DISEÑO, ANÁLISIS E IMPLEMENTACIÓN DE UNA HERRAMIENTA DE SOFTWARE ORIENTADA A GEOLOCALIZACIÓN Y SEGUIMIENTO POR EVENTOS DE TERMINALES MÓVILES.

JESÚS ANDRÉS RUEDA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2011

DISEÑO, ANÁLISIS E IMPLEMENTACIÓN DE UNA HERRAMIENTA DE SOFTWARE ORIENTADA A GEOLOCALIZACIÓN Y SEGUIMIENTO POR EVENTOS DE TERMINALES MÓVILES.

JESÚS ANDRÉS RUEDA

Proyecto de Grado Para Optar al Título de Ingeniero de Sistemas

Director:
MANUEL GUILLERMO FLÓREZ BECERRA
Profesor Titular
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
BUCARAMANGA
2011

AGRADECIMIENTOS

A mi madre, sin cuyo incondicional apoyo y paciencia no hubiera sido posible la realización de este proyecto.

A Aktivia Networks, por estimular mi curiosidad por la programación en dispositivos móviles, y por prestarme los recursos para cultivarla.

A Fabián Jordán, por su amistad y por innumerables gestiones operativas para el desarrollo del sistema.

A Margarita, a quien no hay palabras suficientes para agradecer todo su apoyo y trabajo en la realización del proyecto, su esfuerzo y paciencia, gestión y colaboración, y sobre todo por el entusiasmo puesto en este, sin el cual el proyecto carecería de espíritu.

A Gladys, por su hospitalidad y apoyo en todo momento.

A Laura y Rafa, por su amistad y colaboración prestada durante este tiempo.

CONTENIDO

pág.

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

3G: Abreviación usada para referirse a la tercera generación de la tecnología en transmisión de voz y datos por medio de la telefonía móvil. Se conoce también como UMTS (*Universal Mobile Telecommunications System*). Brinda la posibilidad de transferir no solo voz y datos, sino también datos no-voz que sirven fundamentalmente para actividades asociadas con la conexión a Internet.

A-GPS (*Assisted GPS*): GPS asistido. Sistema que, bajo ciertas condiciones, permite mejorar el desempeño y la precisión en cuanto a la identificación de una posición por parte del sistema de posicionamiento global GPS. Diversos dispositivos móviles dotados con GPS ofrecen también el servicio de GPS asistido por intermedio de la compañía de telefonía celular que les presta servicio de comunicación.

AJAX (*Asynchronous JavaScript And XML*): JavaScript asincrónico y XML. Técnica de desarrollo que permite crear aplicaciones interactivas, las cuales se ejecutan en el cliente mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. Así es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que repercute incrementando la interactividad, velocidad y facilidad de uso de las aplicaciones. La tecnología es asíncrona porque los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página, y sin necesidad de recargarla. .

BLUETHOOTH: Especificación industrial para redes Inalámbricas de área personal (WPANs) que hace posible la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Esta tecnología busca hacer más sencillas las comunicaciones entre equipos móviles y fijos, eliminando la necesidad de conexiones por cable. Diversos dispositivos integran esta tecnología, entre los que se destacan aquellos que pertenecen al sector de las telecomunicaciones y la informática personal.

BTS (*Base Transceiver Station*): Conocida también como estación base, es una instalación fija de radio usada para la comunicación bidireccional. Se usa para comunicar con una o más radios móviles o portátiles. En el contexto de la telefonía móvil, una estación base dispone de equipos transmisores/receptores de radio, en la banda de frecuencias de uso (900 / 1800 MHz) que realizan el enlace entre los usuarios que realizan o reciben llamadas usando teléfonos móviles.

CDC (*Connected Device Configuration*): Configuración de J2ME destinada para dispositivos con mayores capacidades computacionales y de memoria. La CDC usa una máquina virtual parecida a la que usa el J2SE, pero con menores capacidades en lo referente a gráficas y memoria del dispositivo. Para implementar la CDC un dispositivo debe contar con procesador de 32 bits, 2MB o más de memoria total, completa funcionalidad de la Máquina Virtual de Java, y conectividad a algún tipo de red.

CELL-ID: Es un número único utilizado para identificar el área a la que le presta servicio cada BTS (*Base Transceiver Station*) o antena dentro de una red GSM. Cuando el usuario de un dispositivo móvil lo utiliza para comunicarse mediante la red celular, es posible entonces establecer que su posición está dentro del área de cobertura de la BTS mediante la cual se está comunicando.

CLDC (*Connected Limited Device Configuration*): Configuración de J2ME destinada a dispositivos con capacidades limitadas en lo que se relaciona con gráficas, cómputo y memoria; y además, dotados de conexión. Entre estos se cuentan PDAs, teléfonos celulares y similares. Para implementar la CLDC el dispositivo debe contar con procesador de 16 o 32 bits con al menos 25 Mhz de velocidad, y disponer de un mínimo de 128 Kb de memoria no volátil para la Máquina Virtual de Java y las bibliotecas CLDC, y 32 Kb de memoria volátil para la Máquina Virtual en tiempo de ejecución. Además, deben ofrecer bajo consumo puesto que trabajan con un suministro de energía limitado (baterías) y tener conexión sin cable a algún tipo de red.

CLR (*Common Language Runtime*): Lenguaje común en tiempo de ejecución. Componente de la máquina virtual de la plataforma .Net de Microsoft. Es la implementación del estándar CLI (*Common Language Infrastructure*) que define un ambiente de ejecución para los códigos de los programas escritos en Java. El CLR ejecuta una forma de código intermedio llamada *Common Intermediate Language* (CIL), que es la implementación de Microsoft del CLI. La forma en que la máquina virtual se relaciona con el CLR permite a los programadores ignorar muchos detalles específicos del CPU que ejecutará el programa.

COM: Puerto serial de comunicación que transmite datos digitales bit a bit, enviando un solo bit a la vez. Es usado frecuentemente por computadoras y por aparatos auxiliares e independientes que están conectados a la CPU.

CVM (*Compact Virtual Machine*): Implementación de la máquina virtual Java reducida y que está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y memoria total de 2MB o mayor.

EDGE (*Enhanced Data Rates for GSM of Evolution*): Tasas de Datos Mejoradas para la evolución de GSM). También se le conoce por las siglas EGPRS (*Enhanced GPRS*). Es una tecnología de telefonía móvil celular que actúa como puente entre las redes 2G y 3G, sirviendo para la transferencia de datos basada en conmutación por paquetes, como es el caso de la conexión a Internet. Contiene ciertas mejoras frente a la tecnología GPRS (*General Packet Radio Service*), que se reflejan en una mayor velocidad y capacidad para la transferencia de datos. Funciona sobre redes GSM siempre y cuando los operadores de telefonía celular hayan implementado las actualizaciones necesarias; y sólo es accesible para los dispositivos móviles concebidos para trabajar con esta tecnología.

EVENTO: En ciencias de la computación se refiere a una posible acción que el usuario puede realizar que es monitoreada por una aplicación o sistema operativo. Cuando un evento ocurre se invoca un 'manejador' de evento, el cual es una pieza de software que realiza una acción específica con respecto al evento.

GIS (*Geographic Information System*): Sistema de Información Geográfica. Integración organizada de hardware, software y datos geográficos diseñada para almacenar, editar, otorgar y analizar información referenciada geográficamente. Los GIS son herramientas que permiten a los usuarios crear consultas personalizadas, analizar la información espacial, editar datos asociados a ella, presentar mapas y otorgar resultados específicos de todas estas operaciones

GOOGLE MAPS: Servicio brindado por Google de forma gratuita, que consiste en un servidor de aplicaciones de mapas *online*, que permite visualizar mapas desplazables y fotografías satelitales de la superficie terrestre. Este servicio permite su integración parcial con otros sitios web para que estos ofrezcan algunos de los servicios brindados por *Google Maps*.

GPRS (*General Packet Radio Service*): Servicio general de paquetes vía radio. Es una extensión del Sistema Global para Comunicaciones Móviles (*Global System for Mobile Communications o GSM*) para la transmisión de datos no conmutada. Permite velocidades de transferencia de 56 a 144 kbps.

GPS (*Global Positioning System*): Sistema de posicionamiento global. También conocido como NAVSTAR-GPS. Es un sistema global de navegación por satélite que permite establecer la posición de un objeto sobre la esfera terrestre y expresarla en términos de coordenadas geográficas.

GSM (*Groupe Spécial Mobile*): Sistema global para las comunicaciones móviles. Sistema estándar de telefonía móvil celular gratuito. Permite realizar funciones digitales de transmisión de datos utilizando un dispositivo móvil y realizar acciones tales como enviar y recibir correo electrónico o mensajes de texto y navegar por Internet. Es considerado un sistema 2G (de segunda generación) por características tales como su velocidad de transmisión.

HTML (*Hypertext Markup Language*): Lenguaje de marcado de hipertexto. Lenguaje de marcado más común en la elaboración de páginas web. Permite describir la estructura y el contenido del sitio, describir la apariencia de un documento, e incluso introducir un script que afecte el comportamiento de procesadores HTML, tal como son los navegadores web. HTML también se utiliza para referirse a contenidos de tipo MIME (*Multipurpose Internet Mail Extensions*), que tienen mejores posibilidades al transferir contenidos de texto.

J2ME (*Java 2 Micro Edition*): También conocida como Java Micro Edition. Es una especificación de un subconjunto de la plataforma Java que brinda una colección tecnologías y especificaciones que conforman una plataforma apta para el desarrollo de software destinado a dispositivos con recursos restringidos tales como dispositivos móviles.

JAVA CARD: Esta plataforma permite la creación y ejecución de pequeñas aplicaciones sobre tarjetas inteligentes, '*smart cards'*, y dispositivos similares

JAVA EE (*Enterprise Edition*): Es la misma plataforma SE, pero con diversas funcionalidades y APIs para realizar aplicaciones cliente servidor multi-capa.

JAVA SE (*Standard Edition*): Plataforma de desarrollo de Java orientada a dispositivos tales como computadores personales (PC), servidores y similares.

JDK (*Java Development Kit*): Kit de desarrollo de Java. Conjunto de los programas necesarios para desarrollar aplicaciones Java.

JRE (*Java Runtime Enviroment*): Programa que viene con la JVM y el compilador JIT, y que en conjunto con ellos es el encargado de ejecutar los programas que están en *bytecode* sobre el sistema operativo y hardware para el cual están construidos. Para ejecutar aplicaciones desarrolladas en Java sólo es necesario el JRE, mientras que para desarrollar nuevas aplicaciones también es necesario el entorno de desarrollo o JDK.

JSON (*JavaScript Object Notation*): Notación de Objetos de JavaScript. Formato ligero de intercambio de datos independiente de cualquier lenguaje de

programación. Está basado en un subconjunto del Lenguaje de Programación JavaScript, *Standard ECMA-262 3rd Edition*; como subconjunto de la notación literal de objetos de JavaScript no requiere el uso de XML. JSON tiene formato de texto plano, de simple lectura, escritura y generación.

JVM (*Java Virtual Machine*): La máquina virtual de Java es un programa escrito en código nativo de una plataforma, que interpreta y ejecuta código *bytecode* de Java. La JVM es escrita y compilada para un sistema operativo específico, y tiene la responsabilidad de tomar el *bytecode* y ejecutarlo sobre el sistema operativo en el cual se encuentra y para el cual fue creada.

KVM (*Kylo Virtual Machine*): Implementación más reducida de la máquina virtual Java desarrollada por Sun Mycrosistems. Está orientada a dispositivos con bajas capacidades de procesamiento; esta máquina virtual es la que se utiliza en la configuración CDLC.

LBS (*Located Based Services*): Servicios basados en localización. Prestaciones que se proveen al usuario a petición y de modo personalizado, basándose en su posición actual. Dichos servicios determinan la posición del usuario por medio de múltiples técnicas usadas para establecer su ubicación, y usan la información captada para proveer aplicaciones personalizadas y servicios de valor agregado.

LINQ (*Language Integrated Query*): Es un proyecto de Microsoft que agrega consultas nativas semejantes a las de SQL a los lenguajes de la plataforma .NET, definiendo operadores de consulta estándar que permiten a lenguajes habilitados con LINQ filtrar, enumerar y crear proyecciones de varios tipos de colecciones usando la misma sintaxis. Tales colecciones pueden incluir arreglos, clases enumerables, XML, y conjuntos de datos El propósito de LINQ es permitir que todo el código hecho en Visual Studio sea también orientados a objetos, y facilitar y estandarizar el acceso a dichos objetos.

METODOLOGÍA XP (*Extreme Programming*): Enfoque de la ingeniería de software que propone una metodología ágil de desarrollo, que prioriza la adaptabilidad del software a los requerimientos que surgen durante el proceso de construcción del mismo.

MIDP (*Mobile Information Device Profile*): Conjunto de APIs que proveen funcionalidades específicas teniendo en cuenta las características de los dispositivos móviles, y que componen uno de los perfiles creados por Java. Este perfil está construido sobre la configuración CLDC, y fue el primer perfil definido para la plataforma J2ME. Está orientado a dispositivos con capacidad

computacional y de memoria limitada, conectividad limitada, capacidad gráfica reducida y entrada de datos alfanumérica reducida. Entre los dispositivos que se ajustan a las características requeridas por este perfil se cuentan los teléfonos celulares, PDAs y buscapersonas.

PDA (*Personal Digital Assistant*): Computadora de mano que cuenta con un sistema de reconocimiento de escritura. Puede ser usada para realizar diversas labores que incluyen procesamiento de texto, conexión a Internet, y gestión de agenda y calendario. Una PDA típica está dotada con una pantalla táctil gracias a la cual se ingresa la información, un sistema de conexión inalámbrica (Bluetooth, infrarrojo, WiFi) y una tarjeta de memoria. Una de sus funciones esenciales es la sincronización con ordenadores personales para las aplicaciones compartidas (agenda, calendario) para que la información almacenada en ambos coincida.

PDU (*Protocol Data Units*): Unidades de datos de protocolo. Codificación que se aplica al texto de un mensaje para permitir el intercambio de datos entre unidades de igual nivel, o comunicación par-a-par. Existe una codificación para cada una de las capas involucradas en el proceso de comunicación, definidas según el modelo OSI.

POSTBACK: Mecanismo de recarga de páginas introducido por ASP.NET que hace posible recargar la información para ayudar a establecer un estado diferente del inicial, permitiendo separar funciones y hacer el código más legible, estructurado y reutilizable.

REACCIÓN: Para efectos del sistema, es una tarea que debe ejecutarse al ocurrir un evento asignado y que se manifiesta como una acción digital que interactúa con otros servicios informáticos; su función es automatizar el proceso de responder a un evento.

SDK (*Software Development Kit*): Kit de desarrollo de software. Conjunto de herramientas de programación que le permiten a un desarrollador crear aplicaciones para un sistema concreto.

WCF (*Windows Communication Foundation*): Plataforma de mensajería y comunicación diseñada por Microsoft y que forma parte de la API de la Plataforma .NET 3.0. Tiene como propósito permitir el desarrollo de aplicaciones basadas en arquitecturas orientadas a servicios con una API simple y que puede ejecutarse en una máquina local, una red LAN, o sobre Internet de forma segura.

WEBSERVICE: Conjunto de protocolos y estándares que sirven para comunicar datos entre aplicaciones de software. Las aplicaciones que intercambian información pueden haber sido desarrolladas usando diferentes lenguajes de programación y ejecutarse sobre cualquier plataforma; la posibilidad del intercambio de datos depende de la adopción de estándares abiertos. El intercambio de datos se hace entre equipos que estén conectados en una red de computadoras, que puede ir desde una red local hasta Internet.

WI-FI: Marca con que se identifica la *Wi-Fi Alliance*, organización comercial encargada de certificar los equipos que cumplen los estándares de la IEEE 802.11 en relación con redes inalámbricas de área local, fomentando la tecnología inalámbrica y asegurando la compatibilidad de equipos. Una red Wi-Fi, entonces, cumple con unas características especiales en la transmisión de paquetes de datos

XHTML (*Extensible Hipertext Markup Language*): Lenguaje extensible de marcado de hipertexto. Es el lenguaje de marcado concebido para reemplazar a HTML como estándar para la elaboración de páginas web. Tiene las mismas funcionalidades, y además cumple en forma más estricta las especificaciones de XML

XML (*Extensible Markup Language*): Metalenguaje extensible de etiquetas que, no siendo un lenguaje en sí mismo, permite definir lenguajes para diferentes necesidades. Es un estándar para el intercambio de información estructurada entre diferentes plataformas, permitiendo que dos sistemas compartan información de forma ágil y segura. Frecuentemente comparado con JSON, se distingue de este por su mayor nivel de estructuración.

XSL (*Extensible Stylesheet Language*): Familia de lenguajes basados en XML que permite determinar en qué forma la información contenida en un documento XML debe ser transformada para su presentación.

RESUMEN

TÍTULO: DISEÑO, ANÁLISIS E IMPLEMENTACIÓN DE UNA HERRAMIENTA DE SOFTWARE ORIENTADA A GEOLOCALIZACIÓN Y SEGUIMIENTO POR EVENTOS DE TERMINALES MÓVILES*

Autor: **

Palabras clave: Localización, GPS, *Google Maps*, GPRS, Dispositivo móvil, J2ME, ASP.NET, Eventos, Reacciones, Posicionamiento. GIS. LBS (Servicios Basados en Localización).

DESCRIPCIÓN

Este proyecto tiene como propósito llevar a cabo un estudio sobre las tecnologías de localización existentes y su implementación en dispositivos móviles dotados con GPS, con el propósito de desarrollar un prototipo de herramienta de software que permita automatizar los procesos de seguimiento y reacción a eventos de posición. Como resultado se ha diseñado un sistema conformado por tres aplicaciones: LETSClient, software que va en el dispositivo móvil y que se encarga de obtener la posición del usuario, procesarla para determinar si ha ocurrido un evento, y notificar la ejecución de este; LETSMonitor, sitio web que permite la configuración y visualización de usuarios, eventos y reacciones, y LETSServer, quien se encarga de la comunicación de los eventos configurados y ocurridos, así como de la ejecución de las reacciones.

LETSMonitor y LETSClient son aplicaciones web basadas en el .NET Framework de Microsoft, específicamente sobre ASP.NET. La aplicación móvil, LETSClient, está desarrollada en J2ME. El lenguaje de desarrollo empleado para monitor y servidor es C#, y Java para el cliente móvil. El almacenamiento de la información se hace en un motor de base de datos Microsoft SQL Server 2005.

El prototipo se desarrolló utilizando la Metodología XP que estructura un plan de desarrollo dividido en fases con objetivos específicos. La aplicación desarrollada permite seguir a un usuario que porta un dispositivo móvil, configurar eventos de posición que sean notificados al operador del sistema por medio del monitor, y configurar reacciones automáticas que se desencadenen ante la notificación de la ocurrencia de un evento.

---

* Trabajo de grado

** Facultad de Ingenierías Físico-Mecánicas, Escuela de Ingeniería de Sistemas e Informática, Dir. Manuel Guillermo Flórez Becerra

ABSTRACT

TITLE: DESIGN, ANALYSIS AND IMPLEMENTATION OF A SOFTWARE TOOL DIRECTED TO GEOLOCATION AND EVENT TRACKING ON MOBILE DEVICES*

Author: Jesús Andrés Rueda**

Keywords: Localization, GPS, *Google Maps*, GPRS, Mobile device, J2ME, ASP.NET, Events, Reactions, GIS, LBS (Located Based Services).

DESCRIPTION

The purpose of this project is to carry out a study about the existing location technologies and its implementation in mobile devices provided with GPS. The general purpose is to develop a software prototype tool which allows automatizing the tracking and reaction process to location events. The result of this plan was a system formed by three software applications: LETSClient, installed in the mobile device and responsible for obtaining the GPS coordinates, processing those in order to determine if a location event has taken place, and notifying that location event to the server application; LETSMonitor, a website that allows the system operator to configure and visualize users, events and reactions; and LETSServer, a software which communicates the occurred and configured events.

The applications LETSMonitor and LETSClient are based on the Microsoft´s .NET Framework, specifically on ASP.NET. The mobile application, LETSClient, has been developed using J2ME. The development language employed for the applications LETSMonitor and LETSServer is C#, and the language Java was used to develop that mobile application. The storage of the information is done using a database server Microsoft SQL Server 2005.

LETS was developed using the Extreme Programming (XP) methodology which structures a development plan divided into some phases, each one with specific goals. The developed application allows following a user who carries a mobile device, to create and configure location events, and notify to the system operator when the location events happen through the monitor application. It also allows the operator to configure automatic reactions which are triggered in the presence of an occurred event notification.

---

* Graduate thesis project

** Physical Mechanical Engineering Faculty, School of IT Engineering and Informatics, Dir. Manuel Guillermo Flórez Becerra

INTRODUCCIÓN

Los dispositivos móviles cada vez tienen mayor capacidad de gestión y procesamiento de datos y por tanto, pueden soportar programas más complejos que realicen tareas de mayor especificidad. Las redes de transmisión de datos y la capacidad de hardware de estos hace más rápida y eficiente la conectividad. Sus características de hardware les permiten mayor autonomía, pudiendo utilizar dispositivos como GPS o Bluetooth durante periodos de tiempo relativamente largos. Además, los nuevos dispositivos móviles permiten mayor control del hardware por medio del software. En ese sentido, cabe preguntarse hasta donde podría llegarse en el desarrollo de software para dispositivos móviles que habilitara estos para ofrecer prestaciones más complejas, y relacionadas con inquietudes vigentes en el campo de la investigación, como los alcances y limitaciones que se tienen al desarrollar software orientado a localización y seguimiento dentro de nuestro contexto específico.

Un campo ampliamente explorado es el de los usos potenciales que podrían hacerse de un dispositivo capaz de dar cuenta de su posición, tal como el GPS. Actualmente, éste es utilizado para múltiples tareas de localización y seguimiento en la industria del transporte, de la recreación, el turismo, servicios médicos, etc. Dentro de este campo surge el de los LBS (*Located Based Services*, por sus siglas en inglés) que son prestaciones que se proveen al usuario a petición y de modo personalizado, basándose en su posición actual. Dichos servicios determinan la posición del usuario por medio de múltiples técnicas usadas para establecer su ubicación, y usan la información captada para proveer aplicaciones personalizadas y servicios de valor agregado.

En el campo investigativo, los LBS's son considerados frecuentemente como un subconjunto especial de los llamados "Servicios conocedores de su posición". Estos últimos pueden definirse como servicios que adaptan automáticamente su comportamiento, por ejemplo, filtrar o presentar información, a uno o más parámetros reflejando el contexto de aquello que es localizado. Estos parámetros se conocen como información de contexto.

La información de contexto puede ser primaria o secundaria. Es información primaria aquella que se refiere a datos que son seleccionados mediante sensores, por ejemplo, hora, posición, etc. Es información secundaria aquella que resulta del trato de los datos arrojados como información de contexto primaria mediante combinación, deducción o filtrado. La información de contexto secundaria es de

mayor complejidad, y su valor y precisión dependen de los métodos usados para tratar la información primaria.

En este sentido se propone la creación de un software capaz de configurar eventos. La configuración de eventos consiste en determinar uno o varios parámetros que deben cumplirse para que cierta transformación en la posición o en la relación movimiento – espacio del dispositivo sea considerara un fenómeno relevante, es decir, un evento. Identificar eventos permite discriminar los datos de localización y seguimiento al determinar cuáles son más importantes en determinado sentido y por tanto deben ser informados y alertar a destinatarios interesados en ellos. Identificar eventos puede servir para automatizar reglas y acciones, prevenir y/o alertar acerca de situaciones no deseadas o de riesgo , aplicar criterios para discriminar la información de mayor interés, evitar o detectar irregularidades en un proceso común, reaccionar rápidamente en una situación no prevista o fuera de lo normal, etc. En términos generales, el servicio de alerta permite configurar reacciones automáticas previamente parametrizadas que puedan notificar a individuos interesados o a otros sistemas utilizando diversos canales informáticos.

Un escenario potencial de aplicación que daría cuenta de las capacidades del software descrito está enmarcado en el contexto de prestaciones de seguridad y monitoreo a personas. Casos específicos de las mismas son:

a) Alertar cuando una persona entre o salga de uno o varios perímetros previamente establecidos.
b) Localizar por demanda a una persona.
c) Informar si una persona se ha alejado de cierto punto más de una distancia previamente definida.
d) Notificar cuando la velocidad de un individuo o un vehículo supere determinado límite o, por el contrario, cuando decrezca por debajo de cierto punto.
e) Calcular la distancia total recorrida por una persona o vehículo, así como informar cuando esta supere determinado límite.
f) Localizar a una persona basado en intervalos de tiempo.
g) Generar alertas a partir de la combinación de dos o más de los casos arriba mencionados, por ejemplo: Alertar si un vehículo se desplaza durante cierta medida de tiempo a determinada velocidad dentro de un área descrita.

Una de las herramientas de programación más difundida para dispositivos móviles es J2ME, plataforma que funciona en la mayor cantidad de dispositivos móviles en el mercado. J2ME es una colección de tecnologías y especificaciones que

conforman una plataforma apta para las necesidades de dispositivos como PDA's, sistemas embebidos, dispositivos de hogar y teléfonos celulares. La tecnología Java ME, también conocida como J2ME, fue originalmente creada para lidiar con las restricciones asociadas en desarrollar aplicaciones para dispositivos pequeños. Esta plataforma permite construir un entorno de ejecución de Java (JRE) que se ajuste a los requerimientos de un dispositivo en particular del mercado. Así, para explorar las posibilidades de creación de una aplicación de software para la descrita, es necesario investigar y aprender sobre los fundamentos teóricos de la plataforma y el lenguaje que se usa para programar la misma.

1 PRESENTACIÓN DEL PROYECTO

1.1 DESCRIPCIÓN DEL PROYECTO

El presente proyecto se propone crear una herramienta de software que automatice el proceso de seguimiento y reacción a eventos de posición.

El software es apto para realizar las siguientes funciones básicas:

- Señalar la posición de uno o múltiples usuarios en un mapa.
- Procesar y analizar la posición y sus cambios de acuerdo con ciertas condiciones predeterminadas.
- Generar alertas como consecuencia dichos eventos predeterminados y comunicarlas a los usuarios y a sistemas de información externos.

El proyecto se desarrolló siguiendo la metodología de desarrollo ágil XP (*Extreme Programming*). Para crear el software se siguieron los pasos indicados en esta metodología, y el proyecto se estructura de acuerdo con la misma.

Adicionalmente, el proyecto se propone elaborar un tutorial del lenguaje J2ME utilizando la plataforma educativa MeiWeb de la Escuela de Ingeniería de Sistemas e Informática, que de cuenta de la investigación realizada para conocer el lenguaje de programación utilizado en la construcción de la herramienta de software.

1.2 OBJETIVOS

1.2.1 Objetivo general.

Realizar el diseño de una herramienta de software capaz de obtener la posición geográfica de uno o varios terminales móviles dotados de GPS y de procesarla en tiempo real en función de eventos previamente configurados para generar y transmitir alertas; como resultado de la investigación realizada al efecto, elaborar un minitutorial del lenguaje J2ME.

1.2.2 Objetivos específicos.

- Desarrollar una aplicación cliente, en una arquitectura cliente-servidor, que opere en dispositivos móviles dotados de GPS y con plataforma Java j2me que

extraiga la posición geográfica del dispositivo y la procese de acuerdo con eventos previamente configurados.

- Desarrollar una aplicación servidor capaz de:
  - Configurar eventos de acuerdo con criterios predeterminados, para enviarlos a la aplicación cliente y que ésta procese la información de localización de acuerdo con dichos eventos.
  - Recibir la información eventos identificados por el dispositivo móvil.
  - Generar y transmitir alertas sobre los eventos a los destinatarios configurados.
  - Exponer un W*ebService* con funciones suficientes para permitir la comunicación de los eventos generados a los sistemas que lo consuman.

- Desarrollar una interfaz web que permita:
  - Administrar gráficamente: a) la configuración de los eventos, b) los terminales móviles, c) los destinatarios, d) demás parámetros configurables.
  - Visualizar la información relativa a los eventos por medio de un mapa y usando el servicio *Google Maps*.

- Elaborar un tutorial del lenguaje J2ME utilizando la plataforma educativa MeiWeb de la Escuela de Ingeniería de Sistemas e Informática.

1.3 JUSTIFICACIÓN

1.3.1 Descripción del problema.

Los dispositivos móviles cada vez tienen mayor capacidad de gestión y procesamiento de datos y por tanto, pueden soportar programas más complejos que realicen tareas de mayor especificidad. Las redes de transmisión de datos y la capacidad de hardware de estos hace más rápida y eficiente la conectividad. Sus características de hardware les permiten mayor autonomía, pudiendo utilizar dispositivos como GPS o Bluetooth durante periodos de tiempo relativamente largos. Además, los nuevos dispositivos móviles permiten mayor control del hardware por medio del software. En ese sentido, cabe preguntarse hasta donde podría llegarse en el desarrollo de software para dispositivos móviles que habilitara estos para ofrecer prestaciones más complejas, y relacionadas con inquietudes vigentes en el campo de la investigación, como los alcances y limitaciones que se tienen al desarrollar software orientado a localización y seguimiento dentro de nuestro contexto específico.

En el campo investigativo, los LBS's son servicios que determinan la posición del usuario por medio de múltiples técnicas usadas para establecer su ubicación, y usan la información captada para proveer aplicaciones personalizadas y servicios de valor agregado; son considerados frecuentemente como un subconjunto especial de los llamados "Servicios conocedores de su posición". Estos últimos pueden definirse como servicios que adaptan automáticamente su comportamiento, por ejemplo, filtrar o presentar información, a uno o más parámetros reflejando el contexto de aquello que es localizado. Estos parámetros se conocen como información de contexto.

Existen herramientas de geolocalización y seguimiento que brindan la posición del individuo -o del dispositivo móvil- y la de datos estáticos relacionados con este. Un ejemplo son las aplicaciones que muestran al individuo sitios de interés turístico o de utilidad. Estas herramientas no permiten un control mayor de la información mediante el software, dado que esta debe ser procesada directamente por un ser humano, bien sea el individuo que se desplaza o un tercero que recibe su información y que está interesada en ella. En este caso, el análisis de la posición o del desplazamiento que ha realizado el individuo seguido debe realizarse de forma manual, y se requiere que alguien esté dedicado a vigilar los cambios de posición que puedan ser de interés.

Esto es un desaprovechamiento de las posibilidades informáticas que los desarrollos del momento brindan, puesto que limita las posibilidades de análisis de la información y las hace casi completamente dependientes de uno o varios operarios humanos dedicados a ello. El costo en recursos humanos que esto acarrea es desproporcionado, y además puede conducir a que se pierda información de valor o que esta no sea advertida a tiempo.

En este sentido se propone la creación de un software capaz de configurar eventos. La configuración de eventos consiste en determinar uno o varios parámetros que deben cumplirse para que cierta transformación en la posición o en la relación movimiento – espacio del dispositivo sea considerara un fenómeno relevante, es decir, un evento. Identificar eventos permite discriminar los datos de localización y seguimiento al determinar cuáles son más importantes en determinado sentido y por tanto deben ser informados y alertar a destinatarios interesados en ellos. Identificar eventos puede servir para automatizar reglas y acciones, prevenir y/o alertar acerca de situaciones no deseadas o de riesgo , aplicar criterios para discriminar la información de mayor interés, evitar o detectar irregularidades en un proceso común, reaccionar rápidamente en una situación no prevista o fuera de lo normal, etc. En términos generales, el servicio de alerta

permite configurar reacciones automáticas previamente parametrizadas que puedan notificar a individuos interesados o a otros sistemas utilizando diversos canales informáticos.

Para el desarrollo de la herramienta de software descrita se utilizará J2ME, una de las herramientas de programación más difundida para dispositivos móviles, y que es la plataforma que funciona en la mayor cantidad de dispositivos móviles en el mercado. Esta tecnología se muestra como idónea dado que fue originalmente creada para lidiar con las restricciones asociadas en desarrollar aplicaciones para dispositivos pequeños. Por lo tanto, para crear la aplicación de software descrita, es necesario investigar y aprender sobre los fundamentos teóricos de la plataforma y el lenguaje que se usa para programar la misma. Con el objeto de que dicha investigación genere una utilidad práctica, se realizará un tutorial del lenguaje J2ME utilizando la plataforma educativa MeiWeb de la Escuela de Ingeniería de Sistemas e Informática.

Como parte del proceso de socialización de esta investigación, se realizara, en conjunto con el director de este proyecto, un articulo acerca de la concepción y desarrollo de este proyecto con el propósito de ser publicado en una revista de informática o comunicaciones.

1.3.2 Planteamiento del problema.

En el campo investigativo, tanto como en el comercial, se está trabajando ampliamente en los Servicios Basados en Localización (LBS´s) que determinan la posición del usuario por medio de múltiples técnicas usadas para establecer su ubicación, y usan la información captada para proveer aplicaciones personalizadas y servicios de valor agregado. Los dispositivos móviles dotados de GPS son herramientas que por sus características y capacidades y pueden utilizarse para diseñar herramientas de software capaces de ejecutar servicios basados en localización que brinden prestaciones de relativa complejidad. En este proyecto se propone el desarrollo de una herramienta de software que pueda ejecutarse en un dispositivo móvil dotado de GPS y que sea capaz de prestar LBS´s de relativa complejidad, enmarcados en el contexto de prestaciones de seguridad y monitoreo a personas, y dando cuenta de los alcances y limitaciones que se tienen al desarrollar software orientado a localización y seguimiento dentro de nuestro contexto específico.

1.4 VIABILIDAD

- Existen en el mercado dispositivos móviles con GPS integrado, que están disponibles a un costo accesible y tienen instalada la máquina virtual de Java lo cual permite la ejecución de programas realizados en el lenguaje j2me.
- La empresa de telefonía celular COMCEL ofrece servicios de internet inalámbrico a través de su red celular, utilizando protocolos y tecnologías como 3G, GPRS, EDGE, etc. Estos servicios pueden ser utilizados por cualquier particular a cambio de un costo accesible. Gracias al diseño de la aplicación el volumen de datos que es necesario transmitir es relativamente bajo, por lo que permite el uso de un plan de datos económico como el anteriormente mencionado.
- La empresa de telefonía celular COMCEL ofrece el servicio de A-GPS (GPS asistido) con el cual un GPS puede inicializarse y brindar información de su posición en tiempo inferior a un minuto. El uso del servicio de A-GPS para efectos de esta aplicación reduce considerablemente el tiempo necesario para inicializarla, haciéndola más eficiente y coherente con las necesidades de los posibles usuarios.

2 ESTADO DEL ARTE Y LA TÉCNICA

2.1 SERVICIOS BASADOS EN LOCALIZACION

La Asociación GSM, que es un consorcio de 600 Operadores de red GSM alrededor del mundo, define los servicios basados en localización como prestaciones que se proveen al usuario a petición y de modo personalizado, basándose en su posición actual. Ejemplos de este tipo de servicios aquellos que brindan información de restaurantes cercanos, hoteles, u otro contenido específicamente relacionado con la posición como por ejemplo mapas.

En el ámbito se distingue estrictamente entre los Servicios Basados en Localización (LBS) y los Servicios de Localización (LS). Los Servicios de Localización tienen que ver con la ubicación de personas u objetos y con hacer que los datos resultantes estén disponibles para actores externos. Un servicio de localización no implica el procesamiento de datos de localización en el sentido de filtrar o seleccionar información dependiente de la localización o realizar otras acciones de alto nivel (a diferencia de un LBS); el servicio de localización es responsable tan solo de la generación y entrega de datos de posición. Así, los Servicios de Localización contribuyen al funcionamiento de los LBS y pueden ser considerados como un sub-servicio de estos. Sin un servicio de localización, un usuario de un LBS tendría que ingresar los datos de su posición manualmente, lo cual puede ser un proceso tedioso y complicado, especialmente si se hace en dispositivos con interfaces gráficas limitadas. En conclusión, los LBS's y los servicios de localización se complementan.

Los Servicios Basados en Localización responden tres preguntas fundamentales: ¿Dónde estoy?, ¿Qué está a mi alrededor? y ¿Cómo llego allí?; dichos servicios determinan la posición del usuario por medio de múltiples técnicas usadas para establecer su ubicación, y usan la información captada para proveer aplicaciones personalizadas y servicios de valor agregado.

Existen dos enfoques para implementar servicios basados en localización:

- Procesar los datos de localización en un servidor y entregar resultados a un dispositivo.
- Obtener los datos de localización de una aplicación basada en un dispositivo que la usa directamente.

2.1.1 Determinar la localización del dispositivo.

Para descubrir la localización de un dispositivo, un LBS debe capturar la posición en tiempo real. Existen diversos métodos para hacer la captura de posición sus resultados varían, siendo unos más precisos que otros.

La localización puede expresarse de dos formas: en términos espaciales o mediante descripciones textales. Una localización espacial se expresa utilizando el sistema de coordenadas basado en latitud y longitud. La latitud se mide en grados con respecto a la línea del Ecuador, y la longitud se mide, también en grados, con respecto al meridiano de Greenwich. Un parámetro adicional puede ser la altitud, que se expresa en metros sobre el nivel del mar. A diferencia de la localización espacial, la descripción textual utiliza nombres para indicar el lugar en que se encuentra el punto, usando referentes como barrios, ciudades, códigos postales o nombres de calles.

2.1.2 Métodos de posicionamiento.

- Cell-Id: El Cell-Id es un código único que identifica el área cubierta por una BTS dentro de una red GSM. Cuando un individuo se conecta a la red mediante un dispositivo móvil es posible obtener el cell-id del área en que está y así identificar la antena que le está prestando servicio. Conociendo la posición de la antena es posible entonces identificar en que zona está el individuo, y su posición puede refinarse combinando otros criterios como la mayor o menor intensidad de la señal, que es directamente proporcional a la cercanía a la antena. Una BTS en zona urbana puede cubrir un área de entre 400 y 800 m, y en zona rural, de 1 a 3 km. Por tanto, la posición que se obtenga mediante este método siempre será una aproximación.
- GPS: El sistema de posicionamiento global (GPS) funciona gracias a una red de satélites diseñada para obtener posicionamiento espacial en cualquier parte del planeta, compuesta por 24 satélites orbitando alrededor de la Tierra. El sistema determina la posición de un receptor GPS calculando las diferencias en los tiempos que toman en llegar al dispositivo las señales enviadas desde varios satélites. Las señales del GPS están codificadas, y es el receptor GPS quien puede entenderlas. Un receptor GPS puede ir instalado en un teléfono celular, permitiendo así localizar al usuario que porta el teléfono.

El GPS es en la actualidad potencialmente el método con mayor precisión al determinar la localización de un dispositivo, con un margen de error que oscila entre 4 y 40 metros. Su utilización para localizar dispositivos móviles tiene como inconvenientes el costo extra del receptor GPS dentro del dispositivo, un aumento considerable en el consumo de energía por la operación del receptor, y la exigencia de un tiempo inicial de operación para que el receptor ubique correctamente los satélites de los cuales recibirá señales.

- Usar guías de posición de corto rango: En áreas relativamente pequeñas, como por ejemplo un edificio, una red de área local (LAN) puede proveer localización al prestar otros servicios, por ejemplo, equipos provistos adecuadamente pueden usar un trasmisor Bluetooth o Wi-Fi y varios receptores que actuarían como guías.

Adicionalmente, los métodos de localización pueden conectarse a un centro de posicionamiento móvil que provea una interfaz diseñada para que el dispositivo pregunte a la central la información de su posición.

En el campo investigativo, los LBS's son considerados frecuentemente como un subconjunto especial de los llamados "Servicios conocedores de su posición". Generalmente, este tipo de servicios son definidos como servicios que adaptan automáticamente su comportamiento, por ejemplo, filtrar o presentar información, a uno o más parámetros reflejando el contexto de aquello que es localizado. Estos parámetros se conocen como información de contexto. El conjunto de información de contexto potencial es categorizada, y puede ser subdividida en personal, técnica, espacial, social y física. Los contextos también pueden ser clasificados como primarios o secundarios:

- El contexto primario comprende cualquier clase de datos que puedan ser seleccionados de sensores como sensores de luz, micrófonos, acelerómetros, sensores de localización, etc.
- El contexto secundario se conforma por información de contexto de alto nivel, la cual resulta del trato de los datos arrojados como información de contexto primario mediante combinación, deducción o filtrado. La información de contexto secundario puede ser de mayor valor y complejidad que la información de contexto primario.

2.1.3 Estandarización.

El éxito de los servicios basados en localización está basado esencialmente en la disponibilidad de estándares apropiados, los cuales describan los protocolos, interfaces y API's para coordinar la interacción de los entes involucrados. Sin la

existencia de los estándares, estos entes tendrían que comunicarse a través de protocolos y tecnologías privativas, lo cual dificultaría la competencia, el mercado abierto y el éxito de los LBS en general.

2.2 COMPUTACION MOVIL

2.2.1 Introducción.

Se entienden por computación móvil los sistemas de cómputo que pueden ser desplazados fácilmente y cuyas capacidades computacionales puedan ser usadas mientras se mueven. Como ejemplo tenemos a los computadores portátiles, asistentes personales (PDAs) y teléfonos celulares. Los sistemas de cómputo móviles se diferencian de los demás sistemas tradicionales de cómputo en las tareas que se prevé que ejecuten, la forma en que están diseñados y el modo en que se operan. Ejemplos de estos son aspectos como la conectividad inalámbrica, tamaño reducido, la naturaleza móvil de su uso, las fuentes de alimentación (baterías, celdas solares, etc.), las funcionalidades que son peculiarmente adaptadas para un uso móvil, etc.

Los adelantos tecnológicos en materia de hardware y software han permitido la creación de dispositivos de cómputo móviles con capacidades similares a los sistemas de cómputo tradicionales. Esto ha causado el surgimiento de un nuevo paradigma computacional conocido como Computación Móvil. Es de resaltar que el término utilizado ha sido “computación móvil” y no “computación inalámbrica” ya que existe una generalizada y equivocada tendencia a establecer relaciones de implicación entre los términos inalámbrico y móvil.

2.2.2 Móvil e inalámbrico.

La definición de móvil e inalámbrico varía de persona a persona y de organización a organización. En muchos casos ambos términos son usados como sinónimos, pero representan conceptos diferentes.

Móvil es la habilidad de estar en movimiento; así, en un dispositivo móvil su localización no se considera fija.

Inalámbrico se refiere a la trasmisión de voz y datos en ondas de radio, lo cual permite comunicación sin tener una conexión física. Los dispositivos inalámbricos aquellos que usan una red inalámbrica para enviar o recibir datos.

Si bien es cierto que la utilización de redes inalámbricas fortalece el potencial de las aplicaciones móviles, es totalmente errado afirmar que el hecho de que una aplicación haga uso de tecnologías inalámbricas la clasifique como una aplicación móvil. Por ejemplo, una aplicación ejecutada desde un computador de escritorio, que hace uso de una red WLAN, no es una aplicación móvil, pues un computador de escritorio no presenta las características anteriormente mencionadas y que caracterizan la gama de dispositivos móviles.

2.2.3 Ventajas y desventajas.

2.2.3.1 Ventajas.

Como es de esperarse, la computación móvil ofrece una gran cantidad de ventajas. A continuación se enumeran algunas de las más relevantes:

- Portabilidad de hardware, software y datos, y acceso a información en cualquier momento desde cualquier lugar.
- Optimización de procesos de captura de datos críticos – procesos que involucran trabajo de campo.
- Aprovechamiento en ámbitos empresariales de dispositivos móviles de uso personal, con los cuales un gran porcentaje de personas en el mundo se encuentran familiarizadas, como lo son teléfonos celulares, PDAs y buscapersonas.
- Explotación de los distintos sistemas de redes de comunicaciones y tecnologías de comunicación existentes, dando pie al desarrollo de aplicaciones multicanal.
- Creación de redes personales temporales que no requieran de una infraestructura definida.

2.2.3.2 Desventajas.

- Ancho de banda insuficiente: El acceso a internet móvil es generalmente más lento que las conexiones directas por cable, usando tecnologías como GPRS y EDGE y de las más recientes como 3G. Estas redes están disponibles usualmente dentro de la telefonía celular comercial.
- Estándares de seguridad: Cuando se trabaja de forma móvil, se está sujeto a la seguridad de redes públicas.
- Consumo de energía: Cuando una fuente de alimentación continua o un generador portátil no está disponible, los dispositivos móviles dependen

únicamente de su batería. Si tenemos en cuenta que una de las características de los dispositivos móviles es su reducido tamaño, entonces tendremos un tiempo de batería relativamente corto.

- Interferencia en la transmisión: El clima, terreno y el rango que hay desde el punto de señal más cercano pueden interferir con la recepción de señal, así como también las estructuras grandes, techos y túneles.
- Peligros potenciales a la salud: Muchos de los accidentes automovilísticos están relacionados con conductores que se encontraban usando un dispositivo móvil. Además, los teléfonos celulares pueden interferir con algunos dispositivos médicos sensibles.
- Uso del dispositivo: Las pantallas y teclados tienden a ser pequeños, y en la mayoría de los casos, difíciles de usar. Métodos alternativos como el habla o el reconocimiento de escritura, requieren entrenamiento.

2.2.4 Algunas aplicaciones de la computación móvil.

- Servicios de emergencia: Prestar rápida atención, localización y alerta en tiempo crítico, en lugares y terrenos alejados, con o sin intervención humana es definitivamente un campo donde los dispositivos móviles pueden llegar a ser vitales a la hora de reaccionar ante una situación de emergencia.
- Manejo de inventarios: La recolección de información de inventario físico y el control y supervisión de estos se puede lograr en tiempo óptimo gracias al uso de dispositivos móviles. La actualización del estado de inventario en tiempo real es una herramienta fundamental para poder ofrecer servicios de manera oportuna, real y actuar rápidamente ante problemas críticos o situaciones inesperadas (agotamiento de inventario, depreciación, mercancía en mal estado, etc.).
- Manejo de pacientes: La posibilidad de tener el historial médico de los pacientes, de forma instantánea, monitorear en campo signos vitales y poder trasmitir esta información, son habilidades que indudablemente colaboran a los profesionales de la salud a prestar servicios médicos más eficientemente y con un mayor control de su información.
- Ventas directas: Consultar inventarios, realizar pedidos de manera instantánea, revisar ordenes de despacho, devoluciones, cobrar correctamente la cartera, obtener precios y descuentos, etc., de forma instantánea para los vendedores de campo de las empresas de venta de productos, ofrece una ventaja de mercado para aquellas compañías que optan por apoyarse en los sistemas móviles para realizar su labor de fuerza de ventas.
- Servicio a clientes: La asesoría, servicio técnico y consultoría es un área en donde la computación móvil es vital. La consulta a bancos de información, bases

de datos inteligentes, acopio de información actualizada y consulta de especialistas, es sólo una pequeña muestra de todo lo que puede impactar esta tecnología, sin mencionar la posibilidad de contacto permanente con el cliente.

- Personal móvil en oficinas: No es raro encontrar a personal que, pese a encontrarse siempre en el mismo edificio, se muda de lugar con frecuencia para, por ejemplo, dar soporte técnico al personal o revisar proyectos. La computación móvil no sólo les permite ser localizados con facilidad, sino que también les auxilia en la consulta de datos que por lo regular estarían en su oficina.
- Profesionales viajeros. Estos son ejemplos de usuarios potenciales que pueden aprovechar esta herramienta: contadores, gerentes regionales que integran metas empresariales, dirigentes corporativos que requieren información actualizada, periodistas, visitantes médicos, etc.
- Manejo de sucursales. Permiten consolidar la información de las sucursales y oficinas dependientes con la que corresponde a la base de la sede principal
- Grupos de trabajo. La globalización y expansión de empresas hace que sea cada vez más común abordar proyectos con el personal adecuado, y esto puede requerir integrar un grupo de trabajo con profesionales establecidos en diferentes lugares del país o del mundo.

2.3 SISTEMAS DE INFORMACION GEOGRÁFICA

Un Sistema de Información Geográfica (GIS, en su acrónimo inglés, *Geographic Information System*) es una integración organizada de hardware, software y datos geográficos diseñada para almacenar, editar, otorgar y analizar información referenciada geográficamente. Estos sistemas reflejan como un modelo una parte de la realidad que ha sido referida a un sistema de coordenadas terrestres y construida para satisfacer unas necesidades concretas de información. Los GIS son herramientas que permiten a los usuarios crear consultas personalizadas, analizar la información espacial, editar datos asociados a ella, presentar mapas y otorgar resultados específicos de todas estas operaciones[1].

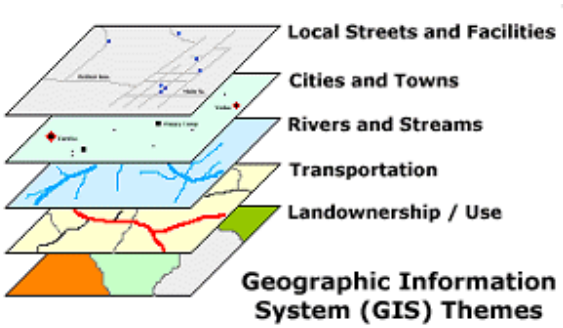
La tecnología de los Sistemas de Información Geográfica puede ser utilizada, entre otros, para gestión de recursos y activos, la recolección de información para investigaciones en ciencias humanas (arqueología, geografía, sociología), mediciones ambientales, actividades de publicidad y mercadeo, entre otros.

[1] GOODCHILD, Michael F y LONGLEY, Paul A (2005). Geographic Information Systems and Science. West Sussex, England. John Wiley & Sons Ltd. P 4

2.3.1 Funcionamiento.

Un Sistema de Información Geográfica usualmente suele mostrar la información en capas concretas, cada capa representa una categoría de información, permitiendo realizar análisis multicriterio complejos. Estas capas normalmente son representadas visualmente y superpuestas unas con otras sobre la misma localidad.

Ilustración 1. Sistemas de Información Geográfica.



Fuente: ITC. University Of Twente

Los GIS funcionan como bases de datos con información geográfica que se encuentra asociada a objetos gráficos de un mapa digital. De esta forma, señalando un objeto se conocen sus atributos e, inversamente, preguntando al GIS por un objeto se puede saber su localización en la cartografía. Ya que poseen la característica de almacenar y manipular información georreferenciada, cada GIS es construido normalmente con una serie de procesos analíticos, operaciones y visualizaciones específicas seleccionadas de acuerdo con los intereses particulares que el sistema ha de satisfacer. Así, cada GIS es único y diferente de los demás y entregará resultados de la misma manera, respondiendo a la necesidad específica, ya sea en forma de mapas, gráficos, reportes, etc.

Las principales cuestiones que puede resolver un Sistema de Información Geográfica, ordenadas de menor a mayor complejidad, son:

- Localización: preguntar por las características de un lugar concreto.
- Condición: el cumplimiento o no de unas condiciones impuestas al sistema.
- Tendencia: comparación entre situaciones temporales o espaciales distintas de alguna característica.

- Rutas: cálculo de rutas óptimas entre dos o más puntos.
- Pautas: detección de pautas espaciales.
- Modelos: generación de modelos a partir de fenómenos o actuaciones simuladas.

Por ser tan versátiles, el campo de aplicación de los Sistemas de Información Geográfica es muy amplio, pudiendo utilizarse en la mayoría de las actividades con un componente espacial. La profunda revolución que han provocado las nuevas tecnologías ha incidido de manera decisiva en su evolución.

2.3.2 Mapas en línea.

Aprovechando el auge de la computación en la nube en los últimos años, compañías y grupos independientes han creado aplicaciones que ofrecen mapas al público en general mapas por medio de una interfaz web, sitios como *Google Maps* y *Bing Maps* otorgan increíbles cantidades de información geográfica de todo el mundo.

Algunos de ellos, como *Google Maps* y *OpenLayers,* presentan al usuario, un conjunto de herramientas de software y librerías, para permitir la creación de aplicaciones personalizadas, usando la infraestructura base creada por ellos. Estas herramientas ofrecen comúnmente mapas de calles, imágenes satelitales y/o áreas, búsquedas textuales de lugares y opciones para generar o calcular rutas.

Con el tiempo y la popularidad que han adquirido estos servicios en línea de mapas, han comenzado por adquirir más y más funcionalidades y características de GIS, donde los usuarios pueden hacer anotaciones en los mapas y compartirlos con otros usuarios.

3 TECNOLOGÍAS UTLIZADAS EN EL DESARROLLO DEL PROYECTO

3.1 *GOOGLE MAPS*

Google Maps (llamado durante un tiempo *Google Local*) es básicamente una tecnología y servicio de mapeo web ofrecido por Google, de forma gratuita, que puede ejecutar diversas prestaciones basadas en mapeo, las cuales incluyen el sitio web de *Google Maps, Google Ride Finder, Google Transit*, y mapas insertados en otros sitios webs mediante la API de *Google Maps*. Ofrece mapas de vías y planeadores de rutas para viajes en vehículos privados, transporte público o caminatas, así como localización de negocios en las ciudades para numerosos países. De acuerdo con uno de sus creadores (Lars Rasmussen) *Google Maps* es "una manera de organizar la información geográfica del mundo".

Google Maps usa la proyección de *Mercator*, por lo que no puede mostrar las áreas alrededor de los polos. Un producto relacionado con *Google Maps* es *Google Earth*, un programa independiente que puede ejecutarse en Microsoft Windows, Mac OS X, Linux, SymbianOS, y iPhone OS y que ofrece mayores prestaciones de vistas satelitales alrededor del globo, incluyendo aquellas que muestran las regiones polares.

3.1.1 Implementación.

Como muchas otras aplicaciones web, *Google Maps* utiliza JavaScript de modo extensivo. Como el usuario se desplaza sobre el mapa, las secciones de la cuadrícula que deben mostrarse son descargadas del servidor e insertadas en la página web. Cuando un usuario busca negocios, los resultados son descargados e insertados en el panel lateral y en el mapa, así, la página no debe recargarse. Las posiciones son dibujadas de modo dinámico al localizarlas con un dibujo correspondiente a un alfiler rojo encima de la imagen.

El sitio utiliza JSON en vez de XML para transmitir datos, por razones de eficiencia. Esta técnica corresponde con el espectro de la metodología Ajax.

3.1.2 Posibilidades de expansión y personalización.

Usando el motor principal y las imágenes de mapas y fotografías satelitales alojadas en Google, muchas herramientas pueden introducir marcadores personalizados de localización, sistemas de coordenadas y metadatos, e incluso personalizar imágenes de mapas dentro de la interfaz de *Google Maps*. La

herramienta de inserción de *scriptsGreasemonkey* ofrece un gran número de *scritps* para personalizar información de *Google Maps* por parte de los usuarios.

Sitios webs que permiten compartir fotos, como Flickr, se combinan con *Google Maps* para conectar fotos con su información geográfica y localizarlas gráficamente.

3.1.3 API de *Google Maps*.

Google creó la API de *Google Maps* para permitir a los desarrolladores integrar esta herramienta en sus sitios web, y utilizar en ella su propia información de localización. Este es un servicio libre y comúnmente no contiene publicidad, sin embargo, en los Términos de Uso Google se reserva el derecho de colocarla.

Usando la API de *Google Maps* es posible introducir completamente el sitio web dentro de un sitio web externo. Para esto, los desarrolladores deben solicitar una llave API, la cual está ligada a la URL del sitio web. La llave API de *Google Maps* no es solicitada para la versión 3 de la API. Para crear una interfaz personalizada que muestre los mapas es necesario utilizar en la página el código JavaScript de Google, así como usar JavaScript para crear funciones y añadir puntos al mapa.

Google Maps promueve activamente el uso comercial de su API. Algunos de sus primeros usuarios a gran escala son los sitios dedicados al comercio y búsqueda de finca raíz.

3.2 SISTEMA DE POSICIONAMIENTO GLOBAL (GPS)

3.2.1 Introducción.

El *Global Positioning System* (GPS) o Sistema de Posicionamiento Global (más conocido con las siglas GPS, aunque su nombre correcto es NAVSTAR-GPS) es un sistema global de navegación por satélite (GNSS) desarrollado por el departamento de defensa de los Estados Unidos de América, el cual permite a un dispositivo electrónico provisto de un receptor GPS determinar su posición al suministrarle información que le hace posible calcular sus coordenadas geográficas.

El GPS funciona mediante una red de 27 satélites (24 operativos y 3 de respaldo) en órbita sobre el globo, a 20.200 km, con trayectorias sincronizadas para cubrir

toda la superficie de la Tierra ubicando 4 satélites sobre cada uno de los 6 planos orbitales. Debido a esta organización se pueden apreciar de 4 a 10 satélites GPS desde cualquier punto de la Tierra[2].

Un receptor GPS determina su posición al calcular los tiempos de llegada de las señales enviadas por los satélites GPS. Cada satélite envía un mensaje que contiene:

- La hora exacta en que el mensaje fue enviado.
- Información orbital del satélite (sus Efemérides).
- El estado del sistema en general y una medida aproximada de cada una de las órbitas de los satélites GPS. El conjunto de estas medidas es conocido como 'el almanaque'.

El receptor utiliza la información del mensaje para determinar su tiempo de transmisión, de esta forma calcula la distancia que existe entre cada satélite y el receptor. Estas distancias y la ubicación de cada satélite en el espacio hacen posible la trilateración para calcular la posición del receptor en términos de longitud, latitud y altitud.

En teoría tres satélites son suficientes para determinar la posición del receptor GPS, pero un error pequeño en la sincronización de los tiempos, multiplicado por la velocidad con que viajan las señales de los satélites (que es la velocidad de la luz) podría resultar en un error muy grande para el sistema. Para evitarlo, el receptor GPS utiliza 4 satélites, o más donde sea posible, para efectuar una medición mucho más precisa.

3.2.2 Funcionamiento.

1.La posición de los satélites es trasmitida en el mensaje con base en las efemérides. La colección de efemérides de toda la constelación de satélites GPS se completa cada 12 minutos y se guarda en el receptor GPS.

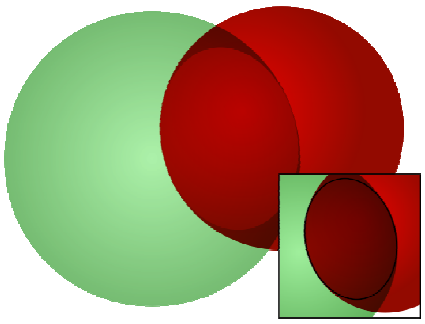
2.El receptor GPS mide su distancia respecto de cada satélite y usa esa información para calcular su posición. El cálculo se hace determinando el tiempo en que la señal tarda en llegar al receptor. Conocido ese tiempo y asumiendo que la señal viaja a la velocidad de la luz (salvo algunas correcciones que se aplican), se puede calcular la distancia entre el receptor y el satélite.

[2] EL-RABBANY, Ahmed(2006). Introduction to GPS: The Global Positioning System. Norwood, Artech house inc. Segunda edicion. Pp 2-3

3.Cada satélite indica que el receptor se encuentra en un punto en la superficie de la esfera, con centro en el propio satélite y de radio la distancia total hasta el receptor

4.Obteniendo información de dos satélites se nos indica que el receptor se encuentra sobre la circunferencia que resulta cuando se intersectan las dos esferas.

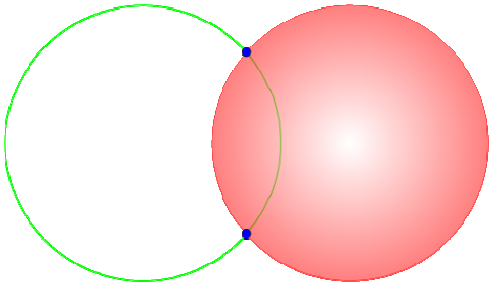
Ilustración 2. Volumen resultante de la intersección de dos esferas



Fuente: Wikimedia Commons.

5. Al realizar los cálculos de un tercer satélite e intersectar su esfera con la intersección resultante de las 2 primeras señales, el resultado arroja 2 puntos. Uno de ellos se puede descartar porque ofrece una posición absurda fuera del globo terráqueo, y el otro corresponde a la ubicación del receptor GPS. De esta manera se obtiene la posición en las 3 coordenadas espaciales. Sin embargo, dado que el reloj que incorporan los receptores GPS no está sincronizado con los relojes atómicos de los satélites GPS, los dos puntos determinados no son precisos.

Ilustración 3. Intersección de una esfera y un círculo



6. Con la información de un cuarto satélite se elimina el error proveniente de la falta de sincronización entre los relojes de los receptores GPS y los relojes de los satélites. Y es en este momento cuando el receptor GPS puede determinar una posición en términos de latitud, longitud y altitud con un alto grado de precisión.

3.2.3 Componentes del sistema GPS.

3.2.3.1 Componente espacial.

El componente espacial está formado por los satélites que están sobre los planos orbitales. Estos planos están centrados sobre el Ecuador, cada satélite orbita aproximadamente a 26,600 Km de altura y realiza 2 orbitas completas cada día sideral (aprox. 23 horas, 56 minutos y 4.091 segundos). Cada satélite tiene un tiempo promedio de vida de 7.5 años.

3.2.3.2 Componente de control.

El componente de control lo conforman:

- Una estación de control maestro (MCS)
- Una estación de control de respaldo
- 4 antenas terrestres dedicadas.
- 6 estaciones de monitoreo.

Toda esta infraestructura de control esta diseñada para mantener registro de las actividades de los satélites en órbita, sincronizar los relojes atómicos a bordo de cada uno de los satélites y ajustar las efemérides de los satélites para

corresponder a su programación de órbita. Al efectuar correcciones en la órbita de un satélite, las señales que este envía a los receptores no son adecuadas para los cálculos de posición, por ello, desde tierra, el satélite cambia su estado (el estado que trasmite a los receptores GPS) como "*unhealthy*" lo cual informa a los receptores GPS que no deben usarlo para sus cálculos.

3.2.3.3 Componente de usuario.

A este componente pertenecen los usuarios del sistema GPS, que incluyen industria militar, investigación científica, el mundo comercial y civil, entre otros. Los receptores GPS que los usuarios tienen en sus dispositivos habitualmente están provistos de antenas ajustadas para la frecuencia específica en la que trasmiten los satélites GPS. Para uso comercial, los fabricantes de componentes electrónicos han provisto a la industria diversos receptores GPS de pequeño tamaño y bajo consumo, ideales para tener dispositivos de mano con acceso al sistema GPS.

Adicionalmente, dispositivos con receptores GPS y tecnologías inalámbricas de conexión han sido difundidos para ser usados con equipos no provistos con receptores, como celulares, cámaras digitales, asistentes personales (PDAs) entre otros. Tal es el caso de los receptores de GPS dotados con conexión Bluetooth.

3.2.4 Aplicaciones.

3.2.4.1 Civiles.

- Navegación terrestre vehicular (y peatonal), marítima y aérea. Bastantes automóviles lo incorporan en la actualidad, siendo de especial utilidad para encontrar direcciones o indicar la posición a la grúa.
- Teléfonos móviles.
- Topografía y geodesia.
- Localización agrícola (agricultura de precisión), ganadera y de fauna.
- Salvamento y rescate.
- Deporte, acampada y ocio.
- Para localización de enfermos, discapacitados y menores.
- Aplicaciones científicas en trabajos de campo.
- Se utiliza para rastreo y recuperación de vehículos.
- Sistemas de gestión y seguridad de flotas.

3.2.4.2 Militares.

- Navegación terrestre, aérea y marítima.
- Guiado de misiles y proyectiles de diverso tipo.
- Búsqueda y rescate.
- Reconocimiento y cartografía.
- Detección de detonaciones nucleares.

3.2.5 GPS asistido (A-GPS).

El GPS Asistido es un servicio que provee una mejora sobre el funcionamiento estándar del sistema GPS usando un canal de comunicación alternativo al que el receptor GPS normalmente usaría, es decir, las señales captadas por la antena conectada al receptor. Este servicio no evita que el receptor GPS reciba y procese las señales de los satélites, sino que facilita la tarea y minimiza la cantidad de tiempo e información requerida para el correcto funcionamiento, aun con intensidades de señal muy bajas.

Para calcular la posición, un receptor GPS debe primero encontrar, identificar y adquirir las señales de cada uno de los satélites y decodificar la información que viene en ellas. Cada satélite GPS trasmite en diferente frecuencia. Debido a la velocidad de movimiento de los satélites se experimenta el 'efecto doppler', el cual induce márgenes de error de las señales. Debido al escaneo de frecuencias que debe hacer el receptor para poder identificar las señales que vienen de cada satélite, sumado a múltiples ensayos que el receptor hace para poder asegurar la posición de los satélites antes de poder calcular la suya propia, se produce un retardo en el inicio de operación normal del receptor GPS conocido como 'calentamiento'. Este proceso puede tardar entre 1 y 10 minutos dependiendo de las condiciones de señal, movimiento y otros factores que afectan el proceso.

El servicio A-GPS provee al receptor GPS la información de señal satelital que este requiere para calcular su posición, entre esta, las frecuencias en que están trasmitiendo actualmente los satélites, las efemérides de estos, etc. Con esta información previamente calculada, el 'calentamiento' pasa a ser un proceso de milisegundos, en lugar de minutos. El conocer la frecuencia exacta a la que está trasmitiendo el satélite permite maximizar la potencia de trasmisión de las señales y lograr que, aún con poca intensidad de señal, sea posible para el receptor GPS calcular su posición de forma correcta[3].

[3] DIGGELEN, Frank van (2009). A-GPS: Assisted GPS, GNS, and SBAS. Boston, Artech House. Pp 2-3

Existen diversos métodos por los cuales el servicio A-GPS obtiene la información necesaria para su funcionamiento, podemos entonces clasificarlos como:

- Métodos *Online*: El A-GPS utiliza una conexión directa a un servidor localizado en una red de información en la que se encuentra vinculado, este servidor otorga a petición del dispositivo la información de los satélites. La tendencia en este tipo de métodos es tener un servidor colocado en internet para tal fin, y acceder a internet a través, ya sea de las difundidas redes celulares (GPRS), o conexiones Wi-fi o Bluethooth que provean acceso a internet. Normalmente, debido a la topología estática de la red a la que está vinculado el dispositivo, es posible tener una referencia cercana de la posición del dispositivo y conocer entonces cuales satélites están visibles desde esa ubicación. Un ejemplo común, es en las redes celulares, conocer la posición de la estación base (antena) que está dado el servicio de telefonía al dispositivo; esta información se envía al servidor A-GPS y este calcula qué satélites están disponibles para esa ubicación.
- Métodos *Off-Line*: El mecanismo usado por estos métodos se basa en obtener, cuando sea posible, la información de un servidor y almacenarla localmente dentro de la memoria del dispositivo para cuando el servicio de GPS del dispositivo sea solicitado, poder brindarle la información para que el proceso sea asistido (A-GPS). Esta información tiene un tiempo de vida válido, pues los datos se vuelven obsoletos con el tiempo.

Algunos dispositivos avanzados utilizan ambos métodos dependiendo del estado de conexión con el servidor.

3.3 JAVA Y J2ME

3.3.1 Java.

Java es un lenguaje de programación de propósito general, basado en clases y orientado a objetos. Fue diseñado para ser sencillo y con una sintaxis muy similar a uno de los lenguajes mas adoptados que existían desde la creación de Java hasta hoy, el lenguaje C/C++. Con una organización un poco diferentes y obviando algunos aspectos del mismo, Java fue concebido como un lenguaje para el área de producción y no para el campo investigativo.

El lenguaje Java se creó con cinco objetivos principales:

- Usar la metodología de la programación orientada a objetos.

- Permitir la ejecución de un mismo programa en múltiples sistemas operativos.
- Incluir por defecto soporte para trabajo en red.
- Ejecutar código en sistemas remotos de forma segura.
- Fácil de usar y toma lo mejor de otros lenguajes orientados a objetos, como C++.

Para conseguir la ejecución de código remoto y el soporte de red, los programadores de Java a veces recurren a extensiones como CORBA (*Common Object Request Broker Architecture*), *Internet Communications Engine* u OSGi respectivamente.

3.3.1.1 Independencia de la plataforma.

Los programas escritos en el lenguaje Java puedan efectuarse igualmente en cualquier sistema operativo, de aquí el famoso lema de la compañía '*write once, run everywhere* '[4].

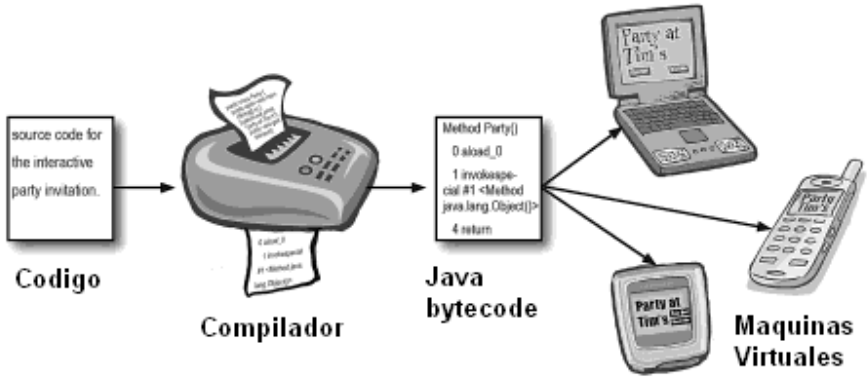
Para lograr este objetivo se compila el código fuente escrito en lenguaje Java. El compilador genera un código conocido como '*bytecode'* (específicamente '*Java bytecode'*), el cual está formado por instrucciones de máquina simplificadas específicas de la plataforma Java. El *bytecode* es un código intermedio entre el código fuente original y el código de maquina del hardware que va a ejecutarlo. El *bytecode* es ejecutado entonces por la máquina virtual (JVM).

3.3.1.2 Máquina virtual de Java.

La máquina virtual de Java es un programa escrito en código nativo de una plataforma, que interpreta y ejecuta código *bytecode* de Java. La JVM (*Java Virtual Machine*) es entonces escrita y compilada para un sistema operativo específico, y tiene la responsabilidad de tomar el *bytecode* y ejecutarlo sobre el sistema operativo en el cual se encuentra y para el cual fue creada.

[4]Trad.: Escríbelo una vez, córrelo donde quieras.

Ilustración 4. Ciclo de ejecución de los programas Java



Adicionalmente se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o *threads*, la interfaz de red) de forma unificada. La JVM interpreta el *bytecode* convirtiéndolo a instrucciones maquina del código nativo por el compilador JIT (*Just In Time*) que viene incluido en la JVM

Sin embargo, mientras el programa en Java es independiente de la plataforma gracias a la arquitectura de máquinas virtuales, el código de la máquina virtual no; cada sistema operativo que soporte Java debe tener una versión propia de su JVM[5].

3.3.1.3 Plataformas Java.

Una plataforma Java es un conjunto de programas relacionados distribuidos como un único paquete por Sun Microsystems, para permitir el desarrollo y ejecución de programas escritos en el lenguaje de programación Java. Cada plataforma corresponde, no a un tipo de procesador o sistema operativo, sino a una JVM y un compilador con una serie de librerías que son específicas para un gama de sistemas operativos y hardware, de forma que los programas creados para un plataforma especifica funcionen en todos los dispositivos que alberge esta plataforma.

Las plataformas actuales son:

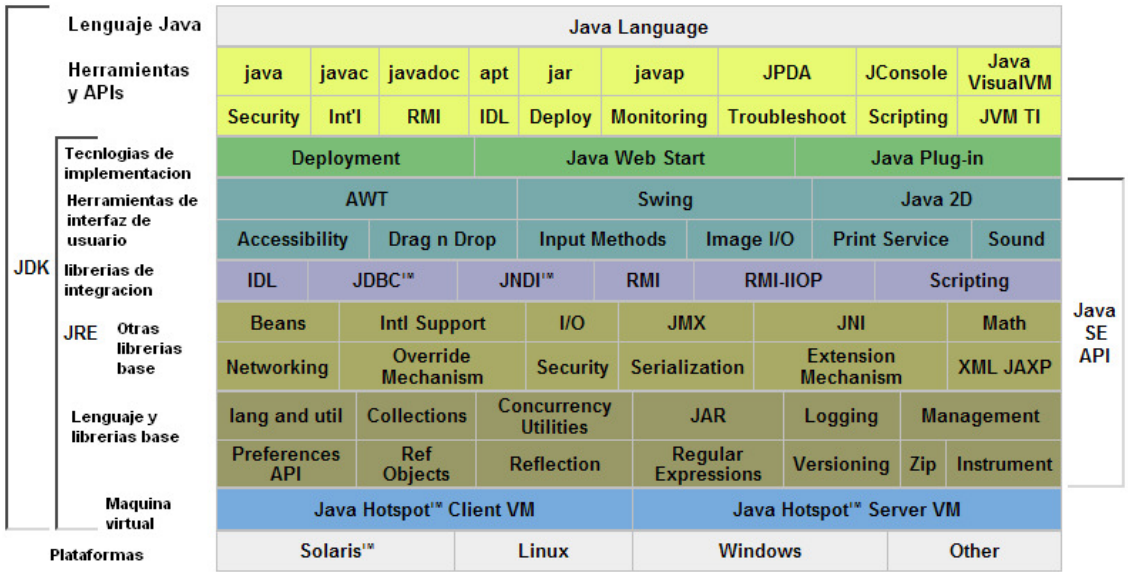
• Java Card: Esta plataforma permite la creación y ejecución de pequeñas aplicaciones sobre tarjetas inteligentes, *'smart cards'*, y dispositivos similares.

[5] KEOGH, James(2003). J2ME: The Complete Reference. Osborne, McGraw-Hill. P 6

- Java ME (*Micro Edition*): Alberga diversos tipos de dispositivos que se destacan por ser limitados en las prestaciones que ofrecen en comparación con ordenadores comunes, está diseñada para dispositivos móviles y otros con bajas capacidades de memoria, almacenamiento y procesamiento.
- Java SE (*Standard Edition*): Para propósito general en computadores personales (PC), servidores y dispositivos similares
- Java EE (*Enterprise Edition*): Es la misma plataforma SE, pero con diversas funcionalidades y APIs para realizar aplicaciones cliente servidor multi-capa.

Cada plataforma se distribuye con diversos programas, donde cada uno provee algunas funcionalidades específicas. Por ejemplo, el compilador de Java que convierte el código Java en *bytecode* Java pertenece a un subconjunto o kit de programas dentro de la plataforma denominado '*Java Development Kit'* o JDK, que es un kit con los programas necesarios para desarrollar aplicaciones Java; otro, el JRE o '*Java Runtime Enviorment*' viene con la JVM y el compilador JIT, este conjunto entonces es el encargado de ejecutar los programas que están en *bytecode* sobre el sistema operativo y hardware para el cual están construidos.

Ilustración 5. Componentes de las plataformas Java



Fuente: Java.

3.3.1.4 Licenciamiento.

Sun Mycrosistems anunció en la conferencia JavaOne en 2006 que Java sería gratuito y software libre. Desde ese entonces ha liberado parte de sus plataformas, gradualmente, debido a que secciones del código no eran de propiedad exclusiva de Sun Microsystems.

3.3.2 J2ME: Java 2 Micro Edition.

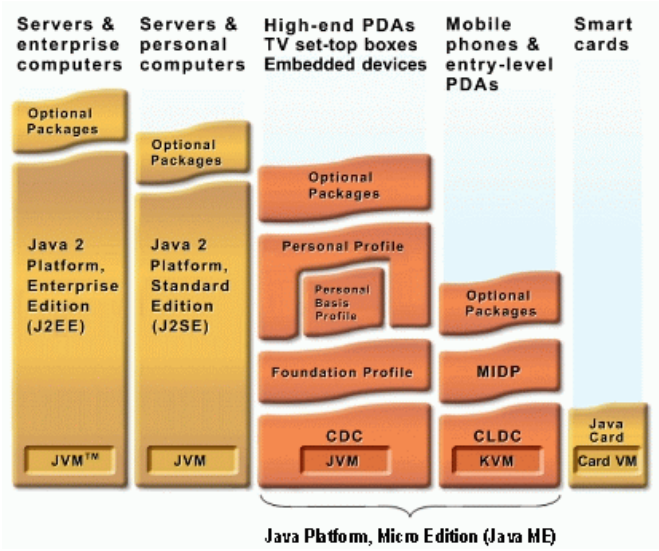
J2ME es una colección de tecnologías y especificaciones que conforman una plataforma apta para las necesidades de dispositivos como PDA's, sistemas embebidos, dispositivos de hogar y teléfonos celulares.

La tecnología Java ME (ahora conocido como J2ME), fue originalmente creada para lidiar con las restricciones asociadas en desarrollar aplicaciones para dispositivos pequeños. Esta plataforma permite construir un entorno de ejecución de Java (JRE) que se ajuste a los requerimientos de un dispositivo en particular del mercado.

La Tecnología Java ME basa en tres elementos:

- Configuración: Una configuración abarca una amplia gama de dispositivos con características similares en recursos, capacidad de procesamiento e interfaz. Cada configuración provee las librerías básicas y características de la maquina virtual específicas para la gama de dispositivos que cubre la configuración. J2ME cuenta con 2 configuraciones disponibles, una para pequeños dispositivos llamada *Connected Limited Device Configuration* (CLDC) donde entran PDA's, teléfonos celulares y similares y *Connected Device Configuration* (CDC) para dispositivos con mayores capacidades.
- Perfil: Un perfil es una serie de APIs que provee soporte específico para un conjunto de dispositivos. Es en el perfil donde se definen las interfaces de usuario para construir aplicaciones con interacción humana.
- Paquetes adicionales: Son conjuntos de APIs específicos de ciertas tecnologías adicionales que no pertenecen a una configuración o a un perfil. Son tecnologías comunes que no son asociadas directamente al tipo de dispositivo, sino que son características que pueden estar o no en un dispositivo sin que este deje de cumplir su propósito general, por ejemplo Bluetooth, mensajería (SMS), localización, entre otras.

Ilustración 6. Plataforma de Java ME

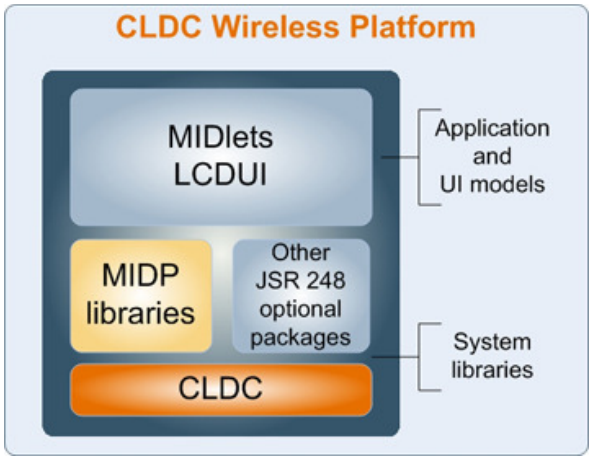


Fuente: Java.

3.3.2.1 Connected Limited Device Configuration (CLDC).

Esta configuración está diseñada para dispositivos con recursos limitados de memoria, almacenamiento, procesamiento y capacidades gráficas. Sobre esta configuración se especifican diversos perfiles que definen una serie de APIs de alto nivel que vendrán a definir el comportamiento de las aplicaciones que se ejecuten. Una combinación que ha tenido gran aceptación y popularidad, es combinar CLDC con el perfil *Mobile Information Device Profile* (MIDP), esta combinación constituye un entorno de ejecución para dispositivos móviles y otros dispositivos con características similares.

Ilustración 7. Plataforma CLDC y MIDP



Fuente: Java

3.3.2.2 Mobile Information Device Profile (MIDP).

Actualmente en la versión 3 (MIDP 3.0), este perfil está diseñado para dispositivos móviles, teléfonos celulares y similares. Cuenta con una API para visualización GUI (*Grafical User Interface*) llamada LCDUI y un API básico de juegos 2D, las aplicaciones creadas para usar este perfil son conocidas como MIDlets. La gran mayoría de teléfonos celulares vienen provistos con una implementación de MIDP y es el estándar de facto para los juegos de teléfonos celulares descargables de internet.

3.3.2.3 Máquina Virtual KVM.

J2ME cuenta con dos implementaciones de máquina virtual. Estas son la KVM (*Kylo Virtual Machine*) y la CVM (*Compact Virtual Machine*). La KVM es la máquina virtual Java más reducida desarrollada por Sun Mycrosistems y está orientada a dispositivos con bajas capacidades de procesamiento, esta máquina virtual es la que se utiliza en la configuración CDLC. La CVM es también una máquina virtual Java reducida y está orientada a dispositivos electrónicos con procesadores de 32 bits de gama alta y memoria total de 2MB o mayor.

3.3.2.4 Paquetes adicionales.

Adicionales a los perfiles, J2ME tiene unos paquetes opcionales que agregan funcionalidades específicas a uno o más perfiles. Algunos de los más conocidos son:

- *Information Module Profile (IMP). JSR 195*: Proporciona un entorno de aplicación para dispositivos embebidos que no tiene grandes capacidades gráficas o con recursos limitados de alguna otra manera: paneles de emergencia, parquímetros, sistemas de alarma domésticos y similares.
- *Multimedia API (MMAPI) JSR 135*: Este paquete extiende la funcionalidad de la plataforma J2ME incorporando soporte de audio, vídeo y otros tipos de datos multimedia basados en tiempo a dispositivos de recursos limitados.
- *Wireless Messaging API (WMA) JSR 120 JSR 205*: El API para mensajería sin cables proporciona acceso independiente de plataforma a recursos de comunicación sin cable como la mensajería SMS (*Short Message Service*).
- *Bluetooth API JSR 082*: Proporciona un estándar para la creación de aplicaciones Bluetooth, de forma que las aplicaciones desarrolladas con el paquete opcional puedan ejecutarse utilizando esta tecnología.
- *Location API for J2ME JSR 179*: Permite la localización de dispositivos móviles para dispositivos con recursos limitados. El API se ha diseñado para generar información sobre la localización geográfica actual del terminal para las aplicaciones Java.
- *Security and Trust Services API JSR 177*: Este paquete amplía las características de seguridad para la plataforma J2ME añadiendo APIs de cifrado, servicio de firma digital y gestión de credenciales de usuario.
- *Mobile 3D Graphics API JSR 184*: Permite generar gráficos tridimensionales a frecuencias de imagen interactivas en dispositivos móviles de recursos restringidos.
- *SIP API for J2ME JSR 180*: El protocolo *Session Initiation Protocol* (SIP) se utiliza para establecer y gestionar sesiones IP multimedia. El API se ha diseñado para permitir que las aplicaciones Java envíen y reciban mensajes SIP.
- *J2ME Web Services APIs (WSA), JSR 172*: Amplía la plataforma de servicios Web para incluir J2ME. Estas APIs permiten que los dispositivos J2ME puedan ser clientes de servicios Web mediante un modelo de programación consistente con la plataforma estándar de servicios Web.
- *J2ME RMI Optional Package, (RMI OP) JSR 066*: Es un paquete que se puede utilizar sobre CDC. EL paquete *RMI Optional* (RMI OP) permite a dispositivos de consumo y aplicaciones embebidas interactuar cómo y con aplicaciones distribuidas.

3.4 JSON

JSON (*JavaScript Object Notation* - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos independiente de cualquier lenguaje de programación. Está basado en un subconjunto del Lenguaje de Programación JavaScript, *Standard ECMA-262 3rd Edition*; como subconjunto de la notación literal de objetos de JavaScript no requiere el uso de XML. JSON tiene formato de texto plano, de simple lectura, escritura y generación. Utiliza convenciones que son ampliamente conocidas por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

3.4.1 Formas de representación.

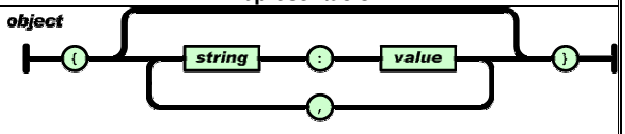
JSON está constituido por dos estructuras:

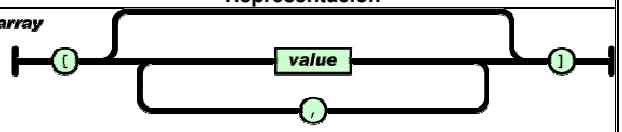
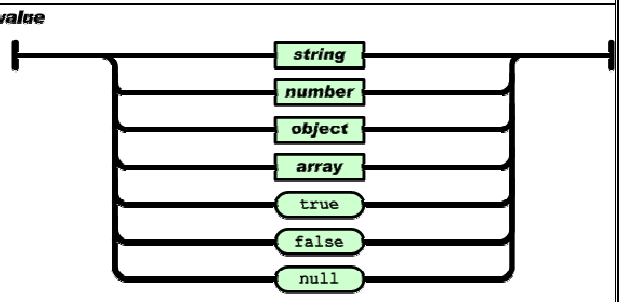
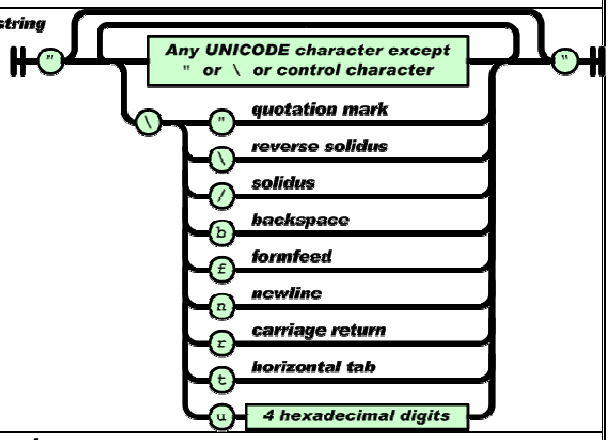
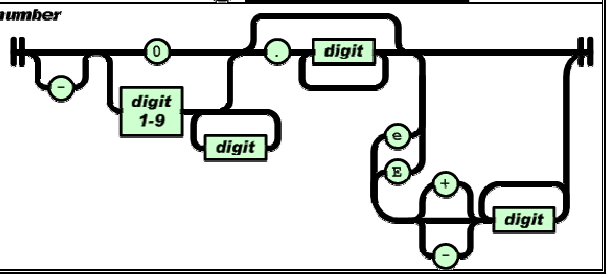
- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, se presentan de estas formas:

Tabla 1. Estructuras de JSON

Estructura	Definición	Representación
Objeto	Conjunto desordenado de pares nombre/valor. Notación: • Empieza con llave de apertura { • Termina una llave de cierre } • Sus propiedades se separan con comas , • El nombre y el valor están separados por dos puntos :	

Estructura	Definición	Representación
Array	Colección ordenada de valores u objetos. Notación: • Inicia con corchete izquierdo [• Acaba con corchete derecho] • Los valores se separan con una coma ,	
Value	Puede ser: • Una cadena de caracteres con comillas dobles • Un número • True, false, null • Un objeto • Un array Estas estructuras pueden anidarse	
String	Colección de 0 o más caracteres Unicode encerrados entre comillas dobles. Los caracteres de escape usan la barra invertida. Es parecida a una cadena de caracteres en C o Java.	
Number	Un *número* es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales	

Fuente: JSON.

3.4.2 Comparación con XML.

Entre sus similitudes se cuenta que ambos son legibles por humanos, tienen una sintaxis muy simple, son jerárquicos, independientes del lenguaje de programación y se pueden usar empleando Ajax.

La ventaja fundamental de JSON frente a XML es la cantidad de caracteres que se emplean para representar información estructurada, significativamente menor en el caso de JSON en la mayoría de los casos. Por lo tanto, al momento de transmitir información aquella codificada en JSON se comunica de forma más veloz y eficiente. En contraposición, XML goza de mayor soporte y ofrece muchas más herramientas de desarrollo (tanto en el lado del cliente como en el lado del servidor). Hay muchos analizadores JSON en el lado del servidor, existiendo al menos un analizador para la mayoría de los entornos. Ambos formatos carecen de un mecanismo para representar grandes objetos binarios.

Sin embargo, para decidir entre XML y JSON es necesario experimentar con el conjunto de datos a tratar para determinar qué formato será más eficiente en términos de tamaño. JSON no es siempre más pequeño que XML. El beneficio de JSON, entonces, no es que sea más pequeño a la hora de transmitir, sino que representa mejor la estructura de los datos y requiere menos codificación y procesamiento. Con independencia de la comparación con XML, JSON puede ser muy compacto y eficiente si se usa de manera efectiva.

Los entornos en el servidor normalmente requieren que se incorpore una función u objeto analizador de JSON. Algunos programadores, especialmente los familiarizados con el lenguaje C, encuentran JSON más natural que XML, pero otros desarrolladores encuentran su escueta notación algo confusa, especialmente cuando se trata de datos fuertemente jerarquizados o anidados muy profundamente.

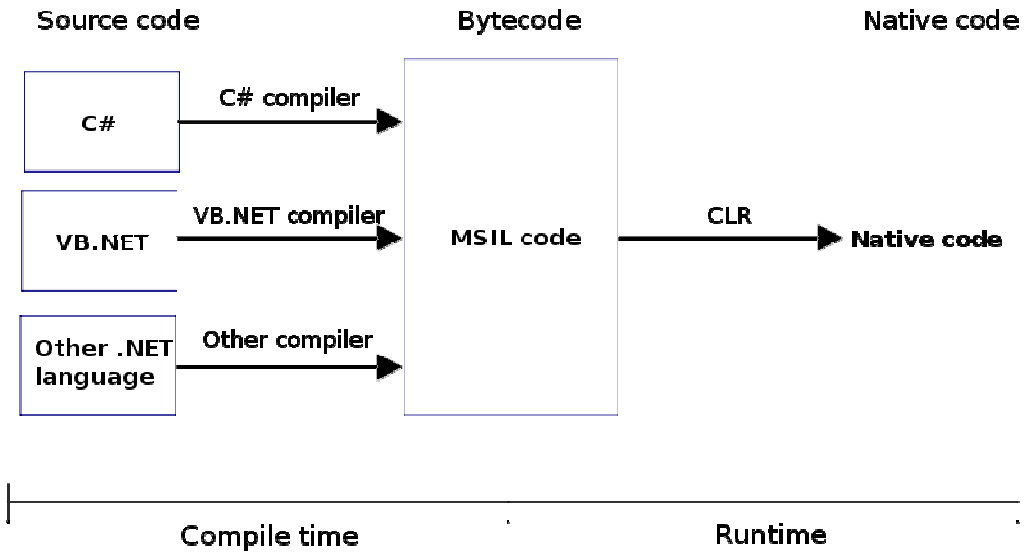
3.5 PLATAFORMA DE DESARROLLO .NET FRAMEWORK

Las aplicaciones del monitor LETSMonitor y del servidor LETSServer requirieron para su desarrollo el uso de la plataforma .NET Framework de Microsoft. Uno de los componentes más importantes del .NET Framework es el CLR o *Common Language Runtime* (Lenguaje común en tiempo de ejecución), que es el componente de máquina virtual de .NET Framework. Este es la implementación del estándar CLI (*Common Language Infraestructure*) que define un ambiente de ejecución para los códigos de los programas. El CLR se ejecuta sólo en sistemas operativos de Microsoft Windows. La forma en que la maquina virtual se relaciona con el CLR permite, durante el proceso de programación, ignorar muchos detalles específicos del CPU que ejecutará el programa.

Los desarrolladores que usan CLR escriben el código en un lenguaje como C# o VB.Net. En tiempo de compilación, un compilador.NET convierte el código a MSIL

(*Microsoft Intermediate Language*). En tiempo de ejecución, el compilador en tiempo de ejecución (*Just-in-time compiler*) del CLR convierte el código MSIL en código nativo para el sistema operativo. Alternativamente, el código MSIL es compilado a código nativo en un proceso separado anterior a la ejecución. Esto acelera las posteriores ejecuciones del software debido a que la compilación de MSIL a nativo ya no es necesaria.

Ilustración 8. Diagrama CLR (*Common Language Runtime*)



Fuente: Wikimedia Commons

LETS fue desarrollado usando el .NET Framework 3.5 entre cuyas características se destacan:

- Integración total de LINQ (*Language Integrated Query*) y del reconocimiento de los datos. Esta nueva característica le permitirá escribir código en idiomas habilitados para LINQ para filtrar, enumerar y crear proyecciones de varios tipos de datos SQL, colecciones, XML y conjuntos de datos usando la misma sintaxis.
- ASP.NET AJAX le permite crear experiencias web más eficaces, más interactivas y con un gran índice de personalización que funcionan con los exploradores más usados.
- Nueva compatibilidad con el protocolo web para generar servicios WCF, como por ejemplo AJAX, JSON, REST, POX, RSS, ATOM y distintos estándares WS- nuevos.

- Compatibilidad total con las herramientas de Visual Studio 2008 para WF, WCF y WPF, incluida la nueva tecnología de servicios habilitados para flujos de trabajo.
- Nuevas clases en la biblioteca de clases base (BCL) de .NET Framework 3.5 que tratan numerosas solicitudes de cliente comunes.

ASP.NET es un *framework* para aplicaciones web construido sobre el *Common Language Runtime*, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework.

Para la construcción de la aplicación LETSMonitor fueron utilizadas las *web forms* (formularios web), páginas de ASP.NET que son el principal medio de construcción para el desarrollo de aplicaciones web. Están contenidos en archivos con una extensión ASPX; estos archivos contienen etiquetas HTML o XHTML estático, y también etiquetas definiendo Controles Web que se procesan del lado del servidor y Controles de Usuario donde los desarrolladores colocan todo el código estático y dinámico requerido por la página web. Para realizar programación dinámica se usa el modelo *code-behind*, o de respaldo, que coloca el código en un archivo separado o en una etiqueta de *script* especialmente diseñada.

4 MARCO METODOLÓGICO

La programación extrema o *eXtreme Programming* (XP) es un enfoque de la ingeniería de software, que se destaca entre los procesos ágiles de desarrollo de software. La metodología de programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Así, a medida que se desarrolla el producto de software los requisitos sobre los que este se ha elaborado pueden transformarse para responder de modo más adecuado a las necesidades del cliente. De este modo, no todos los requisitos para construir el software deben generarse al principio del proyecto.

En *Extreme Programming Explained: Embrace Change* (1999), el primer libro acerca de la metodología XP, su autor Kent Beck definió los principios fundamentales de la programación extrema. Estos son:

- Simplicidad: La simplicidad es el fundamento de la programación extrema. El código es diseñado de forma sencilla y con poca complejidad, y se trabaja en mantener bajo su nivel de complejidad a lo largo del desarrollo del software, para que éste sea más ágil. El principio de simplicidad también se manifiesta en la elección de los nombres para las variables, métodos y clases; y en la insistencia de que la documentación del código, si bien completa, no exceda la justa medida en lo relacionado con los comentarios. El código debe quedar autodocumentado.
- Comunicación: Existe comunicación en diferentes sentidos. Los programadores, que trabajan en parejas, se comunican constantemente entre sí. Por su parte, el cliente forma parte del equipo de desarrollo y por tanto tiene que estar interactuando con éste constantemente, ayudando a establecer prioridades y resolver incertidumbres. El código comunica en cuanto está autodocumentado, y se recomienda reducir los comentarios al máximo dado que pierden validez en el proceso.
- Retroalimentación: En *Extreme Programming*, el trabajo se desarrolla en ciclos cortos y al final de cada uno se revisan los resultados para determinar si estos corresponden a los requisitos del software. El cliente, que está integrado al equipo de desarrollo, brinda su opinión acerca de los resultados para determinar si estos son los deseados o si deben ser modificados. Contar permanentemente con la opinión del cliente es vital para que el equipo de desarrollo direccione sus esfuerzos de la forma más eficiente.
- Coraje: Este principio se asocia con la capacidad del equipo de desarrollo de adoptar los anteriormente mencionados. Cumplir con la programación en parejas,

la retroalimentación constante entre el cliente y el equipo, mantener la simplicidad del código y realizar desarrollos en ciclos cortos son metodologías de desarrollo que, en principio, no se consideran tan eficientes, económicas o rentables como las tradicionales, pero que son defendidas por *Extreme Programming* dado que al final favorecen un desarrollo ágil y sostenido que respeta el trabajo del equipo de desarrollo y las necesidades del cliente.

- Respeto: Este principio fue añadido en la segunda edición de *Extreme Programming Explained*. Se refiere al equilibrio que guardan entre sí los miembros del equipo al trabajar de forma cooperativa y armónica, sin obstaculizarse mutuamente y obrando en conjunto por lograr alta calidad en el producto final.

4.1 CARACTERÍSTICAS DEL MÉTODO XP:

- Simplicidad en el código: El código se desarrolla por ciclos, manejando bajos niveles de complejidad y asegurándose de que cada parte desarrollada funciona adecuadamente y que se presta para hacerle mejoras futuras.
- Corrección de errores: Cada vez que una parte del código es completada esta es probada hasta asegurarse de que funciona correctamente, y sólo después de eso se le añaden nuevas funcionalidades.
- Refactorización del código: Ciertas partes del código, aunque ya funcionen, son reelaboradas para asegurarse de que sean más legibles y su mantenimiento sea más sencillo. Al refactorizar el código la función se mantiene, puesto que ya se ha verificado que es la adecuada, pero este se hace más manejable para los avances futuros.
- Pruebas unitarias continuas: Estas se realizan para probar el correcto funcionamiento de un módulo de código, asegurándose de que cada una de las partes del software funcione correctamente por separado. Esto incluye la realización de 'pruebas de regresión', las cuales están encaminadas a descubrir las causas de nuevos errores (*bugs*), carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software, que han surgido a partir de cambios realizados recientemente en partes de la aplicación que anteriormente a la modificación sufrida no eran propensas a este tipo de error. Esto significa que el error se ha producido por el cambio en el programa. Una vez realizadas las pruebas unitarias se efectúan 'pruebas de integración', en las que las partes del software son probadas en conjunto para asegurar que el sistema total o una gran sección del mismo funcionan adecuadamente.

- Programación en parejas: Se recomienda que la programación del código sea efectuada en grupos de dos personas, donde una persona se encarga de programar y la otra de revisar y discutir el código, intercambiando posiciones cada cierto tiempo. Esto favorece el desarrollo de un código más legible y sencillo.
- Propiedad compartida del código: Todo el equipo de trabajo se involucra –o puede involucrarse- en el desarrollo de los módulos del software. Así este no se divide en subgrupos, y todos están al día de las transformaciones que se hacen en el código.

Reflejo de este conjunto de principios son los pasos que deben seguirse para implementar de manera exitosa la metodología XP. Los presentaremos agrupados en cuatro bloques, donde cada uno se va ejecutando de forma paralela a los demás, evolucionando progresivamente.

Ilustración 9. Mapa conceptual de la metodología de Programación Extrema

4.1.1 Historias de usuario.

Son una descripción informal de las necesidades del usuario, escrita por el cliente en pocas líneas. Se utilizan para estimar tiempos de desarrollo de la parte de la aplicación que describen.

4.1.2 Plan de entregas.

El cliente determina la importancia o prioridad de cada historia de usuario, y por su parte, los programadores realizan una estimación del esfuerzo necesario en el desarrollo de cada una de ellas. A partir de esto se realizan acuerdos sobre el contenido de la primera entrega.

4.1.3 Velocidad del proyecto.

Medida que representa la rapidez con la que se desarrolla el proyecto. Estas medidas permiten determinar cuánto tiempo tomará implementar un conjunto de historias.

4.1.4 Iteraciones.

Se realizan varias iteraciones sobre el sistema antes de ser entregado. Al comienzo de cada iteración el cliente selecciona las historias de usuario que serán implementadas en ella. Los desarrolladores trabajan exclusivamente en las historias seleccionadas por el cliente, evitando avanzar en lo que no ha sido requerido.

4.1.5 Reuniones.

Los desarrolladores se reúnen diariamente para exponer sus problemas, soluciones e ideas en conjunto.

4.2 DISEÑO

4.2.1 Metáfora del sistema.

En esta fase se buscan las frases o nombres que describen y definen cómo funcionan las partes del programa, de modo que la labor que realiza cada una de

ellas se conozca a partir de la denominación. La aplicación de esta labor permite mantener la coherencia entre los nombres que se utilizan.

4.2.2 Soluciones puntuales.

Los problemas se aíslan y se solucionan de forma ajena al contexto, para que las soluciones sean técnicas y para poder predecir cuánto tiempo tomará hallarlas.

4.2.3 Funcionalidad mínima.

Para responder a cada necesidad se busca la solución más sencilla que funcione, y esta se adopta. Los procesos posteriores permitirán refinar y mejorar dicha solución cuando sea oportuno.

4.2.4 Reciclaje.

Implica mantener el código sencillo y fácil de comprender, y reutilizarlo cuando sea oportuno. Además, al revisar el código y mejorarlo debe mantenerse su funcionalidad.

4.3 CODIFICACIÓN

4.3.1 Disponibilidad de clientes.

Los clientes deben estar vinculados en todo momento al desarrollo del software; en la definición de los requisitos necesarios para crear las historias de usuario, al negociar los tiempos en que las funciones que las satisfacen deben ser implementadas, y al realizarse las pruebas que permiten determinar si los desarrollos responden adecuadamente a lo requerido.

4.3.2 Unidad de pruebas.

Se puede dividir la funcionalidad que debe cumplir una tarea a programar en pequeñas unidades, de esta forma se crearán primero las pruebas para cada unidad y a continuación se desarrollará cada unidad. Un punto importante es crear pruebas que no tengan ninguna dependencia del código que en el futuro se evaluará.

4.3.3 Programación por parejas.

El trabajo en pareja involucra dos programadores trabajando en el mismo equipo; mientras uno codifica haciendo hincapié en la calidad de la función o método que está implementando, el otro analiza si ese método o función es adecuado y está bien diseñado. Así se consiguen un código y diseño de calidad.

4.3.4 Integración.

Cada nueva pieza de código es integrada en el sistema una vez que está lista. Esta nueva integración debe aprobar todas las pruebas para que el código sea integrado de forma definitiva al sistema completo. De este modo las incompatibilidades entre lo desarrollado y lo existente se detectan rápidamente.

4.4 PRUEBAS

4.4.1 Implantación.

Cada vez que se consigue codificar una historia de usuario y que funcione, se le entrega el cliente para que la vea, la apruebe y le añada las modificaciones para las siguientes mini-versiones. Cuando se realiza una mini-versión completa del software, reuniendo varias historias de usuario, se le entrega al destinatario final para que trabaje con ella y reporte problemas o proponga mejoras.

4.4.2 Pruebas de aceptación.

Estas pruebas son creadas en el momento en que se definen las historias de usuario elegidas para la iteración, y en ellas el cliente verifica el correcto funcionamiento del módulo o parte del software que se le está entregando y hace pruebas para detectar los posibles casos en que falla. Una historia de usuario se completa cuando pasa las pruebas de aceptación.

4.4.3 Validación.

Se realizarán diversos escenarios de aplicación del sistema con el fin de probar y validar la funcionalidad planeada del prototipo. Dichos escenarios contarán con diversas configuraciones tanto de los eventos a registrar como de las acciones a

tomar con respecto a la ejecución de los eventos. Las pruebas se realizaran simultáneamente con un mínimo de 2 dispositivos móviles dotados de GPS que ejecuten rutas y acciones que desencadenen los eventos previamente configurados, la realización de estas pruebas así como sus resultados y trayectoria serán documentadas.

5 DESARROLLO E IMPLEMENTACIÓN

5.1 CONCEPTO DEL SISTEMA

El desarrollo del proyecto resulta en un prototipo de herramienta de software denominado LETS (*Location Event Tracking System*), que permite capturar uno o varios eventos basados en la posición del usuario y reaccionar ante ellos. Para tal fin el sistema utiliza un dispositivo móvil que captura y notifica tanto su posición como los eventos que han sido preestablecidos para el usuario que lo lleva.

Los eventos son previamente configurados por un operador por medio de la interfaz web del sistema, la cual permite a este administrar los usuarios del sistema, sus eventos y las reacciones que deben ejecutarse cuando estos ocurran. Los eventos posibles no solo cubren situaciones básicas como la proximidad a un punto, la velocidad o el ingreso a un área, sino que, al poseer la capacidad de combinarse entre ellos, ofrecen una amplia gama de posibilidades de configuración.

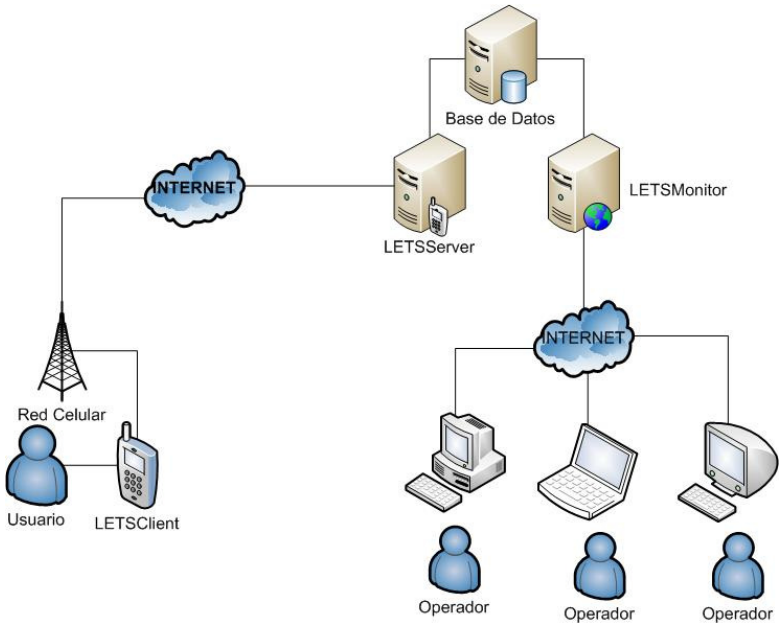
El sistema ofrece una serie de reacciones posibles ante la ocurrencia de un evento, que van desde el envío de un correo electrónico o SMS a una serie de destinatarios hasta la utilización de un *WebService* con los datos del evento ocurrido.

LETS consta de tres partes, cuyas funciones generales se describen a continuación:

- LETSClient (Cliente móvil): Es el software que va en el dispositivo móvil, encargado de obtener la posición del usuario, descargar los eventos que han sido configurados y procesar los diversos cambios en la posición del usuario para determinar si los eventos se han producido. También es responsable de notificar la ejecución de estos eventos y los datos asociados.
- LETSServer (Servidor): Es la pieza central del sistema; se encarga de la comunicación de los eventos a los clientes móviles, y de la recepción y registro de los eventos ocurridos, así como de la ejecución de las reacciones que hayan sido configuradas.
- LETSMonitor (Monitor): Es un sitio web que permite la configuración de los usuarios, eventos y reacciones, además de otros parámetros útiles para el funcionamiento del sistema; también visualiza los eventos ocurridos y los datos asociados a estos.

En la figura se muestra la relación entre las partes del prototipo:

Ilustración 10. Arquitectura del sistema



Se usaron diversas tecnologías para el desarrollo de los componentes. Para el servidor y monitor se utilizo ASP.NET con lenguaje de programación C#, para el almacenamiento de la información se utilizo Microsoft SQL Server 2005, y para el cliente móvil se uso Java J2ME. Como formato de transferencia de los datos entre el cliente y el servidor se uso JSON, y como protocolo de comunicación HTTP. Para la visualización de los eventos y configuración en el monitor se utilizo la API de *Google Maps* para Javascript.

5.2 ANÁLISIS FORMAL DE REQUERIMIENTOS

5.2.1 Requerimientos generales.

Los requerimientos generales que el sistema debe cumplir son:

• El sistema en su modo de operación habitual, no requiere de interacción humana, dado que debe guiarse por los parámetros originalmente configurados. Por tanto, el sistema debe ser autónomo en función de su configuración. Esta característica permite al sistema ser integrado como punto de entrada de otros sistemas, al proporcionar de forma automática datos de los eventos ocurridos en el momento en que suceden, permitiendo así la integración con servicios

informáticos diversos y posiblemente más especializados. Por otra parte, esta característica permite a las personas que deben tomar decisiones basadas en el comportamiento y cambio de posición del usuario reaccionar sólo cuando es realmente necesario; es decir, cuando ocurra el evento que necesita ser atendido, en lugar de analizar todo el tiempo el cambio de posición del usuario.

- El sistema debe ser capaz de cumplir un amplio rango de posibilidades para los eventos. Es decir, que diversos eventos concebibles o imaginables puedan ser efectivamente descritos por los eventos propios del sistema, de forma singular o encadenados. Es por esto que en el sistema se diseñan un conjunto de eventos, cada uno dotado de nombre y propiedades, que pueden ser configurados para el cliente con datos propios y determinados por el operador. Poder describir lo más completamente posible las condiciones que definen al evento es de vital importancia, ya que se pretende que el sistema reaccione sólo cuando ocurra el suceso que es relevante para el operador.
- Para que el software cumpla a cabalidad con su función, es necesario que ofrezca diferentes posibles reacciones que se produzcan ante la ocurrencia de un evento. Además de un conjunto de reacciones estándar, previamente diseñadas y configurables dentro del programa, el sistema ofrece un conjunto de métodos para que los desarrolladores puedan crear sus propias reacciones e incluirlas dentro del funcionamiento del programa. Esto permite el crecimiento del sistema y la adecuación a escenarios no previstos, así como el desarrollo por parte de terceros o comunidades.

5.2.2 Requerimientos para LETSClient.

Nombre: Autentificación de usuario		No. 004
Componente: LETSClient	Categoría: Autentificación	
DESCRIPCIÓN		
El cliente móvil debe estar dotado de un sistema de autentificación para identificar al usuario que accede al terminal, mediante una combinación de *login* y *password*.		
REQUERIMIENTO NO FUNCIONAL		
El terminal móvil debe guardar la información de autentificación e iniciar sesión con ella automáticamente.		

Nombre: Obtención de configuración de eventos	No. 005
Componente: LETSClient	Categoría: Actualización
DESCRIPCIÓN	
LETSClient debe solicitar al servidor la información de los eventos configurados para el usuario que ha iniciado sesión y cargarlos en el terminal móvil cada vez que el usuario inicie sesión.	
REQUERIMIENTO NO FUNCIONAL	
El cliente móvil debe permitir la actualización de la información de los eventos configurados por demanda del usuario.	

Nombre: Evaluación de Eventos	No. 006
Componente: LETSClient	Categoría: Detección de eventos
DESCRIPCIÓN	
LETSClient debe obtener la posición del dispositivo móvil, por intervalos de tiempo predeterminados, evaluar si alguno de los eventos configurados ocurre.	
REQUERIMIENTO NO FUNCIONAL	

Nombre: Notificación de evento al servidor	No. 007
Componente: LETSClient	Categoría: Notificación de eventos
DESCRIPCIÓN	
Una vez ocurrido un evento, el cliente móvil debe transmitir toda la información asociada a este evento al servidor para ser procesada.	
REQUERIMIENTO NO FUNCIONAL	

Nombre: Recepción de mensajes	No. 008
Componente: LETSClient	Categoría: Mensajes
DESCRIPCIÓN	
El cliente móvil debe ser capaz de visualizar en pantalla mensajes provenientes del servidor, en caso de que haya ocurrido un evento y de que la reacción configurada para dicho evento sea un mensaje del servidor al cliente.	
REQUERIMIENTO NO FUNCIONAL	

Nombre: Limpieza	No. 009
Componente: LETSClient	Categoría: Correcta desconexión
DESCRIPCIÓN	
Al cerrarse la aplicación del terminal móvil, esta debe liberar todos los recursos que consume (transmisión de datos, GPS, memoria del dispositivo).	
REQUERIMIENTO NO FUNCIONAL	

5.2.3 Requerimientos para LETServer

Nombre: Autentificación de usuarios	No. 010
Componente: LETServer	Categoría: Autentificación
DESCRIPCIÓN	
LETServer debe autenticar al usuario que se conecta al cliente móvil verificando su información de *login* y *password*.	
REQUERIMIENTO NO FUNCIONAL	

Nombre: Trasmisión de eventos asignados	No. 011
Componente: LETServer	Categoría: Transmisión de eventos configurados
DESCRIPCIÓN	
LETServer debe obtener de la base de datos los eventos previamente configurados para el usuario y transmitirlos al cliente cuando este inicia sesión o cuando el usuario lo demande.	
REQUERIMIENTO NO FUNCIONAL	

Nombre: Registro de eventos ocurridos	No. 012
Componente: LETServer	Categoría: Recepción y registro de eventos ocurridos
DESCRIPCIÓN	
El servidor debe recibir del cliente móvil la información del evento ocurrido y sus datos y guardar registro de ella.	
REQUERIMIENTO NO FUNCIONAL	

Nombre: Ejecución de reacciones	No. 013
Componente: LETServer	Categoría: Ejecución de reacciones
DESCRIPCIÓN	
Una vez ocurrido un evento, el servidor debe ejecutar las reacciones configuradas para el mismo.	
REQUERIMIENTO NO FUNCIONAL	

5.2.4 Requerimientos para LETSMonitor

Nombre: Autentificación del operador	No. 014
Componente: LETSMonitor	Categoría: Autentificación del operador
DESCRIPCIÓN	
El monitor debe permitir al operador el inicio de sesión por medio de la introducción de sus datos de *login* y *password.*	
REQUERIMIENTO NO FUNCIONAL	
Los datos de *login* y *password* de cada operador no pueden estar vacíos. Además, no puede existir el mismo *login* para dos usuarios.	

Nombre: Administración de usuarios	No. 015
Componente: LETSMonitor	Categoría: Administración de usuarios
DESCRIPCIÓN	
LETSMonitor debe permitir la administración de usuarios, esto es, creación, edición y eliminación de usuarios.	
REQUERIMIENTO NO FUNCIONAL	
Todo usuario debe tener un *login* único, un *password* que no puede estar vacío, y un nombre.	

Nombre: Administración de Operadores	No. 016
Componente: LETSMonitor	Categoría: Administración de operadores
DESCRIPCIÓN	
LETSMonitor debe permitir la administración de operadores, esto es, creación, edición y eliminación de operadores.	
REQUERIMIENTO NO FUNCIONAL	
Todo operador debe tener un *login* único, un *password* que no puede estar vacío, y un nombre.	

Nombre: Administración de eventos asignados	No. 017
Componente: LETSMonitor	Categoría: Configuración de eventos
DESCRIPCIÓN	
El monitor debe permitir la creación, eliminación y edición de uno o varios eventos asignados (*AssignedGPSEvent*) a un usuario.	
REQUERIMIENTO NO FUNCIONAL	
Cada evento asignado debe tener un nombre que lo identifique y estar compuesto por al menos un evento. Debe poder habilitarse, es decir, enviarse al cliente móvil, y deshabilitarse.	

Nombre: Edición de Eventos	No. 018
Componente: LETSMonitor	Categoría: Configuración de eventos
DESCRIPCIÓN	
LETSMonitor debe permitir la configuración de las propiedades de los eventos asignados, y de los eventos relacionados a estos en una cadena de eventos.	
REQUERIMIENTO NO FUNCIONAL	
La configuración de los eventos de Área y Proximidad deberá realizarse con la asistencia de un mapa.	

Nombre: Visualización de eventos ocurridos	No. 019
Componente: LETSMonitor	Categoría: Visualización
DESCRIPCIÓN	
El monitor debe permitir visualizar en el mapa los eventos ocurridos para cada evento asignado de un usuario.	
REQUERIMIENTO NO FUNCIONAL	
Al visualizar en el mapa los eventos ocurridos el sistema debe permitir filtrarlos por un rasgo de fechas.	

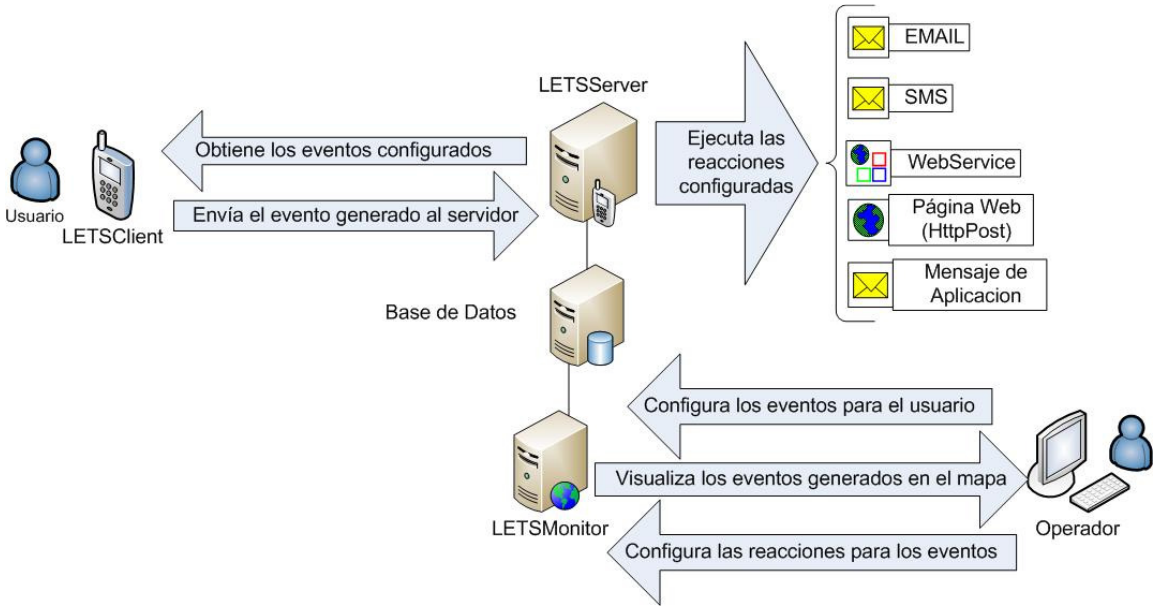
Nombre: Visualización de datos de eventos ocurridos	No. 020
Componente: LETSMonitor	Categoría: Visualización
DESCRIPCIÓN	
LETSMonitor debe mostrar los datos asociados al evento ocurrido.	
REQUERIMIENTO NO FUNCIONAL	
LETSMonitor debe mostrar los datos de configuración asociados a un evento asignado (*assignedEvent*) cuya ocurrencia se ha verificado.	

5.3 DISEÑO

5.3.1 Aspectos generales.

LETS es el resultado del proyecto que se propone el diseño de un prototipo de herramienta de software útil para seguir terminales móviles y generar reacciones de acuerdo con ciertos cambios de posición -eventos- que por sus condiciones particulares se consideran relevantes o de interés para el cliente del sistema. Para llevar a cabo tal fin, se diseñó el prototipo bajo una arquitectura cliente-servidor donde fueron concebidas 3 partes fundamentales para llevar a cabo el propósito del sistema: una parte cliente llamada LETSClient, la cual es responsable de la detección de eventos y de la posición actual del usuario; una parte servidor llamada LETSServer, responsable de comunicarse con la parte cliente, entregar la configuración de eventos a seguir, recibir los datos de los eventos que hayan ocurrido, así como ejecutar las diversas reacciones que se hayan configurado. En conjunto con estas dos es necesario un ambiente donde se puedan configurar los eventos, las reacciones, los usuarios y demás parámetros del sistema para su funcionamiento; esta parte, denominada LETSMonitor, es la que tiene mayor interacción con el cliente, al cual llamaremos de ahora en adelante operador.

Ilustración 11. Relaciones entre las partes del sistema LETS



Dado que se trata de un sistema que reacciona a los cambios de posición del usuario, la aplicación debe cubrir las necesidades de desplazamiento del mismo y

acompañarlo a donde quiera que este vaya. Esto se suma a que la aplicación es cliente-servidor y debe notificar a un servidor central, por lo tanto, la escogencia del canal de comunicación de estos dos juega un papel importante. Se descartan tecnologías de comunicación como redes inalámbricas o sincronización por lotes, como en el caso común de las aplicaciones para *pocket*, dado que las redes inalámbricas (Wi-fi) tienen un corto alcance y no están disponibles en muchas zonas donde se presume el usuario podría desplazarse. Además, la sincronización de las aplicaciones diseñadas para trabajar fuera de línea no es útil dado que se quiere que el sistema ejecute las reacciones en tiempo real. Por tanto, la mejor tecnología que se puede usar son las redes de comunicación celular, que ofrecen una gran cobertura a nivel nacional, tanto en ciudades como en carreteras y zonas rurales.

Las redes celulares ofrecen cobertura, pero hay un cobro por el uso de estos canales de comunicación. La aplicación se diseña en función de evitar un tránsito innecesario de datos sobre el canal, por lo que se requiere que el dispositivo móvil, que da cuenta de la posición, sea quien detecte los eventos y los informe al servidor. Así se transmite solo la información del evento cuando este ocurre, en vez de reporte con la posición del usuario cada cierto tiempo para que el servidor identifique el evento; dejando al terminal ser quien evalúe la ocurrencia del evento reducimos drásticamente el consumo de datos.

Atendiendo a la movilidad del usuario se escoge un sistema de localización que sea fiable y que no esté restringido a ciertas zonas en particular; el método de localización que cumple mejor con estas características es el GPS.

5.3.1.1 Selección del dispositivo móvil.

El diseño general de la aplicación permite establecer tres criterios esenciales para la escogencia del dispositivo:

- Conectividad GPRS (Datos sobre redes celulares).
- Programabilidad.
- Mecanismo de localización GPS.

Los dispositivos que cumplen con estos tres criterios son teléfonos celulares dotados de GPS. Características adicionales favorables son su disponibilidad en el mercado, costos razonables, y que se han consolidado para este tipo de aplicaciones. En función de la programabilidad se seleccionan los celulares dotados con J2ME puesto que es uno de los lenguajes de programación más

difundido entre diversas marcas y modelos de celulares, lo cual permite a la aplicación ser instalada en una gran gama de teléfonos celulares.

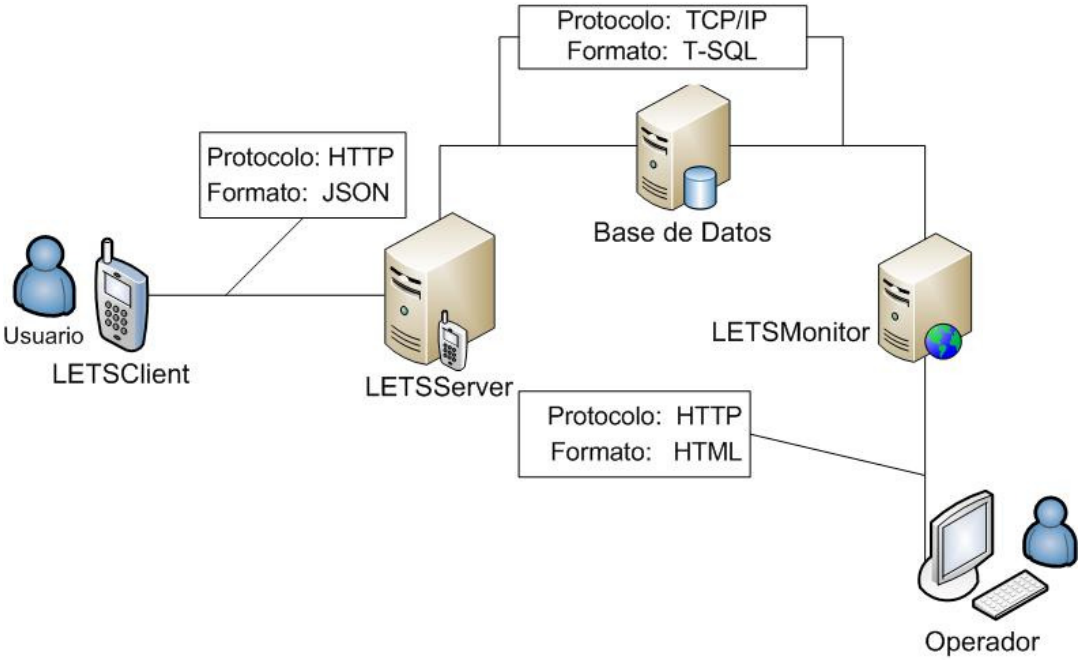
5.3.1.2 Protocolo de comunicación.

Es responsabilidad del servidor comunicarse con la aplicación cliente, para lo cual se diseña el proceso de comunicación entre estas dos aplicaciones. Dado que el acceso a la red celular es gestionado de forma privada por la empresa dueña de esta es complicado pensar en establecer el servidor dentro de esta red; además, en la actualidad las redes celulares se integran con internet, por lo que se sitúa el servidor en Internet. En la naturaleza de la transmisión de datos por redes celulares se sabe que la comunicación es intermitente así que un protocolo de comunicación de forma sincrónica no es funcional, lo cual descarta todos los protocolos de comunicación que cumplen con esta condición. Finalmente, es seleccionado el protocolo de comunicación HTTP por su facilidad en la implementación y por su fiabilidad al ser ampliamente usado en la industria.

En consecuencia, ASP.NET es seleccionado como tecnología de programación en el servidor por ser un lenguaje popular en el medio, con altas características de estructuración de la programación y enfocado para actuar como servidor HTTP.

Una vez seleccionados el canal de comunicación, el protocolo HTTP y la tecnología ASP.NET, solo nos queda definir el formato que se usará. Debido a la naturaleza de los eventos, y de la información que se debe trasmitir cuando un evento es producido, el formato de transmisión debe soportar el envío de información anidada, sin un límite aparente en profundidad, y debe ser liviano en cuanto al tamaño de la información que se trasmite. Con estos 2 criterios se escogió JSON como formato de trasmisión, ya que permite el envío de datos estructurados, es muy ligero comparado con XML, y su implementación es muy sencilla, además de ser uno de los estándares más usados en las tecnologías web.

Ilustración 12. Protocolos de comunicación LETS



5.3.1.3 Almacenamiento de la información.

Para el almacenamiento de datos se utilizó como motor de base de datos Microsoft SQL Server 2005. Su selección obedece, no solo a ser uno de los más usados en la industria, con un gran soporte y un muy buen rendimiento demostrado para aplicaciones de mediana y alta demanda de datos, sino también a su fácil integración dentro del desarrollo de aplicaciones realizadas en Microsoft .NET.

5.3.1.4 Visualización de los datos.

La aplicación monitor LETSMonitor se realiza como un sitio web, ya que los datos que esta gestiona se encuentran en la base de datos que el servidor usa y que también está en Internet. Las dos aplicaciones pueden residir en el mismo equipo servidor. Se construye esta aplicación con el fin de permitir la administración; gestión; visualización y configuración de los eventos, reacciones posibles, usuarios, operadores y demás parámetros que el sistema necesita para funcionar correctamente. Para su construcción se aprovechan técnicas de presentación web como AJAX, que permiten una muy cómoda experiencia de usuario -muy similar a las aplicaciones de escritorio- usando la plataforma ASP.NET, y con la

integración sencilla que ofrece la API de *Google Maps* para utilizar los mapas y construir aplicaciones alrededor de ellos, gracias a un lenguaje intermedio JavaScript.

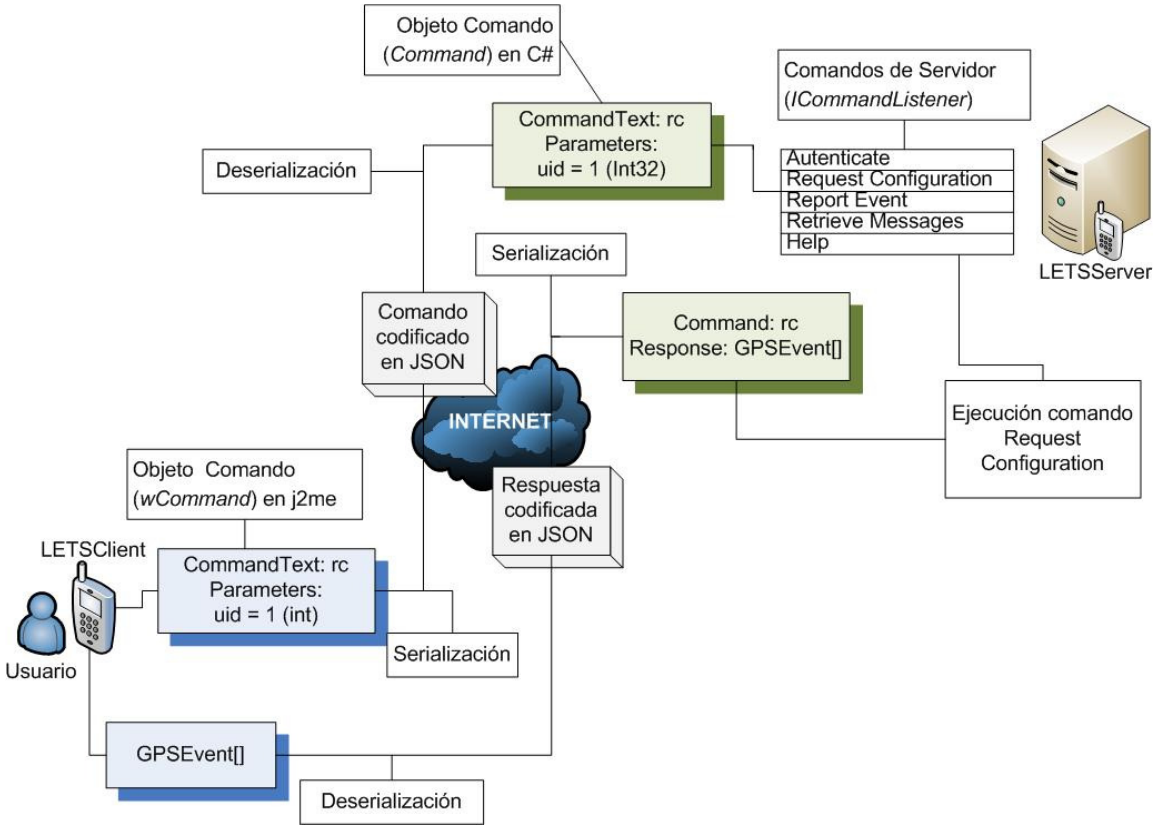
5.3.1.5 Cliente móvil LETSClient.

La aplicación cliente está escrita enteramente en J2ME utilizando los componentes de alto nivel que vienen incluidos dentro del SDK. No se utilizo ningún conjunto de librerías gráficas o componentes personalizados; el modelo de construcción de software es MVC *(Model-View-Controller*) partiendo de un conjunto de clases base construidas para la manipulación de los formularios y demás elementos gráficos.

Debido a la naturaleza cliente-servidor del sistema en general se implementó un modelo de comunicación basado en comandos y parámetros de comando; un comando se entiende como un proceso que se ejecuta en el servidor el cual se invoca con un nombre de comando, y una serie de parámetros cada uno con un nombre asociado y un valor. El comando en sí es entonces el nombre de comando a invocar y el conjunto de los parámetros, que son enviados al servidor el cual emite una respuesta con el mismo nombre de comando ejecutado y un objeto respuesta.

Los parámetros son serializados al enviarlos al servidor con formato JSON, por lo que deben ser parámetros sencillos (primitivos) como números o cadenas de texto. Si representan objetos complejos deben implementar una serie de métodos que permitan descomponerlos en objetos simples. Estos métodos están implementados como una interfaz *ISerializable* creada para este fin. El tipo de respuesta depende del comando ejecutado, es decir, de la respuesta que emita el servidor. Los comandos pueden ser enviados de forma individual, o como un conjunto de estos en una sola petición HTTP (*HttpRequest*).

Ilustración 13. Sistema de comandos LETS



Para el cálculo de la posición del usuario se hace uso de la API de localización incluida en J2ME, que es opcional y que se incluye en los terminales dotados con métodos de localización (GPS, CellID, etc.) Esta API (JSR 179) es creada por la comunidad de Java de forma estándar y es incluida en el terminal por el fabricante del mismo. La API actúa como una capa entre el dispositivo GPS y el programa en J2ME, permitiendo una serie de funciones para el posicionamiento.

Sobre esta API, estrictamente hablando sobre el objeto *LocationProvider*, se crea una clase que gestiona la obtención de la posición GPS del terminal de forma más específica a las necesidades del proyecto, cumpliendo con ciertas prestaciones adicionales.

Los 'eventos', que serán discutidos en detalle en el próximo capítulo, recibirán entonces cada uno la notificación de la nueva ubicación del terminal, y cada uno de ellos será responsable de evaluar si ha ocurrido el evento que él representa, y

por consiguiente, de notificar a la aplicación sobre la ocurrencia del evento, la cual enviará un comando al servidor con los detalles del evento.

Con los elementos anteriormente descritos pueden desarrollarse las siguientes funciones:

- Autentificar al Usuario: Envía al servidor un comando de autentificación, con parámetros como el *login* y el *password*, y un código que identifique al usuario para las siguientes consultas (comandos). El servidor deberá responder si la autentificación es correcta,
- Obtener la configuración de eventos: Envía un comando al servidor solicitando los eventos configurados, los eventos son recibidos y cargados en el cliente móvil para ser analizados y notificar cuando ocurra un cambio de posición.
- Obtener periódicamente la posición: Activa el lector de posición (*GPSReader*) con un intervalo preconfigurado y notifica a los eventos registrados las nuevas posiciones que se van obteniendo.
- Notificar el evento ocurrido: Cuando ocurra un evento, se debe generar el correspondiente objeto que recopila la información del evento ocurrido (*GPSEventArgs*), este objeto se envía al servidor haciendo uso de un comando, para ser registrado y procesado.
- Recibir mensajes: Una de las posibles reacciones que el sistema ofrece, frente a la ocurrencia de un evento, es el envío de un mensaje al usuario a través de la aplicación (no SMS); el programa cliente está atento a recibir como respuesta de cualquier comando un mensaje por parte del servidor y lo visualiza en pantalla para alertar al usuario.

Funciones adicionales del cliente son mostrar la posición obtenida en el momento, el nombre de los eventos que están siendo procesados, y adicionalmente se permite la visualización de los datos de los eventos, es decir, en forma grafica o texto, dependiendo del tipo de evento. Estos datos de evaluación permiten determinar la posible ejecución del evento; por ejemplo, en el caso de un evento de proximidad a un punto, se muestra la ubicación actual, la distancia de ella al punto, el punto y el círculo que forma el radio configurado del evento. En el caso de un evento de velocidad, se muestra únicamente la velocidad.

Adicionalmente, el cliente permite la configuración de datos básicos o parámetros necesarios globales para la ejecución del sistema, como son el intervalo al cual obtendrá las lecturas de GPS (*GPSRead*), la dirección URL del servidor, y un indicador de si el sistema debe guardar las credenciales del usuario para iniciar sesión automáticamente la próxima vez.

5.3.1.6 Servidor LETSServer.

El servidor es el componente sin interfaz gráfica del sistema. Este componente es encargado de atender los comandos realizados por la aplicación cliente, dar respuesta a estos y ejecutar las reacciones configuradas para el evento que ocurra.

De forma mas detallada, el servidor es una aplicación ASP.NET usando C# como lenguaje de programación; está creada como un servidor de comandos que recibe parámetros y genera respuestas y acciones de acuerdo al conjunto de comandos que tiene configurado (*ICommandListener*); el servidor de comandos es configurado con un formato de respuesta que para este caso es JSON (*JSONFormatter*).

La aplicación ASP.NET hace uso de un objeto *IHttpHandler* provisto por las librerías .NET, con el cual recibe peticiones HTTP con parámetros por GET o POST. En la configuración del sitio se le asigna a este objeto una URL (dentro del sitio) a la cual deben dirigirse las peticiones de comandos de los clientes. Como se dijo anteriormente, las peticiones pueden ser singulares (un solo comando por petición) o múltiples (varios comandos en la misma petición). Las respuestas vendrán con el nombre del comando ejecutado. La respuesta enviada por este objeto es una respuesta HTTP, como una pagina ordinaria HTML, con el texto en el formato definido.

Cada comando es validado con respecto al tipo y cantidad de parámetros obligatorios, de acuerdo al tipo de comando, y tanto la petición como la respuesta obedecen al formato que se haya especificado en su configuración.

Los comandos disponibles son:

Ilustración 14. Sistema de Comandos de LETS

Comando	**Parámetros**	**Respuesta**	**Descripción**
rc	- *uid (Int32):* El identificador del usuario del cual se requiere obtener la configuración de eventos.	*GPSEvent[]:* Colección de eventos, un vector con todos los eventos que están habilitados para este usuario, cada evento es del tipo GPSEvent.	*Request Configuration*: Entrega los eventos que estén configurados para un usuario, cada evento es serializado con todas sus propiedades.

Comando	Parámetros	Respuesta	Descripción
Au	- *l (String)*: L*ogin* del usuario. - *p (String)*: *Password* del usuario.	*object[]*: Un vector de 2 posiciones, la primera tiene el id del usuario (Int32) y la segunda posición el nombre del usuario.	*Autenticate*: Autentica un usuario usando un desafío de *login* y *password*.
Re	- *args (GPSEventArgs)*: El objeto argumento con toda la información de la ocurrencia de un evento.	*Nullable*: No tiene respuesta.	*Report Event*: Almacena el evento generado, y procesa todas las reacciones que lleve a lugar dicho evento.
Rm	- *uid: (Int32)*: El id del usuario.	*Parameters[]:* Un vector de parámetros. cada parámetro representa un mensaje, contiene un objeto *Parameters*, que tiene como claves: *id,created,text,sender;* con el id del mensaje, la fecha de creación, el texto del mensaje y el nombre de quien lo envía respectivamente.	*Retrieve Messages*: Obtiene los mensajes que están pendientes por enviar al usuario, identificado por el parámetro uid.
Help	- *cmd (String):* Opcional. Nombre del comando del cual se desea obtener ayuda.	*Parameters*: Un objeto con las llaves, *name,description,parameters,returns* con el nombre, descripción del comando, información de los parámetros que recibe e información del valor que retorna.	*Help*: Provee ayuda. Cuando el parámetro cmd es incluido devuelve la ayuda especifica de cada comando, en caso contrario devuelve una lista con todos los comandos disponibles y su descripción.
Echo	(Sin parámetros)	*Parameters:* Retorna un objeto *Parameters* con los mismo parámetros que fue enviada la petición, más la llave echo con el nombre del servidor.	*Echo*: Comando de Eco. Devuelve la respuesta, útil para comprobar comunicación con el servidor.

El comando ‘re’ (*Report Event*) es encargado de recibir de la aplicación cliente los datos del evento ocurrido. Estos datos son almacenados en la base de datos junto con la posición del evento ocurrido, la fecha y hora y el tipo de evento. Posterior a este almacenamiento se ejecutan las reacciones que han sido configuradas para el evento en particular. Este comando es fundamental y

encapsula la segunda funcionalidad principal del sistema, la capacidad de reaccionar ante los eventos ocurridos.

Los eventos son objetos dentro del sistema que tienen una funcionalidad primordial, pero a su vez cada tipo de evento presenta características diferentes, propiedades y comportamientos que difieren de uno a otro. Las reacciones a estos eventos también son objetos diferentes entre sí; para el almacenamiento de los eventos y de las reacciones en la base de datos se optó por una técnica basada en la serialización de objetos.

La serialización consiste en el proceso de codificar un objeto en un medio físico de almacenamiento (memoria, *buffer*, archivo...) para poder trasmitirlo; este conjunto de *bytes* codificados pueden ser usados para recrear el objeto en su estado original (antes de la serialización). Esta técnica además permite almacenar con el mismo formato diferentes objetos con diversas propiedades así como objetos para los cuales el almacenamiento no haya sido diseñado estrictamente. Las tablas usadas en la base de datos, una para los eventos y otra para las reacciones, permiten el almacenamiento no sólo de la información del evento en un registro con un formato común, sino también el almacenamiento de un evento o reacción que no esté dentro del conjunto inicial con el que fue diseñada la aplicación.

Para el almacenamiento y control de la información asociada a un evento, o propiamente a los eventos de un usuario, se crearon una serie de objetos y conceptos para relacionarla. Se llama Usuario (*User*) al objeto que representa a un usuario, es decir la persona o entidad que debe ser seguida y monitoreada por los cambios en su posición; un Evento (*GPSEvent*) es uno de los posibles eventos que el sistema es capaz de reconocer con respecto al movimiento del usuario; un Evento Asignado (*AssignedGPSEvent*) es un evento que ha sido asignado a un usuario, el cual puede estar habilitado o no (para efectos de control y manejo del sistema), y posee una colección de reacciones (*Reaction*) que deben ser ejecutadas al ejecutarse el evento.

Cuando ocurre un evento, y este es reportado por medio del comando ‘re’ se obtiene un objeto con los datos del evento ocurrido -Argumentos del Evento (*GPSEventArgs*)- el cual es creado, serializado y enviado desde el cliente. Cuando esto ocurre se crea un Evento Ocurrido (*OcurredGPSEvent*) el cual almacena los Argumentos del Evento junto con la información del evento asignado (*AssignedGPSEvent*).

Con fines de interoperabilidad entre sistemas, y de permitir diversas técnicas de comunicación, se planea la creación de un servicio web (*WebService*) que

entregue información acerca de los eventos ocurridos. Este funciona como respuesta a una consulta que debe indicar un rango de fechas y un código distintivo de usuario, el cual deberá ser configurado al crear cada usuario. Este servicio web se descompone en 4 funciones diseñadas para entregar de manera simple pero efectiva información relacionada al evento ocurrido así:

Tabla 2. Funciones del WebService

Función	**Parámetros**	**Salida**	**Descripción**
GetUsers	• *codesearch (String)*: Texto de búsqueda para el código del usuario. • *namesearch (String)*: Texto de busqueda para el nombre del usuario. • *imeisearch (String)*: Texto de búsqueda para el Imei del celular registrado al usuario. • *phonesearch (String)*: Texto de búsqueda para el número teléfónico del usuario.	*DataTable*: Retorna unatabla de 2 columnas con el código y nombre de los usuarios que cumplen con uno o varios de los textos de búsqueda.	Este método sirve para obtener códigos de usuario que cumplan con los criterios especificados por *codesearch*, *namesearch*, *imeisearch* y *phonesearch*. Cada código identifica a un usuario. Los criterios de búsqueda pueden ser cadenas vacías; esto le dirá a la función que no debe utilizar este parámetro como criterio.
GetUserData	• *usercode (String)*: El código del usuario del que se desea obtener información.	*string[]:* Retorna un vector de cadenas de texto, con la información del usuario así:En la posición 1 el código del usuario, en la posición 2 el nombre, en la posición 3 el *login*, en la posición 4 si se encuentra actualmente habilitado, en la posición 5 el Imei y en la posición 6 el número de teléfono asociado.	Este método es útil para conocer información adicional de un usuario. Si el método retorna *null* indica que el código de usuario ingresado no existe.

Función	Parámetros	Salida	Descripción
GetAssginedEvents	• *usercode (String)*:El código del usuario del que se desea obtener información.	*DataTable*: Retorna una tabla de datos con la información de los eventos que han sido configurados para este usuario. Las columnas de esta tabla son: • *name:* El nombre del evento asignado. • *root_event_type:* El tipo de evento principal que tiene este evento.	Este método es útil para conocer el número, nombre y tipo de los eventos que tiene un usuario actualmente habilitados para su seguimiento.
GetOcurredEvents	• *usercode (String)*: El codigo del usuario que e se desea consultar. • *from (DateTime):* La fecha y hora desde la cual se quieren obtener los eventos ocurridos. • *to (DateTime)*: La fecha y hora hasta la cual se desea obtener los eventos ocurridos.	*DataTable*: Retorna una tabla con la información de los eventos ocurridos. La tabla posee las siguientes columnas: • *event_name:* El nombre del evento asignado. • *data:* La fecha en la que ocurrió el evento. • *latitude:* La latitud en la que ocurrió el evento. • *longitude:* La longitud en la que ocurrió el evento. • *root_type_event:* El tipo del evento principal que ocurrió.	Con este método se puede obtener y filtrar información de los eventos que han ocurrido para el usuario especificado con el código, dentro del rango de fechas.

La funcionalidad más inmediata de este servicio web es ofrecer la posibilidad de realizar consultas periódicas para detectar nuevos eventos ocurridos, y permitir que sistemas de terceros reaccionen ante la aparición de estos.

5.3.1.7 Monitor LETSMonitor.

El componente monitor es el encargado de la configuración del sistema en general. Es un sitio web creado en ASP.NET cuyo fin es asistir en los procesos de configuración de usuarios, operadores, eventos asignados, y eventos ocurridos, así como en otros aspectos de configuración del sistema.

Fue diseñado de una manera sencilla y clara, haciendo uso de las técnicas y practicas de la programación en ASP.NET, con la asistencia de librerías de controles de código AJAX abierto como el *AjaxToolkit*. Esto con el fin de mejorar la experiencia de usuario y dar la sensación de estar trabajando en una aplicación de escritorio y simultáneamente aprovechar las ventajas únicas que proporcionan las tecnologías web, como la portabilidad, facilidad de mantenimiento, actualización y diseño.

Se crearon un conjunto de controles bajo los patrones de diseño de ASP.NET para lograr la integración fácil y coherente de ASP.NET con *Google Maps* (Versión 3); estos componentes proveen la funcionalidad de dibujar diversas formas sobre el mapa (áreas, líneas, marcadores, círculos), usar mecanismos de selección (áreas dinámicas, rangos dinámicos y puntos concisos), así como mostrar el mapa sobre el HTML de la pagina; adicionalmente, ya que se requiere que la experiencia de usuario se enriquezca con el uso de AJAX, el control del mapa no realiza *PostBack* alguno, por lo que se mantiene la página ligera al volver a cargar los datos de imágenes y *scripts* de Google.

Basándose en el trabajo creado para el componente servidor, haciendo uso de las librerías específicas creadas para modelar y encapsular la lógica del sistema, las tareas que el monitor debe realizar se dividieron en módulos de la siguiente forma:

- *Usuarios*:Este módulo administra la información básica de los usuarios de Nombre, *Login* y *Password*. En adición a las funciones de ingresar, editar y eliminar usuarios (sólo cuando no hay registros de eventos ocurridos) este módulo es el punto de enlace para las 2 opciones más importantes del monitor, que son: a) la configuración de eventos, y b) el monitoreo de eventos. Se presentan como vínculos ya que cada uno de estos módulos se dan por usuario, es decir, sólo se puede configurar los eventos de un usuario, o monitorear los eventos ocurridos de un usuario a la vez.
- *Configuración de Eventos*:En este módulo se configuran los eventos asignados a un usuario. Para esto se presentan una serie de controles que permiten de forma fácil la configuración; cada control de estos puede abrir cuadros de control

secundarios los cuales se presentan a manera de ventanas flotantes sobre el resto de la página, permitiendo múltiples ventanas abiertas.

1) Control de eventos asignados: Desde este se pueden crear nuevos eventos asignados, eliminar los existentes, o seleccionarlos para su edición.

2) Editor de propiedades: En diversos casos muestra las propiedades de lo que el operador haya seleccionado; así, al seleccionar el evento asignado, en el control de propiedades se muestra la información para su edición. Desde el editor de propiedades se permite la configuración de las reacciones, la configuración de los eventos y sus propiedades, y en general la configuración del evento asignado (*AssignedGPSEvent*).

El editor de propiedades es un control desarrollado para el proyecto. Es una herramienta poderosa ya que muestra las propiedades de un objeto de programación y permite la edición del mismo, haciendo posible modificar las propiedades de cualquier objeto creado (salvo algunas excepciones por la naturaleza de las propiedades del objeto).

3) Control de la cadena de eventos: Es un control herencial. La cadena de eventos, como se describirá más adelante con más detalle, es una secuencia de eventos que debe producirse para considerar que un evento ha ocurrido. Posee unas jerarquías para determinar el principio de la cadena y el curso posible de esta. Este control proporciona una representación visual de la cadena para la edición de cada evento dentro de la misma, donde cada elemento es un evento que puede ser seleccionado para su correspondiente edición en el control de propiedades de la página.

4) Control de mapa: Está en el área central de la página. Permite ver de forma gráfica, sobre el mapa, algunos parámetros de configuración de los eventos. Se aplica para el caso de los eventos de área (*GPSAreaEvent*) y de proximidad (*GPSProximityEvent*) mostrando el área de influencia del evento y el rango de proximidad respectivamente. Además de mostrar la información, el control de mapa es interactivo. En el caso del evento de proximidad permite escoger el origen y distancia que cubrirá el evento sobre el mapa; y para el evento de área permite definir el área de interés al seleccionar los vértices de la misma sobre el mapa con el mouse.

5) Control informativo: En adición a los anteriores controles se presenta el control de datos de evento, el cual despliega información relevante sobre cada evento dentro de la cadena de eventos y el color con que cada uno de los eventos es

representado en el mapa. Esto último, con el propósito de distinguirlos y tener una visión más clara de los parámetros de la cadena de eventos.

- *Monitor de Eventos*:Este módulo permite visualizar la ocurrencia de los eventos asignados a un usuario a lo largo del tiempo. La selección de eventos que se muestran sobre el mapa puede restringirse mediante un rango de fechas y la elección de un evento asignado del usuario escogido.

Se presentan ante el operador los controles para seleccionar el evento a monitorear. Al elegir el evento, el control de mapa -que se encuentra en la mayor área de la página- se actualiza y muestra un marcador sobre cada punto donde ocurrió el evento. Al seleccionar cada marcador se despliega toda la información relacionada (*GPSEventArgs*) con el evento ocurrido, incluyendo fecha, hora, datos de posición, etc.

En adición a la información anterior, se muestra en un control la información del evento asignado, esto es, la información de configuración que permite determinar cuándo y como ha ocurrido un evento.

La información que se muestra en el monitor se actualiza cada 2 minutos sin necesidad de recargar la pagina. De esta forma el monitor puede utilizarse para visualizar en tiempo real los eventos ocurridos.

- *Dashboard*:Esta es la primera página que ha de ser mostrada al operador cuando inicie sesión en el monitor. En ella se presentan los 10 últimos eventos ocurridos en el sistema en forma de tabla y también sobre un mapa. Adicionalmente se muestran el número de eventos ocurridos que se han producido para cada usuario desde principio de mes.
- *Configuración*: En este modulo se administran algunas opciones globales del sistema, como son las especificaciones de correo electrónico (Servidor SMTP) y carpeta para guardar datos temporales (la cual debe tener permisos de escritura)

5.3.2 Eventos de posición.

Un evento se define como ‘algo que pasa’; en ciencias de la computación se refiere a una posible acción que el usuario puede realizar que es monitoreada por una aplicación o sistema operativo. Cuando un evento ocurre se invoca un ‘manejador’ de evento, el cual es una pieza de software que realiza una acción específica con respecto al evento. Esta definición se ajusta perfectamente al concepto de evento al que hace referencia este proyecto, pero restringida a

acciones que tengan que ver con el cambio de posición del usuario; a estos se eventos se les denomina 'eventos de posición'.

Para el desarrollo del proyecto se diseñaron un conjunto base de eventos de posición que pueden ser configurados y detectados por el sistema. Estos eventos son:

5.3.2.1 Evento de Proximidad (GPSProximityEvent).

Este evento ocurre cuando el usuario se encuentra a una distancia 'r' determinada de un punto dado. Esta distancia es en línea recta medida desde el punto hasta la posición actual del usuario. Se puede configurar de 3 modos posibles: *Near*, *Away*, *Always*. En el modo *Near* el evento ocurre cuando el usuario se encuentra a menor o igual distancia 'r' del punto dado. En modo *Away* el evento ocurre cuando el usuario se encuentra a mayor distancia que 'r' del punto dado. El modo *Always* ocurre siempre, independiente de la distancia a la que se encuentre. Este último modo es útil cuando se quiere usar este evento para reportar en todo momento la distancia que existe entre el punto y el usuario.

Para medir la distancia que existe entre 2 posiciones GPS se usó la 'fórmula de Haversine' para calcular distancias sobre un gran círculo conociendo sus coordenadas en latitud y longitud.

5.3.2.2 Evento de Área (GPSAreaEvent).

Este evento ocurre dependiendo de la posición de un usuario con respecto a un área determinada, esta área (*GPSArea*) se define como un polígono cerrado con un conjunto de vértices. Para determinar las condiciones de ocurrencia se presentan 5 modos diferentes: *Outside*, *Inside*, *Change*, *Enter*, *Exit*. En el modo *Outside* el evento ocurre cuando el usuario se encuentra fuera del polígono. El modo *Inside* se configura para que el evento ocurra cuando el usuario esté dentro del polígono. En el modo *Change* el evento ocurre cuando el usuario pasa de estar dentro del polígono a fuera, o viceversa, en otras palabras, cuando ocurre un cambio de estado con respecto al área. En el modo *Enter* el evento ocurre cuando el usuario entra al área, y en el modo *Exit* cuando éste sale del área.

Para determinar si la posición del usuario está dentro o fuera del área se uso un algoritmo conocido como PIP (*Point In Polygon*) en geometría, el cual se basa en

determinar el número de veces que la recta horizontal que pasa por el punto corta los segmentos de recta que forman los vértices consecutivos del polígono antes de llegar efectivamente al punto[6].

5.3.2.3 Evento de Distancia (GPSDistanceEvent).

Este evento ocurre cuando el usuario ha recorrido una distancia determinada, medida en metros.

5.3.2.4 Evento de Velocidad (GPSSpeedEvent).

Este evento lleva registro de la velocidad actual del dispositivo, y ocurre basado en un valor límite de velocidad y 4 modos posibles: *Any*, *Above*, *Equals*, *Below*. En el modo *Any* el evento ocurre sin importar la velocidad actual; al igual que el evento de proximidad en modo *Always*, este modo sirve para registrar la velocidad independientemente del límite en todo momento. En el modo *Above* el evento ocurre si la velocidad actual del dispositivo es mayor que el límite; en Equals si la velocidad es estrictamente igual al límite, y *Below* si la velocidad es menor que el límite.

5.3.2.5 Evento de Tiempo (GPSTimeEvent).

El evento de tiempo ocurre cuando se supera un intervalo de tiempo previamente especificado (en segundos). La ocurrencia del evento se detecta cuando el tiempo entre lecturas de posición supere o iguale el intervalo de tiempo determinado. El evento de tiempo tiene un modo que puede activarse y desactivarse: cuando el modo *repeat* se activa, el evento se dispara repetidas veces cada vez que el intervalo de tiempo se produce; si el modo *repeat* se desactiva, el evento ocurrirá una única vez, cuando se supere el intervalo de tiempo, y continuará detectándose como ocurrido cada vez que se evalúe.

Este evento posee una propiedad especial llamada *ClearEventIndex* que hace que el conteo de tiempo se reinicie ante la ocurrencia de otro evento que, en una cadena de eventos, está encadenada al evento de tiempo. Esto es útil en casos

[6] BOURKE, Paul. 1987. "Determining if a point lies on the interior of a polygon". Consultado en: http://local.wasp.uwa.edu.au/~pbourke/geometry/insidepoly/)

donde se requiera que de acuerdo a una condición, por ejemplo, que el usuario salga de un área, el conteo de tiempo que lleva el evento se reinicie.

5.3.2.6 Evento de Fecha/Hora (GPSDateTimeEvent).

El evento de fecha/hora no ocurre en presencia de cambios de posición como los anteriores eventos; este evento ocurre de acuerdo a la fecha en la que la lectura de la posición es realizada. Fue creado como un evento auxiliar que puede ayudar a la configuración de eventos más complejos. Este evento se configura basado en 2 parámetros, el modo y la condición. El modo en este caso determina la parte de la fecha/hora que se va a utilizar para la evaluación de la ocurrencia, y existen 3 modos posibles *Full_DateTime*,*Time*,*Date*. En *Full_DateTime* se utilizará toda la fecha y la hora para la evaluación; en *Time* se usará solo la hora, minutos y segundos de la fecha, y en *Date* se usará solo la fecha, sin importar la hora. El segundo criterio es la condición, elegible entre 3 condiciones:*After*,*Before*,*Equal*. En la condición *After* el evento ocurrirá si la parte de la fecha actual es mayor que la parte de la fecha límite determinada por el evento; en el caso de *Before* será si es menor, y en el caso *Equals* si son iguales ambas partes.

5.3.2.7 Evento de Conteo (GPSCounterEvent).

Este evento por sí sólo no posee significado; cuando se usa en una cadena de eventos, este evento ocurre cuando un límite 'n' de eventos ha ocurrido.

5.3.2.8 Objetos asociados a la producción de eventos.

Los eventos creados son objetos que se encuentran codificados de igual forma tanto en el cliente (LETSClient) como en el servidor (LETSServer). Sin embargo, a pesar de ser iguales en su estructura difieren en sus funciones. El objeto del cliente tiene la responsabilidad de evaluar si el evento ha ocurrido basado en las lecturas de GPS (*GPSRead*) que el sistema le otorga, mientras que el objeto del servidor existe para propósitos de configuración de un evento que ha ocurrido o puede ocurrir. Cuando un evento ha ocurrido genera un objeto argumento (*GPSEventArgs*), que es específico para cada tipo de evento, con los datos asociados a la ocurrencia del mismo. Este objeto también se encuentra codificado tanto en el cliente como en el servidor; es creado en el cliente y luego es serializado para transmitirlo al servidor, quien lo interpreta y almacena.

Todo objeto argumento de evento (*GPSEventArgs*) posee como propiedades: la fecha y hora en que el evento fue producido, la lectura de posición GPS en la cual ocurrió, y el identificador del evento que fue producido.

Tabla 3. Propiedades de los Objeto Argumento de eventos

Nombre	Evento que lo produce	Datos
GPSAreaEventArgs	Evento de Área	*State (AreaEventState):* Puede ser uno de 3 estados posibles, *Unknown, Inside, Outside*.
GPSCounterEventArgs	Evento de Conteo	*Times (Int32):* El número de veces que ocurrieron los eventos en la cadena de eventos.
GPSDateTimeEventArgs	Evento de Fecha/Hora	*Date (DateTime):* La fecha en la cual la evaluación del evento fue positiva.
GPSDistanceEventArgs	Evento de Distancia	*Distance (double)*: La distancia que el usuario a recorrido
GPSProximityEventArgs	Evento de Proximidad	*Distance (double):* La distancia a la que actualmente se encuentra el usuario del punto.
GPSSpeedEventArgs	Evento de Velocidad	*Speed (double)*: La velocidad actual del usuario.
GPSTimeEventArgs	Evento de Tiempo	*TimePass (Int32)*: El número de segundos que ha pasado desde la evaluación del evento.

5.3.2.9 Cadena de eventos.

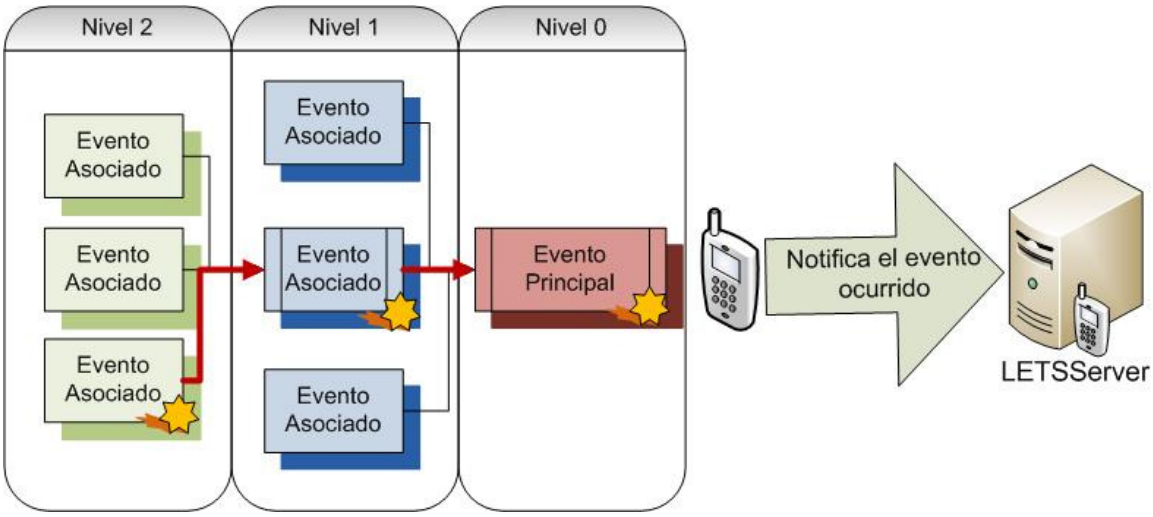
Aunque el catálogo de eventos proporcionado cubre los aspectos básicos de un evento que se desearía seguir, se requiere la posibilidad de diseñar eventos complejos cuya ocurrencia esté dada por una serie de condiciones previas y en muchos casos complejas, dado que el sistema pretende automatizar el análisis de eventos de posición. Se crea entonces el concepto de cadena de eventos.

Cada evento individual puede contener una serie de eventos asociados o 'eventos hijos', cuando un evento posee eventos asociados el proceso de evaluación para

determinar la ocurrencia del evento varía; en lugar de evaluar la posición entregada por el lector GPS (*GPSReader*) directamente, se evalúa el evento sólo si alguno de sus eventos asociados ha ocurrido. Visto de otra forma, cuando un evento asociado ocurre le notifica al evento principal que ha ocurrido, y el evento principal evaluará si él también ha ocurrido basado en la lectura GPS (*GPSRead*) con la cual ocurrió el primer evento.

Un evento asociado puede constituirse a su vez en una subcadena de eventos, donde el evento asociado es el evento principal de esa cadena y tiene eventos asociados frente a cuya ocurrencia se evalúa si se ha producido o no el evento principal.

Ilustración 15. Cadena de eventos



De esta forma, se puede usar la cadena de eventos para generar eventos mas específicos que serán desencadenados en orden, siguiendo el patrón diseñado por la cadena de eventos.

Como un ejemplo práctico de la cadena de eventos se presenta la siguiente situación: Se desea que se genere un evento cuando un usuario salga de la oficina principal y tarde más de media hora en volver.

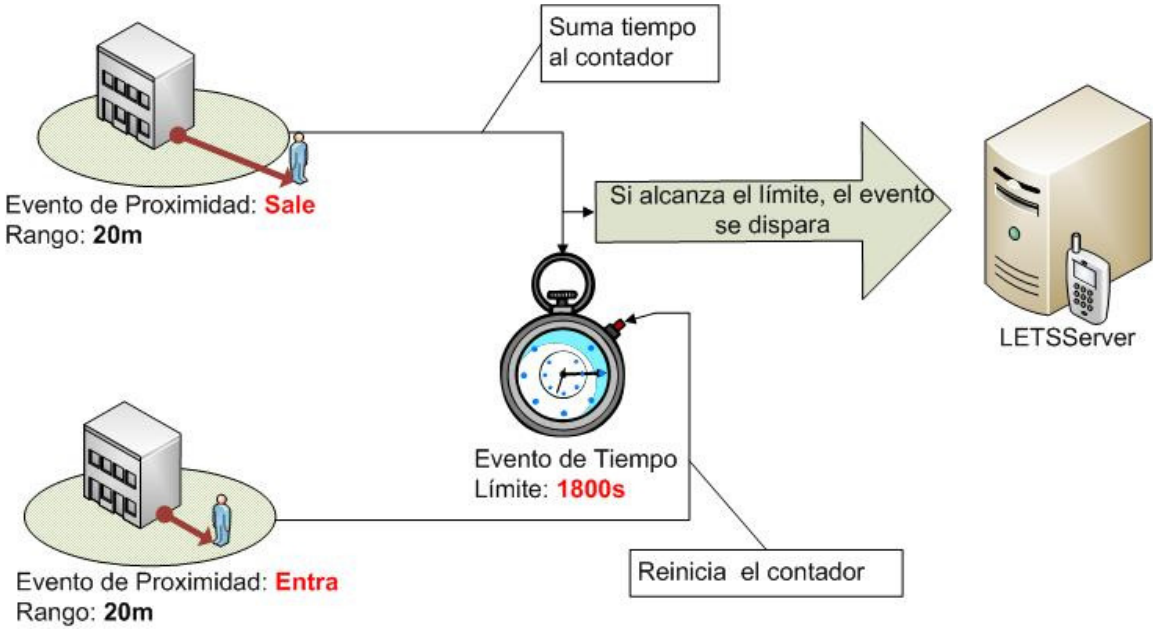
Para el caso anterior ninguno de los eventos base se ajusta al requerimiento exacto, pero al usar la cadena de eventos podríamos configurar un evento de la siguiente forma dadas las siguientes consideraciones:

Sea GPSOficina las coordenadas del centro de la oficina y considérese que dentro de un radio de 20 metros de dichas coordenadas el usuario está adentro; si está en un punto fuera de dicho radio, se considerará que está afuera.

Ilustración 16. Cadena de eventos representada en una estructura de árbol

En esta configuración de ejemplo el evento se evalúa desde los eventos más externos. Esta (en la gráfica) es una estructura de árbol en teoría computacional. La evaluación inicia desde las hojas del árbol; en este caso, si el usuario se encuentra fuera del área se dispara el 'Evento de Proximidad Sale', y continuará disparándose mientras el usuario esté por fuera del radio de 20 m previamente establecido. Cada vez que el 'Evento de Proximidad Sale' se dispare el 'Evento de Tiempo' se evaluará en función del tiempo transcurrido desde la primera vez que se detectó que el usuario salió del área, hasta que se completen 1800 segundos, momento en el que el 'Evento de Tiempo' se considera ocurrido. En el caso de que el usuario entre al área antes de que se completen los 1800 segundos (30 minutos) se disparará el 'Evento de Proximidad Entra', el cual hará que el 'Evento de Tiempo' reinicie su contador, dado que la propiedad *ClearEventIndex* esta en 1, indicando que cuando el evento con índice 1 ocurra (los índices inician desde 0) reiniciará el contador, dejando el evento listo para ocurrir cuando el usuario vuelva a salir del área.

Ilustración 17. Cadena de eventos ejemplo



La cadena de eventos es una herramienta muy útil para dar especificidad a los eventos, poner condiciones y atenuantes a la ocurrencia de los eventos. Es de vital importancia recordar que su funcionamiento inicia en los nodos hojas o eventos asociados y avanza hacia los hacia los nodos padres o evento principal.

5.3.3 Reacciones.

Una reacción, para efectos del sistema, es una tarea que debe ejecutarse al ocurrir un evento asignado. Las reacciones son diseñadas en el servidor (LETSServer) y son ejecutadas por este mismo, su función es automatizar el proceso de responder a un evento, que normalmente podría ser realizado por un operador cuando el evento se detectase. Las reacciones son acciones digitales que interactúan de alguna forma con otros servicios informáticos para ejecutar alguna tarea más compleja, fuera del sistema, aunque pueden usarse para ejecutar acciones también sobre el servidor.

El sistema cuenta con un conjunto de reacciones básicas configurables, estas son:

5.3.3.1 ***Reacción de Correo Electrónico (EmailReaction).***

Esta reacción envía un correo electrónico a los destinatarios configurados en formato HTML y utilizando una plantilla para el título del email y otra para el contenido del mensaje. La utilización de plantillas permite que el mensaje enviado tenga contenido estático o dinámico. Para el correcto funcionamiento de esta reacción debe configurarse correctamente el servidor SMTP en la configuración global del sistema. (Ver 5.3.3.7 Plantillas)

5.3.3.2 Reacción de Petición a Página Web (HttpRequestReaction).

Este tipo de reacción genera una petición a una página web determinada por el parámetro URL de la reacción; la petición puede ser por *GET* o *POST* y los parámetros que han de usarse son obtenidos del evento ocurrido y pueden ser estáticos o dinámicos; el uso de parámetros dinámicos se definirá en la sección 5.3.3.7.

5.3.3.3 Reacción de mensaje (MessageReaction).

Envía por medio del sistema un mensaje a la aplicación cliente (LETSClient), la cual lo mostrará en pantalla y alertará al usuario. El mensaje puede definirse para varios usuarios, configurados en la lista de destinatarios, y el contenido del mensaje puede ser estático o dinámico de acuerdo a la plantilla que se use. (Ver 5.3.3.7 Plantillas)

5.3.3.4 Reacción de Mensaje de Texto (SMSReaction).

Esta reacción hace uso de un dispositivo celular conectado físicamente en el servidor en un puerto serial de comunicación (COM). La aplicación envía por medio de este dispositivo, y mediante comandos AT en formato PDU, el mensaje a los números telefónicos configurados. El mensaje puede ser estático o dinámico según la plantilla usada. Para el uso de esta reacción debe asegurarse la correcta comunicación entre el equipo usado como servidor y el dispositivo celular, y la posibilidad del dispositivo celular para enviar mensajes de texto (disponibilidad de red, saldo, batería etc.).

5.3.3.5 Reacción a Servicio Web (WebServiceReaction).

Ofrece la posibilidad de notificar a un servicio web (*WebService*). Para esto debe especificarse la URL del servicio web así como el nombre del método que se usara, y los parámetros que se enviarán. Los parámetros pueden ser estáticos o dinámicos según la plantilla utilizada. Para este tipo de reacción hay que asegurarse de que el directorio temporal en la configuración general del sistema esté disponible en lectura y escritura, y que los datos del servicio web estén correctos. Se prefiere el uso de parámetros sencillos (primitivos).

5.3.3.6 Plantillas y Transformaciones para contenido Dinamico.

Debido a la naturaleza diversa de los eventos, y dado que el sistema pretende comunicarse por diversos medios con otros sistemas, se presenta una alternativa en cuanto al contenido de los datos de las reacciones. De esta forma, se crea un sistema de plantillas para transformar el contenido de acuerdo a los datos que vengan en el evento ocurrido (*OcurredGPSEvent*).

- *PlantillaBásica*: Dentro de las propiedades de las reacciones en las que se permite contenido dinámico, las cuales son de tipo texto (*String*), el sistema revisará y tomará como plantilla cualquier campo que se encuentre entre caracteres '#event:<identificador>#'' donde identificador es el nombre del campo propiedad del objeto que se desea reemplazar en el texto, partiendo desde el objeto evento ocurrido (*OcurredGPSEvent*) y usando el caracter punto '.' como delimitador de objeto-propiedad. Por ejemplo, si en una reacción de email se desea en el titulo enviar el nombre del usuario que realizó el evento, se debe poner en el campo *Subject* el texto "Reacción generada por el usuario #event:user.name#", en este ejemplo el sistema reemplazara #event:user.name" por la propiedad name del objeto user del evento ocurrido.
- *PlantillaAvanzada*: Las plantillas avanzadas hacen uso de las transformaciones sobre XML, las transformaciones XSL. El evento ocurrido es serializado en XML y se le aplica la transformación XSL que el usuario escoja; esto, a diferencia de la plantilla básica, permite acceder por los datos de los eventos encadenados y aplicar plantillas a los mismos para comunicar sus datos, obteniendo mayor control sobre la transformación de contenido que se desee aplicar.

Las reacciones que soportan transformaciones XSL tienen una plantilla base para XSL la cual proporciona un ejemplo de como hacer las transformaciones.

5.4 IMPLEMENTACIÓN

Durante el desarrollo de la aplicación del presente proyecto y por la misma metodología de desarrollo de software utilizada, el sistema se construyo por fases obteniendo prototipos que fueron implementando las funciones requeridas. Cada prototipo busca demostrar la posibilidad de la función más crítica que el sistema necesita para su funcionamiento y su progreso hacia la siguiente fase.

A continuación se detallan las características y alcances de cada uno de los prototipos.

5.4.1 Primer prototipo.

La implementación de este primer prototipo abarca 2 funciones fundamentales: la obtención de la posición del dispositivo, y la comunicación con el servidor.

En la aplicación móvil (LETSCLient) se crea la clase *GPSReader* la cual es la responsable de la comunicación con el GPS del dispositivo por medio del objeto *LocationProvider*. Esta clase gestiona la obtención de la posición a intervalos regulares de tiempo y notifica esta posición a los objetos que implementen la interfaz *IGPSReadListener* que se encuentren suscritos al objeto *GPSReader*. La posición descrita se representa mediante un objeto *GPSRead* que contiene la información de latitud y longitud de la lectura, así como de la fecha y hora en que fue tomada la lectura. Se crea también la clase *CommandClient* que es encargada de la creación, envío y serialización de los comandos que hayan de ejecutarse en el servidor, así como de recibir las respuestas y notificarlas; un objeto *CommandClient* envía comandos, los cuales están representados por la clase *wCommand*. Los parámetros que reciben los comandos deben ser variables primitivas u objetos que implementen la interfaz *ISerializable*. Cuando el servidor envía la respuesta del comando, el *CommandClient* notifica a los objetos que implementen la interfaz *ICommandListener* que hayan sido registrados en el.

En un formulario básico se muestran la latitud, longitud y estado del GPS obtenidos por el *GPSReader*. El formulario incluye opciones para iniciar las lecturas y detenerlas; también una opción para enviar la posición actual al servidor en forma de un comando y mostrar la respuesta en pantalla.

Ilustración 18. Formulario básico del cliente móvil

En el servidor se crea la aplicación LETSServer. Se configura el *IHttpHandler* para recibir los comandos del cliente; posteriormente, se crea el objeto *CommandServer* el cual es el encargado de des-serializar los comandos e invocar al objeto *ICommandListener* que se encuentra registrado en él, y que corresponde con el texto de comando que viene en el comando del cliente (*Command*). Se crea un *ICommandListener* de prueba para recibir la lectura GPS (*GPSRead*) del cliente y devolver el texto, "Lectura registrada con éxito".

Ilustración 19. Respuesta al envío de la lectura

5.4.2 Segundo prototipo.

El segundo prototipo demuestra la posibilidad de la utilización de la API de *Google Maps* dentro del sistema, más específicamente en el área del monitor (LETSMonitor), y la utilización del almacenamiento de datos por parte del servidor y monitor.

La API de *Google Maps* para visualizar y manipular mapas está escrita enteramente en JavaScript, por lo que para integrarla de una forma coherente dentro de la filosofía de construcción de sitios web de ASP.NET se construyeron controles de ASP.NET (*ScriptControl*) para crear una interfaz entre el código de ASP.NET y Javascript. De esta forma es más sencillo el desarrollo y reutilización de estos componentes en futuros proyectos. Las librerías *OpenSource* con controles para *Google Maps* disponibles a la fecha de realización del proyecto no contemplaban el uso de la versión 3 de *Google Maps*, por lo que se vio la necesidad y oportunidad de crear controles personalizados.

Estos controles permiten visualizar Mapas (*MapControl*), dibujar superposiciones (Polígonos, Polilíneas, Marcadores, Imágenes, Círculo) en el mapa (*MapDrawControl*) y seleccionar elementos sobre el mapa (*MapSelectionControl*). También permiten reaccionar ante eventos como *click* y *drag&drop* sobre el mapa o superposiciones.

lustración 20. Controles de dibujo del mapa

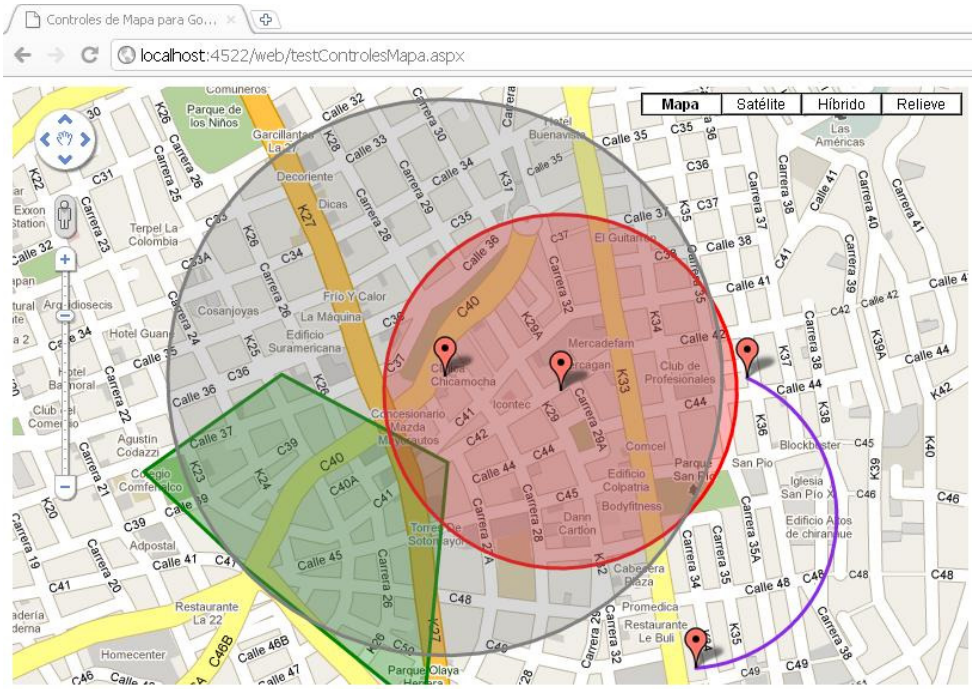
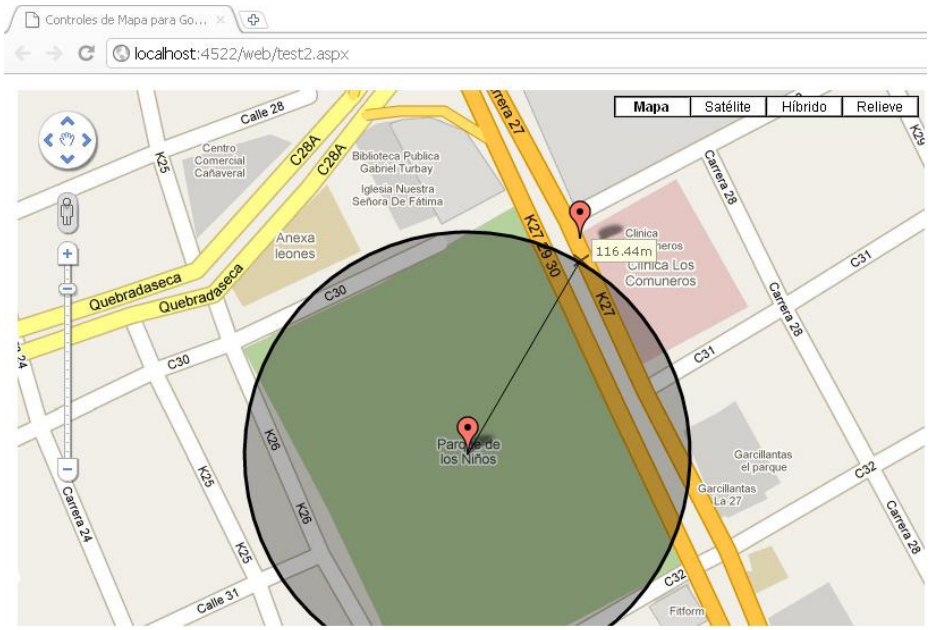


Ilustración 21. Controles de selección del mapa



Se crean tanto para el servidor (LETSServer) y monitor (LETSMonitor) objetos para manejar el acceso a datos (*DataBaseSource*) y se establece un modelo para los objetos que vendrán a representar la lógica de funcionamiento del sistema (*Entity*, *EntityProvider*, *EntityManager*).

Se añade la funcionalidad de almacenar la posición (*GPSRead*) al Comando de Servidor (*ICommandListener*) que se creo en el prototipo anterior, la cual después es recuperada por una página del sitio monitor (LETSMonitor), y mostrada por medio del uso de los controles previamente creados.

5.4.3 Tercer prototipo.

En este prototipo se crean los objetos que representan los eventos (*GPSEvent*) tanto en la aplicación cliente (LETSClient) como en la aplicación servidor (LETSServer), los cuales incluyen los algoritmos necesarios para evaluar si el evento ha ocurrido, y la notificación de esta ocurrencia.

Inicialmente se crea la definición base de evento (*GPSEvent*), el cual permite registrar una serie de objetos que serán notificados cuando el evento ocurra.

Estos objetos deberán implementar la interfaz *IGPSEventListener* para ser notificados. Cada evento posee un conjunto de eventos hijos (*GPSEventCollection*), esto con el fin de representar la cadena de eventos que debe ejecutarse antes de ocurrir el evento. Para notificar al evento de cuando uno de sus eventos asociados ha ocurrido cada evento implementa para sí mismo la interfaz *IGPSEventListener*, y para poder evaluar si el cambio de posición hace que el evento ocurra, cada evento también implementa la interfaz *IGPSReadListener* de esta forma obtiene la posición por medio de un *GPSReader*.

Entonces, de acuerdo al diseño, se implementan los eventos *GPSProximityEvent*, *GPSAreaEvent*, *GPSDistanceEvent*, *GPSSpeedEvent*, *GPSTimeEvent*, *GPSDateTimeEvent* y *GPSCounterEvent*, así como los argumentos de cada evento (*GPSEventArgs*), que son los objetos que almacenan la información del evento cuando ocurre.

Adicionalmente, los eventos deben ser deserializados dado que provienen del servidor, y los argumentos de evento deben ser serializados para enviarlos al servidor como notificación de que un evento ha ocurrido. Por esto tanto los eventos (*GPSEvent*) y los argumentos de evento (*GPSEventArgs*) implementan la interfaz *ISerializable*.

Ilustración 22. Eventos cargados en el móvil desde el servidor

Se crea aquí un comando de servidor, el comando *RequestConfigurationCommand*, que trasmitirá los eventos configurados, y otro comando *ReportEvent* que almacenará los argumentos de evento que reciba como reporte de la ocurrencia de un evento.

Para el almacenamiento de los eventos y argumentos se crean los objetos *AssignedGPSEvent* y *OcurredGPSEvent* con sus respectivos objetos en el modelo de negocio de la aplicación (*AssignedGPSEventProvider*, *AssignedGPSEventManager*, *OcurredGPSEventProvider*, *OcurredGPSEventManager)*.

Finalmente, se elabora una opción en la aplicación cliente para visualizar los efectos de la evaluación de cada evento. Para esto se crea una superficie de dibujo bajo coordenadas GPS usando la clase *Canvas* de J2ME, esto con el fin de verificar la correcta evaluación de cada evento.

Ilustración 23. Menú inicial del cliente móvil

Ilustración 24. Visor de eventos

Ilustración 25. Representación visual del Evento de Distancia

Ilustración 26. Representación visual del Evento de Proximidad

5.4.4 Cuarto prototipo.

En este prototipo se elaboran las reacciones, su estructura base y el conjunto de reacciones que proporcionara el sistema. También se definen el concepto de usuario, operador en el monitor, las páginas para configurar los eventos y consultar los eventos ocurridos.

Se crea la clase que representa una reacción (*Reaction*); esta clase es la base para construir las reacciones propias del sistema. Con ella se crean las clases del modelo de negocio *ReactionProvider* y *ReactionMananger* , un objeto reacción es notificado de la ocurrencia de un evento gracias a la implementación de la interfaz *IReaction*. Adicionalmente, a un evento asignado (*AssignedGPSEvent*) se le adhiere como propiedad una colección de reacciones (*ReactionCollection*) a ejecutar para cuando ocurra el evento.

Aquí se implementan las reacciones *EmailReaction*, *SMSReaction*, *MessageReaction*, *WebServiceReaction* y *HttpRequestReaction*. Cada reacción implementa una transformación de contenido, es decir, el uso de plantillas para modificar el texto de acuerdo a los datos del evento ocurrido (*OcurredGPSEvent*). Para ofrecer esta funcionalidad en común todas las reacciones heredan de la clase *ContentTrasnformableReaction*, la cual provee métodos para transformar el

texto de acuerdo a la plantilla, y también ofrece métodos para aplicar una plantilla XSL a la transformación XML del evento ocurrido.

Los objetos que representan al usuario y los necesarios para su modelamiento en el sistema (*User*, *UserProvider*, *UserManager*) son implementados; y se le adhiere como propiedad un objeto *User* al objeto *AssignedGPSEvent* para determinar a qué usuario se le ha de asignar el evento en cuestión.

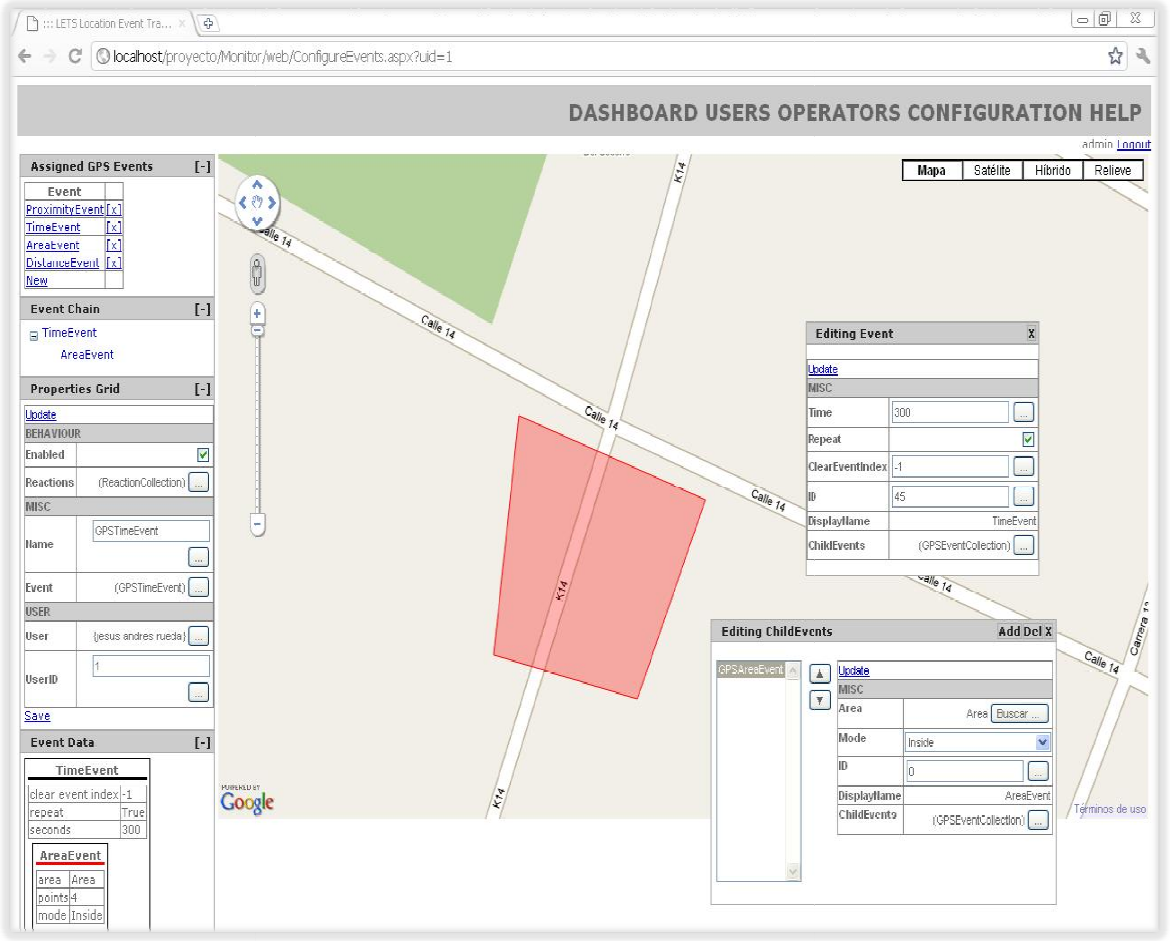
En esta fase es delimitado el concepto de operador, y los objetos que lo modelan (*Operator*, *OperatorProvider*, *OperatorManager*). Dado que el operador tendrá participación en el sistema sólo desde la aplicación monitor (LETSMonitor), esta se implementa bajo el modelo de ASP.NET para usuarios Web *MembershipProvider* y *RoleProvider*; esto con el fin de agilizar el desarrollo del sitio usando controles y metodologías propios de ASP.NET.

Para las páginas de configuración de eventos (*ConfigureEvents.aspx*) se crea el control editor de propiedades (*PropertiesGrid*), el sistema de ventanas flotantes de contenido (*WindowControl*) y el control de visualización de la cadena de eventos (*GPSEventTreeControl*). Estos se integran y asocian con las acciones de los demás controles de mapa para proveer la funcionalidad requerida de editar las propiedades de un evento asignado.

Para la pagina de visualización de eventos (*Monitor.aspx*) se utilizan los controles de mapas (*MapControl*, *MapDrawControl*) para visualizar la información de los eventos ocurridos. Tanto la página de monitoreo como la de configuración reciben como parámetro (por medio de *QueryString*) el id del usuario del cual se desea mostrar o configurar los eventos.

Se implementa aquí el servicio web, *LETSWebService*, sobre el sitio del Servidor (LETSServer). Este *WebService* esta creado con las herramientas que son provistas por ASP.NET y realizado bajo la especificación de SOAP, dando como resultado el archivo *LETSWebService.asmx.* .

Ilustración 27. Controles para la edición de eventos en el monitor

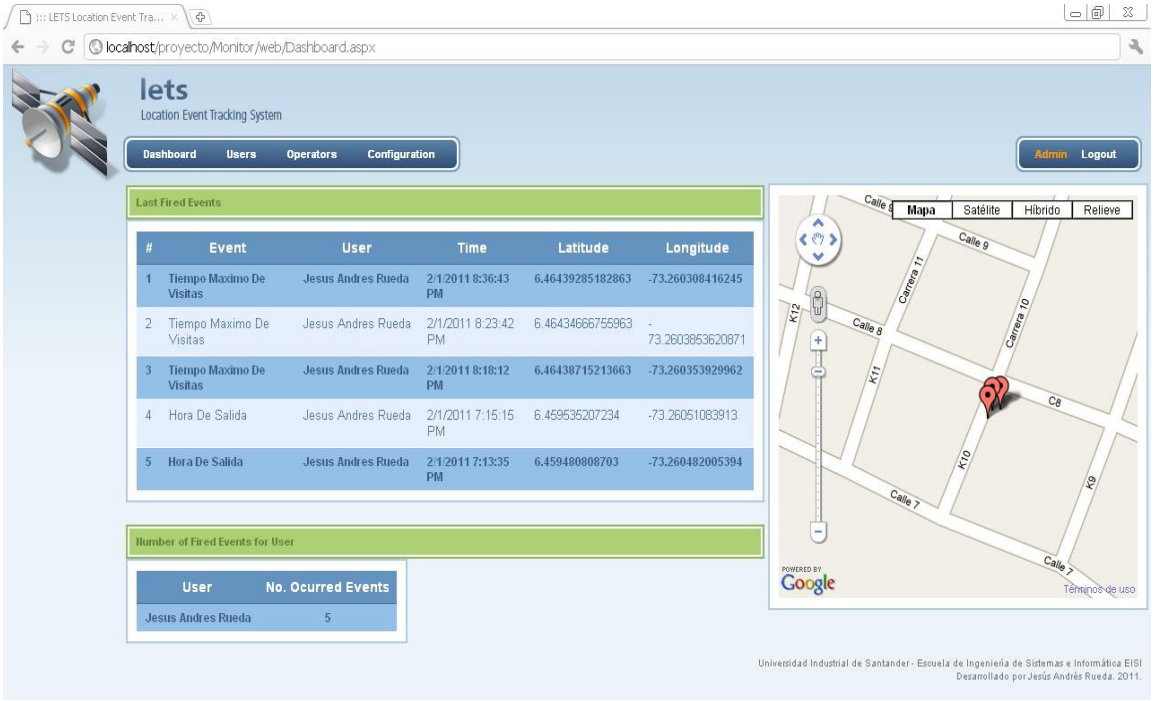


5.4.5 Quinto prototipo.

En este prototipo se termina la construcción de la aplicación estableciendo el diseño gráfico definitivo sobre la LETSMonitor, implementando las funciones de AJAX sobre el mismo y definiendo el diseño gráfico definitivo sobre LETSClient.

Aquí se crean las páginas de administración de usuario (Users.aspx) y operadores (*Operators.aspx*), se elabora el menú principal, se establece el diseño con el uso de CSS y *Themas* de ASP.NET, y se implementa la página de configuraciones globales (*Configuration.aspx*).

Ilustración 28. Dashboard LETS



Usando las tecnologías AJAX implementadas para ASP.NET como *UpdatePanel* y *UpdateProgress* se trabajó en hacer más atractivo el uso de la interfaz. También se incluye la librería de controles de código abierto *AjaxToolKit* para el uso de controles como paneles colapsibles y calendarios de fecha para enriquecer la experiencia del usuario.

En el cliente se crean el formulario de *login* y el formulario de configuración de los parámetros de la aplicación, y se eliminan los controles que mostraban datos de depuración.

Ilustración 29. Monitor LETS

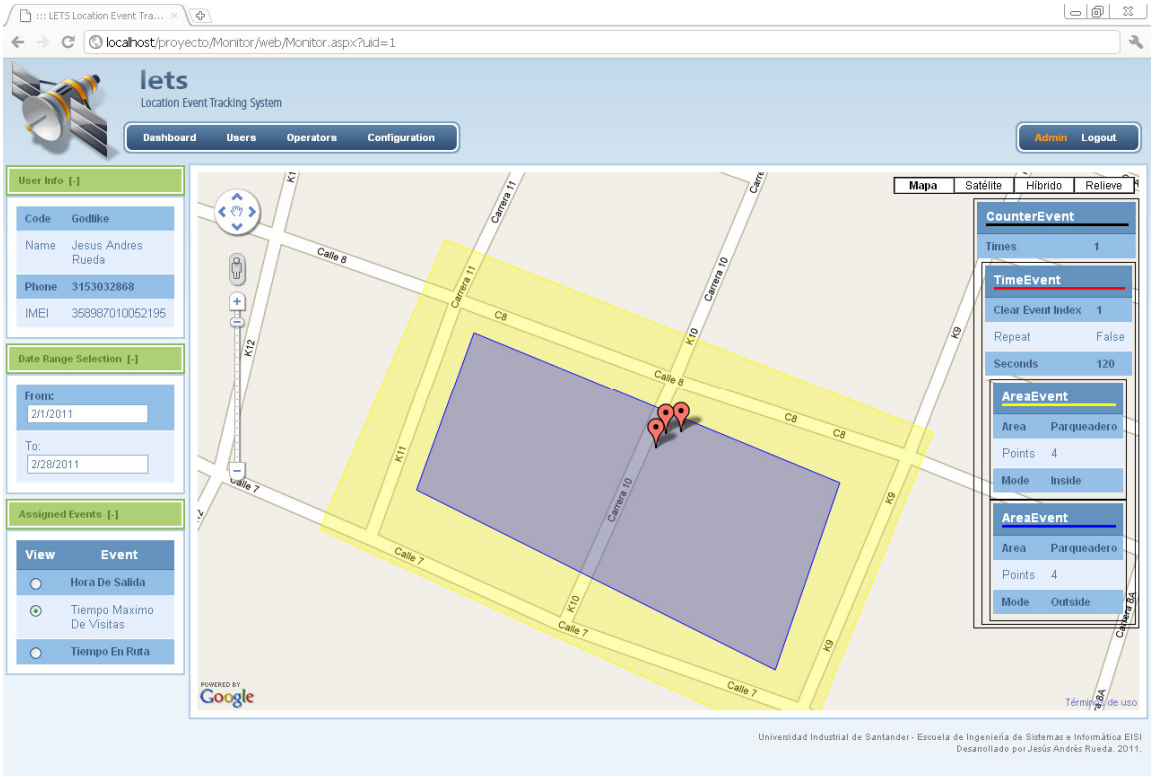
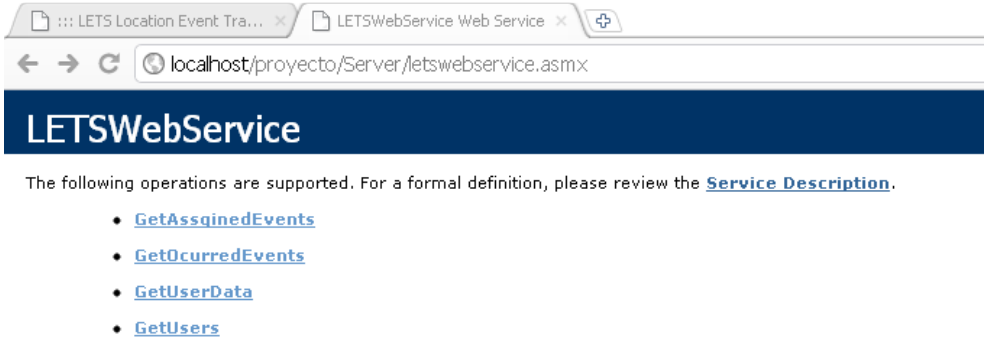


Ilustración 30. Webservice LETS



6 CONCLUSIONES

El objetivo del proyecto "*Diseño, análisis e implementación de una herramienta de software orientada a geolocalización y seguimiento por eventos de terminales móviles*" fue el desarrollo de un conjunto de aplicaciones prototipo capaces de automatizar los procesos de seguimiento y reacción a eventos de posición, al procesar y analizar la posición y sus cambios de acuerdo con ciertas condiciones predeterminadas, para generar reacciones como consecuencia de dichos eventos y comunicarlas a los usuarios y a sistemas de información externos.

Al finalizar el proyecto se han alcanzado los objetivos planteados al inicio, elaborando un prototipo del sistema LETS conformado por las aplicaciones LETSMonitor, sitio web que permite la configuración y visualización de usuarios, eventos y reacciones; LETSClient que se encarga de obtener la posición del usuario, procesarla para determinar si ha ocurrido un evento, y notificar la ejecución de este; y LETSServer, quien se encarga de la comunicación de los eventos configurados y ocurridos, así como de la ejecución de las reacciones.

En el transcurso del proyecto:

- Se hizo un análisis de las redes de comunicación que podrían emplearse para reportar la posición y cambios de posición de un usuario, y de los sistemas de localización disponibles, determinando que el mejor canal de comunicación es el GPRS, y que el sistema de localización es GPS. La aplicación, por tanto, se desarrolló para dispositivos móviles de gama media dotados con GPS.
- Se diseñó, desarrolló e implementó un conjunto de aplicaciones de software que realizan acciones diversas orientadas procesar y analizar la posición de un usuario de acuerdo con criterios predeterminados, para detectar así la ocurrencia de eventos y notificar los mismos a fin de generar reacciones automáticas previamente configuradas.
- Se implementó un evento diseñado para probar el software y se documentó el comportamiento del mismo a fin de presentarlo como evidencia del sistema.
- Se diseñó un tutorial de J2ME utilizando la plataforma MeiWeb de la Escuela de Ingeniería de Sistemas dando cuenta de la investigación realizada sobre el lenguaje de programación utilizado en el diseño del cliente móvil.

Del desarrollo del proyecto se concluye que:

- Los dispositivos móviles de gama media dotados con GPS y actualmente en el mercado han probado ser capaces de soportar programas que desarrollan funcionalidades complejas, toda vez que la aplicación del cliente no sólo obtiene la posición del dispositivo sino que además descarga eventos desde el servidor y detecta si estos han ocurrido para comunicarlo al servidor, por no mencionar otras funciones. El desarrollo de programas con funcionalidades complejas para dispositivos móviles es un amplio campo que permite satisfacer múltiples demandas del mercado, y cuya implementación se facilita por el costo accesible de los terminales requeridos.
- La medotología XP (*eXtreme Programming*) permite desarrollar programas de forma ágil y contando con pocos programadores. Además, permite la evolución constante del programa gracias a las sucesivas evaluaciones y correcciones, así como a la refactorización del código.
- La plataforma de diseño .NET Framework 3.5 de Microsoft probó sus capacidades para el desarrollo de librerías y de las aplicaciones de servidor y monitor, soportando un lenguaje de programación estructurada como es C#. Por su parte la versión Micro Edition de Java J2ME demostró gran funcionalidad y capacidad a la hora de desarrollar aplicaciones para dispositivos móviles, que cumplen características especiales de agilidad y ligereza dada la naturaleza de los terminales en que se ejecutan. Finalmente, JSON se probó como un formato de comunicación compacto, ágil, y por tanto adecuado para este tipo de aplicaciones.
- Aunque no fue diseñado específicamente para resolver una demanda comercial, el prototipo LETS podría ampliarse para implementarse en el campo comercial como un sistema capaz de automatizar procesos de seguimiento y reacción a usuarios que portan terminales móviles. Podría utilizarse en campos variados como, por ejemplo, prestaciones de seguridad, o aumentando la eficiencia de empleados que desarrollan labores de campo.

BIBLIOGRAFÍA

B'FAR, Reza y FIELDING, Roy T. Mobile Computing Principies: Designing and Developing Mobile Applications with UML and XML. Cambridge, Cambridgge University Press,2005.

BARTH, Matthew y FARRELL, Jay.El sistema de posicionamiento global GPS. New York, McGraw-Hill Companies Inc., 1999.

DIGGELEN, Frank van.A-GPS: Assisted GPS, GNS, and SBAS. Boston, Artech House, 2009.

EL-RABBANY, Ahmed. Introduction to GPS: The Global Positioning System. Norwood, Artech House Inc.,2006. Segunda edición.

FORUM NOKIA. Location-Based Services (En Línea) : http://www.forum.nokia.com/Technology_Topics/Mobile_Technologies/Location-Based_Services/

_____. Resources for Mobile Application Developers (en línea).http://www.forum.nokia.com/.

FROUFE QUINTAS, Agustín; JORGE CÁRDENAS, Patricia.J2ME Java 2 Micro Edition, Manual de Usuario y Tutorial. Editorial Ra–ma, 2003

FUSTER ESCUDER, Jose Miguel y MARTINEZ ROSIQUE, Juan Antonio. El sistema de posicionamiento global GPS. Valencia, Universidad Politécnica de Valencia, 1995.

GLOBAL POSITIONING SYSTEM (En Línea) http://www.gps.gov/

GOODCHILD, Michael F y LONGLEY, Paul A (2005). Geographic Information Systems and Science. West Sussex, England. John Wiley & Sons Ltd.

GPS Explained (En Línea) http://www.kowoma.de/en/gps/index.htm

Introducción a JSON (En línea). http://www.json.org/json-es.html

J2ME and Location-Based Services (En Línea) http://developers.sun.com/mobility/apis/articles/location/

JAVA TECHNOLOGY, Sun Developer Network (En línea). http://java.sun.com

KEOGH, James. J2ME: The Complete Reference. Osborne, McGraw-Hill, 2003.

KÜPPER, Axel. Location-Based Services: Fundamentals and Operation. England, John Wiley & Sons Ltda, 2005.

MALLICK, Martyn. Mobile and Wireless Desing Essentials. Indianapolis,Indiana, Willey Publishing Inc., 2003

Microsoft .NET Framework 3.5 (En línea). http://www.microsoft.com/downloads/details.aspx?familyid=333325fd-ae52-4e35-b531-508d977d32a6&displaylang=es

OVERVIEW (MID profile). Javadocs. (En línea): http://download.oracle.com/javame/config/cldc/ref-impl/midp2.0/jsr118/index.html

TOMLIN, C.Dana. Geographic Information Systems and Cartographic Modelling. New Jersey, Prentice Hall, 1991.

ANEXO A

SEGUIMIENTO GPS BASADO EN EVENTOS: UNA NUEVA APLICACIÓN PARA TELÉFONOS CELULARES.

Jesús A. Rueda, Manuel G. Becerra Flórez.

Resumen—Este artículo tiene como propósito dar a conocer la herramienta de software LETS (Location Event Tracking System) diseñada para automatizar los proceso de seguimiento y reacción por eventos a terminales móviles.

Índice de Términos—Dispositivos móviles, Eventos, Geolocalización, GPS, J2ME, Reacciones.

I. INTRODUCCIÓN

Entre todos los dispositivos móviles existentes, los teléfonos celulares se cuentan entre aquellos que han alcanzado mayor desarrollo en tiempos recientes. Estos dispositivos han sido enriquecidos en sus capacidades de procesamiento y en el tipo de servicios que pueden incorporar, a fin de brindar al usuario una experiencia cada vez más amplia y satisfactoria. Entre las recientes mejoras a los teléfonos celulares se cuentan capacidades como reproducción de música, la inclusión de cámaras fotográficas y de video dentro de los dispositivos móviles, una mayor capacidad de conectividad, etc. Uno de elementos que se integran a algunos teléfonos celulares para diversificar sus capacidades es un receptor GPS capaz de comunicarse con el teléfono y transmitirle la información. Los celulares, además, son capaces de manipular los datos provistos por el receptor y combinarlos con otra información. Por ejemplo, el celular Sony Ericsson C-702, que viene dotado con un receptor GPS y cámara fotográfica, al capturar instantáneas graba estas junto con la información de la posición en que fueron tomadas.

Los dispositivos móviles con estas características fueron de acceso restringido en el país durante mucho tiempo. Las políticas de las compañías operadoras de telefonía celular, a cargo también de la comercialización de dispositivos móviles, exigían al cliente cumplir abundantes requisitos antes de venderle un teléfono celular de alta tecnología, y además demandaban que adquiriese junto con éste un plan de datos o un plan de voz de costo elevado; así, los celulares con características tecnológicas más desarrolladas contaban con poca difusión y no eran aptos para ser utilizarse en soluciones de software dirigidas al público general.

En los últimos cinco años esta situación ha cambiado gradualmente, y en la actualidad es posible para una persona de estrato medio -o para una empresa- adquirir un teléfono celular de gama media o alta a un costo razonable e incluso sin necesidad de adquirir un plan de datos o de voz. Además, los servicios de conectividad ofrecidos por las empresas proveedoras de telefonía celular se han ampliado, y hacen posible que un cliente adquiera Internet pagando sólo el día o el tiempo que desea utilizar, sin cláusulas de permanencia.

Dado que estos dispositivos se han integrado al mercado, y tienen costos muchas veces inferiores a los de otros dispositivos móviles -como son las PDAs, pockets, iPads, etc.- es posible pensar en ellos como herramientas de hardware sobre las cuales pueden diseñarse soluciones de software encaminadas a satisfacer necesidades personales o empresariales. En este contexto se concibió la aplicación de software LETS como una herramienta portable, para ser instalada en teléfonos celulares de gama media y alta dotados con GPS.

II. SERVICIOS BASADOS EN LOCALIZACIÓN

El sistema GPS permite detectar la posición geográfica de un receptor GPS y expresarla en términos de latitud y longitud [1]. Los receptores GPS son dispositivos de tamaño reducido, que pueden obtenerse solos o integrados dentro de otros más grandes, como los teléfonos celulares y los equipos AVL (Automatic Vehicle Location) que se instalan en algunos vehículos para auxiliar el proceso de conducción[2].

Un receptor GPS, sin embargo, no puede hacer mucho más aparte de reportar su posición, y con esta información el máximo alcance sería visualizar punto a punto la posición del usuario en un mapa. Para que el conocimiento de la posición geográfica de un terminal tenga un mayor sentido, es necesario que esta

información geográfica pueda procesarse en combinación con otros datos tales como fecha, hora, velocidad, lugar, para identificar cuándo ha sucedido algo importante aplicando criterios predefinidos. Por ejemplo: La combinación de un área predeterminada o geocerca con la posición de un usuario permite establecer si el usuario está dentro de la zona delimitada o fuera; saber esto podría servir si el área delimitada es insegura y se quiere que el usuario no entre a ella, o que salga de ahí lo más pronto posible [3].

Cuando los datos de posición son tratados de esta forma pueden prestarse al usuario servicios conocidos como LBS (Located Based Services), servicios conocedores de su posición, que se definen y configuran de acuerdo con necesidades del usuario[4].

No solo quien porta el dispositivo móvil puede beneficiarse de estas prestaciones. Una tercera persona también puede desear ser notificada de los acontecimientos relevantes en lo que se refiere a la posición del usuario, o incluso, puede desear que frente a estos acontecimientos se produzcan otras reacciones automáticas diferentes de la notificación[5].

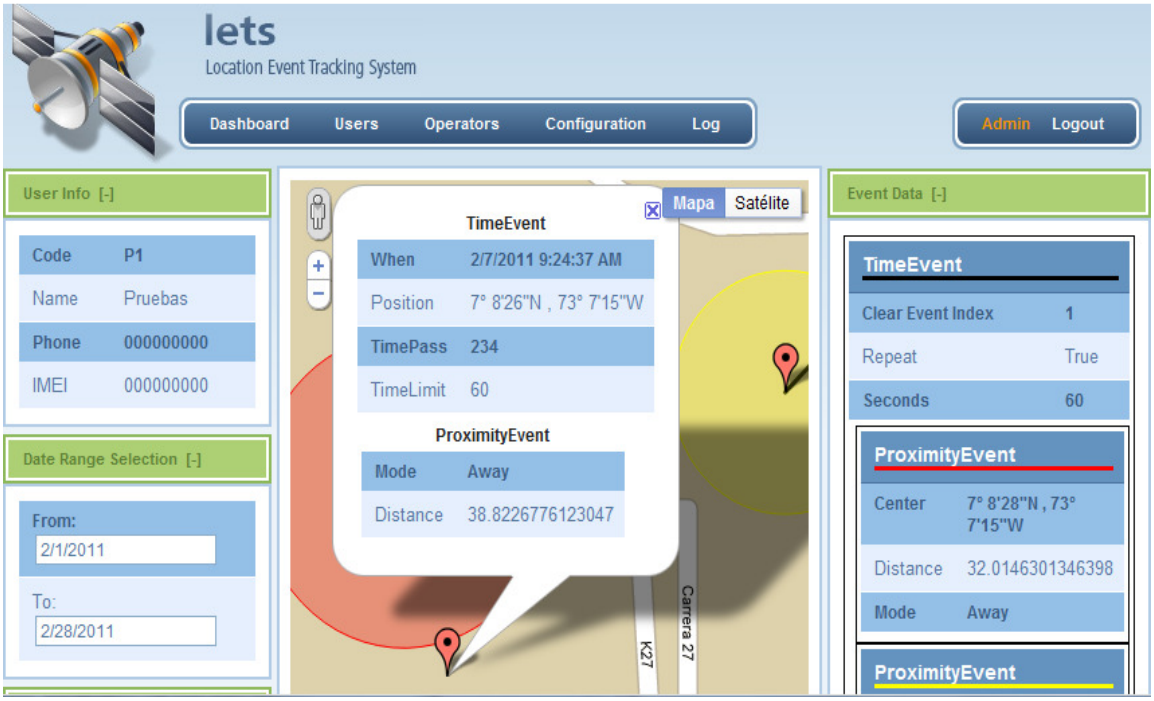


Figura 1. Representación de un evento en LETSMonitor

III. LETS: LOCATION EVENT TRACKING SYSTEM

Con el propósito de satisfacer necesidades que podrían surgir en escenarios como los anteriormente referidos, se diseñó una herramienta de software denominada LETS (Location Event Tracking System) cuyo propósito es automatizar los procesos de seguimiento y reacción a eventos de posición. Un evento de posición se define como algo que sucede en relación con la posición geográfica de un usuario el cual, combinándose con otros criterios, se considera un acontecimiento relevante frente al cual debe producirse algún tipo de reacción automática - detección, notificación, alerta, etc.- En el diseño del programa se concibieron un conjunto limitado de eventos, cada uno de los cuales se compone de ciertos parámetros a los cuales deben asignarse valores concretos a fin de poder establecer cuándo ocurre un evento. Se diseñaron un conjunto de reacciones automáticas que serían las que se desencadenarían frente a la ocurrencia de un evento. Se implementó un encadenamiento de varios eventos de posición para definir uno de mayor complejidad, y en establecer un conjunto de posibilidades en la edición de las reacciones para que su alcance sea mayor al de una simple alerta.

El diseño del sistema está conformado por tres módulos principales:

A. LETSMonitor: Es una aplicación web que permite visualizar en un mapa los eventos de posición y administrar todo lo relacionado con estos. Desde esta aplicación se establece a quién se va a seguir, cuales eventos de posición se quiere conocer, y se visualizan y editan los parámetros de configuración.

B. LETSServer: Esta aplicación es el corazón del programa; reside en un servidor web y se conecta tanto con la aplicación monitor como con la aplicación del cliente móvil. Guarda la información de los eventos y reacciones configurados y la transmite al celular; además, guarda la información de posición que captura el celular y los eventos que detecta y los comunica al monitor para que éste le permita al usuario visualizarlos.

C. LETSClient: La aplicación del cliente móvil obtiene la información del receptor GPS, obtiene los parámetros de configuración de los eventos definidos usando el monitor, y procesa la información de posición detectando cuándo ocurre un evento para notificárselo al servidor. La aplicación también le permite al usuario visualizar su posición y determinar su estado en relación con los eventos configurados.

Figura 2. Representación del evento de proximidad

IV. EVENTOS Y CADENA DE EVENTOS

Para poder automatizar los procesos de seguimiento, el sistema LETS concibe los acontecimientos relevantes en términos de eventos de posición. Un evento de posición puede entenderse como algo que sucede y que implica la posición geográfica del individuo. Un ejemplo de evento de posición puede ser un evento de área, es decir, cuando la posición geográfica del individuo está dentro de un área predeterminada.

Dentro del sistema LETS se ha definido e implementado un conjunto limitado de eventos de posición: de área, proximidad, tiempo, fecha, velocidad, contador, distancia. Cada uno de estos puede configurarse con los parámetros que indican su ocurrencia, y la posición del usuario combinada con estos parámetros permite saber si el usuario ha producido o no los eventos. Por ejemplo, configurando el evento velocidad, el sistema detectará cuándo el usuario está viajando a más de 60Km/h.

En ciertas ocasiones, los eventos que se desea monitorear son más complejos que los ya mencionados, o son una combinación de ellos. Continuando con el ejemplo anterior, puede que lo que desee saberse es cuándo el usuario está viajando a más de 60 Km/h dentro del perímetro urbano. Para estas circunstancias se diseñó el concepto de cadena de eventos: Un evento principal puede tener encadenado uno o varios eventos secundarios, de forma que cuando el evento secundario se dispare, el evento principal se evalúe

TABLA 1.

DEFINICIÒN DE LOS EVENTOS DE POSICIÓN

Evento	Propiedades	Descripción
GPSProximityEvent	• *Radius*: Define la el punto centro y la distancia desde el mismo. • *Mode:* Define 3 modos posibles, *Near, Away, Always*.	El evento de proximidad se evalúa en función de un punto y la distancia del usuario hasta este. Cuando el modo es *Near* el evento ocurre si se está a menor o igual distancia que el radio definido, si el modo es *Away* el evento ocurrirá si se encuentra a una distancia mayor, y en el caso de *Always* el evento ocurrirá independientemente de la distancia.
GPSAreaEvent	• *Mode*: 5 modos posibles, *Outside*, *Inside*, *Change*, *Enter*, *Exit*. • *Area*: Define un conjunto lineal de puntos que conforman el perímetro del area.	El evento de área evalúa la posición del usuario con respecto a un área; si el modo es *Outside* el evento ocurre cuando el usuario se encuentra fuera del área, *Inside* cuando está dentro, *Exit* cuando sale del área, *Enter* cuando entra y *Change* cuando entre o salga de ésta.
GPSDistanceEvent	• *Limit*: Define el valor en metros del recorrido.	El evento de distancia ocurre cuando el usuario ha recorrido una distancia mayor que el límite definido.
GPSSpeedEvent	• *SpeedLimit*: Define el límite de velocidad en metros por segundo. • *Mode*: Uno de los posibles modos *Any*, *Above*, *Equals*, y *Below*	El evento de velocidad evalúa la velocidad con que se mueve el usuario, cuando se encuentra en modo *Above* el evento ocurrirá si el usuario se mueva a una velocidad mayor del límite, *Equals* cuando la velocidad sea igual al límite, *Below* cuando sea menor, y *Any* ocurrirá sin importar la velocidad actual.
GPSTimeEvent	• *Time:* El número de segundos del contador de tiempo. • *Repeat*: Indica si el evento debe ocurrir cíclicamente. • *clearEventIndex*: Índice del evento que reiniciará el contador	El evento de tiempo funciona como un contador de tiempo, al llegar al valor límite el evento se declara como ocurrido, cuando este evento posee una cadena de eventos asociada el tiempo sumará de acuerdo a la ocurrencia de sus eventos asociados. Se puede configurar para que uno de esos eventos reinicie el contador estableciendo la propiedad *clearEventIndex.*
GPSDateTimeEvent	• *DueDate*: Hora y fecha límite del evento. • *Condition*: Condición para evaluar el evento, puede ser *After,Before,Equal* • *Mode*: Modo en que se usará el valor de la fecha limite, puede se *Full_DateTime, Time o Date*	El evento de Fecha/Hora ocurre con respecto a la fecha y hora actual y una fecha hora límite. Cuando se usa el modo *Full_DateTime* la comparación se realiza con la fecha y hora, cuando es *Time* sólo se hará por la hora y en modo *Date* se hará por la fecha. La comparación puede ser cuando sea después (*After*), antes (*Before*) o igual (*Equal*).
GPSCounterEvent	• *Times*: Número de veces.	El evento de conteo ocurre cuando se ha producido un número de veces fijo alguno de los eventos de su cadena de eventos.

a su vez para saber si ha ocurrido o no. El evento principal sería un evento de área que define el perímetro urbano, y el evento secundario sería un evento de velocidad que se dispara cuando el usuario sobrepasa los 60 Km/h. Así, cada vez que el sistema detecta que el usuario está viajando a una velocidad superior a la indicada evalúa si el usuario está dentro del perímetro urbano. Si es así, el

sistema informa que el evento configurado, que es el resultado de la combinación de los dos anteriores, ha ocurrido.

En una cadena de eventos, el evento principal puede evaluarse cuando ha ocurrido uno o varios de sus eventos asociados, según la forma en que se configure la cadena. El evento contador ha sido concebido como una herramienta para auxiliar la configuración de cadenas de eventos y lograr que estas incluyan más posibilidades.

V. REACCIONES

En la automatización de procesos de seguimiento es necesario que el sistema no sólo detecte cuándo ha ocurrido algo, sino que sea capaz de ejecutar acciones al respecto. En respuesta a un evento, la reacción por defecto es su reporte a la base de datos, que permite visualizar en el monitor el lugar y fecha en que ocurrió el evento, señalando la posición en que se disparó con un marcador que aparece sobre un mapa.

El sistema está provisto con otras reacciones que se establecen al configurar el evento, y que se producen cuando este ocurre; entre ellas se cuentan el envío de correos electrónicos, envío de SMS, un mensaje que se envía al individuo que porta el celular, la notificación a un WebService [6], o la transmisión de información de la ocurrencia del evento a una página web externa al sistema. Cada una de estas reacciones puede dirigirse a uno o varios destinatarios. Su propósito esencial es alertar sobre la ocurrencia del evento a quienes puedan estar interesados; sin embargo, pueden servir a otros propósitos, como la acumulación de estadísticas en bases de datos.

VI. APLICACIONES

La aplicación LETS tiene un evidente uso en lo relacionado con labores de seguimiento: para prestar seguridad a un sujeto, vigilar el cumplimiento de labores de un trabajador de campo, generar y evaluar estadísticas de actividades que impliquen desplazamiento, automatizar seguimiento vehicular; sin embargo, hay que tener en cuenta que para la instalación y utilización de un software como el creado es indispensable contar con el consentimiento de la persona que lleva el teléfono móvil y que es monitoreada, y esta tiene que estar en conocimiento de la aplicación y de sus alcances.

VII. TECNOLOGÍAS EMPLEADAS

Para el desarrollo de la herramienta LETS se utilizó el lenguaje de programación J2ME en el desarrollo de la aplicación del cliente móvil[7], y se utilizó el Framework 3.5 de .NET para el desarrollo de las aplicaciones del servidor y del monitor, utilizando la tecnología y herramientas de ASP.NET y el lenguaje de programación C#[8]. Para enriquecer la experiencia de usuario se utilizaron los controles de AJAX en la aplicación del monitor. Finalmente, el protocolo de comunicación elegido para la transmisión de datos entre las tres aplicaciones que conforman el software es JSON por su mayor agilidad al momento de transmitir datos[9].

VIII. CONCLUSIONES

La plataforma de diseño .NET Framework 3.5 de Microsoft probó sus capacidades para el desarrollo de librerías y de las aplicaciones de servidor y monitor, soportando un lenguaje de programación estructurada como es C#. Por su parte la versión Micro Edition de Java J2ME demostró gran funcionalidad y capacidad a la hora de desarrollar aplicaciones para dispositivos móviles, que cumplen características especiales de agilidad y ligereza dada la naturaleza de los terminales en que se ejecutan. Finalmente, JSON se probó como un formato de comunicación compacto, ágil, y por tanto adecuado para este tipo de aplicaciones.

Aunque no fue diseñado específicamente para resolver una demanda comercial, el prototipo LETS podría ampliarse para implementarse en el campo comercial como un sistema capaz de automatizar procesos de seguimiento y reacción a usuarios que portan terminales móviles. Podría utilizarse en campos variados como, por ejemplo, prestaciones de seguridad, o aumentando la eficiencia de empleados que desarrollan labores de campo.

REFERENCIAS

[1] A. El-Rabanny, Introduction to the GPS: The global positioning system. Norwood, Artech House Inc. 2006. Segunda edición. pp 2 – 3.

[2] M. Barth, Matthew y J. Farrell, Jay. El sistema de posicionamiento global GPS. New York, McGraw-Hill Companies Inc., 1999. pp. 15 – 17.

[3] M. Goodchild y P. Longley. Geographic Information

Systems and Science. West Sussex, England. John Wiley & Sons Ltd. 2005. pp. 62 – 74.

[3] Forum Nokia. Location-Based Services (En línea).

Disponible en: http://www.forum.nokia.com

[5] D. Tomlin. Geographic Information Systems and Cartographic Modelling. New Jersey, Prentice Hall, 1991. pp. 112.

[6] A. Ferrara y M. MacDonald. Programming .NET Web Services. Sebastopol, O'Reilly & Associates, Inc. 2002. pp. 37 – 39.

[7] A. Froufe Quintas y P. Jorge. J2ME Java 2 Micro Edition, Manual de Usuario y Tutorial. Editorial Ra–ma, 2003. pp. 54 – 126.

[8] M. MacDonald. Asp.NET 3.5 in C# 2008. Berkeley, Apress, 2007. pp. 432 – 627.

[9] Introducción a JSON (En línea). (En línea). Disponible en: http://www.json.org/json-es.html