Geolocalización de Sitios de Interés Para Aplicaciones Móviles

G-SIAM

Estado del Arte

VERSIÓN 1.3

Universidad de la Empresa
Soriano 959 - Tel.: 2900 2442
Montevideo - Uruguay | info@ude.edu.uy

Contenido

1 INTRODUCCIÓN 3

2 PLATAFORMA DE DESARROLLO 3

2.1 JAVA MICRO EDITION 3
2.2 SISTEMAS OPERATIVOS MÓVILES 4
2.3 CONCLUSIONES 9

3 OBTENCIÓN DE PUNTOS DE INTERÉS 10

3.1 INTRODUCCIÓN 10
3.2 SERVICIOS WEB DE TERCEROS 11
3.3 BASE DE DATOS PROPIA 12
3.4 CONCLUSIONES 13

4 CREACIÓN DE SERVICIOS WEB 14

4.1 INTRODUCCIÓN 14
4.2 WSDL + SOAP 14
4.3 REST 15
4.4 CONCLUSIONES 16

5 INTERFAZ GRAFICA 16

5.1 INTRODUCCIÓN 17
5.2 COMPONENTES POR DEFECTO 17
5.3 GREENDROID 17
5.4 CONCLUSIONES 18

6 REFERENCIAS 19

1 Introducción

Este documento pretende dar un cierre a la etapa de investigación de las tecnologías a utilizar en el proyecto, haciendo un análisis de las posibles alternativas en cuanto a sistemas operativos móviles existen en el mercado y por qué se eligió Android como plataforma a utilizar.
Ya que el proyecto cuenta también con una aplicación web, que es la que expone los servicios que consumirá el dispositivo móvil, se cree conveniente exponer en este documento las tecnologías investigadas para llevar a cabo esta aplicación.
Una de las características principales y más importantes del proyecto es de donde obtener los puntos de interés que se mostraran en el dispositivo móvil, se investigaron varias alternativas desde consumir servicios externos hasta implementar una base de datos propia de puntos de interés, analizando ventajas y desventajas de cada una.
Por último se analizaron las diferentes alternativas para la creación de la interfaz grafica.

2 Plataforma de Desarrollo

2.1 Java Micro Edition

La plataforma Java Micro Edition (JME) es un subconjunto de la plataforma Java para dispositivos con poca memoria y escasa capacidad de procesamiento como son PDA, electrodomésticos, teléfonos móviles entre otros.
En un entorno de ejecución JME aparecen diferentes componentes.

- Distintas maquinas virtuales (MV) dependiendo de la capacidad de procesamiento de cada dispositivo.
- Las configuraciones son el conjunto básico de librerías. Existen dos configuraciones:
  - CLDC (Connected Limited Device Configuration) enfocada a dispositivos con escasa capacidad de memoria como teléfonos móviles.
  - CDC (Connected Device Configuration) para dispositivos con mayor capacidad que la CLDC
- Los perfiles son librerías mas especificas de cada dispositivo orientadas a desarrollar funcionalidades de más alto nivel.

Al existir dos configuraciones diferentes cada una va a requerir su propia maquina virtual. La MV de la configuración CLDC se denomina KVM y para la configuración CDC se denomina CVM.
La Kilobyte virtual machine (KVM) adopta su nombre porque requiere muy poca memoria (entre 40Kb y 80Kb). Fue desarrollada por completo en C y con muy pocas líneas de código. Esta MV es la adecuada para dispositivos con muy poca capacidad de procesamiento.
La compact virtual machine (CVM) es una maquina virtual que posee todas las características de la MV estándar de Java y está pensada para dispositivos con una capacidad de procesamiento mayor como podrían ser electrodomésticos.
El objetivo principal de las configuraciones es el de garantizar la portabilidad entre diferentes dispositivos con escaza memoria y definir las funcionalidades generales. Las funciones mas especificas son definidas en los perfiles.

Un Midlet es una aplicación Java desarrollada sobre el perfil MIDP con la configuración CLDC.
Cabe aclarar que la capacidad de los dispositivos móviles ha ido en aumento y hoy por hoy existen dispositivos de gran capacidad, llamados Smartphone (teléfonos inteligentes) que permiten correr aplicaciones mucho más ricas en términos de capacidad de procesamiento e interfaz gráfica.
Estos dispositivos cuentan con un sistema operativo que los dota de características propias del fabricante perdiendo quizás homogeneidad entre los dispositivos de diferentes SO pero ganando en la calidad de las aplicaciones que corren en estos.
JME al no ser un sistema operativo para dispositivos móviles no puede dotar a los Smartphone de aplicaciones de gran calidad por lo que ha quedado rezagado en la lucha de este mercado que tanto está creciendo.
Por lo dicho anteriormente y dadas las características de la aplicación a desarrollar que requiere de una interfaz grafica atractiva y funcionalidades tales como GPS y una capacidad de procesamiento razonable damos se descarta la elección de JME como plataforma a desarrollar el proyecto.
A continuación se analizarán los principales sistemas operativos para dispositivos móviles y se profundizará en el estudio de Android que fue el elegido para desarrollar el proyecto de grado.

2.2 Sistemas Operativos Móviles

Para la elección del sistema operativo móvil (desde ahora SO) en el cual se desarrollara el proyecto, el equipo tuvo que investigar los distintos SO que están presentes en el mercado.
La lista a continuación abarca los principales SO hoy en día, cabe aclarar que la no presencia de Symbian en esta selección es por su unión con Windows y su inminente desaparición en un futuro no muy lejano.

2.2.1iOS

iOS es el SO desarrollado por Apple para dotar a sus terminales iPhone y actualmente a las cada vez más popular iPad.
Está basado en el kernel de Mac OS lo que lo dota de un SO robusto y una interfaz gráfica que se caracteriza por su fácil manejo.
La arquitectura del iOS está conformada por cuatro capas, el núcleo del sistema, la capa de servicios, la capa de comunicación y la capa de Cocoa Touch que es un conjunto de librerías para crear aplicaciones en iOS.
Una de las desventajas de iOS es que no soporta Java, en su lugar se usa un lenguaje de programación llamado Objective-C que es un súper conjunto de C orientado a objetos.
Pero la gran contra de este SO y en particular de los productos de Apple, es que son códigos cerrados y se debe contar con el hardware y software especifico para desarrollar para estos productos, por esto fue que se descarto como plataforma a utilizar en el proyecto.

2.2.2Windows Phone

Es el SO de Windows para dispositivos móviles, que anteriormente se llamaba Windows Mobile. Se basa en el sistema Windows CE y se caracteriza por sus similitudes con las versiones de escritorio para facilitar la tarea al usuario.
Este SO será según los expertos el que experimentará mayor crecimiento en cuota de mercado en el periodo 2011 – 2015.
Si bien no se opto por este SO para el proyecto por razones de tiempo y de que no es Open Source, es una alternativa a evaluar en un futuro si se quiere seguir con el desarrollo de este proyecto.
Otra de las razones por la cual no se elijo este SO es porque es el que menos cuota de mercado tiene actualmente. La nueva versión Windows Phone 7, la cual pretende competir con los SO de mayor peso en el campo de los dispositivos móviles como Android, iPhone, BlackBerry, esta aun muy verde en comparación a los mencionados anteriormente.

2.2.3Android

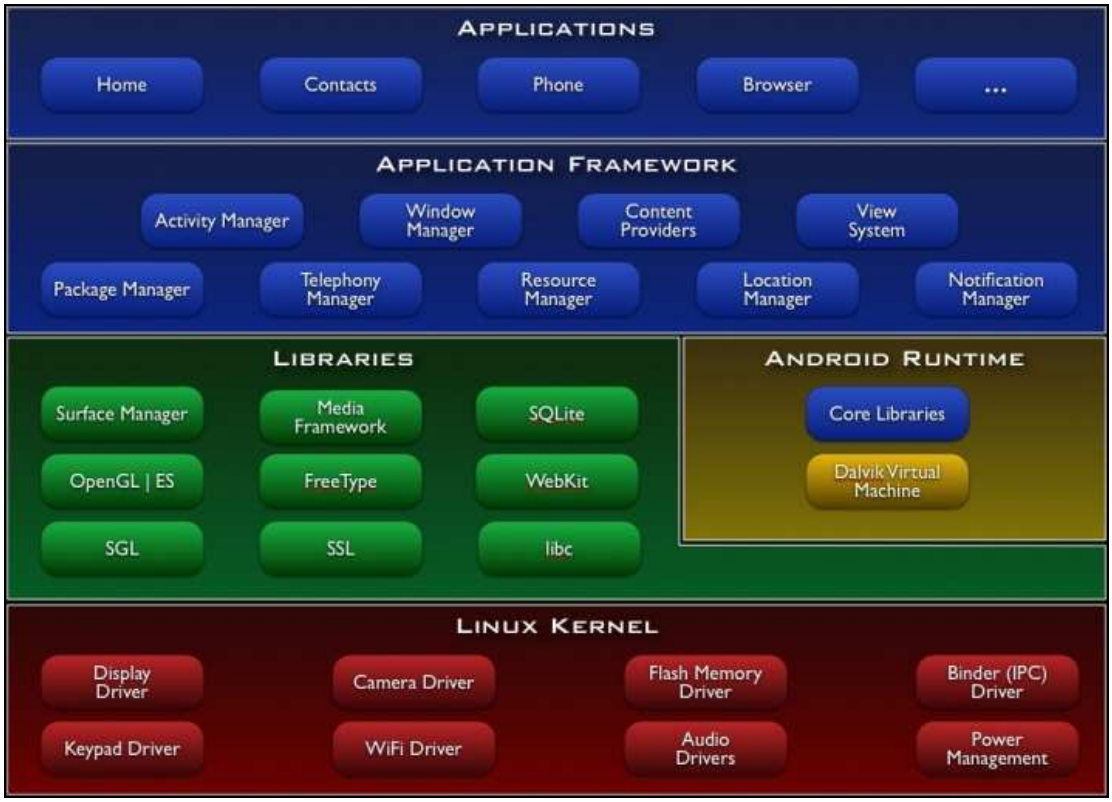
Android es una plataforma abierta basada en el kernel de Linux 2.6 para dispositivos móviles desarrollado inicialmente por Android Inc y luego adquirido por Google. Actualmente se encuentra bajo la Open Handset Alliance que está compuesta por más de 50 empresas de desarrollo de hardware, software y operadores de servicios.
Esta diseñado en una arquitectura en capas donde el kernel de Linux hace de una capa de abstracción entre el hardware y el sistema.
En esta capa se encuentran los distintos servicios que ofrece esta plataforma como son la seguridad, gestión de memoria y procesos, manejo de drivers y protocolos de red.
Por encima de esta capa se encuentran el conjunto de librerías, que es la base para el desarrollo de aplicaciones sobre esta plataforma.
El runtime de Android es un conjunto de bibliotecas que determinan la base para las librerías escritas en Java.
Uno de los principales componentes del runtime es la Maquina Virtual de Dalvik. Esta permite que cada aplicación Android corra su propio proceso en una maquina virtual Dalvik completamente independiente de los demás procesos. Esta máquina virtual ejecuta archivos .dex los cuales son optimizados para un bajo consumo de memoria.
Cuando el código se compila se generan archivos .class que luego son transformados archivos .dex.
Al mismo nivel del runtime se encuentran un conjunto de librerías escritas en C/C++ que son accedidas mediante la capa de aplicación. Sus funcionalidades abarcan desde la persistencia de datos mediante SQLite pasando por librerías multimedia para audio y video y manejo de aplicaciones graficas 2D y 3D mediante OpenGL.
La siguiente capa es quizás la más importante para los programadores ya que es con la que tienen contacto directo para el desarrollo de aplicaciones Android. Esta capa es el framework de Android y es la que utiliza las características antes planteadas.

Esta capa contiene varios componentes:

- Un conjunto de vistas que permite desarrollar la interfaz gráfica.
- Los proveedores de contenidos son el método de intercambio de información entre aplicaciones.

- Administrador de recursos que permite acceder a recursos como strings, gráficos, archivos de layout, etc
- El gestor de notificaciones permite generar alertas personalizadas al sistema.
- El gestor de aplicaciones es el encargado de manejar el ciclo de vida de las actividades.

La última capa, es la capa de aplicaciones que vienen por lo general ya instaladas en los sistemas operativos Android. Esta capa es desarrollada con todas las aplicaciones antes mencionadas e incluye cliente de correo, calendario, navegador, agenda de contactos entre otros.
A continuación se muestra la arquitectura de Android:



2.2.3.1Funcionalidades Generales

Las aplicaciones en Android se construyen en base a bloques de software elementales. Aunque no es necesario que una aplicación los utilice todos.

- **Activity:** Una actividad es sin duda el componente más utilizado en una aplicación Android. Se puede definir como una tarea que se lleva a cabo en la aplicación y tiene una interacción con el usuario, en otras palabas se puede decir que una actividad es una pantalla de la aplicación. Las aplicaciones estarán compuestas por muchas actividades, que relacionadas entre sí y cada vez que se accede a una nueva actividad, la anterior es

almacenada en una pila que gestiona el ciclo de vida de las actividades. Gracias a esto el usuario podrá navegar a una actividad anterior.

- **Intent:** Un intento como dice su nombre es la intención por hacer algo como puede ser "abrir el navegador" o "enviar un SMS". Android por su parte buscara la aplicación más adecuada para llevar a cabo esta petición. Estos intentos pueden ser los predefinidos por Android como se menciono anteriormente o los creados por el programador para navegar entre distintas actividades.

- **Service:** Un servicio es un proceso que se ejecuta en background sin necesidad de que haya interacción con el usuario y probablemente por un largo periodo de tiempo. Se podría decir que es equivalente al daemon de Unix. Por ejemplo un servicio podría estar ejecutando el reproductor de música, realizar operaciones de entrada/salida o enviar un SMS en background.

- **Content Provider:** Los proveedores de contenido recuperan y almacenan la información haciéndola accesible para otra aplicación. Es la forma que Android tiene para compartir información ya que no hay una zona donde todos paquetes puedan acceder.

2.2.3.2Ciclo de Vida de las Aplicaciones

Android maneja una jerarquía de procesos para determinar cuál debe continuar y cual debe matar en base a los componentes y el estado que se están ejecutando.

- **Proceso en primer plano:** Un proceso en este estado es aquel en el cual el usuario esta interactuando. Este será la última opción cuando se tenga que eliminar uno.

- **Proceso visible:** Este es un proceso que es visible por el usuario pero este no está interactuando directamente con él. Esta situación se puede dar por ejemplo cuando hay una confirmación en una ventana emergente. Al igual que el anterior, un proceso en este estado será de los últimos en ser elegidos para su eliminación por falta de memoria.

- **Proceso de servicio:** Aunque un proceso de servicio no es visible por el usuario y no necesitaría interacción de este, si hace cosas que al usuario le importa cómo puede ser el reproductor de música.

- **Proceso de segundo plano:** Estos procesos no se encuentran visibles por el usuario y el sistema podría matarlo en cualquier momento para recuperar memoria para otro proceso.

- **Proceso vacio:** Los procesos vacios son aquellos que no contienen ningún componente activo y son frecuentemente eliminados por el sistema. La única razón para mantenerlos activos es para manejar un cache y mejorar los tiempos de inicio.

2.2.3.3Ciclo de Vida de las Actividades

El ciclo de vida de una actividad es una parte muy importante para desarrollar aplicaciones de alto rendimiento y flexibles. Las actividades se van almacenando en una pila de actividades, cuando se llama a una actividad esta pasa a la parte superior de la pila y la anterior queda por debajo de esta en background hasta que vuelva nuevamente a la parte superior o sea eliminada por falta de memoria.
Los estados de una actividad son:

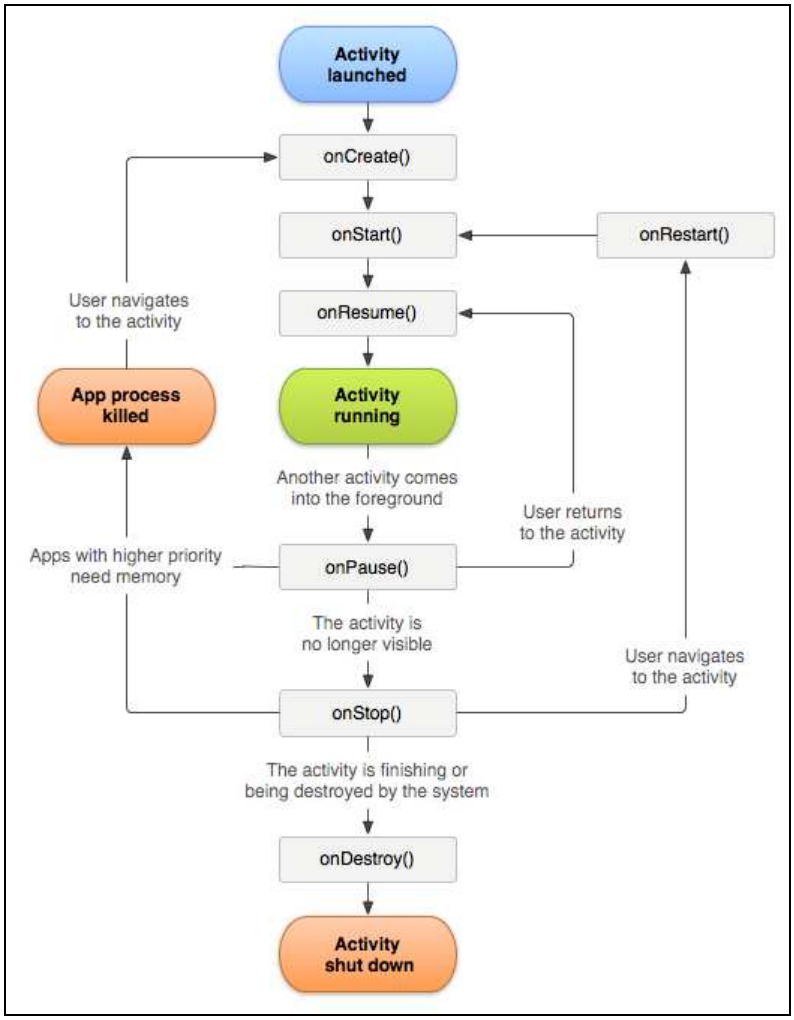
- **En ejecución:** La actividad se encuentra en el tope de la pila y tiene el foco del usuario.

- **En Pausa:** Una actividad está en pausa cuando deja de ser la principal pero aun es visible por el usuario. Esto es posible cuando se llama a una ventana emergente transparente de confirmación que deja ver la actividad anterior. Cuando la ventana emergente se cierra la anterior continúa con sus datos en el estado de ejecución.

- **Parada:** En este estado la actividad no es visible por el usuario pero sigue manteniendo toda la información. Sin embargo una actividad en este estado es elegible para ser eliminada de la pila si es que el sistema necesita memoria para otra aplicación.

El ciclo de vida de una aplicación está muy relacionado con el ciclo de vida de una actividad.
Android da prioridad a las actividades que están en interacción con el usuario. Para manejar el ciclo de vida de una actividad en Android existen los métodos de la clase Activity que son invocados con los cambios de estados correspondientes. Los métodos son: OnCreate(), OnDestroy(), OnPause(), OnStop(), OnFrezze(), OnResume() , OnRestart() y Finish().

A continuación se muestra un diagrama del ciclo de vida de una Activity:

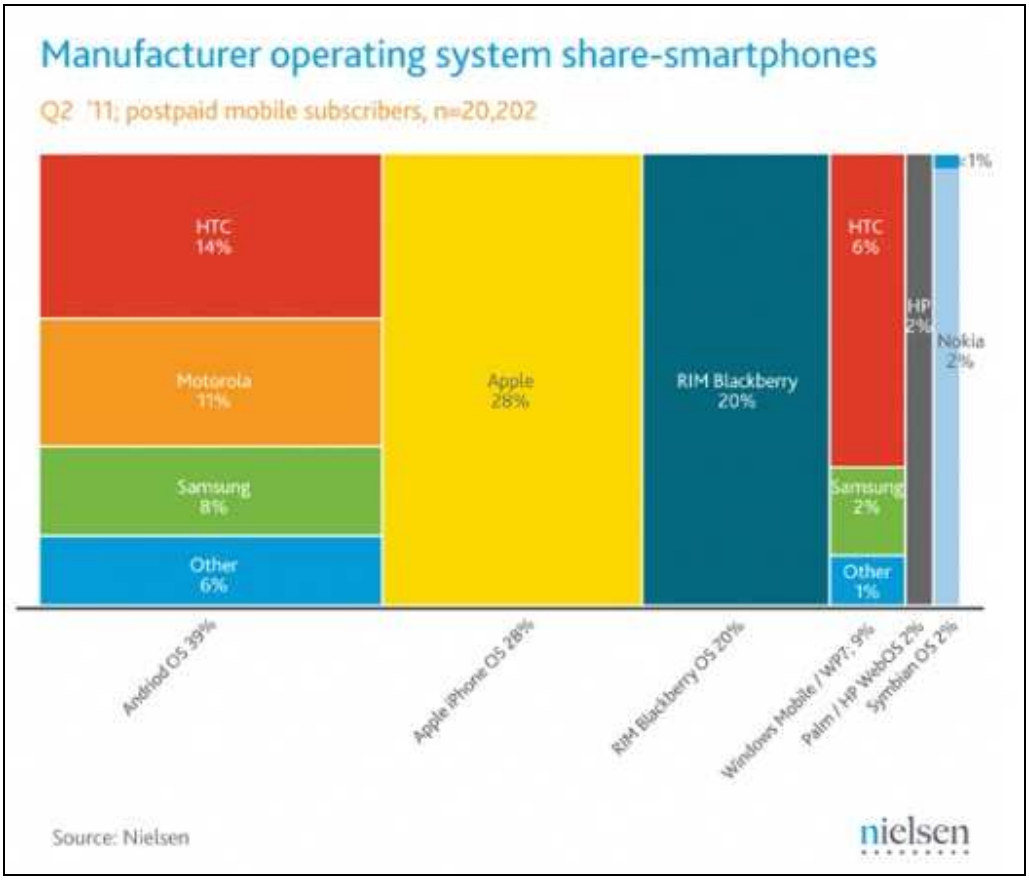


2.3 Conclusiones

La elección de Android como plataforma en la cual desarrollar el proyecto se debe básicamente a tres puntos

- Es Open Source, cumpliendo uno de los requisitos planteados del proyecto.
- Además de ser el líder en cuota de mercado de hoy en día, las proyecciones lo dan como el ganador a futuro con el 43.8% de mercado en el 2015.
- Por último dado la experiencia de los integrantes del equipo en el lenguaje de programación Java y dado que Android utiliza este lenguaje es otro de los puntos a favor que determinaron en la elección de esta plataforma.

La siguiente infografía muestra la cuota de mercado de los sistemas operativos para dispositivos móviles más importantes.



3 Obtención de Puntos de Interés

3.1 Introducción

La obtención de puntos de interés es uno de los pilares fundamentales del proyecto y uno de los que genero mayor investigación.
Un punto de interés desde ahora en más PDI, se entiende como cualquier sitio físico que pueda ser localizado en un mapa por un par de coordenadas geográficas y resulte de utilidad para el usuario.
Se analizaron distintas formas de obtener dichos puntos, desde servicios web de terceros hasta la creación de una base de datos propia con los PDI.
A continuación se detallan las diferentes alternativas investigadas.

3.2 Servicios Web de Terceros

Una forma de obtener los PDI es consumiendo servicios de terceros mediante un llamado por RESTful al recurso que se quiere obtener. Dentro de los proveedores más importantes se encuentra Simplegeo y Geonames.
Ambos son bases de datos con millones de PDI que pueden ser consultados por llamadas a los servicios que exponen.
.

3.2.1SimpleGeo

Simplegeo es una base de datos con millones de PDI que expone mediante servicios web, además ofrece apis de desarrollo para diferentes lenguajes de programación como Java, Object-C, Java Script, Python lo que la convierte en una alternativa interesante.
Simplegeo expone tres tipos de productos:

- Simplegeo Place está basado principalmente en la obtención de PDI.
- Simplegeo Context obtiene datos del contexto en el cual nos encontramos (temperatura, datos del clima, etc).
- Simplegeo Storage permite almacenar información en la nube de forma rápida y en tiempo real.

3.2.1.1Ventajas

Dentro de las ventajas que se encontraron se debe destacar que es una base de datos con millones de registros de rápido acceso y robustez.
Permite consumir servicios web con diferentes criterios de búsqueda como puede ser PDI cercados a la ubicación del dispositivo a un radio dado.
Otra de las ventajas es que permite ingresar nuevos PDI, esto es de gran importancia para el proyecto ya que uno de los requerimientos es el de ingresar nuevos sitios, además de que pose muy pocos PDI para Uruguay.

3.2.1.2Desventajas

La gran desventaja que se encontró es que no es un servicio del todo gratis ya que limitan el número de llamadas por día, cosa que para esta instancia no es de mayor importancia pero si se quiere continuar con este proyecto es un tema a tener presente.
Otra de las grandes desventajas al consumir servicios de terceros es la disponibilidad de los mismos y que la aplicación queda atada al buen funcionamiento de estos.

3.2.2Geonames

Geonames es otra base de datos con millones de registros PDI al igual que Simplegeo, también expone los recursos mediante servicios web con un alto grado de disponibilidad y robustez.
También ofrece una gran cantidad de librerías escritas en diferentes lenguajes de programación como son Java, Objective-C, Ruby, Python, entre otras para acceder a estos servicios.

3.2.2.1Ventajas

Una de las ventajas de utilizar Geonames es que es una base de datos con millones de PDI que se pueden consultar mediante servicios web con diferentes clientes.

3.2.2.2Desventajas

La gran desventaja que se encontró en esta alternativa es que no permite por código ingresar nuevos PDI y este es un requerimiento importante dentro del proyecto.

3.3 Base de datos Propia

Otra de las alternativas que se investigo y por la cual se opto es la de tener una base de datos propia donde guardar los PDI.
Para poder manejar estos datos de una forma rápida y eficiente es necesario manejar datos espaciales para saber la posición geográfica del PDI. Se opto por usar PostgreSQL con un plugin llamado Postgis para el manejo de datos geográficos.

3.3.1Funcionalidades generales

PostgreSQL es una base de datos Open Source que cuenta con una gran cantidad de desarrolladores. El hecho de que sea Open Source es de gran importancia para el proyecto ya que este es un pilar importante del este.
Postgis es un plugin de PostgreSQL que permite manejar y realizar consultas sobre datos espaciales (GIS). Con Postgis se pueden manejar todos los objetos definidos en la especificación OpenGIS como son puntos, líneas, polígonos, multipuntos, etc. En el proyecto solo se manejaran puntos para representar a los sitios de interés pero cabe la posibilidad de utilizar otros objetos en futuros trabajos.
Los datos espaciales son guardados en la base de datos en una columna de tipo GEOMETRY que es agregada a PostgreSQL al momento de instalar el plugin.
A continuación se muestran ejemplos de consultas SQL para el manejo de datos espaciales:

```
INSERT INTO SPATIALDATABASE(THE_GEOM,THE_NAME)
VALUES(GeometryFromText('POINT(-126.4 45.32)',312),'Un Lugar')
```

Esta consulta es un ejemplo de cómo insertar un punto en una tabla de la base de datos que permite manejar datos GIS

```
SELECT *
FROM GEOTABLE
WHERE
GEOM && GeometryFromText('BOX3D(900 900,1100 1100)',-1)
AND
Distance(GeometryFromText('POINT(1000 1000)',-1),GEOM)<100;
```

Esta consulta muestra como seria la forma de recuperar los objetos que se encuentran a menos de 100 metros de distancia del punto POINT (1000, 1000).

3.3.2Ventajas

Entre sus principales ventajas se encuentran:

- Alta concurrencia mediante un sistema denominado MVCC (Acceso concurrente multi-versión) que permite que mientras un proceso escribe en una tabla, otro proceso podrá acceder a la misma tabla sin necesidad de bloqueos. Esta característica es superior a los bloqueos de tabla o fila que aplican otros sistemas de gestión de base de datos.

- Debido a que es un proyecto Open Source y que hay una gran cantidad de desarrolladores detrás de este, existen muchos plugins para instalar como el que se utilizara en el proyecto para manejar datos espaciales.

Quizás la gran ventaja de que Gsiam maneje los datos de los PDI y usar PostgreSQL para esto, es que se tiene absoluto control y no se depende de ningún servicio externo que podría ocasionar problemas de disponibilidad del servicio. Como por ejemplo en el caso de Simplegeo se corre el riesgo de caer en gastos que no son admitidos en este proyecto ya que se trata de que sea un servicio completamente gratis para el usuario.

3.3.3Desventajas

La desventaja más importante que se encontró en manejar los PDI es la inexperiencia del equipo en este tipo de tecnología. Esto genera que se tenga que invertir una cantidad de tiempo considerable en el aprendizaje de base de datos espaciales y consultas espaciales que se utilizaran en el desarrollo del proyecto.

3.4 Conclusiones

Por lo expuesto anteriormente se opto por manejar los PDI en una base de datos propia PostgreSQL con el plugin PostGis para el manejo de datos geográficos.
Con esta elección se tendrá más control sobre los datos manejados y no se dependerá de servicios externos.
Para el acceso a datos se usara JDBC ya que JPA, que permite una integración con EJB de manera clara, no soporta consultas espaciales.
Si bien se manejaran datos geográficos en la base de datos estos serán solo para almacenar la ubicación geográfica de los PDI ya que manejarlos sin esta característica resultaría por demás costoso en cuanto a rendimiento.

4 Creación de Servicios Web

4.1 Introducción

Dado que se decidió manipular los datos en una base de datos propia es necesario construir y exponer un conjunto servicios web para poder acceder a la información. Si hablamos de servicios web en Java, no se puede pasar por alto el API JAX-WS. Esta API Java esta dentro de la especificación JEE 5 y permite la definición de servicios web mediante anotaciones de una forma simple. El intercambio de información con un cliente es a partir de mensajes XML como lo hace SOAP. JAX-WS no permite, de forma estándar el intercambio de información mediante el formato JSON que es más liviano que el XML.
Es de vital importancia lograr una comunicación rápida y eficiente entre el dispositivo móvil y el servidor. Las dos formas más utilizadas hoy en día para construir servicios web son: mediante la tecnología SOAP + WSDL o basados en la arquitectura REST. Es por esto que surge la interrogante acerca de la elección de la misma.

4.2 WSDL + SOAP

Una de las primeras alternativas que surge al momento de hablar de consumir servicios es la utilización de SOAP y WSDL que son dos de los estándares más usados de la especificación WS.
Por un lado SOAP, es un estándar que permite invocar servicios remotos mediante XML. Otra de sus características es que es independiente de la arquitectura de red por cual se puede invocar servicios sobre cualquier protocolo como puede ser SMTP, MQ entre otros aunque en el proyecto el único que interesa es HTTP.
Por otro lado, WSDL es el estándar en XML que permite definir servicios como un contrato que el cliente debe saber para lograr la comunicación.

Las funcionalidades generales de estos dos estándares son la de consumir servicios sobre casi cualquier arquitectura. Esto en una arquitectura compleja y que requiera de una integración con sistemas legados podría ser la opción adecuada.

4.2.1Ventajas

Quizás la gran ventaja que se encuentra al utilizar SOAP y WSDL es la madurez con la que cuentan estos estándares, documentación y productos que hay para su desarrollo.

4.2.2Desventajas

Ya grandes proveedores de Web 2.0 están migrando a una arquitectura REST, incluyendo a Yahoo, Google, eBay y Facebook, quienes marcaron como obsoletos a sus servicios SOAP y WSDL y pasaron a usar un modelo más fácil de usar, orientado a los recursos.
Esto se debe quizás a la complejidad que se tiene para desarrollar y consumir servicios con SOAP y WSDL y al ancho de banda que consumen.

4.3 REST

En una arquitectura basada en REST todo es un recurso. Un recurso se accede a través de una interfaz común basada en los métodos estándar HTTP. En una arquitectura REST se suele tener un servidor REST que proporciona el acceso a los recursos y al cliente REST que accede y modifica los recursos REST. Todos los recursos deben soportar las operaciones HTTP comunes.
Los recursos son identificados por un Global ID (por lo general son URIs).
REST permite que los recursos tengan diferentes representaciones, ej. texto, XML, JSON, etc.
El cliente REST puede pedir una representación específica a través del protocolo HTTP (Negociación de Contenido).

RESTFul: Son aquellos servicios Web que funcionan bajo REST.
Los Web Service REST proporcionan acceso a través de los métodos GET y POST de HTTP.
GET: En accesos vía GET, tanto las operaciones como los parámetros se pasan por la URL. Sólo soporta argumentos con tipos simples. Es utilizado para obtener un recurso del servidor
POST: En este tipo de acceso, la información no viaja en mensajes SOAP Envelope, sino directamente en el payload del mensaje. Se utiliza para agregar un recurso en el servidor
PUT: Este método es usado para cambiar el estado de un recurso o actualizarlo
DELETE: Este método es utilizado para eliminar un recurso.
Una implementación concreta de un servicio web REST sigue cuatro principios de diseño fundamentales:

- Utiliza los métodos HTTP de manera explícita
- No mantiene estado
- Expone URIs con forma de directorios
- Transfiere XML, JavaScript Object Notation (JSON), o ambos

4.3.1Ventajas

- Simples de implementar.

- Es bueno cuándo es importante que la comunicación sea ligera en términos de bytes transmitidos debido al coste.
- Eficientes cuándo el consumidor tiene limitaciones de ancho de banda ej: móviles, PDAs, etc.
- Ya varios grandes proveedores de Web 2.0 están migrando a esta tecnología, incluyendo a Yahoo, Google, eBay y Facebook, obteniendo un modelo más fácil de usar orientado a los recursos.

.

4.3.2Desventajas

- Gran número de objetos.
- Manejar el espacio de nombres (URIs) puede ser engorroso.
- La descripción sintáctica/semántica muy informal (orientada al usuario).
- Pocas herramientas de desarrollo.

4.4 Conclusiones

Tomando en cuenta que la aplicación está enfocada en dispositivos móviles se determina que la mejor opción es utilizar REST dado que es ideal para este tipo de aplicaciones como se describió anteriormente. Además los servicios web en REST son muy fáciles de implementar y dado que la complejidad principal del proyecto es Android no existe la necesidad de sumar complicaciones técnicas.
RestEasy de Jboss es el framework elegido para la implementación de los servicios web RestFul. RestEasy es una implementación de la especificación Java de JAX-RS. Dicha especificación define una API Java sobre protocolo HTTP que permite definir servicios web RestFul. Se escogió este framework ya que además de ser Open Source, su integración con el servidor Jboss es muy sencilla aunque puede correr en cualquier contenedor de Servlets como podría ser Tomcat, lo cual lo dota de una gran portabilidad.
Ya que JAX-RS es una especificación del lado del servidor RestEasy permite definir su contraparte del lado del cliente y así implementar los métodos RestFul mediante anotaciones definidas en la especificación JAX-RS.

5 Interfaz Grafica

5.1 Introducción

Al momento de comenzar con los prototipos de pantallas el equipo encontró que la parte grafica no era sencilla, ya que cada componente se maneja de forma particular y cuentan con cierta complejidad.
Dentro de los prototipos de pantallas se tienen componentes del estilo Tabs, ActionBar, List, Maps, SegmentedHost, entre otros.
Es por esto que surge la necesidad de análisis de la interfaz grafica. Además de los componentes que vienen por defecto dentro del SDK Android se investiga una librería llamada GreenDroid. Esta librería intenta ayudar a los desarrolladores codificar aplicaciones altamente funcionales así como también aprovechar toda la potencia de Android. En el mercado se encuentran varias aplicaciones que utilizan esta librería.

5.2 Componentes por Defecto

Android proporciona componentes por defecto, llamados widgets o las herramientas necesarias para crearlos.
Para desarrollar la interfaz grafica, se puede utilizar ficheros xml con un editor grafico proporcionando por el framework, o desarrollar la interfaz en base a código Java.

5.2.1Ventajas

Utilizar los componentes por defecto del framework tiene la gran ventaja de contar con el testing necesario para su óptima utilización. Además de contar con una cantidad de ejemplos y tutoriales para su aprendizaje.

5.2.2Desventajas

Los componentes estándares tienen un aspecto básico y no son por demás atractivos para el usuario final. Esto es importante en una aplicación para dispositivos móviles, donde los usuarios buscan interfaces atractivas y sencillas.
Otra desventaja que presenta es que se repite mucho código para implementar interfaces sencillas.

5.3 GreenDroid

GreenDroid es una librería para Android, que facilita el desarrollo de la interfaz grafica además de hacerla mucho más amigable para el usuario. Uno de los propósitos de GreenDroid es evitar la pérdida de tiempo en la copia de los mismos fragmentos de código una y otra vez. Dentro de sus principales características, se destaca la inclusión de una barra superior (ActionBar) con diferentes funcionalidades configurables que le dan al usuario mayor usabilidad. Otra de las características es la facilidad de creación y manejo de pestañas (tabs) dentro de una

aplicación. Para utilizarlo en un proyecto Android, solo basta con incluir el proyecto GreenDroid al workspace y luego incluirlo como librería al proyecto. A continuación se muestran ejemplos.

5.3.1 Ventajas

Una de las grandes ventajas es que es fácil de utilizar. Al utilizar Greendroid se garantiza un diseño y estilo único en toda la aplicación. Es Open Source y como se menciono anteriormente es de gran importancia para el proyecto.

5.3.2 Desventajas

Por tratarse de una librería relativamente nueva no se encuentra mucha información, solo se encuentra la documentación del sitio y ejemplos en el Blog del autor.

5.4 Conclusiones

Por lo expuesto anteriormente se opto por el uso de GreenDroid para el proyecto.
Debido a que Gsiam apunta a ser una herramienta de uso cotidiano, es primordial que sea amigable. Además al ser un prototipo es de suma importancia que sea mantenible y reutilizable para poder ser modificado en el futuro de manera sencilla.
En los prototipos de pantallas se ven varios componentes los cuales al ser tratados con esta librería se logra un aspecto más profesional, entre ellos se encuentran los tabs, actionBar y list.
A continuación se muestra un prototipo de tab con los componentes por defecto y otro con GreenDroid para notar la diferencia visual:

6 Referencias

Android: http://es.wikipedia.org/wiki/Android (último acceso 21/12/2011)
Java Mobile: http://www.oracle.com/technetwork/java/javame/javamobile/training/jmesdk/index.html (último acceso 21/12/2011)
IOS: http://es.wikipedia.org/wiki/IOS_(sistema_operativo) (último acceso 21/12/2011)
IOS: http://developer.apple.com/technologies/ios/ (último acceso 21/12/2011)
Windows Phone: http://es.wikipedia.org/wiki/Windows_Phone (último acceso 21/12/2011)
IDC - Press Release: http://www.idc.com/getdoc.jsp?containerId=prUS22871611 (último acceso 21/12/2011)
The Developer's Guide: http://developer.android.com/guide/index.html (último acceso 21/12/2011)
Porcentaje de mercado: http://www.idc.com/getdoc.jsp?containerId=prUS22871611 (último acceso 21/12/2011)
Android Estadísticas: http://blog.nielsen.com/nielsenwire/online_mobile/in-u-s-smartphone-market-android-is-top-operating-system-apple-is-top-manufacturer/ (último acceso 21/12/2011)
Simplegeo: https://simplegeo.com/ (último acceso 21/12/2011)
Geonames: http://www.geonames.org/ (último acceso 21/12/2011)
Datos espaciales: http://es.wikipedia.org/wiki/Base_de_datos_espacial#Puntos (último acceso 21/12/2011)
PostgreSQL: http://es.wikipedia.org/wiki/PostgreSQL (último acceso 21/12/2011)
PostGis: http://es.wikipedia.org/wiki/PostGIS (último acceso 21/12/2011)
Building Web Services with JAX-WS: http://docs.oracle.com/javaee/5/tutorial/doc/bnayl.html (último acceso 21/12/2011)
JAX-WS: http://en.wikipedia.org/wiki/Java_API_for_XML_Web_Services (último acceso 21/12/2011)

JSON: http://es.wikipedia.org/wiki/JSON (último acceso 21/12/2011)
SOAP: http://es.wikipedia.org/wiki/Simple_Object_Access_Protocol (último acceso 21/12/2011)
WSDL: http://www.w3.org/TR/wsdl (último acceso 21/12/2011)
Rest: http://es.wikipedia.org/wiki/Representational_State_Transfer (último acceso 21/12/2011)
Restful: http://www.ibm.com/developerworks/webservices/library/ws-restful/ (último acceso 21/12/2011)
GreenDroid: http://greendroid.cyrilmottier.com/ (último acceso 21/12/2011)