# Diseño e implementación de un servicio de localización y visualización de mapas utilizando J2ME para dispositivos móviles y herramientas de libre distribución

*Renzo C. Bertuzzi Leonelli*

Universidad Austral de Chile, Valdivia, Chile. Ingeniero Civil en Informática
e-mail: rbertuzzi@inf.uach.cl

*Juan P. Salazar Fernández*

Universidad Austral de Chile, Valdivia, Chile. Ingeniero Civil en Informática
e-mail: juansalazar@uach.cl

El siguiente documento expone un proyecto de tesis que propone el diseño e implementación de un servicio que permita la localización y visualización de mapas. El software construido tiene como plataforma de ejecución los teléfonos celulares, para lo cual se implementó utilizando J2ME. La aplicación entrega al usuario el medio para conocer su ubicación, desplazarse por el mapa y conocer la ubicación de puntos de interés (restoranes, hospitales, hoteles, etc.). Esta aplicación interactúa con un servidor de mapas y una base de datos geográfica. Para crear el servicio se utilizaron herramientas de libre distribución tanto para la base de datos geográfica como para el servidor de mapas.

*Palabras clave:* GIS, SIG, J2ME, java, mapas, localización, móviles.

*The present paper exposes a thesis project that proposes the design and implementation of a service that allows localization and visualization of maps. The software developed has mobile phones as deployment platform, so it was implemented with J2ME. The application provides to user a means of knowing his position, moving through the map and knowing the localization of special sites (restaurants, hospitals, hotels, etc.). The application communicates with a map server and a geographic data base. To build this service open source tools were used as for the geographic database and for the map server.*

*Key words:* GIS, SIG, J2ME, java, maps, localization, mobile phones, l

## 1. INTRODUCCIÓN

Con las nuevas tecnologías y dispositivos móviles cada vez con mayor capacidad de procesamiento, almacenamiento y despliegue de gráficos es un hecho que se puede ampliar en gran medida la funcionalidad de los teléfonos celulares sin necesidad de realizar cambios a nivel de hardware, utilizando las herramientas de software adecuadas y teniendo en cuenta las capacidades y limitaciones de los equipos actuales, por ejemplo: SMS, WAP, Bluetooth, etc.

Bluetooth [9] es una tecnología orientada a redes inalámbricas de poco alcance que utiliza

bajos consumos eléctricos por lo que es muy adecuada para los dispositivos móviles pequeños. La mayor parte de los teléfonos celulares actuales soportan Bluetooth, de esta manera esta tecnología se ve como una forma viable de transferir datos, por ejemplo: información de mapas hacia un teléfono móvil cuando el usuario se encuentre en el rango de una red de este tipo.

Otra forma de alimentar con información un dispositivo móvil es a través de WAP, un protocolo parecido al HTTP pero diseñado para dispositivos móviles. Una ventaja de WAP es que el usuario tendrá acceso a éste mientras tenga cobertura en su móvil. Opcionalmente a WAP, los dispositivos también cuentan con la capacidad de realizar conexiones HTTP directas; de esta forma, es posible acceder, por ejemplo a un webservice para obtener algún tipo de información.

La plataforma J2ME (Java 2 Micro Edition) [1] proporciona un entorno de desarrollo para crear aplicaciones destinadas a familias de dispositivos móviles, contando con las librerías de clases necesarias para construir interfaces gráficas, almacenamiento persistente, manejo de eventos, conectividad de red y juegos.

Los sistemas de información geográfica permiten representar la realidad y almacenar esta información para su análisis y edición. Los sistemas GIS permiten, en general, describir aspectos de la geografía en cuanto a posición, altitud, describir accidentes geográficos, agrupar información temática o temporal. Para guardar toda esta información necesitan de un repositorio de datos que por lo general son bases de datos relacionales que utilizan algún formato de representación espacial específico [2], como mapas vectoriales, rasters, mixtos, etc.

Con base en las tecnologías ya nombradas, y adaptando un formato de representación espacial para el proyecto, se creó una aplicación que permite al usuario saber su ubicación actual dentro de un mapa, conocer puntos de interés en el lugar donde está y navegar por el mapa. Para ello se utilizó servicio de mensajería SMS, bluetooth y consultas a un webservice vía HTTP.

## 2. PLATAFORMA J2ME

### 2.1. *Arquitectura* J2ME

La plataforma de desarrollo para dispositivos móviles J2ME posee 3 elementos principales que permiten la ejecución de aplicaciones en dispositivos con recursos limitados, como se especifican a continuación:

1) Máquina Virtual: al igual que las demás plataformas de java, las aplicaciones J2ME son compiladas a bytecode y luego son interpretadas por una máquina virtual. Existen dos máquinas virtuales, la KVM y la CVM. La KVM es la más pequeña de todas y requiere de 40kb a 80kb de memoria.
2) Configuración: es un conjunto de clases básicas para orientadas a una misma familia de dispositivos. Existen dos configuraciones, CLDC y CDC. La primera es la configuración utilizada para aplicaciones sobre teléfonos móviles.
3) Perfil: es un conjunto de clases que proveen funcionalidades más específicas para distintos dispositivos de una misma familia. Los teléfonos móviles emplean el perfil MIDP.

### 2.2. MIDlets

Los MIDlets son aplicaciones construidas usando el perfil MIDP. Un MIDlet pasa por 4 estados de ejecución (cargado, activo, pausa, destruido) que son controlados por un gestor de aplicaciones llamado AMS.

Una interfaz gráfica de un MIDlet se estructura en base a pantallas, donde cada pantalla contiene tareas posibles de ejecutar. Se accede a las funcionalidades de la aplicación navegando a través de las distintas pantallas, ya que en un determinado momento sólo puede mostrarse una única pantalla en el LCD del dispositivo. Se pueden crear dos tipos de interfaces:

1) *Interfaz de Alto Nivel*: este tipo de interfaz es dependiente del dispositivo. No se tiene control sobre la apariencia de los objetos en pantalla, pero tiene la ventaja de ser muy portable.
2) *Interfaz de Bajo Nivel*: es menos portable y es la aplicación la encargada de dibujar los objetos en pantalla. Proporciona acceso a eventos de bajo nivel como la pulsación de teclas.

## 2.3. *Almacenamiento Persistente*

El perfil MIDP cuenta con un sistema de administración de registros (RMS) para almacenar datos de forma persistente; es decir, permite guardar información que se conservará en la memoria física del dispositivo después que la aplicación finalice su ejecución, y estarán disponibles cuando el MIDlet sea ejecutado nuevamente.

El RMS se estructura en base a RecordStores equivalentes a una tabla de un RDBMS. Cada RecordStore es una colección ordenada de registros de dos campos, un identificador asignado por el RMS y un campo binario de longitud variable.

## 2.4. *Conectividad*

La conectividad es un aspecto importante dentro de J2ME, por ello el perfil MIDP cuenta con acceso a variados tipos de conexiones que incluyen sockets, HTTP, SMS, Bluetooth, entre otros. El conjunto de clases que permite realizar conexiones se denomina Generic Connection Framework (GCF).

Para abrir cualquier tipo de conexión se usa la instrucción *Connector.open* (url), y el GCF retornará un objeto del tipo adecuado para manejar la conexión.

Las conexiones que se usaron en el proyecto son:

### 2.4.1. *Conexiones HTTP*

Son creadas usando una url de conexión de tipo *http://servidor:puerto*. El objeto devuelto por la llamada a *Connector.open* tiene métodos para efectuar peticiones y obtener información de la conexión.

### 2.4.2. *Conexiones Bluetooth*

Son creadas usando una url de conexión de tipo *btssp://direccion:canal*, donde *direccion* es la dirección del dispositivo remoto y *canal* es el canal en el cual se ofrece algún servicio.

### 2.4.3. *Conexiones SMS*

Se pueden enviar y recibir SMSs Son creadas usando una url de conexión de tipo *sms://telefono:puerto*.

## 3. SISTEMAS GIS

Un Sistema de Información Geográfico o GIS, puede ser definido como "un sistema de información basado en computadores que intenta capturar, almacenar, manipular, analizar y visualizar datos tabulares asociados y referenciados espacialmente para resolver problemas complejos de administración, planeamiento e investigación" [10].Un GIS tiene cuatro elementos principales:

1.*Hardware:* es el equipamiento electrónico sobre el que opera el GIS. En los inicios de los GIS el componente hardware condicionaba en gran medida la construcción de estos sistemas de información debido al poder de procesamiento que requieren. Actualmente, el hardware ya no es una limitación.

2.*Software:* son todos los programas necesarios para manejar el hardware y manipular la información. El software GIS cuenta con las funcionalidades para capturar, almacenar, manipular, analizar y visualizar información referenciada geográficamente.

3.*Datos Espaciales:* es la información geográfica que compone el GIS. Es el componente más importante y a la vez son los que en la actualidad consumen la mayor parte de las inversiones en términos económicos y de tiempo. La información georreferenciada de un GIS proviene de dos fuentes principales: datos espaciales y datos temáticos. Los datos espaciales son la información proveniente de la geografía del terreno, y los datos temáticos corresponden a la información de otras fuentes como son datos de clientes, población, etc. que están relacionados a los datos espaciales.

4.*Personal:* son las personas encargadas de desarrollar, administrar y operar el sistema. Se identifican los roles de director de proyecto, desarrolladores, DBA, técnicos y usuarios.

### 3.1. *Representación de Datos Espaciales*

Los datos que almacena la base de datos geográfica de un GIS deben modelar el mundo real y a la vez estar representados de una forma entendible para las máquinas. Para simplificar la información del mundo real ésta se divide en capas, donde cada capa representa un tema de interés para el GIS, por ejemplo, carreteras, viviendas, hidrografía, etc.

Según la forma en que se almacenan los datos geográficos y las relaciones o topología entre ellos se tienen tres grupos principales de representaciones [3]: Vectoriales, Rasters y Orientadas a Objetos. Cada tipo de representación tiene utilidades específicas que dependen de cada sistema GIS en particular, siendo los vectoriales y rasters los más usados en la actualidad.

#### 3.1.1. *Representación Vectorial*

Utiliza pares de coordenadas para formar vectores con los que se modelan los objetos geográficos esencialmente en base a su forma geométrica.

La topología arco-nodo que usan los sistemas vectoriales estructura la información geográfica en base a pares de coordenadas para formar puntos o nodos, y uniendo puntos forma líneas o arcos, con los que a su vez se forman polígonos o áreas.

Esta información se puede almacenar en una base de datos relacional donde se utilizan tablas separadas para guardar los puntos, líneas y polígonos.

#### 3.1.2. *Representación Raster*

Las representaciones rasters toman una muestra del espacio geográfico y la dividen en pequeñas celdas o pixeles ordenados en una matriz regular, y a cada pixel se le asigna un valor numérico que indica su grado según el tema que representa, altura, densidad, humedad, etc. Cada

pixel de un modelo raster tiene un tamaño constante, y su posición en relación al mundo real es conocida a través de la fila y columna que ocupa en la matriz.

### 3.1.3. *Representación Orientada a Objeto*

Mientras los modelos vectoriales y rasters usan capas para representar los distintos niveles de información geográfica, el modelo orientado a objetos plantea almacenar la información geográfica en el propio objeto, y modelar la topología en base a relaciones entre objetos. De esta forma cada objeto tendrá sus propiedades y comportamiento particulares los cuales pueden transmitir a objetos descendientes gracias al esquema de herencia de la orientación a objeto.

Una gran ventaja de este modelo es que permite dinamismo en los objetos al incorporar comportamientos, dando la posibilidad de cambio a los objetos según el espacio, tiempo y las relaciones con los demás objetos.

### 3.1.4. *Lenguaje de Consulta Espacial*

Un sistema de información geográfico debe proveer los medios para acceder a la información espacial almacenada, lo cual se efectúa a través de un lenguaje de consulta espacial o SSQL el cual es una extensión de SQL. Se optó por extender el lenguaje SQL y no crear un lenguaje nuevo por la gran estandarización y aceptación que tiene SQL, además que las bases de datos de los sistemas GIS contienen datos geográficos como no geográficos.

Un lenguaje SSQL permite crear instrucciones del tipo “SELECT-FROM-WHERE” para obtener datos de tres tipos [11]:

1. *Consultas exclusivas de propiedades espaciales*. Por ejemplo: “Obtener todas las ciudades cruzadas por un río”.
2. *Consultas no espaciales*. Por ejemplo: “Cuál es la población de la ciudad X”.
3. *Consultas combinadas*. “Obtener todos los vecinos del sector X”.

Además de permitir estos tres tipos de consultas, el lenguaje SSQL debe ser capaz de devolver resultados de forma distinta a la conocida forma tabular, ya que muchas consultas devolverán datos espaciales que deben ser presentados en forma gráfica.

## 4. SISTEMAS DE POSICIONAMIENTO

Los sistemas de posicionamiento se refieren al método por el cual se puede determinar la posición de un dispositivo, y utilizando servicios adicionales se puede responder preguntas como ¿Dónde estoy? ¿Qué hay cerca? ¿Cómo llego allá?

La ubicación puede ser expresada como una descripción textual o en términos espaciales [12]. Una ubicación expresada como un texto es por lo general la dirección de una calle, la ciudad, comuna, código postal, etc. Una ubicación espacial puede ser expresada en términos de coordenadas geográficas usando latitud-longitud-altitud (esta última opcional). La latitud se expresa en grados de 0-90 al norte o sur del ecuador, la longitud en grados de 0-180 al este u oeste del meridiano de Greenwich, y la altitud en metros sobre el nivel del mar.

Existen varios métodos [13] para determinar la ubicación de un dispositivo, los cuales varían en las tecnologías empleadas y en la precisión con que entregan el resultado. Dentro de los

tipos de tecnologías empleadas están las redes telefónicas, celulares, de área local, Wi-Fi , Bluetooth y GPS entre otros.

### 4.1. *Método* Cell-ID

Es un método de bajo costo que utiliza la red GSM para determinar la ubicación de un teléfono celular. El identificador de la antena o célula (Cell ID) a la que está conectado el dispositivo es usado para aproximar la ubicación del usuario. La precisión de este método depende del tamaño de la célula: unos 500 metros de exactitud dentro de una ciudad y unos 10 kilómetros en zonas rurales. El método mejorado puede obtener precisiones de 150 a 500 metros.

### 4.2. *Método Bluetooth*

Dado que bluetooth es una red de corto alcance este método provee posicionamiento de buena precisión, con máximo de error de unos 10 metros, además de poder entregar posicionamiento vertical. Es muy adecuado para obtener la ubicación dentro de edificios o zonas urbanas pequeñas.

## 5. Estándar GML Y CGML

GML es un lenguaje de marcas construido en XML para el modelado, transporte y almacenamiento de información geográfica [14].

GML proporciona una gran variedad de objetos para describir información geográfica como entidades (features), sistemas de coordenadas de referencia, geometrías, topología, tiempo, unidades de medida y valores generales.

### 5.1. *Estructura de un documento* GML

Un documento GML es un conjunto de entidades geográficas denominadas features. Una entidad geográfica o feature es “una abstracción de un fenómeno del mundo real; es una entidad geográfica si está asociada a una posición relativa en la Tierra” [14].

A este conjunto de entidades se le denomina FeatureCollection y constituye el elemento raíz del documento. Dentro de este elemento están contenidas todas las entidades que contiene el documento GML y además existe un elemento adicional, gml:Box, que es un rectángulo que enmarca a todas las entidades presentes en el documento.

Una feature está compuesta por propiedades y una geometría como se ve en la siguiente figura:

Figura 5.1.1.
Feature GML

```
<gml:featureMember>
  <sql_statement fid="265">
    <name>Valdivia</name>
     <ogr:geometryProperty>
       <gml:Point>
<gml:coordinates>-70.66,-30.74
</gml:coordinates>
       </gml:Point>
     </ogr:geometryProperty>
  </sql_statement>
</gml:featureMember>
```

## 5.2. *Tipos de entidades* GML

Algunos tipos de objetos geométricos de GML son:

1) *Point:* corresponde a un punto adimensional sobre la tierra. Está compuesto por un par de coordenadas.
2) *LineString:* conjunto de pares de coordenadas que definen una línea compuesta por uno o más segmentos.
3) *Polygon:* área con un borde exterior y opcionalmente un borde interior.
4) *MultiGeometry:* conjunto de una o más entidades.

## 5.3. cGML

El cGML es una especificación XML orientada a proveer de mapas digitales a dispositivos inalámbricos pequeños [4].

cGML está basado en GML 2.1.2. cGML logra disminuir el tamaño de los documentos GML alrededor de un 64% [15] al reemplazar los nombres de los elementos en GML por nombres más cortos, remover los elementos vacíos que sólo se usan para contener otros elementos y reducir el número de atributos disponibles. Otro factor que influye en la disminución del tamaño es la conversión de las coordenadas en punto flotante con un promedio de unos 9 dígitos a números enteros de dos o tres dígitos.

Aunque los nombres de los elementos son más cortos éstos aún preservan el sentido de lo que representan por lo que son fácilmente legibles por una persona.

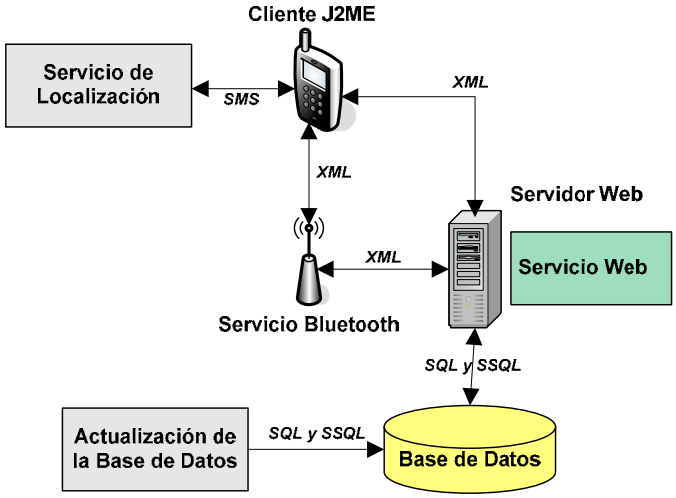
# 6. DISEÑO Y DESARROLLO DEL PROTOTIPO

En este capítulo se presentará el diseño de arquitectura e implementación de los distintos componentes necesarios para crear el proyecto propuesto.

## 6.1. *Arquitectura*

Para implementar el prototipo se hace necesario dividir el problema en varios módulos que interactúen entre sí, desde la extracción de datos desde la base de datos geográfica hasta la visualización gráfica de los datos en la pantalla del móvil.

Figura 6.1.1.
Arquitectura




El modelo tiene como ente iniciador de actividad al Cliente J2ME, el cual es el que gatilla acciones en los demás módulos. Desde el cliente se puede consultar la posición o proceder a descargar un mapa.

Si el cliente decide obtener su ubicación puede hacerlo usando el servicio bluetooth o bien a través de mensajes SMS. Para ello envía una petición hacia el Servicio de Localización en forma de SMS o bien se conecta al Servicio Bluetooth y luego espera por una respuesta, SMS o un flujo de bytes respectivamente.

Para la descarga de mapas el cliente envía una petición de descarga de mapas directamente al Servidor Web o al Servicio Bluetooth. En el segundo caso el Servicio Bluetooth sirve de puente entre el cliente y el Servidor Web. Luego el Servidor Web procesa la petición para lo cual debe conectarse a la Base de Datos para extraer la información necesaria, trasformarla en un documento XML y devolverla al cliente.

La Base de Datos se alimenta de información desde el módulo de Actualización que es el que se encarga de agregar nuevos mapas y actualizar los existentes.

## 6.2. *Base de Datos*

Es el elemento más importante de la arquitectura, ya que maneja toda la información del sistema.

Por el alcance del proyecto no se proporcionó la implementación del módulo de actualización de base de datos.

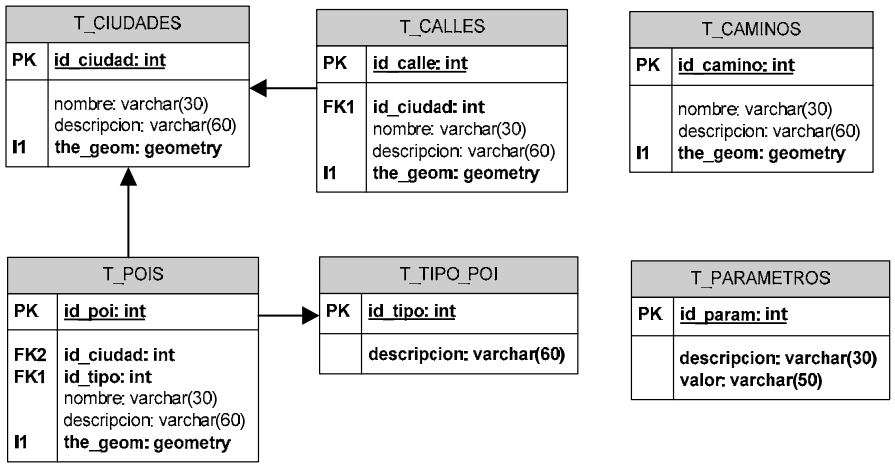
Como motor de base de datos se usó PostgreSQL 8.0 con la extensión PostGIS para permitir el almacenamiento y manejo de información geográfica.

Dado que la aplicación permite ver mapas de ciudades, donde se pueden observar lugares de interés tales como hospitales, restaurantes, cines, etc, denominados POI's[1] por su sigla en inglés, las calles de las ciudades y caminos entre ciudades, cada uno de estos elementos se dejó como una capa del mapa final. Las tablas necesarias para representar estos elementos son:

Figura 6.2.1.
Tablas

| *Nombre* | *Tipo Geométrico* | *Descripción* |
|---|---|---|
| T_CIUDADES | MULTIPOLYGON | Áreas de las ciudades |
| T_CALLES | MULTILINESTRING | Líneas que componen las calles de la ciudad |
| T_POIS | POINT | Puntos de Interés de una ciudad |
| T_CAMINOS | MULTILINESTRING | Caminos |
| T_TIPO_POI | | Tipos de puntos de interés |
| T_PARAMETROS | | Parámetros de sistema |

Figura 6.2.2.
Modelo de Datos




### 6.2.1. *Poblado de la Base de Datos*

Para realizar el poblado de datos se utilizó el cargador shp2pgsql de PostGIS. Este cargador toma un archivo geográfico en formato ShapeFile y genera un archivo sql con instrucciones para la inserción de los datos.

Para generar el archivo ShapeFile se siguieron los siguientes pasos:

1) *Creación de Mapa Vectorial*: utilizando el software fGIS se importó una foto de un plano de Valdivia. Sobre esta imagen se crearon 4 capas (contorno, POIs, calles, caminos). Se trazaron los objetos sobre las capas teniendo como guía la imagen. Se exportó cada capa a un archivo MIF.

[1] POI: Point Of Interest: Punto de Interés.

2) *Corrección de Coordenadas*: las coordenadas de los objetos quedaron relativas al tamaño de la imagen por lo que fue necesario rectificarlas. Para ello se creó una pequeña aplicación en Visual Basic 6.0 que tenía como entrada un archivo MIF y un rango de coordenadas. El rango de coordenadas usado fueron las coordenadas reales de Valdivia.
3) *Conversión a formato SHP:* Cada archivo MIF luego fue convertido a formato ShapeFile utilizando nuevamente fGIS.
4) *Carga en BD:* Usando el cargador sho2pgsql cada archivo SHP fue convertido en instrucciones sql que fueron ejecutadas contra la BD.

### 6.2.2. *Formato de consulta espacial*

Para obtener un rendimiento óptimo se crearon índices en los campos con datos geográficos de las tablas. Cabe señalar que los índices en campos espaciales solo son usados por el planificador de consultas cuando se emplean los operadores && (intersección), ~= (igual) o = (rectángulo exterior igual).
Dado que la consulta más frecuente en la base de datos será "obtener todos los objetos que estén en un radio de r metros del punto (x,y)", para aprovechar los índices se construyeron las consultas con el operador && de la siguiente forma:

1) Obtener todos los objetos que estén sobre un cuadrado de lado 2*r centrado en (x,y). Esta operación usa && por lo que utiliza el índice en la tabla.
2) De los resultados anteriores seleccionar aquellos cuya distancia al punto (x,y) sea menor o igual a r.
   La instrucción SSQL equivalente se observa en la siguiente figura:

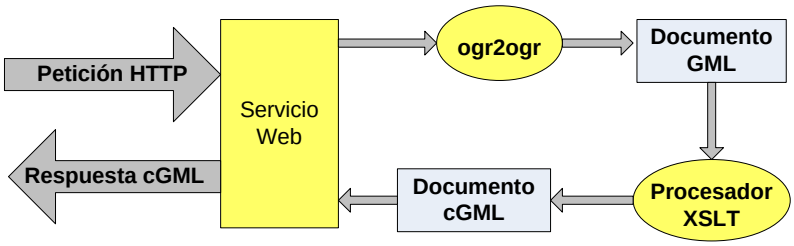
Figura 6.2.2.1.
Consulta SSQL

```
select
      a.nombre, a.descripcion,
      b.descripcion as tipo,
      a.the_geom
from
      T_POIS a join T_TIPO_POI b
      using(id_tipo)
where
      a.the_geom &&
      'BOX3D(x-r y-r, x+r y+r)'::box3d
      and distance(a.the_geom,
      GeomFromText('POINT(x y)', -1)) < r
```

## 6.3. *Servidor Web*

Este módulo es un intermediario entre el Cliente J2ME y la base de datos. Dadas las grandes restricciones de procesamiento y memoria de los dispositivos móviles, el módulo web se encarga de recibir las peticiones del cliente móvil, conectarse a la base de datos y procesar la información con el propósito de evitar que el cliente tenga que realizar cálculos o conexiones que necesitan muchos recursos.

Figura 6.3.1.
Módulo Servidor Web




Se utilizó Apache 2.0.54 con PHP 4.3.1 como servidor web, y para el acceso a la base de datos y para las consultas espaciales se empleó la utilidad ogr2ogr [5] que tiene integrado el driver de conexión a PostgreSQL.

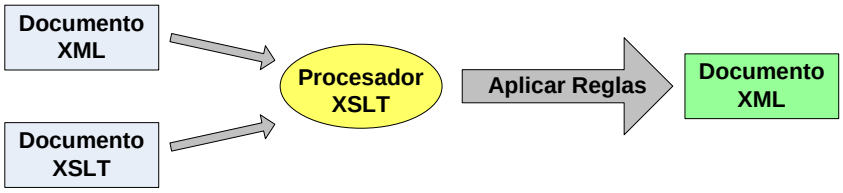
Para dejar la aplicación J2ME simple y pequeña el servicio web es sólo una pagina web que recibe parámetros por GET.

El servicio web está encargado de recibir las peticiones del cliente, convertir estas peticiones en consultas SSQL y pasárselas al componente ogr2ogr que retorna los resultados como un documento GML. Luego el servicio web invoca al componente de generación de cGML para transformar el documento GML en cGML, el cual es enviado al cliente.

El componente encargado de generar cGML a partir del documento GML se implementó como una plantilla de transformación XSLT.

Como procesador XSLT se utilizó la extensión Sablotron [6] para PHP, incluida desde la versión 4 de PHP.

Figura 6.3.2.
Proceso XSLT



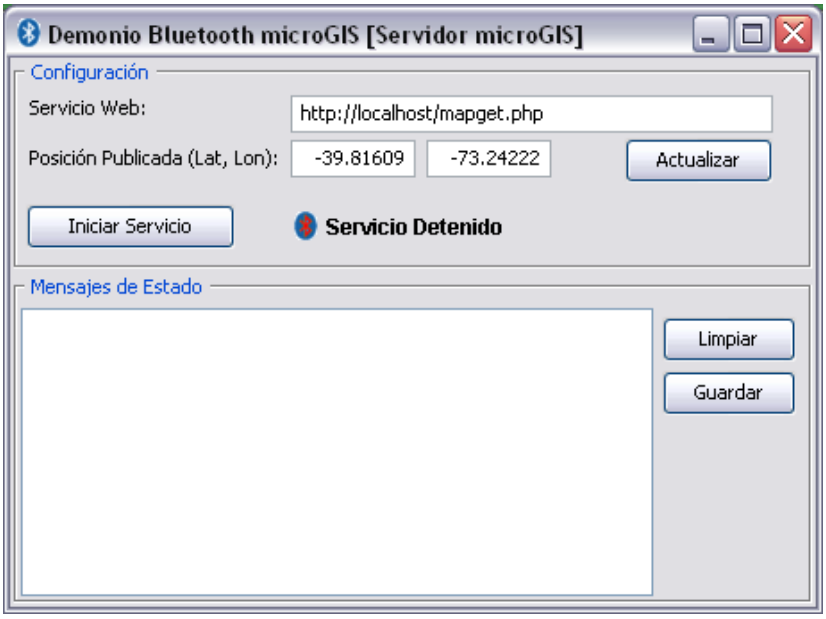

## 6.4. *Servicio Bluetooth*

Se creó un servicio bluetooth que está esperando conexiones de clientes para la descarga de mapas. Cuando un cliente se conecta al servicio bluetooth le envía una petición para descargar un mapa, luego el servicio bluetooth se conecta al servidor web para obtener el documento cGML y retornar este documento al cliente. De esta forma el servicio bluetooth funciona como un puente entre el cliente móvil y el servidor web.

Este módulo fue implementado con Java J2SDK, utilizando la librería BlueCove versión 1.1.1 [7]. BlueCove es una implementación de código abierto de la API Bluetooth JSR-82 para aplicaciones java J2SE. Aunque BlueCove no implementa íntegramente la API JSR-82, permite de manera simple migrar un código escrito para J2ME a una aplicación J2SE.

Para poder acceder al dispositivo bluetooth a través de J2SE BlueCove cuenta con una DLL intebth.dll, la cual es invocada usando JNI .
Primero se creó un servidor bluetooth utilizando el emulador de J2ME WT se Sun que cuenta con soporte para conexiones bluetooth simuladas. Para ello se creó un MIDlet que recibía las peticiones del cliente empleando la API JSR-82. Una vez depurado el servidor se creó la aplicación final en J2SE con BlueCove para que se ejecute en un PC. La interfaz gráfica de este módulo se muestra en la figura a continuación:

Figura 6.4.1.
Servicio Bluetooth




Para iniciar el servidor se debe especificar la dirección del servidor de mapas en el campo *Servicio Web*, y la posición geográfica que retornará el servicio en el campo *Posición Publicada*. Para iniciar el servicio se presiona el botón *Iniciar Servicio*. Al lado de este botón hay un icono y un mensaje de estado. Además en la sección *Mensajes de Estado* se guarda un registro de la actividad del servicio (inicio, detención, errores, conexiones, etc.).

## 6.4. *Servicio de Localización*

El servicio de localización se implementó en dos formas, una como un centro receptor de mensajes de tex

El centro de mensajes emplea el método de Cell ID para resolver la posición del cliente. Recibe las peticiones de los clientes vía SMS, usando el número telefónico que recibe en el mensaje consulta a qué antena celular está conectado ese número, y luego retorna las coordenadas de posición (longitud y latitud) de la antena en un mensaje SMS de vuelta al cliente.

La información respecto a qué antena está conectado un teléfono móvil es manejada por la empresa de telefonía. Es por esta razón que, para efectos del proyecto, el centro receptor de mensajes, una vez recibida la petición del cliente, simula la consulta para conocer la antena a la cual está conectado el móvil, y retorna un mensaje SMS con coordenadas de posición ficticias.

## 6.5. *Cliente* J2ME

Este módulo, que corresponde a una aplicación MIDlet, es la cara visible del proyecto, de aquí se generan las peticiones de los usuarios para descargar un mapa o conocer su posición, peticiones que ponen en marcha los demás módulos.
Dentro de las opciones disponibles para el usuario en el MIDlet están:

a) *Configurar:* aquí se puede configurar parámetros de la aplicación.
b) *Obtener Posición*: el usuario puede obtener su posición a través de 3 opciones: bluetooth, SMS o ingresarla manualmente.
c) *Descargar Mapa*: hay dos opciones para descargar mapas: usando HTTP y usando bluetooth.
d) *Eliminar Mapa:* permite eliminar un mapa de la memoria local del teléfono.
e) *Ver Mapa:* el usuario puede ver un mapa nuevo o bien ver mapas guardados localmente.
f) *Modo Mover:* con esta opción el usuario puede mover el mapa hacia arriba, abajo, izquierda o derecha. Esta función es útil para desplazar el mapa cuando el mapa completo no es visible en la pantalla.
g) *Modo Cursor:* con esta opción el usuario dispone de un cursor que puede posicionar en cualquier lugar del mapa.
h) *Capas:* en esta sección del menú se pueden mostrar u ocultar los textos e iconos de las capas y mostrar u ocultar las propias capas. Además se pueden ordenar las capas para definir la posición de las capas en el mapa.
i) *Buscar:* con esta opción el usuario ingresa un texto de búsqueda y se resaltan en el mapa todos los objetos que coincidan con el criterio usado.
j) *Guardar:* permite que el usuario guarde en la memoria local del dispositivo un mapa descargado desde la red.
k) *Medir:* esta opción permite que el usuario mida la distancia entre dos puntos del mapa.
l) *Ruta:* permite obtener la ruta óptima entre dos puntos y resaltar esta ruta en el mapa.
m) *Ver Info:* permite consultar información detallada de un objeto del mapa.
n) *Ayuda:* muestra la pantalla de ayuda.

### 6.5.1. *Herramienta de Desarrollo*

Para implementar la aplicación MIDlet se empleó la suite J2ME Wireless Toolkit v2.2, que está disponible para descarga gratuita desde el sitio de Sun.

Se empleó esta suite porque cuenta con una serie de características que hacen más fácil el desarrollo de aplicaciones J2ME. Entre otros permite:

a) Automatizar los pasos necesarios para crear un MIDlet. Automáticamente compila, preverifica y empaqueta.
b) Se pueden seleccionar las características del dispositivo destino, por ejemplo la configuración CLDC, perfil MIDP, tamaño de memoria, APIs disponibles, etc.
c) Dispone de emuladores para probar las aplicaciones.
d) Tiene utilidades para simular envío de mensajes SMS.

### 6.5.2. *Conectividad*

El componente de conectividad es invocado cuando el usuario decide descargar un mapa o consultar su posición.

En el caso de descargar un mapa el componente de conexión recibe los argumentos de la petición (dirección del servidor, posición, tamaño de pantalla, radio de cobertura), se conecta al servidor y devuelve un flujo de bytes con la respuesta entregada por el servidor. Esta conexión puede ser a través de bluetooth o directamente al servidor web a través de HTTP.
En el caso de consultas de posición envía una petición al servidor bluetooth o centro de mensajes, recibe una respuesta y luego devuelve las coordenadas de posición.

Hay que hacer notar que todas las conexiones son realizadas por threads o hilos separados del hilo de ejecución principal del MIDlet, puesto que éstas pueden ser lentas y dar la impresión que la aplicación está colgada.

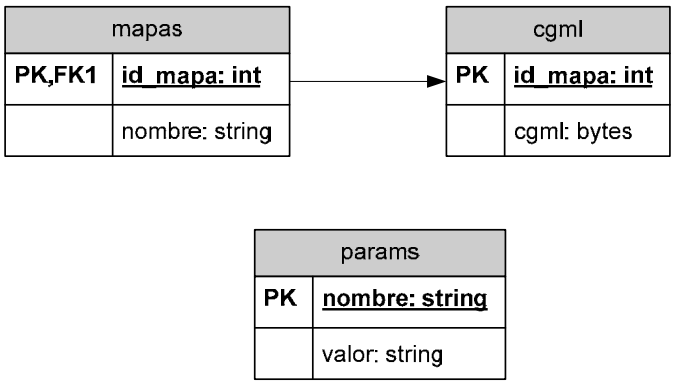
### 6.5.3. *Almacenamiento Local*

La aplicación cuenta con parámetros configurables y además con la opción de guardar los mapas descargados.

Para almacenar los parámetros de la aplicación se creó un RecordStore llamado params donde en cada registro se guarda un par parámetro:valor. Para almacenar los mapas se crearon dos RecordStore, mapas donde se guarda el nombre de los mapas y cgml donde se almacena el documento cGML que conforma el mapa en formato cGML comprimido.

Una vista como un diagrama relacional de las tablas empleadas en la memoria local se presenta en la figura:

Figura 6.6.2.1.
Almacenamiento Local




Por un tema de rendimiento se separó el documento cgml del nombre del mapa, ya que los registros en la memoria son sólo una serie de bytes, por lo que si se quisiera guardar el nombre y el documento cgml en la misma tabla estos dos datos quedaría en el mismo campo, por lo que para leer el nombre de un mapa sería necesario leer todo el registro, incluido el documento cgml, lo que haría lento el despliegue de la lista de mapas.

Para optimizar el espacio requerido para guardar los mapas y a la vez no requerir de código adicional para esta tarea se aplicó una compresión al documento cGML con lo que se consiguió disminuir en alrededor de un 35% el tamaño.

En pruebas realizadas se obtuvo un promedio de 37% y 89% de reducción de tamaño al usar cGML comprimido en comparación a cGML y GML respectivamente.

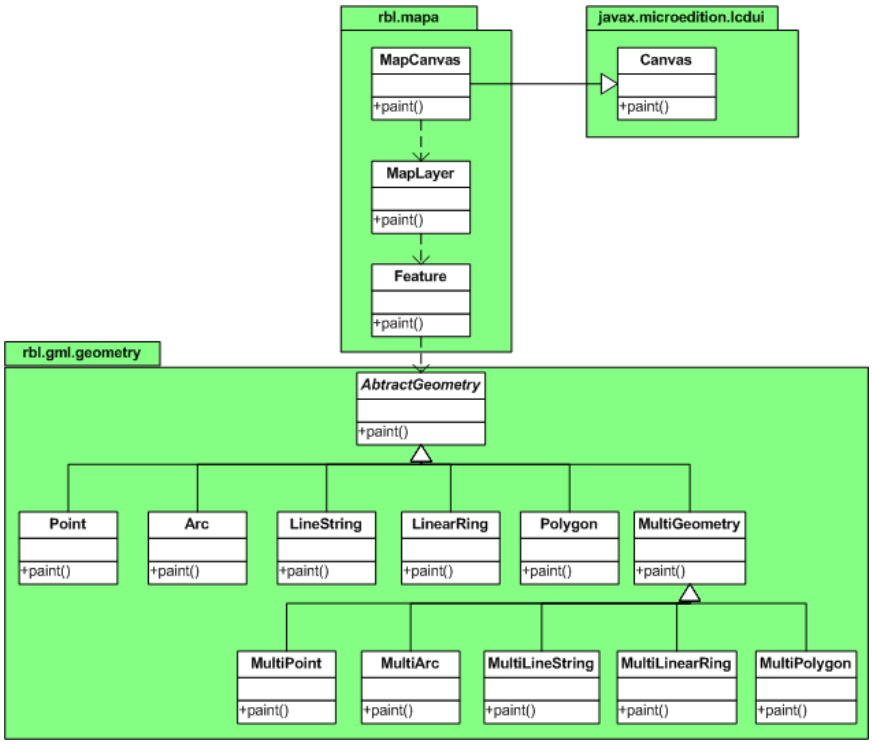
### 6.5.4. *Despliegue de Mapas*

Para mostrar mapas en la pantalla del teléfono móvil es necesario leer el documento cGML o cGML comprimido e interpretar estos datos para producir los objetos que finalmente ve el usuario.

Para la lectura de los documentos cGML y cGML comprimido se empleó el parser de XML kXML [8] el que posee la capacidad de procesar documentos XML normales y binarios, por lo que el uso de este parser permite trabajar con los documentos cGML recibidos desde el servidor y con los documentos cGML comprimidos guardados en la memoria persistente del dispositivo.

Para modelar un mapa y permitir su despliegue se creó una jerarquía de clases donde cada objeto geográfico de cGML tiene un equivalente objeto java.

Figura 6.6.4.1.
Diagrama de Clases de un Mapa




### 6.5.5. *Ruta Óptima*

Para resolver este problema se empleó el algoritmo más eficaz y eficiente conocido, el algoritmo de Dijkstra.

Para implementar el algoritmo de Dijkstra para encontrar la ruta óptima entre dos puntos del mapa es necesario construir primero un grafo dirigido donde los nodos son las intersecciones de calles, las aristas son las calles y el peso de las aristas es la longitud de las calles. Luego se aplica el algoritmo de Dijkstra al grafo generado.

Sólo es necesario construir el grafo de un mapa cuando se ejecuta por primera vez la obtención de ruta óptima del mapa. Para posteriores consultas de ruta óptima el grafo ya está

generado y es directamente utilizado por el algoritmo, por lo que el tiempo de ejecución de las sucesivas peticiones de ruta óptima es considerablemente menor.

Para construir el grafo se utilizan todos los objetos LineString del mapa, dado que éstos representan calles y caminos, además la clase creada LineString cuenta con la propiedad sentido, que permite construir un grafo dirigido tomando en cuenta el sentido de tráfico en cada calle.

#### 6.6.6. *Publicación del* MIDlet

Para construir el MIDlet y dejarlo listo para descarga e instalación en un teléfono se siguieron los pasos:

1) *Compilación:* en el J2ME Huirles Toolkit se definió como perfil destino el perfil MIDP 2.0 con soporte para las APIs WMA 1.1 (JSR 120 para envío y recepción de SMSs) y Bluetooth (JSR 082).
2) *Permisos:* se definieron los permisos necesarios para el uso de bluetooth, http y sms.
3) *Empaquetado y Ofuscación:* todos los archivos que componen el MIDlet fueron empaquetados en un único archivo JAR, el cual además fue ofuscado, operación que permite disminuir el tamaño del archivo JAR final y proteje los archivos class para dificultar su decompilación.
4) *Firma digital:* se generó un certificado de prueba que permite que el MIDlet haga uso de los permisos nombrados sin la confirmación del usuario.

## 7. PRUEBAS DEL PROTOTIPO

Se creó un ambiente de pruebas que permitió validar el prototipo en emuladores y en un teléfono móvil.

### 7.1. *Pruebas de Servidor*

#### 7.1.1. *Confiabilidad*

Estas pruebas son básicamente para testear que el servidor de base de datos, el servidor web y el servicio web de generación de mapas están funcionando correctamente.

Se creo una página web que invoca al servicio web y muestra el documento cGML generado, y se comparó el contenido de este documento con los resultados que entrega una consulta equivalente directamente en la base de datos y con la salida de la utilidad ogr2ogr bajo la misma consulta. De esta forma se puede comprobar que el documento cGML está correctamente generado, tiene el número de entidades esperado y la conversión y escalamiento de coordenadas se realiza correctamente.

#### 7.1.2. *Rendimiento*

Se midió el rendimiento del servicio web en cuanto a tiempo de respuesta para la generación de mapas cGML.

Figura 7.1.2.1.
Rendimiento del Servidor

| Radio(km) | N Entidades | Tiempo | S | S/T |
|---|---|---|---|---|
| 0.1 | 11 | 1.96 | 0.03 | 0.02 |
| 0.2 | 31 | 2.05 | 0.03 | 0.01 |
| 0.3 | 52 | 2.15 | 0.07 | 0.03 |
| 0.4 | 72 | 2.20 | 0.11 | 0.05 |
| 0.8 | 124 | 2.34 | 0.03 | 0.01 |
| 1.0 | 160 | 2.48 | 0.02 | 0.01 |
| 1.2 | 183 | 2.61 | 0.06 | 0.02 |
| 1.5 | 234 | 2.73 | 0.05 | 0.02 |
| 2.0 | 277 | 2.86 | 0.03 | 0.01 |
| 2.5 | 347 | 3.12 | 0.06 | 0.02 |
| 3.0 | 423 | 3.36 | 0.05 | 0.01 |

Aplicando un análisis de regresión a la tabla se obtiene una curva de aproximación lineal con un error típico de 2.3% y con ecuación:

*Tiempo = 0.0033 * Entidades + 1.9497*

De la ecuación se deduce que el tiempo base para generar un mapa es de 1.95 segundos aproximadamente y que éste crece de forma lineal a una tasa de 0.0033 segundos por cada entidad del mapa.

## 7.2. *Pruebas de Funcionalidad*

Se efectuaron pruebas en tres emuladores (dos de SUN y uno de Nokia) y en un teléfono real (Nokia 6230 v05.5).

En la figura 7.2.1 se ve la aplicación en un emulador a la izquierda y en el teléfono a la derecha.

Figura 7.2.1.
Pruebas de funcionalidad

Puede apreciarse cómo cada mapa se adapta al tamaño y color de la pantalla, y además la forma en que se despliegan los menús varía de acuerdo al dispositivo.

### 7.3. *Pruebas de Rendimiento del* MIDlet

Se puede destacar que la velocidad de ejecución de la aplicación en el teléfono móvil fue un tanto superior a la obtenida en los emuladores, lo que demuestra la buena implementación de la máquina virtual, y permite esperar los mismos tiempos de respuesta, o incluso mejores, en dispositivos reales de lo que se percibe durante las pruebas en emuladores, siendo un caso muy puntual el cálculo de la ruta óptima entre dos puntos, donde en los emuladores tomaba unos 7 segundos en construir el árbol y encontrar una solución, en el teléfono móvil este tiempo bajó a unos 4 segundos aproximadamente.

Se detectó una restricción en el espacio de almacenamiento del teléfono. Si bien el teléfono dispone de más de 1 Mb de memoria persistente, sólo permite almacenar como máximo 32 kb por cada RecordStore; es decir, solo se puede tener una tabla con 32 kb máximo.

## 8. RESUMEN

En el presente proyecto se presentó el desarrollo de una aplicación destinada a dispositivos móviles que soporten Java J2ME que permite usar el teléfono móvil como un visor de mapas. Para ello se montó una base de datos geográfica, un servidor web y un servidor bluetooth que permiten alimentar de información al dispositivo móvil destino.

Se creó un modelo de una arquitectura constituida por cinco módulos que interactúan entre sí para llevar la información geográfica al usuario final.

Se montó una base de datos PostgreSQL más la extensión PostGIS, con lo cual se obtuvo una base de datos geográfica que cumple el estándar de la OGC.

Se implementó un servicio web en php sobre un servidor web Apache que permitió hacer consultas a la base de datos para generar mapas vectoriales en el formato estándar GML. Luego estos mapas se tranforman en documentos cGML utilizando XSLT.

Se creó un servidor bluetooth que actúa como sistema de localización y al mismo tiempo como puente entre el dispositivo final y el servicio web. Este servicio bluetooth se conecta al servidor web, obtiene un mapa en cGML y lo envía al cliente móvil.

Se implementó una aplicación MIDlet capaz de leer mapas en formato cGML, con conectividad web y bluetooth, que permite ver e interactuar con los mapas descargados y también almacenarlos en forma local.

## 9. CONCLUSIONES

Los resultados obtenidos permiten evidenciar los siguientes aspectos:

La importancia de implementar un sistema de información geográfico, ya que permite contar con información en un formato compatible con la mayoría de los programas, haciendo de esta manera más fácil el mantenimiento del sistema. Esto se reflejó en la base de datos usando PostGIS,

ya que soporta consultas y datos del estándar OGC, lo que facilitó la carga del mapa creado en formato ShapeFile a la base de datos.

La inconveniencia de los mapas en formato GML. En efecto, dado que la memoria, procesamiento y almacenamiento son recursos escasos en los dispositivos móviles que soportan el perfil MIDP los mapas en formato GML no son muy convenientes dado su gran tamaño. Se comprobó que el empleo de un compact GML o cGML fue una muy buena alternativa, dado que permitió bajar el tamaño de un mapa en alrededor de un 80%, y al mismo tiempo ayudar a ahorrar recursos de procesamiento en el cliente ya que las coordenadas son convertidas y escaladas en el servidor para que se adapten a la pantalla del móvil, liberando al dispositivo de estas costosas tareas.

Finalmente, sobre la base de los resultados evidenciados en este proyecto se ve una gran tendencia
en desarrollos de aplicaciones y servicios en el área de sistemas de información geográfica y sistemas de posicionamiento; por lo que se espera que aumenten en gran medida los proyectos similares al presentado en este trabajo, así como también, proyectos de índole comercial.

## REFERENCIAS

[1] Sun Microsystems. *Homepage for the J2ME platform*. Disponible: http://java.sun.com/j2me
[2] SIGweb. *La naturaleza de la información geográfica y su gestión mediante SIG*, cap. Modelos y estructuras de datos. Disponible: http://155.210.60.15/Geo/SIGweb/ Tema_2.htm
[3] Topologías, Modelos de Datos y Tipos de SIG. Disponible: http://recursos.gabrielortiz.com/ index.asp?Info=012
[4] cGML. *The compact GML for mobile devices*.
Disponible: http://www.crs4.it/nda/cgml/idea.html
[5] GDAL/OGR Utilities.
Disponible: http://www.gdal.org
[6] Sablotron extension for PHP
Disponible: http://www.gingerall.com/
[7] BlueCove open source implementation.
Disponible: http://sourceforge.net/projects/bluecove
[8] XML Parser for J2ME
Disponible: http://kxml.enhydra.org
[9] H. Mahmoud Qusay. *Wireles Application Programming with J2ME and Bluetooth.*(2003, Feb)
[10] Fischer and Nijkamp. *GIS Systems.* (1992).
[11] Max J. Egenhofer. *Spatial SQL: A Query and Presentation Language*. (1994)
[12] H. Mahmoud Qusay. *J2ME and Location-Based Services*. (2004 Mar)
[13] Nadège Faggion y Suresh Leroy. *Alcatel Location-based Services Solution*. (2005 Sept)
[14] OpenGis Consortium, Inc. *Geography Markup Language (GML). Implementation Specification*. (2004 Feb)
[15] Andrea Piras, Roberto Demontis, Emanuela De Vita y Stefano Sanna. *Compact GML: merging mobile computing and mobil cartography*. (2004 Jul)