via web.

El servicio WMS produce mapas de datos referenciados espacialmente, de forma dinámica a partir de información geográfica.

JavaScript, es un lenguaje de scripting basado en objetos no tipado y liviano, utilizado para acceder a objetos en aplicaciones

OpenLayers es una biblioteca javascript que nos permite elaborar nuestros mapas haciendo uso de su propia base de información cartográfica.

PostgreSQL, es un sistema gestión de base de datos relacional orientada a objetos

Sistema Operativo Windows

Sistema Operativo Windows o Linux

Sistema Operativo Linux

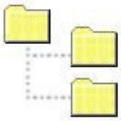
El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix y Winodws, que implementa el protocolo HTTP/1.11 y la noción de sitio virtual.

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas.

Software de Logic IO, para comunicar los equipos

Terminal movil

Tortoise, es un cliente de Subversion para windows



Repositorio_SVN

Repositorio de subversion

Es un plataforma de desarrollo la cual permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos.

Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características

Se trata de una RTCU, unidad remota de telemetría

GPS, de la marca GARMIN interconectado a la RTCU

Figura 9: Leyenda

5.2. Arquitectura del Entorno

Tras disponer de una clara visión global de la arquitectura empleada, damos paso a hablar con mayor detalle de la arquitectura del entorno tal y como se observa en la imagen anterior.

Se puede distinguir entre dos partes que engloban todo el conjunto, la visión del entorno de desarrollo y el entorno de producción.

5.2.1. Entorno de desarrollo

En el entorno de desarrollo tal y como podemos observar en la siguiente imagen, el principal actor involucrado es el **programador**, el cual bajo un entorno Windows[14] o Unix/Linux[15], utiliza el software necesario para trabajar en local y hacer las pruebas de software pertinentes en su máquina antes de subir los cambios al servidor de producción Landis.

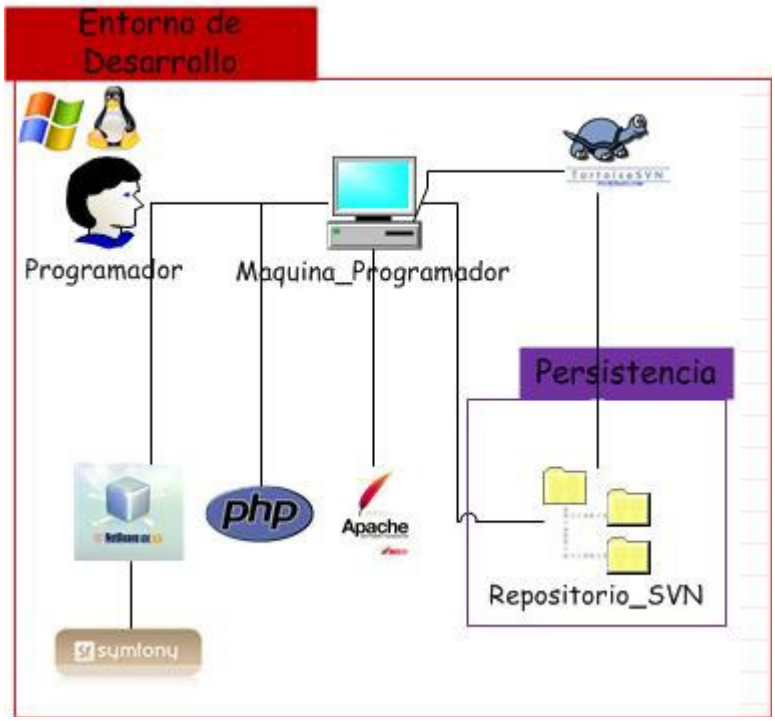


Figura 10: Entorno de desarrollo

14 **Windows**: Son los nombres de los sistemas operativos desarrollados por microsoft desde el año 1981. (http://www.microsoft.com/spain/Windows/)

15 **Unix**/**Linux**: Términos empleados para referirse a la combinación del núcleo o kernel libre.(http://www.linuxes.org/)

En cuanto al **Hardware** utilizado para la máquina del programador, no se precisa de ningún Hardware específico ya que cualquiera de los ordenadores que hay hoy en día son capaces de sobra para soportar las aplicaciones que se detallarán a continuación.

Sin embargo y para que quede constancia se proporcionan los principales componentes Hardware que componen esta máquina:

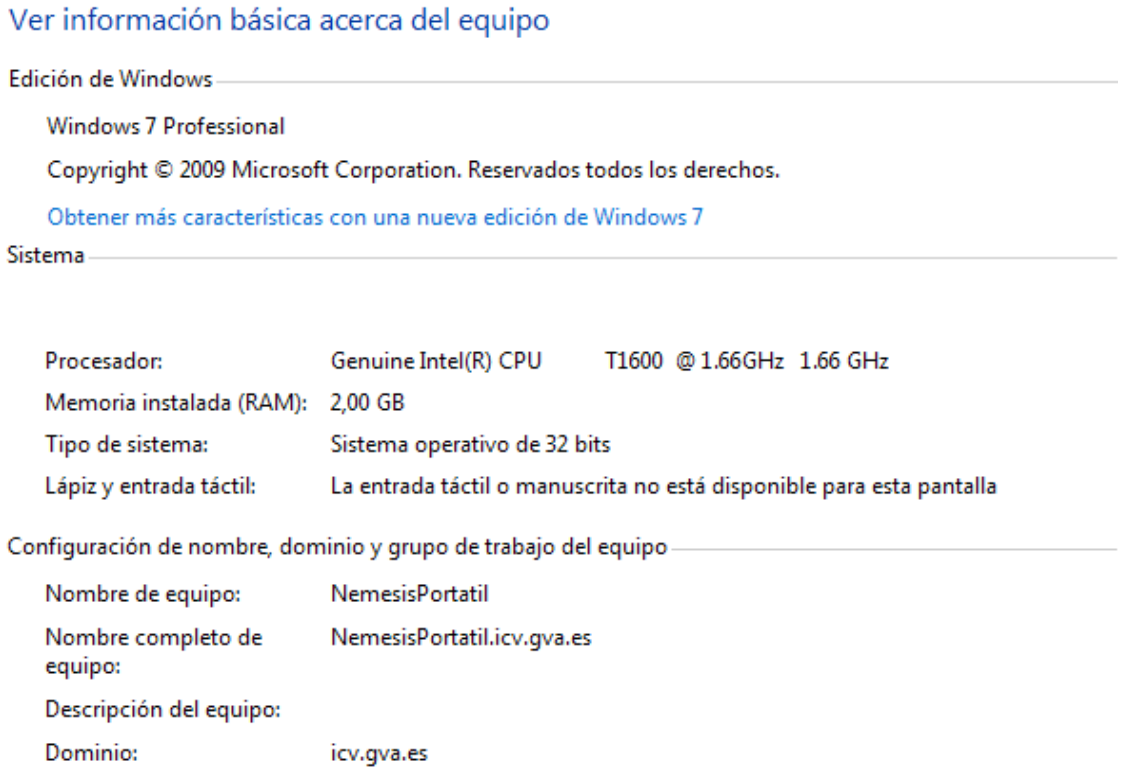


Figura 11: Hardware máquina desarrollo

El Software utilizado en la máquina del programador es:

- **Servidor Apache + PHP:** El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.),Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

  El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache software Foundation. Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

  De entre las ventajas que goza el servidor apache podemos destacar:
  • Multi-plataforma
  • Extensible
  • Popular (fácil conseguir ayuda/soporte)
  • Modular

• Código Abierto

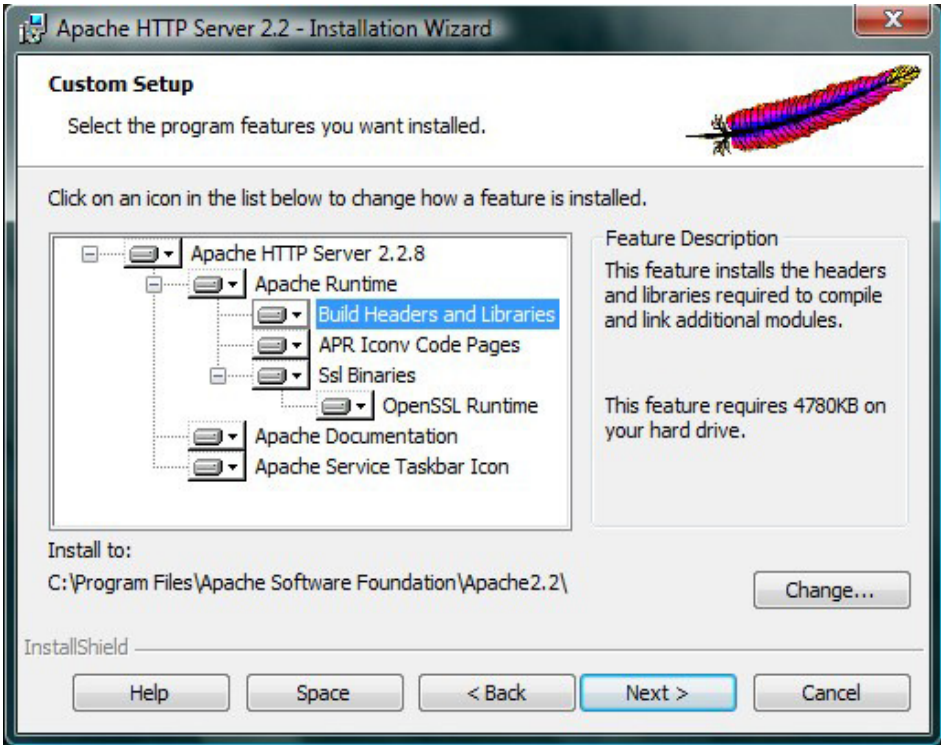


Figura 12: Instalador Apache

- **Cliente SVN Tortoise**[16]

  Tortoise SVN es un cliente gratuito de código abierto para el sistema de control de versiones Subversión. Esto es, Tortoise SVN maneja ficheros y directorios a lo largo del tiempo.

  Los ficheros se almacenan en un repositorio central. El repositorio es prácticamente lo mismo que un servidor de ficheros ordinario, salvo que recuerda todos los cambios que se hayan hecho a sus ficheros y directorios. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos, y quién hizo el cambio.

16 **Tortoise**: Es un cliente SubVersion para Windows. Esta liberado bajo licencia GNU GPL. (http://tortoiseSVN.net/downloads)

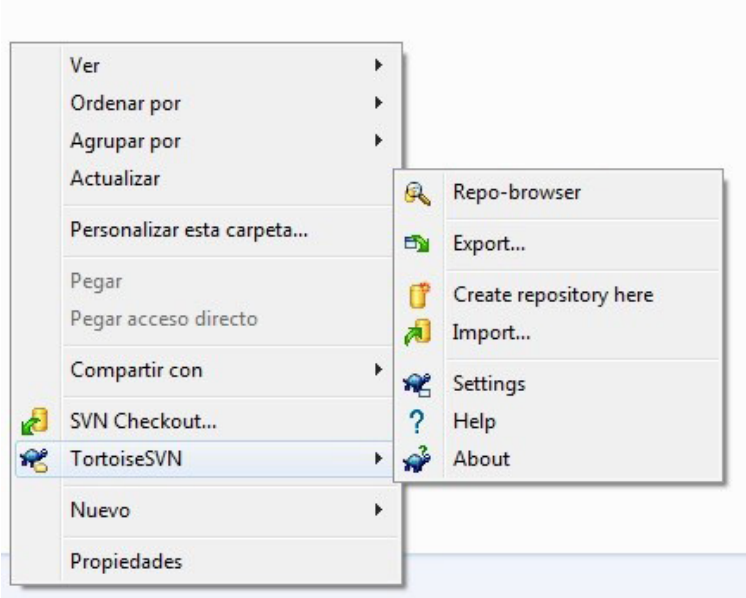


Figura 13: SVN: Tortoise

- **NetBeans + Symfony**

  La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

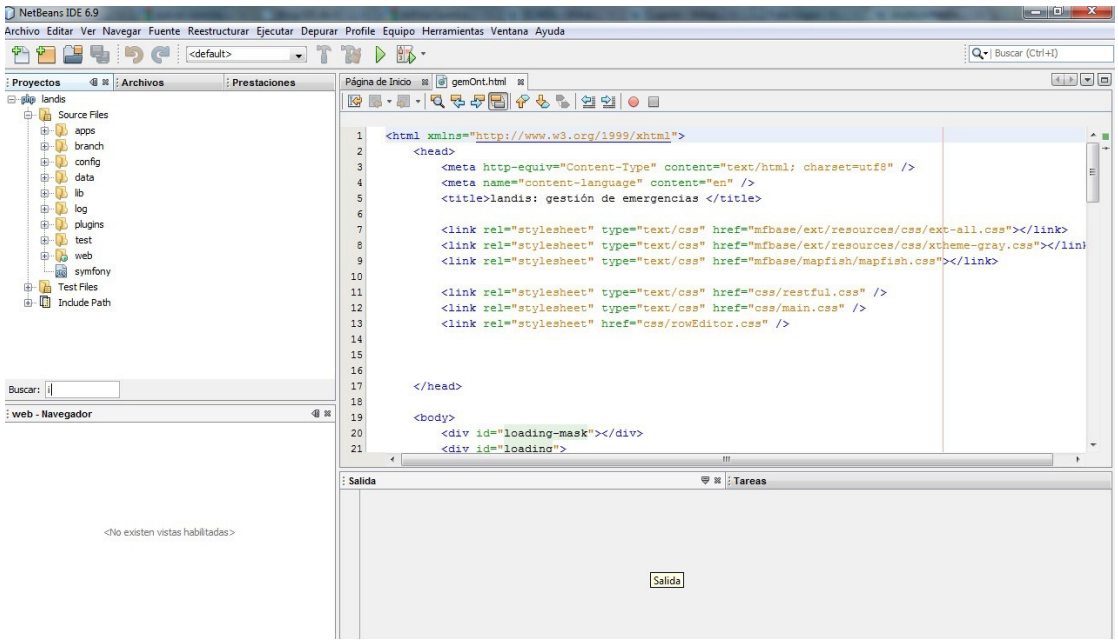


Figura 14: Entorno desarrollo NetBeans

Observamos una captura de nuestro entorno de desarrollo NetBeans sobre el cual tenemos instalado el framework de Symfony.

Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente en PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL[17], PostgreSQL, Oracle[18]. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y Unix estándares).

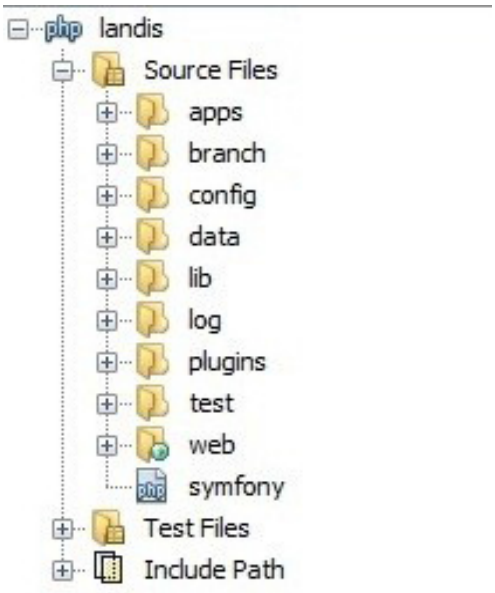


Figura15: Entorno Symfony (Estructura de datos en un entorno Symfony)

17 **MySQL**: Es un sistema gestor de bases de datos relacional, multihilo y multiusuario.(http://www.mysql.com/)

18 **Oracle**: Es un sistema gestor de bases de datos relacional desarrollado por Oracle corporation. (http://www.oracle.com/index.html)

Entre sus características cabe destacar:

- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de *"convenir en vez de configurar"*, en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de *mejores prácticas* y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

El utilizar un framework como symfony tiene sus ventajas ya que nos permite integrar módulos que serán necesarios para el correcto funcionamiento de la aplicación como serían el modulo que integra para MapFish.

MapFish es un framework para manejo de mapas vía Web. Está compuesto de un servidor hecho en Phyton, con compatibilidad con PHP; y de un cliente Javascript que une EXTjs[19], OpenLayers[20], GeoExt[21], además de objetos propios.

El plug-in de MapFish que tenemos integrado en nuestro framework de symfony nos permite tener una mayor interacción con la base de datos y el código en PHP de nuestra aplicación.

---

19 **Extjs:** Sencha (anterior ExtJS)es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Originalmente construida como una extensión de la biblioteca YUI, en la actualidad puede usarse como extensión para las bibliotecas jQuery y Prototype. (http://www.sencha.com/)

20 **OpenLayers:** OpenLayers es una librería de JavaScript de código abierto bajo una derivación de la licencia BSD para mostrar mapas interactivos en los navegadores web.(http://openlayers.org/)

21 **GeoExt**: La librería GeoExt es una herramienta utilizada para construcción de aplicaciones web de mapas basada en javascript. Es una herramienta OpenSource e incluye una gran variedad de ejemplos de como utilizar la librería.(http://www.geoext.org/)

OpenLayers es una librería de JavaScript[22] de código abierto bajo una derivación de la licencia BSD para mostrar mapas interactivos en los navegadores web. OpenLayers ofrece un API para acceder a diferentes fuentes de información cartográfica en la red: Web Map Services, Mapas comerciales (tipo Google Maps, Bing, Yahoo), Web Features Services[23], distintos formatos vectoriales, mapas de OpenStreetMap, etc.

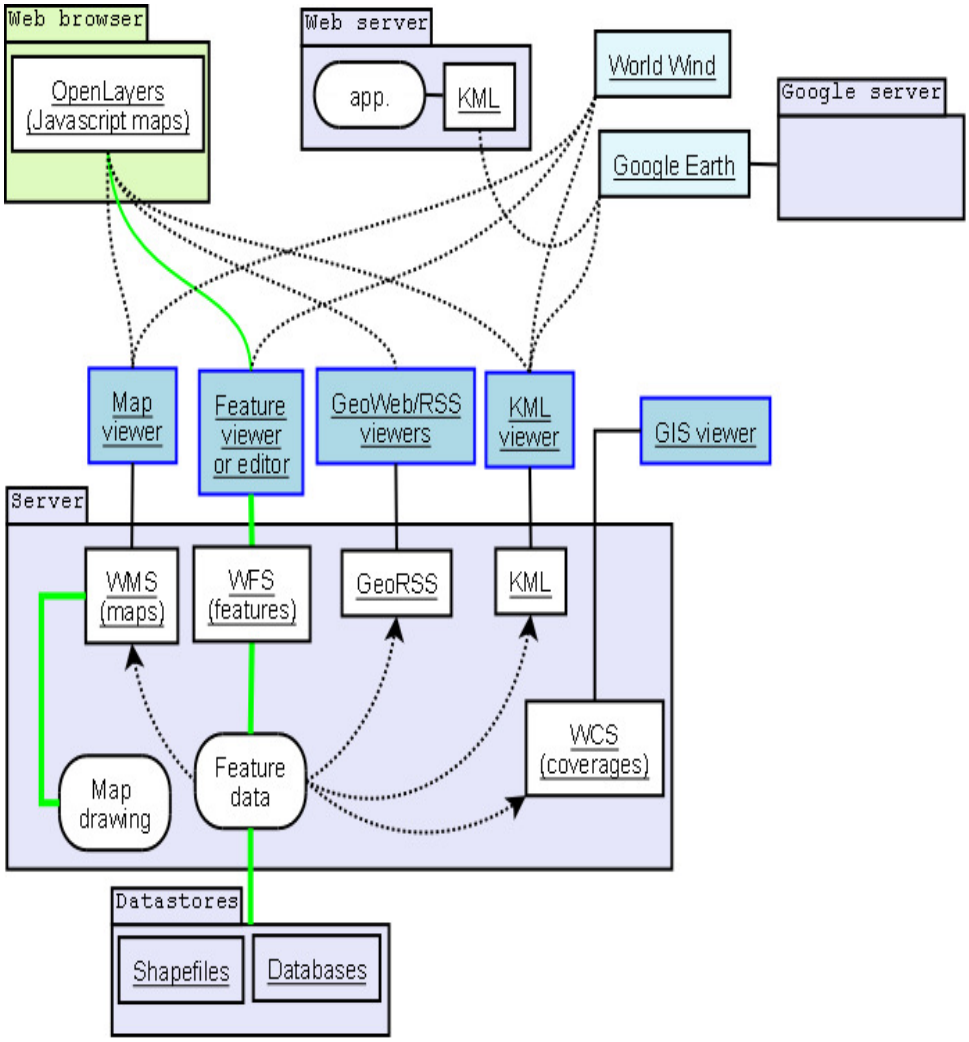


Figura 16: OpenLayers

22 **JavaScript**: Es un lenguaje de scripting basado en objetos no tipeado y liviano, utilizado par acceder a objetos en aplicaciones(http://www.javascript.com/)

23 **Web Feature Service** o **WFS**: es un servicio estándar, que ofrece una interfaz de comunicación que permite interactuar con los mapas servidos por el estándar WMS.

5.2.2. Entorno de Producción

En el entorno de Producción cabe distinguir tres partes principales: por una parte estarían los servidores encargados de la capa de lógica de negocio donde tenemos (Landis, Fina y Gateway), y por otra parte, la capa de persistencia a la que pertenecería la base de datos y el repositorio que, aunque esté situado en el entorno de desarrollo, proporciona persistencia a los datos y distintas versiones que se van desarrollando de la aplicación.

Y por último, pero no menos importante, nos encontramos con la capa de presentación, en la que se encuentra la interfaz de usuario.

5.2.2.1 Modelo o lógica de negocio.

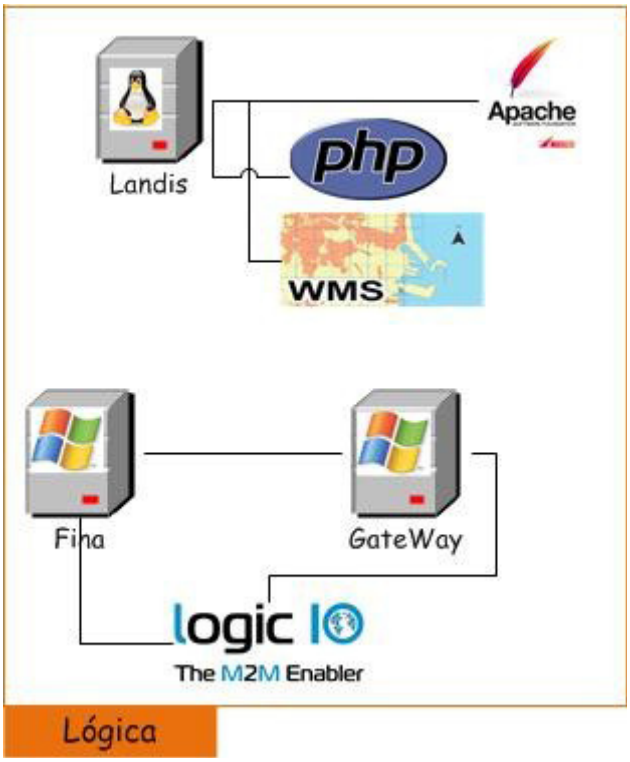


Figura 17: Lógica de negocio

En esta capa de negocio encontramos tres servidores los cuales vienen detallados en la figura 17. Por lo que se refiere a requerimientos tanto de software como de hardware disponen de las siguientes configuraciones:

- **Landis:**

  Se trata de un servidor Linux con los siguientes componentes Hardware:

Sistema Operativo Ubuntu 7.10	Intel® Core™ i3 2,93 Ghz	Ram=4 GB	Hdd=5TB

El software del que dispone el servidor Landis viene detallado a continuación:

-Servicios WMS para proporcionar la cartografía a la aplicación CEM.

El servicio Web Map Service (WMS) definido por el OGC (Open Geospatial Consortium) produce mapas de datos referenciados espacialmente, de forma dinámica a partir de información geográfica. Este estándar internacional define un "mapa" como una representación de la información geográfica en forma de un archivo de imagen digital conveniente para la exhibición en una pantalla de ordenador. Un mapa no consiste en los propios datos. Los mapas producidos por WMS se generan normalmente en un formato de imagen como PNG, GIF o JPEG y opcionalmente como gráficos vectoriales en formato SVG (Scalable Vector Graphics).

El estándar define tres operaciones:

-Devolver metadatos del nivel de servicio.

-Devolver un mapa cuyos parámetros geográficos y dimensionales han sido bien definidos.

-Devolver información de características particulares mostradas en el mapa (opcionales).

Las operaciones WMS pueden ser invocadas usando un navegador estándar realizando peticiones en la forma de URLs (Uniform Resource Locators). El contenido de tales URLs depende de la operación solicitada. Concretamente, al solicitar un mapa, la URL indica qué información debe ser mostrada en el mapa, qué porción de la tierra debe dibujar, el sistema de coordenadas de referencia, y la anchura y la altura de la imagen de salida. Cuando dos o más mapas se producen con los mismos parámetros geográficos y tamaño de salida, los resultados se pueden solapar para producir un mapa compuesto. El uso de formatos de imagen que soportan fondos transparentes (Gif o PNG) permite que los mapas subyacentes sean visibles. Además, se puede solicitar mapas individuales de diversos servidores.

El servicio WMS permite así la creación de una red de servidores distribuidos de mapas, a partir de los cuales los clientes pueden construir mapas a medida. Las operaciones WMS también pueden ser invocadas usando clientes avanzados SIG[24], realizando igualmente peticiones en la forma de URLs. Existe software libre, como las aplicaciones gvSIg, Kosmo y otros, que permite este acceso avanzado a la información remota, añadiendo la ventaja de poder cruzarla con información local y disponer de una gran variedad de herramientas SIG.

24 **SIG**: Un Sistema de Información Geográfica es una integración organizada de Hardware, Software y datos geográficos.

-Servidor Apache + PHP:

Al igual que en el entorno de desarrollo y en la máquina del programador, el servidor
Apache con PHP tendrá la misma configuración y requisitos que los establecidos anteriormente.

- **Fina**

Se trata de un servidor Windows XP con los siguientes componentes Hardware:

Sistema Operativo	Intel® Core™	Ram=4 GB	Hdd=250GB
Windows XP SP-3	i2 2,93 Ghz		

En el servidor Fina disponemos de un software proporcionado por el fabricante de los dispositivos Rtcu LOGIC IO, llamado GPRS Tracking el cual mediante una librería de sistema Windows llamada (vsmsgw.dll) permite la interconexión entre el Gateway y la base de datos.

Este software ha sido modificado de manera que permite que la información que recibe del Gateway la transmite a la Base de Datos en Postgres siguiendo unos criterios los cuales se verán con mayor detalle en apartados posteriores.

Si observamos la siguiente captura de la aplicación se pueden observar los distintos apartados, detallados en la imagen que hay a continuación.

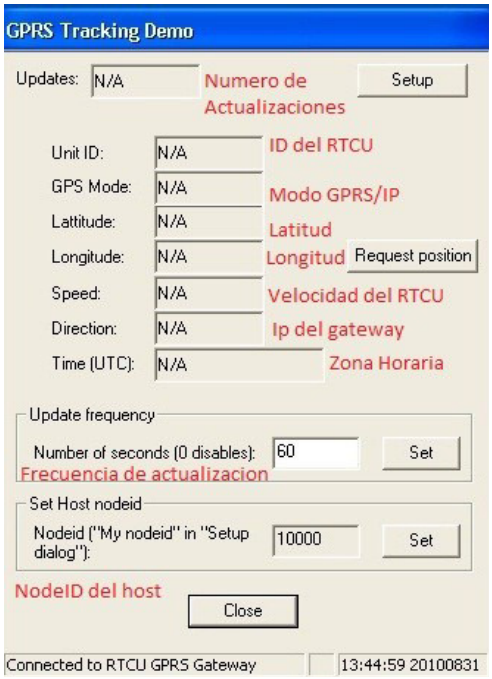


Figura 18: GPRS Tracking 1

- **Gateway**

Se trata de un servidor Windows XP con los siguientes componentes HARDWARE:

Sistema Operativo Windows XP SP-3	Intel® Core™ i2 2,93 Ghz	Ram=4 GB	Hdd=250GB

En el servidor Gateway disponemos de un software proporcionado por el fabricante de los dispositivos Rtcu LOGIC IO, llamado Rtcu Gateway monitor Tool, el cual mediante una una librería de sistema Windows llamada (vsmsgw.dll) permite la interconexión entre el Gateway y los distintos dispositivos Rtcu.

Este software ha sido modificado de manera que permite que la información que recibe del Gateway la transmite al servidor Fina siguiendo unos criterios

A continuación observamos los distintos apartados que componen la aplicación:

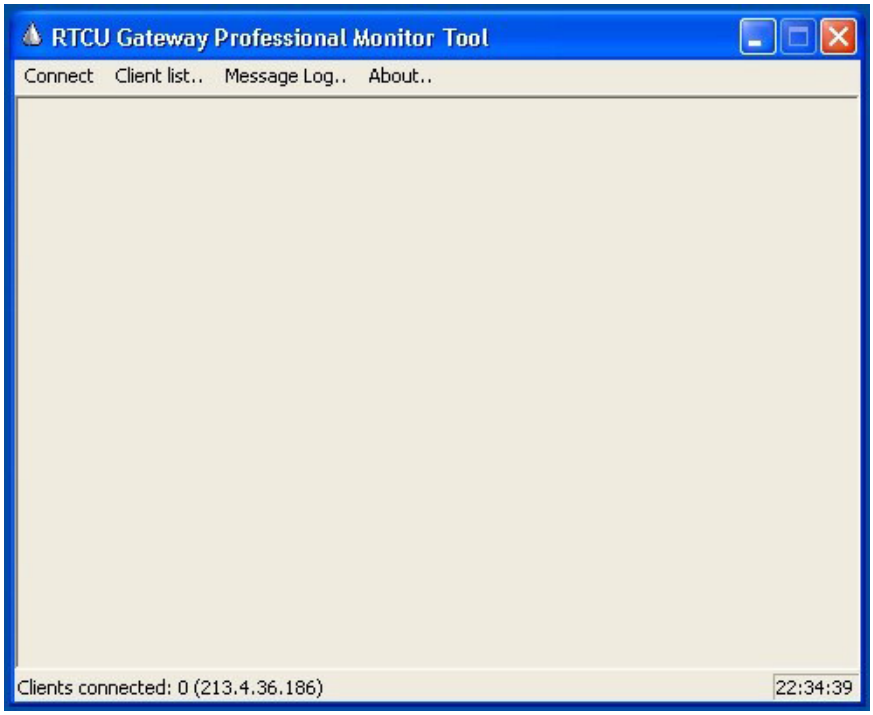


Figura 19: RTCU Gateway

Figura 20: RTCU Gateway conexión

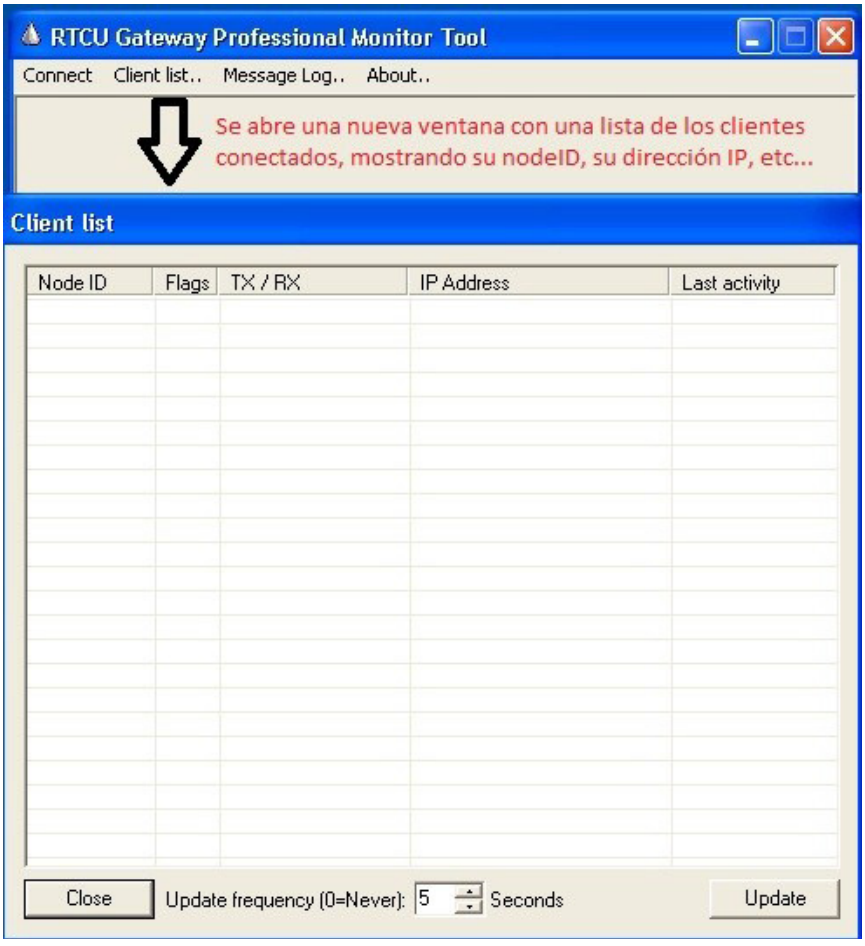


Figura 21: RTCU Gateway client list

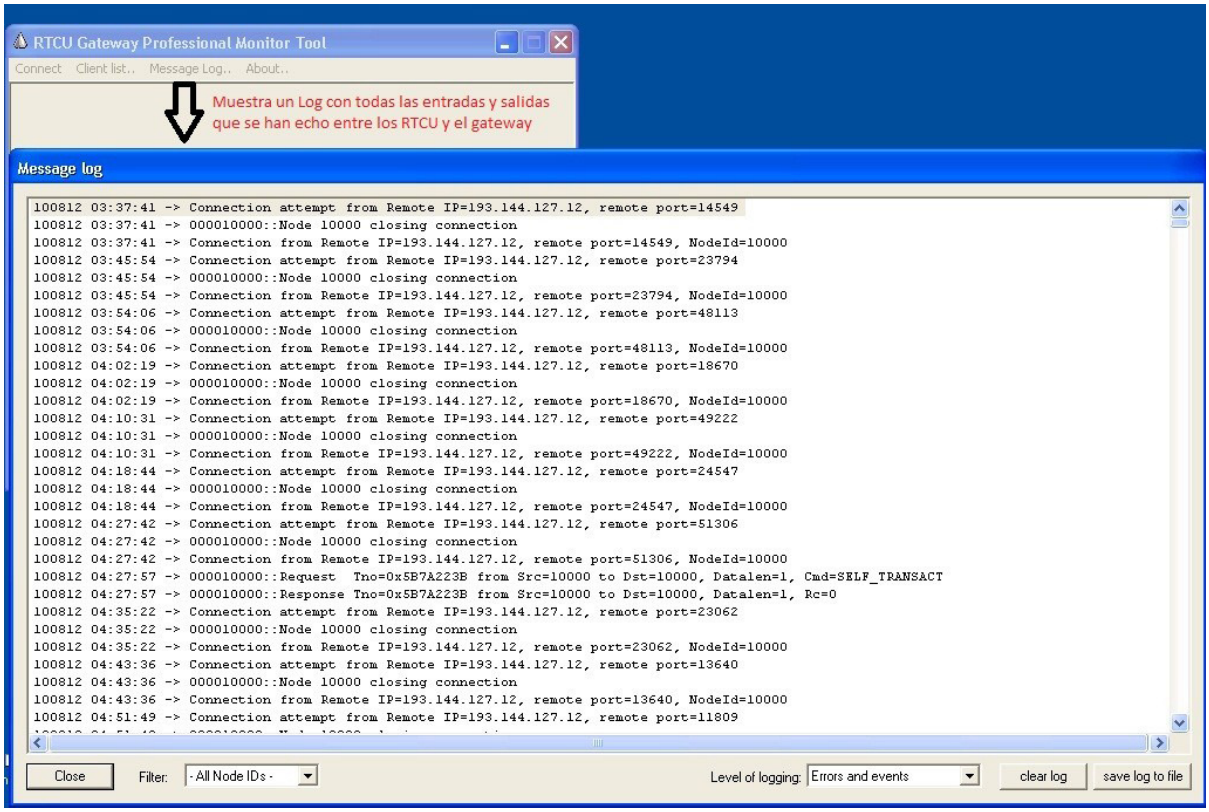


Figura 22: RTCU Gateway log

5.2.2.2. Persistencia de datos

En este apartado pasamos al análisis de la capa de persistencia, en la cual se encuentran por una parte la Base de Datos de Postgres que contiene todos los datos de la aplicación, las alarmas, vehículos, edificaciones, lugares de interés cartografía base, posicionamiento de los vehículos en todo momento, etc. y por otro el repositorio SVN que se encuentra en el entorno de desarrollo.

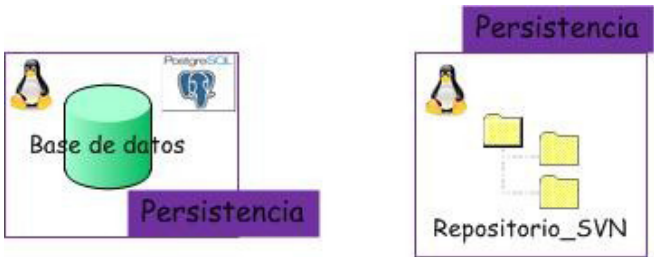


Figura 23: Persistencia

En lo que se refieres al HARDWARE del servidor Landis de Base de Datos tenemos:

Sistema Operativo Ubuntu 7.10	Intel® Core™ i3 3,10 Ghz	Ram=6 GB	Hdd=5TB

En lo referente al software ubicado en esta máquina se dispone de un servidor Postgres conectado a Landis y un Cliente pgAdmin III[25], para poder acceder y visualizar los datos de la base de datos desde una interfaz gráfica.

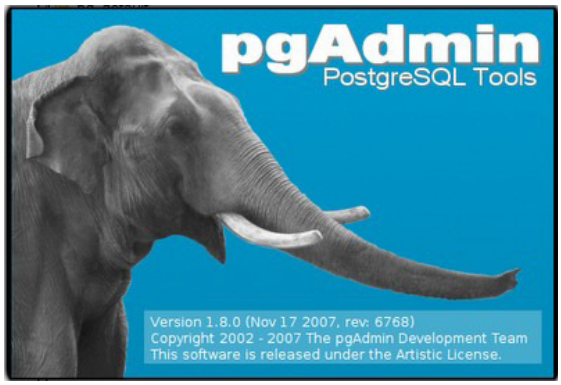


Figura 24: PgAdmin

5.2.2.3. Cliente o Interfaz de usuario

A continuación, damos paso al análisis de la interfaz de usuario en la que vemos como el operario del sistema CEM interactúa con el mismo mediante un navegador web, en principio los requisitos Hardware para esta máquina no están establecidos pero bastaría con la misma configuración de la que dispone el PC del programador que es donde ha sido probado en primera instancia.

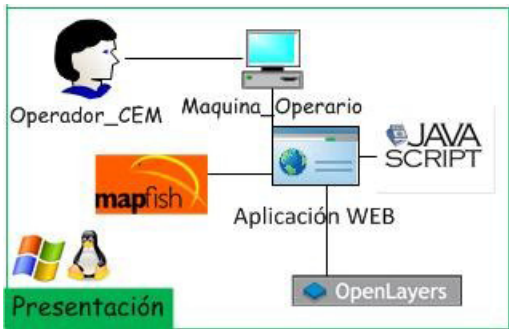


Figura 25: Presentación

25 **PgAdmin**: Es un cliente para base de datos en postgres tanto en sistemas Windows como Linux (http://www.pgadmin.org/)

El navegador puede estar situado tanto en un entorno Windows como Linux, ya que lo único importante en esta parte es que el navegador soporte JavaScript, ya que mediante JavaScript que se carga en el navegador permite la utilización de los mapas con OpenLayers y el plug-in MapFish integrado en nuestra aplicación.

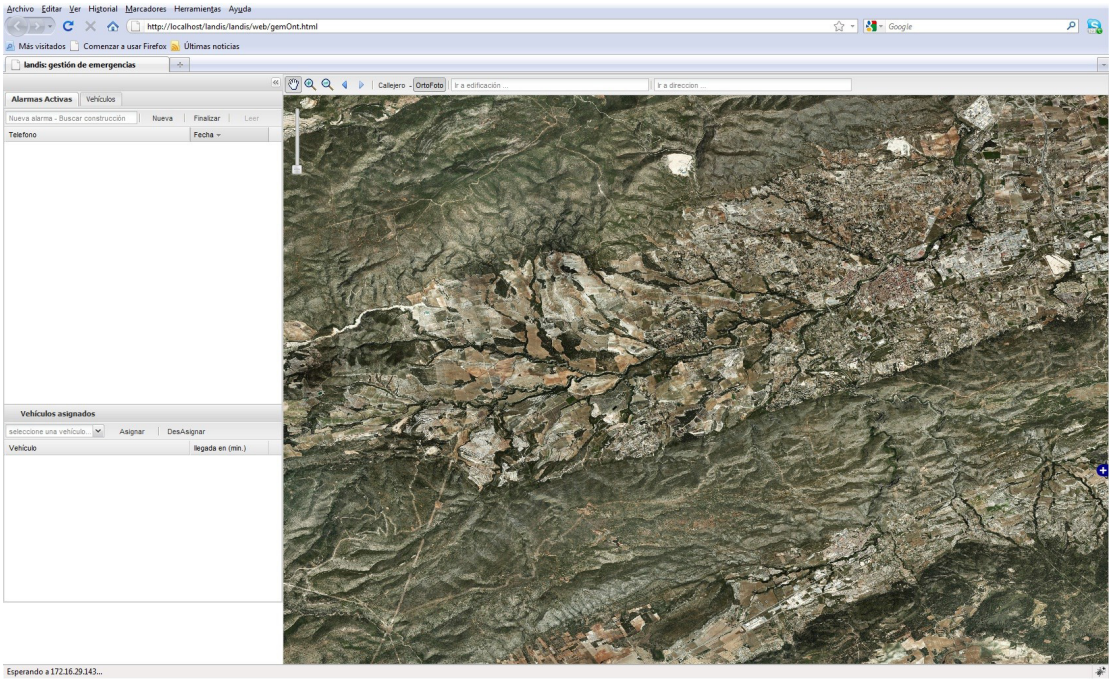


Figura 26: CEM ventana principal

5.3. Arquitectura de la aplicación

A continuación, vamos a detallar en grandes rasgos la arquitectura de la aplicación empleada para el desarrollo de toda la aplicación. Cabe destacar que la aplicación ha sido desarrollada haciendo uso de un framework llamado Symfony el cual hemos definido en puntos anteriores.

Hemos elegido este framework por su sencillez pero a la vez potencia ya que nos permite integrar PHP, plug-in de extJs y MapFish.

Entre sus principales características destaca que nos permite generar un modelo a partir de la abstracción de una base de datos, en nuestro caso en Postgres, crear vistas y controles para la misma.

La arquitectura de la aplicación se puede separar en tres puntos principalmente:

-MVC (Model view Controller). El modelo es la abstracción que hace symfony de nuestra base de datos, de manera que se crea el modelo en PHP al cual accederemos y con el que interactuaremos.

-Vista. La vista, es generada automáticamente por el symfony tomando como base el modelo generado, aunque en nuestro caso no hemos partido de la vista que genera symfony para desarrollar la aplicación, sino más bien ha sido desarrollada una vista desde cero.

-Control. En ella tenemos todos los modules, que forman la aplicación y que vienen a ser las funciones que podemos hacer con la misma. Se encargan de recibir peticiones y dar respuesta a las mismas. Se crea un modulo para cada acción que queramos que lleve a cabo la aplicación, o bien que de respuesta a la misma, así pues tal y como podemos observar en la siguiente imagen, tenemos módulos para:

-La asignación de las alarmas a los vehículos.
-Comprobar las alarmas activas.
-Comprobar la posición de las alarmas, de los vehículos.
-Mostar las construcciones, etc...

6. Diseño e Implementación

Tras haber adquirido una visión completa del sistema en cuanto a los actores que hay en el mismo, las acciones que efectúan y los distintos entornos en los que estás ubicado. Pasamos a entrar al mayor nivel de detalle, en que se especifica el diseño y posterior implementación de la solución, veremos la metodología de desarrollo seguida para llevar a cabo la solución de la aplicación.

6.1. Entorno de Producción

Una vez llegados a este punto, podemos observar como dentro del entorno de producción, tenemos 3 capas bien diferenciadas, la capa de presentación, la capa de persistencia y la de lógica.

A continuación, detallaremos cada una de ellas, tanto su diseño como la implementación que ha sido necesaria para llevar a cabo la aplicación.

6.1.1. Capa de presentación

En esta capa de presentación se encuentra la maquina del operario, el cual visualiza la aplicación por medio de un navegador Web. En principio el sistema operativo puede ser un sistema operativo tanto Windows, como Linux o MAC, y el navegador puede ser cualquiera siempre y cuando tenga disponibles los plug-in[26] de JavaScript pertinentes.

26 **Plug-in:** Un plug-in es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API.

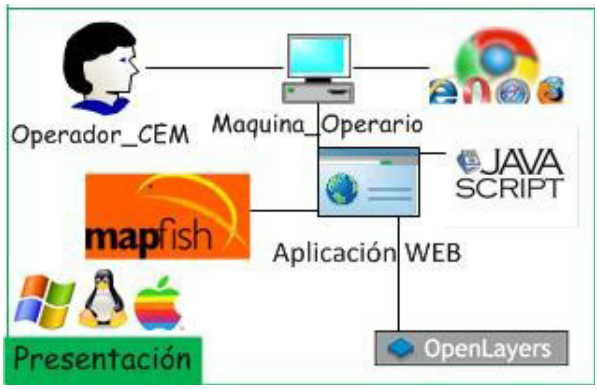


Figura 27: Capa de presentación

El perfil del operador CEM, debe ser de un usuario nivel medio, con conocimientos web y de cartografía en general además de disponer de los medios y conocimientos necesarios para saber que hacer en caso de emergencia.

En principio no ha de tener ningún conocimiento técnico de los elementos que han sido empleados para el desarrollo del CEM, ya que estos son transparentes al usuario final.

Observaremos los distintos paneles de la aplicación y para qué sirven cada uno de ellos.

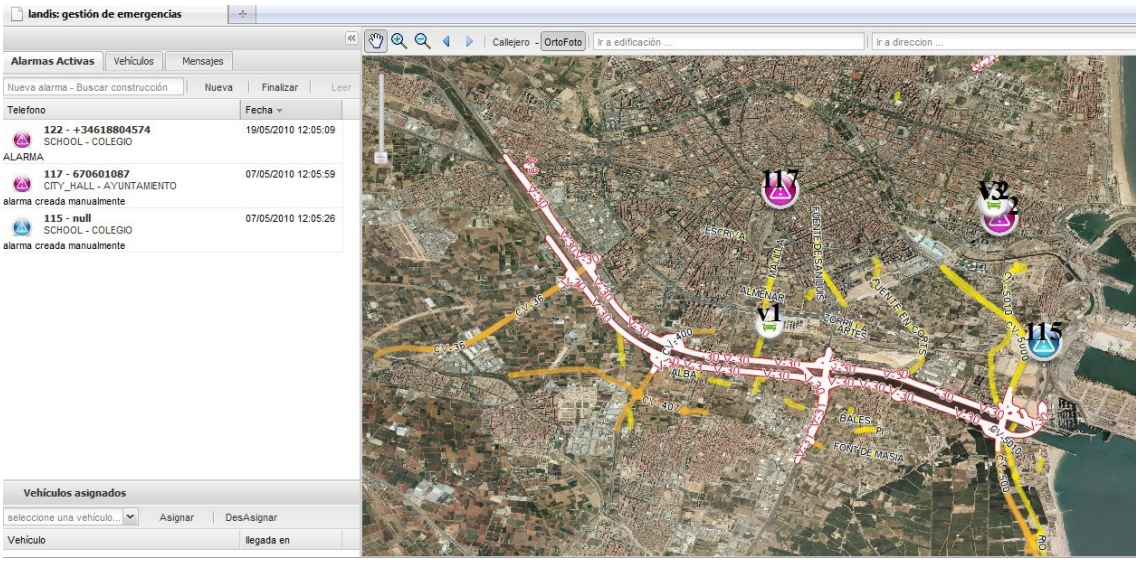


Figura 28: Ventana principal

En ella podemos ver los distintos vehículos que están dados de alta en el sistema y las alarmas que han sido creadas y asignadas a los vehículos para probar su correcto funcionamiento.

Pasamos a hacer un análisis de la aplicación por partes:

En la figura 29 se observa una presentación de los paneles/contenedores de información también llamados Grids:

Se pueden distinguir claramente las tres pestañas principales que componen la aplicación: alarmas nuevas, vehículos y mensajes, al pinchar cada una de ellas nos permite movernos por los paneles correspondientes a esa sección.

Así pues, este es el panel correspondiente a la **creación, búsqueda y finalización de alarmas**, se pueden observar distintos elementos de entre los cuales cabe destacar los siguientes:

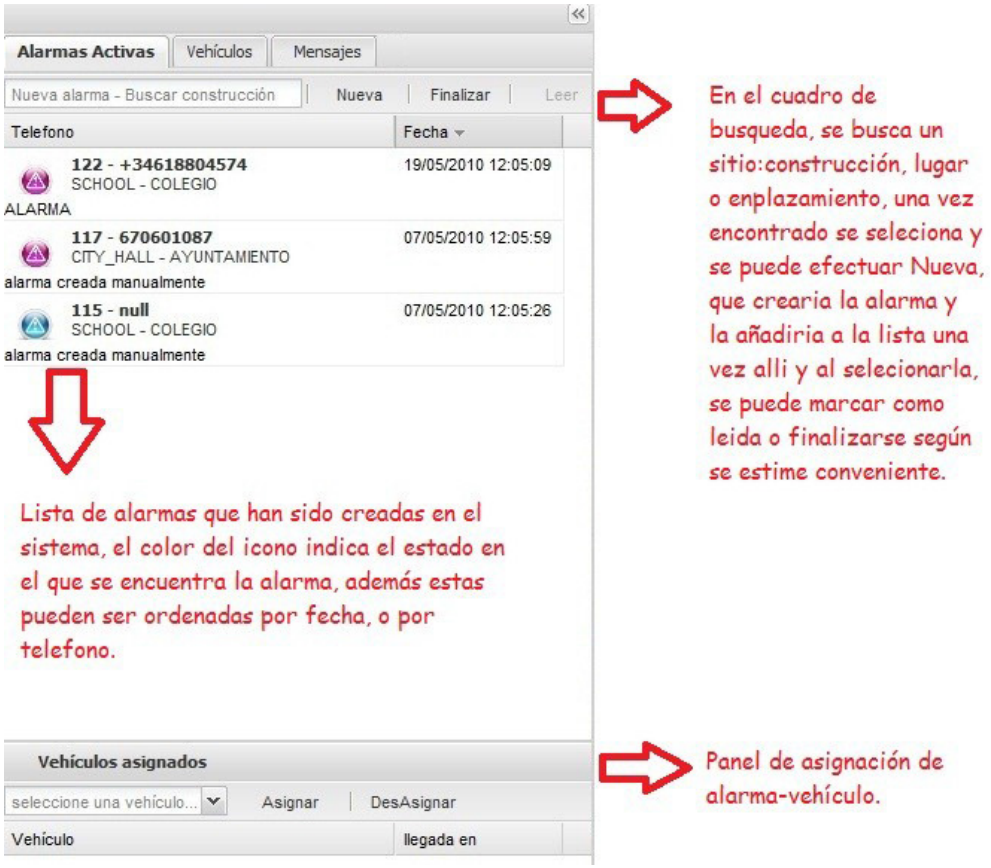


Figura 29: Grid alarmas

Funcionamiento:

Desde el cuadro de búsqueda, se introduce el número de construcción o parcela a buscar o en su defecto el número de teléfono del propietario, incluso permite la búsqueda por nombre si es un lugar público. Suele mostrar las coincidencias que concuerdan con el patrón de búsqueda introducido. De este patrón de búsqueda realizamos la selección del emplazamiento sobre el cual se quiere crear la alarma, una vez creada la alarma esta se agrega a la lista de alarmas con un estado sin asignar y aparece el icono en el mapa en la posición donde pertenece.

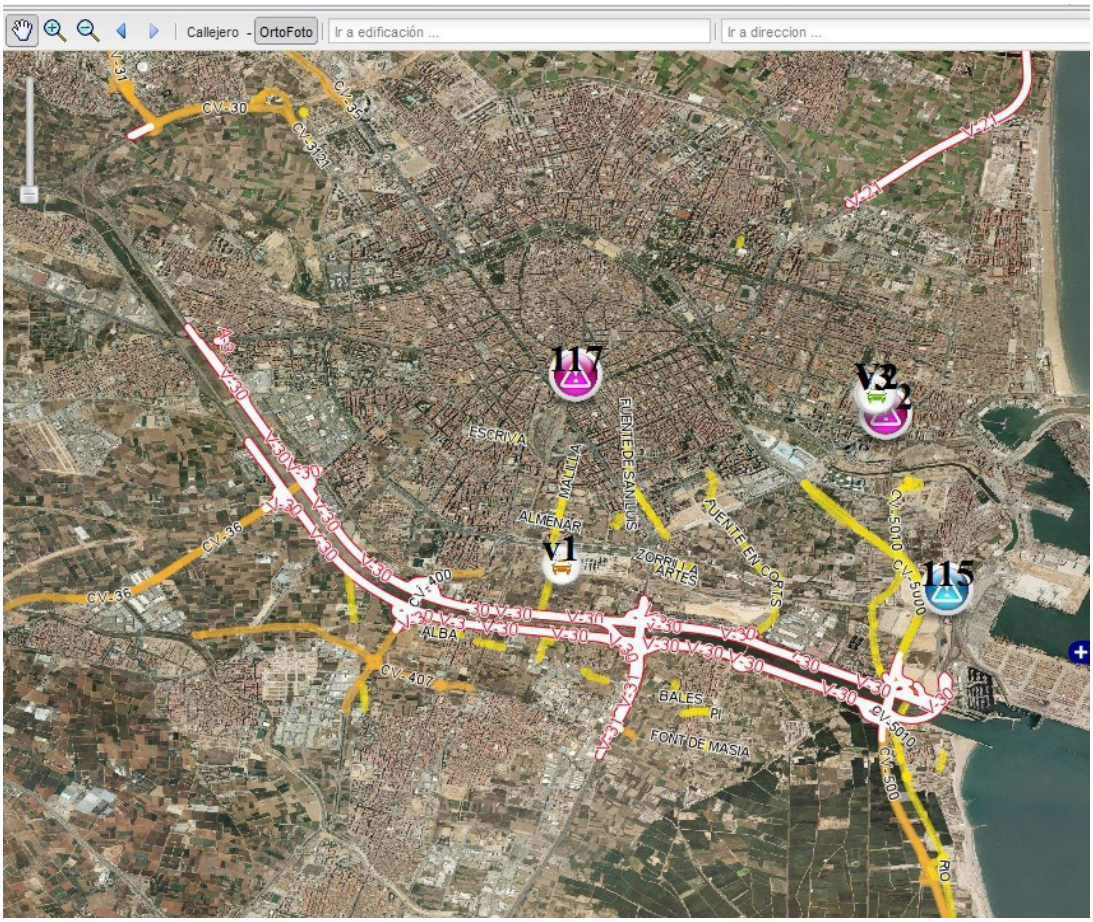


Figura 30: Vista cartografía

Por ejemplo si, creásemos la alarma 117, al mismo tiempo que se nos agregase en la lista en el panel de alarmas, se agregaría al mapa. Véase a continuación la **simbología** de las alarmas:

Figura 31: Simbología de las alarmas

Analizamos el panel de **vehículos asignados,** que se encuentra en la parte inferior tanto en la pestaña de alarmas como en la pestaña de vehículos:

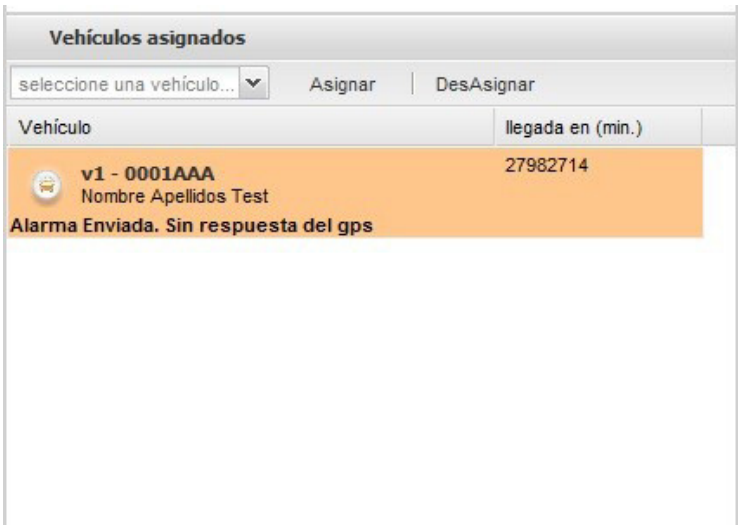


Figura 32: Vehículos asignados

Para poder asignar una alarma a un vehículo, se han de seguir unos pasos determinados, una vez creada la alarma esta se pone en estado nuevo, color azul.

Si deseamos asignar esta alarma a uno de los vehículos que tengamos disponibles para las emergencias tan solo hemos de teniendo la alarma seleccionada abrir el menú desplegable de vehículos en la parte inferior del panel y seleccionar el vehículo pertinente al que le será asignada la alarma y pinchar en el botón asignar, en ese momento se creará la asociación alarma-vehículo y se enviará al dispositivo GPS del vehículo la alarma y la ruta que ha de seguir para llegar a la misma, si deseamos dar más información acerca de esta se puede enviar un mensaje al vehículo.

Al mismo tiempo el estado del vehículo al que se le asigna la alarma cambia y por consiguiente el color del icono del mismo que pasa de ser verde a color naranja, aunque hasta que el vehículo no acepta la alarma esta no cambia su estado en el mapa y en la lista.

Figura 33: Grid vehículos

En la siguiente imagen pasamos a analizar la segunda pestaña de la aplicación, correspondiente a los vehículos. En ella se observa la información acerca de los vehículos disponibles en el sistema. Los vehículos destinados a emergencias son elementos externos, estos no se pueden crear ni finalizar, simplemente nos vienen dados en el sistema, previamente introducidos en la base de datos.

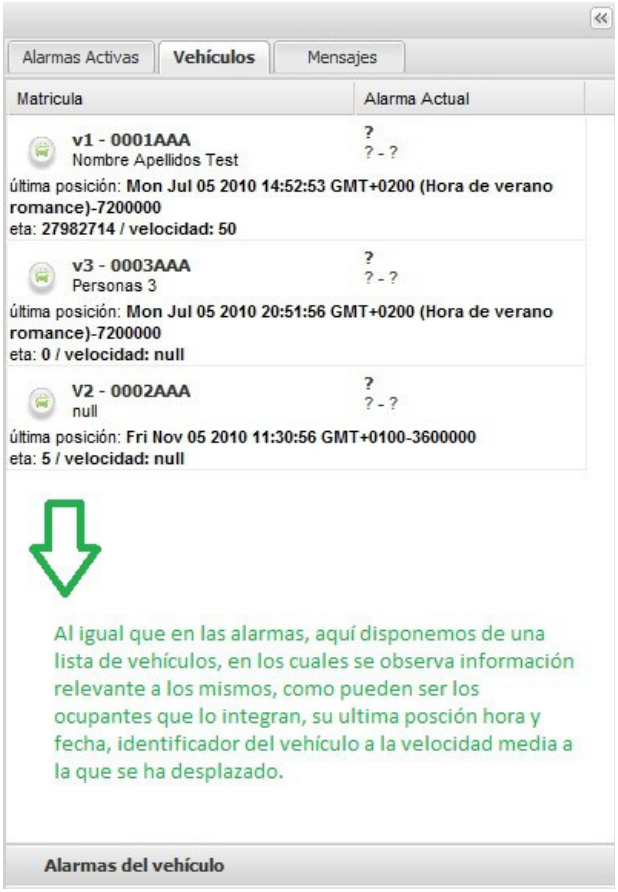


Figura 34: Panel vehículos

En este panel observamos la lista de vehículos que tenemos disponibles en el sistema, si cambiase su estado cambiaría el color de icono del vehículo, tanto en la lista como en el mapa, en la parte inferior de este panel tenemos la misma asignación alarma-vehículo al igual que en la pestaña de alarmas con el mismo funcionamiento, por lo que no pasaremos a detallarla.

Grid de Mensajes. En la última de las tres pestañas esta implementado el envió y recepción de mensajería entre el vehículo y la aplicación. Dentro de esta pestaña tenemos dos pestañas más, la de mensajes enviados y mensajes recibidos cada una de ellas compuestas por dos casillas de selección de fecha en las cuales se puede seleccionar la fecha.

O bien se puede introducir la fecha manualmente, en cualquier caso ambas casillas sirven para acotar la búsqueda de mensajes a un período de tiempo determinado, la primera de las casillas de fecha indica la fecha desde la que empieza la búsqueda (fecha inicio) y la segunda hasta cuando ha de realizarse la búsqueda (fecha fin).

Al pulsar el botón refrescar y dependiendo de en cual de las dos pestañas del panel nos encontremos mostrará los mensajes enviados y/o recibidos para esa período de fechas seleccionado.

Para no poder saturar el sistema con peticiones y puesto que la base de datos se supone que guarda un histórico de mensajes enviados y recibidos, se ha acotado esta búsqueda a las últimas cien ocurrencias, es decir sacara los últimos cien mensajes tanto en enviados como en recibidos.

6.1.2. Lógica de negocio

En este punto se detalla el análisis de la capa de lógica de negocio propiamente dicha, en la que encontramos los distintos servidores: Landis, Fina, Gateway, además se han incluido los dispositivos Rtcu y los GPS.

Diseño.

Se ha precisado del diseño de un entorno que permita la interconexíon de cada uno de los elementos esmentados en la imagen anterior.

Si partimos desde los propios vehículos de emergencias observamos como estos se comunican con el CEM mediante los dispositivos GPS de Garmin debidamente conectados a sus unidades Rtcu, es muy importante que para cada dispositivo GPS le corresponda una y sola una unidad Rtcu ya que será la única forma de identificar como es debido un vehículo.

Una misma unidad Rtcu no sirve para dos vehículos distintos, ya que cada una de estas dispone de un nodeID único el cual nos servirá en etapas posteriores para identificar el vehículo y permitir la comunicación entre el vehículo y la aplicación. Los dispositivos Rtcu se conectan mediante la red GPRS a Internet y tienen configurada la IP del Gateway en su software interno el cual permite que transmisión de información entre Rtcu y Gateway, veremos como esta información será tratada en etapas posteriores por el servidor Gateway o Fina.

El software específico instalado en el servidor Gateway permite la transmisión de los datos que reciben de Internet a Fina mediante la librería específica del Gateway, una vez en el servidor Fina los datos son debidamente almacenados en la Base de Datos de Postgres, la cual alimenta a su vez el servidor Landis que es el que da servicio a la aplicación CEM.

El servidor Landis, es el que mantiene la comunicación entre el CEM y la Base de Datos en Postgres del sistema, así mismo da servicios de cartografía WMS a la aplicación CEM y hace de servidor HTTP(Apache) para la aplicación. Tras una breve explicación de la visión global de la capa de lógica que disponemos, pasamos a analizar con mayor detalle la implementación y puesta a punto que ha sido necesaria para conseguir el correcto funcionamiento de sistema

Implementación.

-GPS de Garmin y Rtcu

Siguiendo el mismo orden que en diseño, pasamos al análisis de la implementación que ha sido necesaria para la puesta a punto de los dispositivos de GPS de Garmin y de las unidades de Rtcu, ambas unidades han sido distribuidas y programadas por el fabricante Logic Io el cual nos ha proporcionado el software necesario que hemos empleado para el Gateway y para Fina, por tanto la configuración en estos dispositivos a sido mínima, tan solo ha habido que configurar la ip del Gateway, para que los datos que transmiten cada uno de los dispositivos Rtcu sean enviados a esta máquina específicamente.

Funcionalidad del centro de mensajes del GPS Garmin SMS_Garmin:

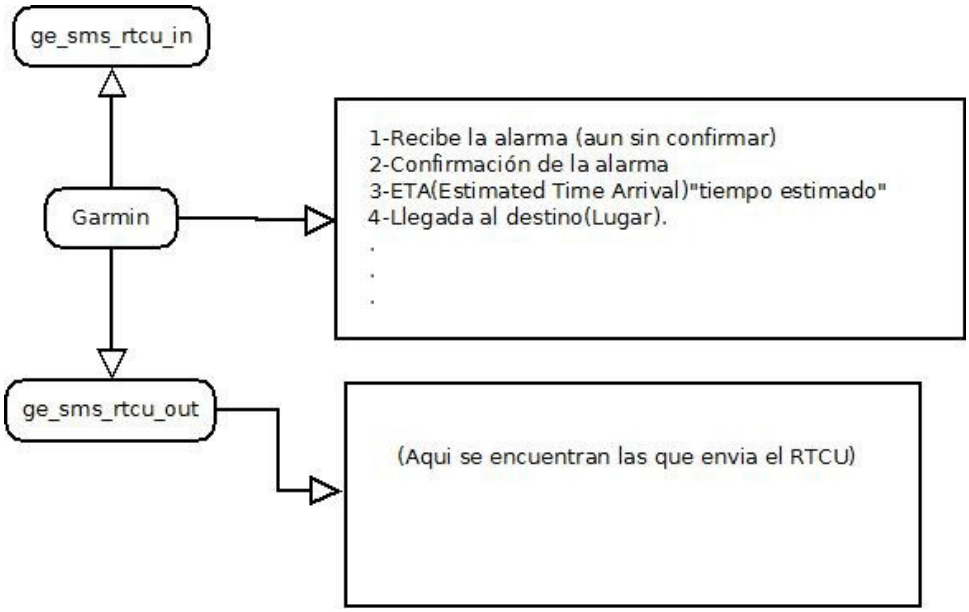


Figura 35: GPS Garmin

La finalización y/o confirmación de la alarma debe de ser manual por parte del personal que atiende la emergencia.

Las coordenadas que se enviadas al GPS de Garmin son coordenadas semicirculares, que son las que entiende el GPS para ir al lugar. Al mismo tiempo al GPS le llega la información acerca de la construcción y/o teléfono del lugar en donde se ha producido la alarma.

-GateWay

Si analizamos el entorno del servidor Gateway observamos que este equipo tiene como sistema operativo un Windows XP, ya que las aplicaciones de Logic IO no funcionan bajo Linux, debido a que hacen uso de una librería vsmsgw.dll que es la que hace de enlace entre esta máquina y Fina.

En el servidor de Gateway está continuamente ejecutándose la aplicación RTCU Gateway Monitor Tool de Logic IO, cuenta además de conexión a Internet con una IP fija, cuya Ip es (213.4.36.186).

Tal y como podemos observar en la siguiente figura observamos como el Gateway sirve de comunicación entre los dispositivos Rtcu y los clientes, en nuestro caso la base de datos en postgres.

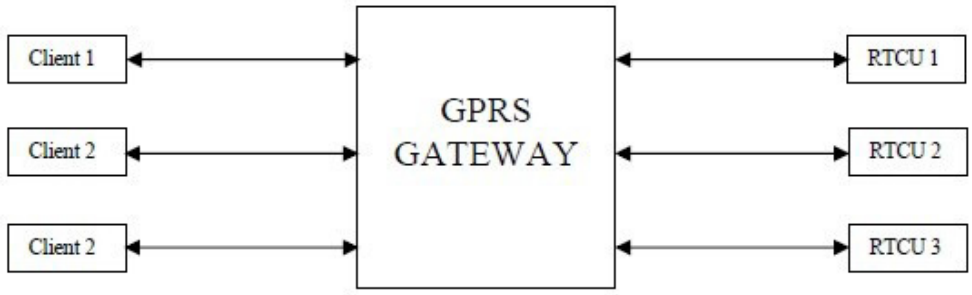


Figura 36: Gateway

Aunque tanto el servidor de Fina como el Gateway podrían estar físicamente en la misma máquina, se ha optado por un patrón de diseño en el que estén en dos máquinas distintas para así tener una mayor seguridad y consistencia en las transacciones que se efectúan entre la base de datos y el GPRS Tracking (Fina). Así pues una vez visto el porque de esta solución de diseño, pasamos al análisis de la aplicación cuyas principales funciones son las siguientes:

-RTCU Gateway Monitor Tool, en el observamos las diferentes opciones:
En la 1ª opción de Connect podemos configurar el modo de conexión de las RTCU al gateway monitor tool. Una vez está establecida la conexión del gateway con las RTCU, esta hace de enlace con el servidor de Fina y permite la comunicación entre los dispositivos RTCU y la Base de datos.

Fina
El servidor Fina esta configurado bajo un entorno Windows XP, ya que las aplicaciones de Logic IO no funcionan bajo linux, debido a que hacen uso de una librería vsmsgw.dll que es la que hace de enlace entre esta máquina y el gateway.

Se ha optado por separar las máquinas del gateway y Fina en dos máquinas distintas, para que los datos de la base de datos y toda la información referente a estos no quede expuesta a Internet y sea tan solo accesible en local, así ganamos en seguridad y evitamos posibles ataques, por lo tanto en el servidor Fina tenemos las siguientes aplicaciones ejecutándose:

Software de Gprs Tracking
Principalmente la aplicación de GPRS Tracking se encarga de recibir y enviar información entre el gateway y transmitirla a la base de datos de postgres en landis

La aplicación GPRS Tracking de Logic IO básicamente está constantemente pidiendo información acerca de un nodeId por un puerto determinado de la máquina, esta información que recibe la compara y dependiendo del valor que tenga se guardará en una tabla u otra de la base de datos, tal y como observamos en el siguiente código extraído del código fuente del programa GPRS Tracking:

```
static int _cdecl cbText(int HisNodeID, char *str, void* arg) {
 char sm[300];
   char mensaje[120];
  const char *conninfo;
  char *cod1,*cod2,*cod3,*cod4,*codmensaje;
  PGconn     *conn1;
  PGresult   *res1;
  conninfo = "hostaddr=172.16.29.143 dbname=landis user=postgres password=icvicv";
  strcpy(mensaje,str);
  conn1 = PQconnectdb(conninfo);
  cod1 = strtok(str," ");
  cod2 = strtok(NULL," ");
  cod3 = strtok(NULL," ");
  cod4 = strtok(NULL," ");
     /* Check to see that the backend connection was successfully made */
     if (PQstatus(conn1) != CONNECTION_OK)
    {
     fprintf(stderr, "Connection to database failed: %s",
     PQerrorMessage(conn1));
     exit_nicely(conn1);
     }
     else
     {

         //si cod1=message sm INSERT INTO ge_sms_rtcu_in HisNodeID,mensaje mensaje = str - cod1
         //si cod1 es ETA update nodeid cod2
         if (!strcmp(cod1, "message"))
             sprintf(sm,"INSERT INTO ge_sms_rtcu_in (nodeid,cod1,mensaje) values (%i,'%s','%s');",HisNodeID,cod1,mensa
         else if (!strcmp(cod1,"ETA"))
             sprintf(sm,"INSERT INTO ge_sms_rtcu_in values (%i,'%s','%s','%s','%s');UPDATE ge_datosgps_tr SET eta='%i'
             else
             sprintf(sm,"INSERT INTO ge_sms_rtcu_in values (%i,'%s','%s','%s','%s');",HisNodeID,cod1,cod2,cod3,cod4);

        res1=PQexec(conn1 ,sm);
        PQclear(res1);
        PQfinish(conn1);
     }
    return 0;
}
```

Figura 37: Código fuente del programa GPRS Tracking

Si analizamos en profundidad el código extraído de la aplicación observamos como al llamarse la función(cdecl_cbText(nodeID,str,arg)) se le pasa el nodeID y el código del mensaje que posteriormente será comprobado para así efectuar una acción u otra en la base de datos, ya que no es lo mismo que el contenido del mensaje sea del tipo “message” o que sea del tipo “ETA” estos han de codificarse de forma distinta y por tanto habrá que darles un tratamiento distinto.

Por otra parte en el código se muestra como configurar la librería vsmsgw.dll para que haga de enlace entre la máquina de GPRS tracking y la maquina de gateway con el RTCU monitor tool, en ella se configura la IP de la máquina a la que se tiene que acceder,el nodeID que se utilizará, el puerto y las opciones con las que queremos configurar la librería de acceso al
gateway.

Incluso la frecuencia de refresco que está establecida en 30 segundos se puede cambiar a la que mejor nos convenga.

```
//=============================================================================================================
//-----------------------------------------------------------------------------
// Initialize the VSMSGW library, establishing a connection to the GPRS Gateway
// with the appropriate parameters
//-----------------------------------------------------------------------------
int rc=vsmsgwInit(AfxGetApp()->GetProfileInt("Options", "My nodeid", 10000),
                                (LPCTSTR)AfxGetApp()->GetProfileString("Options", "GW IP", "213.4.36.186")
                                  AfxGetApp()->GetProfileInt("Options", "GW Port", 5001),
                                  key,
                  cbText,
                  cbPDU,
                  this
                  );
//=============================================================================================================

m_nodeid=AfxGetApp()->GetProfileInt("Options", "My nodeid", 10000);
m_updatefreq=AfxGetApp()->GetProfileInt("Options", "Updatefreq", 30);
UpdateData(false);
```

Figura 38: Código fuente para establecer la frecuencia de refresco

Así pues tal y como se observa tenemos un refresco de 30 segundos establecidos en la variable m_updatefreq, un m_nodeid=10000 y el puerto elegido para la comunicación es el 5001.

Landis
El servidor Landis es el que proporciona toda la información necesaria de la que se provee la aplicación web del operador CEM, por tanto detallaremos a continuación las aplicaciones empleadas para proporcionar estos servicios, de entre las cuales cabe destacar las siguientes:

-Servidor Apache + PHP
El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual.

6.1.3. Persistencia de datos

Una de las partes más importantes del sistema es la persistencia de los datos en nuestro caso utilizamos 2 métodos para garantizar la persistencia de los datos, por un lado la base de Datos Postgres, y por otro el repositorio de SVN que veremos en mayor detalle en la parte del entorno de desarrollo puesto que es allí donde se utiliza, así pues pasamos a nos centramos en la base de Datos en postgres.

PostgreSQL es un sistema de gestión de base de datos relacional orientado a objetos y libre, publicado bajo la licencia BSD. Se ha decidido implantar este sistema gestor de base de datos en otros motivos porqué es potente, rápido y cumple las propiedades ACID de los SGBD, las cuales consisten en: atomicidad, consistencia, Aislamiento y durabilidad.

Aparte la mayoría de usuarios están familiarizados con el entorno que proporciona Postgres, si bien hemos hecho uso de su interfaz para usuario pgAdmin para la creación de las tablas y vistas, todas las acciones podrían haberse realizado por línea de mandatos.

Tablas
SI observamos la base de datos está formada por un conjunto de tablas las cuales empiezan casi todas por (ge) de “gestión de emergencias”, el nombre de cada tabla indica el contenido que tendrá.

Vistas
Una vista de base de datos es un resultado de una consulta SQL de una o varias tablas; también se le puede considerar una tabla virtual.

Al tener una estructura fija el esquema relacional, en varias ocasiones nos hemos visto con la necesidad de para obtener alguna información tener que hacer unas vistas que nos proporcionen la información que precisamos.

6.2. Entorno desarrollo

La metodología seguida para el desarrollo en concreto de la aplicación es la de implementar toda la aplicación en local mediante un servidor Apache local con las mismas prestaciones que hay en Landis y tras probar los cambios y si todo es correcto almacenarlos en el repositorio de SVN y subir al mismo tiempo los cambios a Landis en producción para así que sean visibles al resto de usuarios.

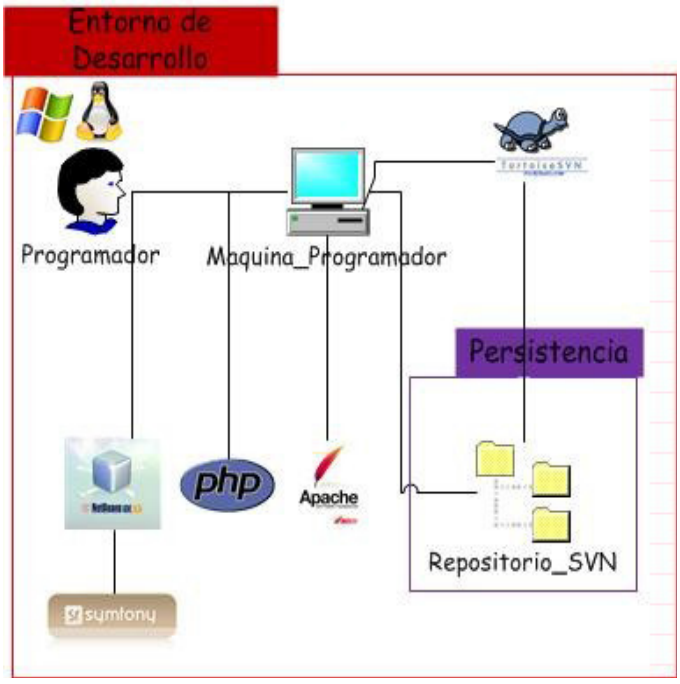


Figura 39: entorno de desarrollo

7. Observaciones y conclusiones

Para poder realizar el TFG de "Programación de una plataforma RTCU IDE para controlar una flota de vehículos a través de GSM, Bluetooth, WiFi y GPS", primero me he tenido que documentar revisando documentación de la página del fabricante.

Una vez tenía toda la documentación necesaria empecé ha estudiar todos los catálogos y manuales viendo ejemplos que proporciona el fabricante para aprender a programar sobre la plataforma RTCU IDE.

A continuación el Instituto Cartográfico Valenciano (ICV) me proporciono un ordenador, un dispositivo RTCU y una fuente de alimentación de 12V para alimentar al dispositivo RTCU. Con estos elementos compilé y probé los pequeños ejemplos que suministra el fabricante sobre el dispositivo RTCU para poder entender mejor su funcionamiento.

Gracias a lo que aprendí de programación en los primeros cursos de carrera me ha sido fácil entender como funcionaban los programas que proporciona el fabricante de la plataforma RTCU IDE los cuales hemos usado para montar este proyecto.

Juan Ybarra (el informático del ICV) diseño todo lo relacionado con los servidores a la vez que yo le ayudaba me iba explicando como lo realizaba y las ventajas de porque lo hacia así y no de otra forma.

Una vez estaban todos los servidores montados se programó la aplicación definitiva, que es el resultado de unir los diferentes ejemplos de programas que proporciona el fabricante en un solo programa.

A continuación, se procedió a comprobar el correcto funcionamiento de la aplicación instalando equipos en vehículos y realizando algún recorrido para validar el adecuado funcionamiento de la aplicación.

También quiero comentar que no todo ha funcionado a la primera, han ido surgiendo fallos y problemas que conforme iban surgiendo se miraba cual podía ser la causa del problema hasta que se solucionaba dicho problema. Uno de los problemas que se dio fue que no recibíamos los datos de la unidad RTCU y después de mirar que le llegaban los 12V de alimentación y teníamos conexión con dicha unidad me di cuenta que el problema era una mala conexión del conector de la antena GPS que tienen. Se procedió a soldar con estaño el cable que estaba suelto y empezó a recibir y enviar datos sobre su posición.

Existen diversos servicios de gestiones de emergencias que proporcionan algunas de las funcionalidades necesarias para atender estas emergencias, el gestionarlas se vuelve a veces una tarea dificultosa debido a la naturaleza misma de las situaciones y lugares en donde se puede producir una emergencia. Por lo tanto se precisa de un centro de emergencias que disponga de unas cualidades que permitan la rápida y precisa actuación por parte de los efectivos que hayan de tratar la emergencia (personal sanitario, bomberos, policía, etc.).

Debido a las carencias observadas en algunos de los sistemas de emergencias actuales, el instituto cartográfico valenciano en colaboración con el cuerpo de policía ha dispuesto un proyecto para el seguimiento y tratamiento de las emergencias, en el que ha sido empleada la cartografía propia del instituto con las ventajas asociadas que ello conlleva. Así mismo al disponer de esta cartografía personalizada se tiene pleno conocimiento no solo de toda la cartografía base que tiene disponible cualquier centro

de emergencia, sino también se dispone de las ubicaciones de las sendas rurales y carreteras no asfaltadas, además de todas las construcciones y edificaciones diseminadas en el territorio las cuales se encuentran georeferenciadas únicamente en nuestra aplicación.

Se ha conseguido crear una aplicación que proporciona un servicio rápido, seguro y preciso, requisitos indispensables que ha de tener un buen gestor de emergencias, ya que cuando se produce un accidente o se sufre una emergencia una rápida y concisa actuación junto con una buena comunicación entre el centro de emergencias y los efectivos destinados a su tratamiento puede suponer la diferencia entre salvar una vida o fracasar en ello, y no hay nada tan importante como la vida. Por lo que toda ventaja que se pueda tener para aumentar las probabilidades éxito a la hora de atender una emergencia es de suma importancia y ha de tenerse en consideración.

Como buen gestor de emergencias proporciona un servicio real para cualquier ciudadano que pueda precisarlo. Un servicio en el que, mediante un plan de actuación rápido y preciso, es capaz de atender y dar asistencia a todos los que la soliciten, incluso en las condiciones más adversas y con mayor dificultad.

Por otro lado, el haber trabajado en grupo en este proyecto me ha ayudado bastante ha aprender a trabajar en equipo y saber elegir la mejor propuesta de todas las que teníamos sobre la mesa después de estar horas discutiendo cual era la mejor solución a tomar.

Como conclusión final me ha parecido un proyecto bastante completo y bueno ya que aparte de tocar temas de informática que se vieron en la asignatura de Arquitectura y Redes telemáticas de 2º curso también se ha trabajado con temas de electrónica y antenas, a la vez que he aprendido a trabajar en grupo.

8. Bibliografía

Teoría de RTCU: http://www.logicio.com/introduction.htm

Documentación de la plataforma RTCU: http://www.logicio.com/downloadfiles.htm

Software para programar el RTCU: http://www.logicio.com/download_files.htm

Teoría vista en la asignatura de Arquitectura y Redes telemáticas *(11301)* de 2º curso para saber cómo funciona un servidor de datos.

Teoría vista en la asignatura de Antenas y radiopropagación *(11291)* de 3º curso para saber cómo funciona una antena.

Teoría vista en la asignatura de Programación 1 *(11267)* de 1º curso para entender cómo funcionan los programas que proporciona el fabricante de la plataforma RTCU.

Teoría vista en la asignatura de Circuitos electrónicos *(11271)* de 2º curso para saber cómo funciona un circuito electrónico y saber como realizar una medida (tensión o corriente).

Documentación ofrecida por el ICV (Instituto Cartográfico Valenciano): http://www.icv.gva.es/es