Cómo introducir semántica en las aplicaciones SIG móviles: expectativas, teoría y realidad

Laia Descamps-Vila[1], Joan Casas[2], Jordi Conesa[2], A.Pérez-Navarro[2]

[1] I.C.A. Informática y Comunicaciones Avanzadas, S.L., C/ Almogàvers 107-119, 08018 Barcelona, ldescamps@grupoica.com.
[2] Estudis d'Informàtica, Multimèdia i Telecomunicació, Universitat Oberta de Catalunya, Rambla Poblenou 156, 08018 Barcelona, [aperezn|jconesac|jcasasrom]@uoc.edu.

RESUMEN

Con la aparición de los GPS y la difusión de los smartphones (iPhone, Android, etc.) han proliferado aplicaciones que tratan con información geográfica. Muchas de estas aplicaciones ya empiezan a proporcionar información semántica a través de: 1) la clasificación de puntos de interés turísticos en categorías (tesauros turísticos, categorías de OpenStreetMap, etc.), y 2) la representación de sus metadatos (LinkedGeoData, Cruzar, etc.). La pregunta es: si dicha información ya está disponible, ¿por qué los dispositivos móviles no la utilizan para ofrecer servicios geográficos de forma más "inteligente"? Una posible respuesta está en la limitación de recursos en los dispositivos móviles, tanto a nivel físico (memoria y velocidad de procesador) como a nivel lógico (inexistencia de programas o módulos necesarios, como PostGIS o Jena).

Este trabajo presenta un exhaustivo estudio sobre los principales formalismos que permiten representar conocimiento y las herramientas que los utilizan disponibles en dispositivos móviles. El artículo presenta, para cada uno de los formalismos y herramientas, una descripción, sus actuales implementaciones en dispositivos móviles y las limitaciones de dichas implementaciones. Este estudio permite vislumbrar rápidamente las distintas opciones disponibles para implementar aplicaciones geográficas semánticas en dispositivos móviles. El articulo muestra también un caso práctico donde se han utilizado ontologías para aportar semántica a un asistente de viaje que se ejecuta en los dispositivos móviles.

***Palabras clave**: personalización, móvil, SIG, web semántica.*

ABSTRACT

With the advent of GPS and the spread of smartphones (iPhone, Android, etc.) have proliferated applications dealing with geographic information. Many of these applications are beginning to provide semantic information through: 1) the classification of tourist attractions in categories (thesauri tourism categories OpenStreetMap, etc.), And 2) the representation of metadata (LinkedGeoData, Crossing, etc.). The question is: if such information is already available, why mobile devices does not use it to

provide geographic services in more "intelligently"? One possible answer lies in the limited resources on mobile devices, both physical (memory and processor speed) and logical level (no programs or modules needed, as PostGIS or Jena).
This paper presents a comprehensive study on the main formalisms which represent knowledge and tools available on mobile devices. The article presents, for each of the formalisms and tools, a description, its current implementations on mobile devices and the limitations of such implementations. This study provides a glimpse into the various options available to implement semantic geographic applications on mobile devices. It also shows a case study where ontologies have been used to provide semantics to a travel assistant that runs on mobile devices.

***Keywords**: customization, movile devices, GIS, web semantics.*

1 INTRODUCCIÓN

El creciente interés y progreso tecnológico de los dispositivos móviles y sistemas basados en la localización (LBS) ha llevado a la aparición de muchas aplicaciones móviles basadas en Sistemas de Información Geográfica (SIG) como gvSIG Mini [1], OsmAnd, Google Maps 5 y Wikitude. Estas aplicaciones proporcionan información y servicios acerca de los recursos geográficos tales como puntos de interés (POI), rutas de cualquier tipo (turísticas, comerciales, etc.), navegación por la calle y realidad aumentada.

Sin embargo, los dispositivos móviles actuales todavía presentan limitaciones para gestionar la información geográfica, y son incapaces de gestionar la información semántica. Las aplicaciones actuales salvan estos escollos mediante una conexión a internet continuada que permita ejecutar las aplicaciones en el servidor.

Así, la mayoría de los SIG que se ejecutan en dispositivos móviles siguen una arquitectura cliente-servidor donde el servidor es el responsable de almacenar y gestionar toda la información y realizar la mayoría de las operaciones geográficas. La ventaja es que se mejora el tiempo de proceso debido al mayor rendimiento de los servidores; el inconveniente es que necesitan de una conexión a Internet continua para proporcionar todos los servicios necesarios y, además, el rendimiento global de la aplicación está condicionado por la calidad de la conexión. Como ejemplo de SIG con está arquitectura se puede ver el proyecto CsxPOI [2].

Ejecutar las aplicaciones en el servidor parece una solución satisfactoria, a priori. Sin embargo, en el ámbito español, por ejemplo, la cobertura 3G o HSDPA (3.5G) es inexistente en la mayor parte del territorio, como podemos ver en el mapa proporcionado por las diferentes compañías telefónicas nacionales [3,4,5]. La figura 1 muestra, en particular, el mapa de cobertura de la compañía telefónica Orange en el área de Catalunya: los lugares donde hay cobertura 3G están pintados de color púrpura. Se puede ver que prácticamente sólo cubren las áreas densamente pobladas. La poca cobertura de Internet no es crítica para aplicaciones SIG que siempre se usan en una de estas áreas, pero es realmente problemática si es necesario desplazarse por el territorio, como sería el caso de un turista o de un operario de mantenimiento de la red eléctrica, por poner dos ejemplos.

Es por tanto, necesario, ejecutar las aplicaciones SIG en el dispositivo móvil de forma independiente del servidor. Sin embargo, las limitaciones en rendimiento y capacidad de almacenamiento de estos dispositivos dificultan esta opción. Es por tanto necesario buscar una solución que permita al usuario disponer en el móvil sólo de aquél volumen de información que realmente necesite, es decir, que permita al

usuario filtrar los datos según sus necesidades y sus preferencias. Así, la arquitectura cliente-servidor seguiría vigente, pero sólo sería necesaria en breves períodos de tiempo ya que su objetivo es "personalizar" el SIG para el usuario: la información enviada sería diferente de acuerdo a los intereses del mismo (historia, naturaleza, compras) o teniendo en cuenta otros factores tales como el número o tipo de usuarios (viaja con la familia o con amigos).

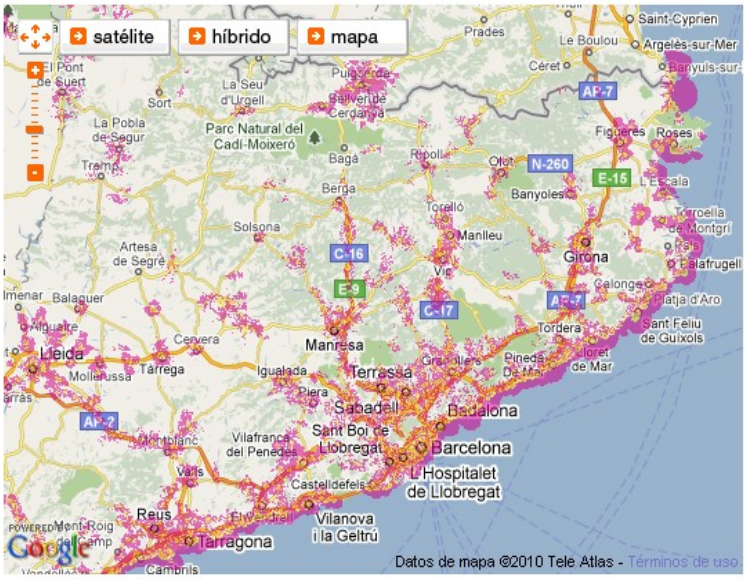

Figura 1: *Cobertura 3G (color lila) en Catalunya, proporcionada por la compañía Orange.*

La personalización de las aplicaciones SIG móvil se puede realizar mediante el uso de la tecnología de Web Semántica, que proporciona diferentes herramientas para almacenar información relacionada con las preferencias del usuario e inferir información acerca de esta información. La integración entre la Web Semántica y los SIG en el dispositivo móvil es un foco actual de estudio [6,7].

En resumen: con el fin de desarrollar aplicaciones SIG turísticas que sean eficaces, necesitamos crear aplicaciones que funcionen en cualquier lugar y que ayuden al usuario a descartar la información irrelevante para él/ella de forma automática. Para hacer eso, las aplicaciones deben ejecutarse por completo en el móvil mediante el almacenamiento de datos espaciales y la ejecución de operaciones espaciales en el dispositivo. También deben proporcionar las funciones de la Web Semántica con el fin de filtrar los datos enviados a los usuarios.

El objetivo de este artículo es estudiar cómo incluir en el dispositivo móvil la ejecución de las aplicaciones y las funciones de la web semántica para desarrollar aplicaciones móviles funcionales y eficientes. Así, este estudio no se centra en una aplicación móvil que ofrece rutas generales a todos los usuarios, sino que se trata de un sistema que genera rutas personalizadas: cada vez que un usuario desea comenzar un viaje se diseña una nueva ruta especialmente adaptada a él/ella, y todo esto se hace sin necesidad de tener conexión a Internet.

Para lograr este objetivo, la aplicación tiene que cumplir los siguientes requisitos:

a) mostrar información turística personalizada basada en las preferencias del usuario,

b) almacenar y administrar información geográfica en el dispositivo móvil,

c) ejecutar todos los procesos geográficos en el móvil, como la gestión de datos espaciales y el recalculo de rutas cuando sea necesario.

El artículo se estructura en siete secciones: En la sección 2 se presenta un estado del arte donde se explican algunos conceptos necesarios para entender todos los pasos del estudio, en la sección 3 se explican los requisitos necesarios para alcanzar los objetivos del documento, en las secciones 4 y 5 se estudia cómo implementar un prototipo que cumpla los requisitos descritos en la sección 3 y también se realizan algunas pruebas funcionales, en la sección 6 se discuten los resultados obtenidos en las pruebas anteriores y, finalmente, en la sección 7 se recogen las conclusiones.

2 ESTADO DEL ARTE

En esta sección se describen algunos conceptos importantes que son necesarios para entender todos los pasos de esta investigación. Para ello, se hará una breve introducción a la Web Semántica en general y se mostrará qué es una ontología, los datos RDF y el lenguaje SPARQL; a continuación, se hará un breve repaso a las bases de datos espaciales, desde el punto de vista del presente estudio.

2.1. Lenguajes de ontologías: definición i utilización

Las ontologías son modelos conceptuales de parte de la realidad escritos en un lenguaje interpretable por un programa.[8]

Al estar enfocadas a computadoras deben estar escritas en un idioma que un programa puede ser capaz de entender. Hoy en día, los lenguajes de ontologías más utilizados son RDF (Resource Description Framework) y OWL (Ontology Web Language).

RDF es un modelo estándar para el intercambio de datos en la Web. Se basa en tripletas que representan expresiones del tipo sujeto-predicado-objeto (**quién** hace **qué** con **algo**). OWL es el lenguaje propuesto por W3C para representar ontologías en la web semántica, y se ha construido como una extensión de RDF. Con el fin de gestionar los datos RDF, existen algunos lenguajes de consulta, entre los que destaca SPARQL. SPARQL es el equivalente a SQL en bases de datos (BBDD) relacionales, pero con operaciones de consulta limitadas. Por lo tanto, usar SPARQL es la forma más conveniente de obtener y administrar los datos RDF.

2.2. OWL/RDF Frameworks

Para almacenar los datos RDF hay dos sistemas de almacenamiento principales: TDB y SDB.

TDB es una base de datos no transaccional integrada que se utiliza para el uso de conjuntos de datos muy grandes. Este sistema almacena los datos RDF como un grupo de datos en un solo directorio, en un sistema de ficheros, llamado Dataset. SDB es un sistema de almacenamiento que utiliza bases de datos SQL. SDB permite operaciones transaccionales y tiene soporte para PostgreSQL, MySQL, Oracle, SQL server, H2.

Con la aparición de las tecnologías de la Web Semántica, han surgido diferentes frameworks RDF para trabajar con las ontologías, los datos RDF y las consultas SPARQL. Un framework RDF permite el almacenamiento, la inferencia y la consulta de datos RDF. Aquí se describen los frameworks más populares, que son Jena y Sesame. Ambos son muy similares, están escritos en Java y tienen soporte para los lenguajes RDF y OWL.

Jena ofrece los componentes TDB y SDB (descritos a continuación) y Sesame tiene almacenamiento nativo y en BBDD relacionales. Todos los sistemas de

almacenamiento están diseñados para ser consultados utilizando el lenguaje SPARQL.

2.3. Bases de datos espaciales

Cuando se utilizan datos geográficos las BBDD relacionales se vuelven insuficientes ya que tienen limitaciones en el almacenamiento de datos espaciales, no permiten usar las localizaciones espaciales como índices y no proporcionan funciones espaciales. Por ello, es preferible utilizar BBDD espaciales, que están optimizadas para trabajar con datos espaciales.

Las BBDD espaciales tienen dos características importantes: proporcionan operaciones espaciales y usan índices espaciales (sobre todo R-Tree y QuadTree) . Algunas librerías espaciales también proporcionan algunas de estas características, que son una colección de recursos que dan soporte a las aplicaciones espaciales, por ejemplo JTS Topology Suite o GeoTools [9]. Estas permiten tratar con información espacial sin necesidad de utilizar BBDD espaciales. Esto es útil cuando la persistencia no es necesaria o cuando no es posible utilizar BBDD espaciales.

Las librerías JTS Topology Suite y GeoTools están escritas en Java y proporcionan implementaciones de todos los “Simple Features” que se encuentran en las especificaciones del OGC (Open Geospatial Consortium).

Una vez establecido el panorama actual, en el apartado siguiente se mostrarán las características del sistema propuesto.

3 SISTEMA ESPERADO

El objetivo del presente proyecto es disponer de un sistema SIG en el dispositivo móvil que esté personalizado para el usuario y que pueda funcionar desconectado de internet.

Para ello se ha desarrollado una aplicación SIG móvil que funcione en un sistema operativo móvil Android, con la versión 2.1 - Eclair o superior (todos los teléfonos móviles pueden actualizarse hasta 2.1). El móvil utilizado para hacer las pruebas de este trabajo tiene un procesador de 600 MHz y 250 MB de memoria RAM.

Para alcanzar el primer objetivo (personalización), el dispositivo debe ser capaz de almacenar y consultar datos semánticos en la aplicación móvil. Para ello se ha creado una ontología turística. Ésta es una extensión de la ontología LinkedGeoData [10] de LinkedData y utiliza las características de OpenStreetMap, pero adaptada para describir recursos turísticos. Con el fin de sacar provecho de los datos creados por otros usuarios y hacer que los resultados obtenidos sean interoperables, los datos semánticos se almacenan con formato RDF y se utiliza un framework RDF para tratarlos.

Para conseguir que la aplicación funcione completamente en el dispositivo móvil, es necesario almacenar datos espaciales y usar funciones geográficas en el móvil. Estos datos y funciones se utilizaran para seleccionar los POI y para el algoritmo de cálculo de rutas. Para ello, es necesario almacenar la información espacial en el móvil.

En resumen: se dispone en el móvil de una ontología almacenada, mecanismos de consulta de datos de la misma, BBDD espaciales y un algoritmo de cálculo de rutas en el móvil. Con todo esto ya se podría disponer de la aplicación, pero para que sea viable es necesario que las operaciones con información semántica y espacial sean eficientes para ofrecer una experiencia de usuario satisfactoria.

En las dos secciones siguientes se estudia cómo implementar un prototipo que satisfaga estos requisitos. En primer lugar se presenta el objetivo a lograr. A

continuación, se explican las pruebas que se realizan para encontrar una solución y los resultados obtenidos. Cuando los resultados son insatisfactorios, se presentan soluciones alternativas hasta que se da con una solución viable.

Este estudio tiene dos partes diferenciadas, con dos conjuntos de pruebas: primero uno para tratar con datos semánticos en el móvil y después otro para tratar con datos espaciales. Se sigue este orden porque el trabajo con datos semánticos en el móvil es más complicado y puede llevar a hacer algunos cambios en la aplicación que pueden afectar a la segunda parte del estudio.

4 ALMACENAR Y TRATAR DATOS SEMÁNTICOS EN EL MÓVIL

En esta sección se analiza la forma de almacenar y administrar datos semánticos en el dispositivo móvil. Como se ha dicho, el estudio se centra en los datos almacenados en formato RDF y se hacen las consultas utilizando el lenguaje SPARQL.

A continuación se presenta un ejemplo de una consulta SPARQL (Figura 2). Este ejemplo consulta las subclases de la clase *instrumentos de viento* de una ontología de la temática de música.

La primera solución que se propone es utilizar un framework de Web Semántica en el móvil, de la misma manera que se utilizan frameworks de Web Semántica como Jena o Sesame en una aplicación Java que corra en un ordenador.

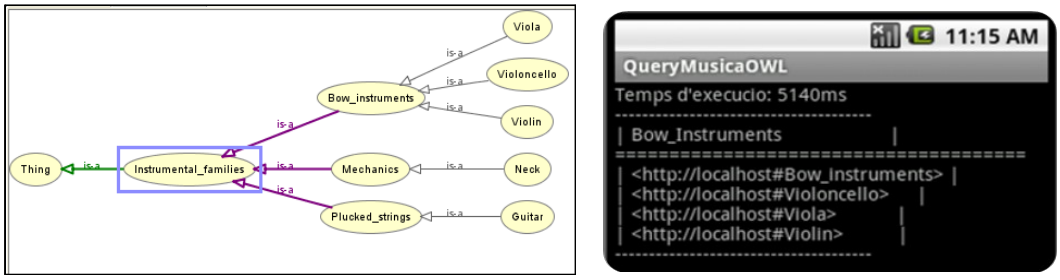

Figura 2: *Modelo conceptual de la ontología de música. Consulta SPARQL. Resultados de la consulta con el tiempo de ejecución, más de 5 segundos.*

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX : <http://localhost#>
        SELECT * WHERE {
        ?Bow_Instruments  rdfs:subClassOf <http://localhost#Bow_instruments>}";
```

Prueba1A. Utilizar el Framework de Sesame

La primera opción que se considera es utilizar el framework de Sesame. Como se ha dicho, Sesame se utiliza para almacenar, hacer inferencias y consultar datos RDF. El framework de Sesame es ampliamente utilizado y tiene buen soporte de la comunidad.

Problema: Sesame no está adaptado a Android, por lo que es necesario adaptar sus librerías para utilizarlas en este sistema operativo, e incluso en ese caso no se puede asegurar que funcionen. Por tanto, esta solución no es viable y se buscan otras opciones.

Prueba1B. Androjena + ARQoid + TDBoid

Una solución para el problema de la *Prueba1A* es buscar un framework RDF que ya esté adaptado a Android. Hoy en día, la única opción que se ha encontrado con un buen soporte por parte de la comunidad es Androjena [11]. Androjena es una adaptación del popular framework de Jena a la plataforma Android. Incluye un motor de consulta SPARQL (ARQoid) y un sistema de almacenamiento de base de datos no transaccionales que almacena los datos en un sistema de ficheros, un Dataset (TDBoid).

Problema: Androjena aún no tiene soporte para trabajar con BBDD relacionales, que es una funcionalidad necesaria en nuestro sistema.

Androjena no se ha adaptado completamente a Android, ya que sigue sin tener un sistema de almacenamiento de BBDD relacional. Jena tiene un componente llamado SDB, que se usa para el almacenamiento de datos RDF en BBDD relacionales, pero tampoco ha sido adaptado a Android todavía. Sólo se ha adaptado un sistema de almacenamiento, el TDB.

En nuestra aplicación se tienen que utilizar y conectar dos tipos de datos distintos: datos semánticos con los recursos turísticos y datos espaciales con información del mapa. Cuando la aplicación genera una ruta con puntos de interés, los datos de la ruta, que serán semánticos y espaciales, necesitan unirse para seleccionar los puntos de interés a visitar y para generar la ruta que pasa por todos estos puntos. Los datos espaciales se deben almacenar en una base de datos espacial o en una base de datos relacional extendida con funcionalidades espaciales. Por lo tanto, se necesita el componente SDB de Jena, que no está adaptado a Android, para acceder a datos espaciales que se encuentren en una base de datos relacional.

Otro punto a tener en cuenta es la edición de los datos turísticos. Esto es muy importante porque hoy en día el éxito de un proyecto depende de la colaboración de diferentes personas, por lo que la capacidad de integración de datos es una necesidad. La incorporación y el mantenimiento de datos en una BBDD relacional es más fácil y más eficiente que en un archivo. Por ejemplo, si un museo desea añadir información nueva sobre su POI o se abre un nuevo restaurante en una región, necesitamos un sistema de almacenamiento con una forma fácil de hacer la edición de la información de los POI.

También existe la posibilidad de almacenar parte de la información en una base de datos relacional (como datos no-RDF) y tener otra parte en un Dataset (como datos RDF). Ambos conjuntos de datos se pueden combinar usando las aplicaciones D2RQ, Triplify o Jibbering, que hacen un mapeo de la base de datos relacional hacia datos RDF. Algunos proyectos como DBpedia [12] o LinkedGeoData [10] utilizan estos softwares de mapeo. Desafortunadamente, cuando estas aplicaciones se ejecutan en dispositivos móviles obligan a trabajar conectadas con el servidor.

Prueba1C. Androjena + ARQoid + BBDD relacional

Otra solución puede ser utilizar un framework de Web Semántica y una BBDD relacional. En el contexto del móvil consiste en utilizar Androjena junto con el motor de consulta SPARQL (ARQoid) y una BBDD relacional.

La ventaja de esta solución es que tenemos datos RDF en una base de datos y podemos hacer consultas SPARQL. Además, es posible añadir índices a la base de datos para mejorar la eficiencia y la gestión de todo el sistema. De hecho hay una aplicación, RDF on the Go [13], que ya se sigue esta arquitectura.

Problema: Realizar consultas SPARQL en el móvil es muy ineficiente.

Al utilizar Androjena, los datos RDF se representan como tripletas (sujeto-predicado-objeto) y las consultas deben consultar cada tripleta de la ontología. Se trata de una gran cantidad de información, lo que puede llevar a tiempos de consulta elevados.

Para verificar si lo tiempos de respuesta son aceptables, se crea una aplicación para Android que ejecuta consultas SPARQL sobre distintas ontologías. Las ontologías se almacenan en la tarjeta SD del dispositivo y, con el fin de obtener una visión más completa, cada una tiene un número distinto de tripletas. Después de realizar las pruebas, se obtiene, por ejemplo, que una consulta sobre las subclases de una clase tomó 5 segundos para una ontología de 18 tripletas, y 80 segundos para una ontología de 402 tripletas. Éste es un tiempo de ejecución de las consultas demasiado elevado. Los tiempos se acortan cuando la ontología está en memoria, pero aún así siguen siendo ineficientes: 1 segundo para la primera ontología y 50 segundos para la segunda. Dado que la ontología que se utiliza en el presente trabajo tiene alrededor de 400 tripletas, esta solución no es adecuada.

Solución elegida

Como se ha visto, usar datos RDF en el móvil lleva a perder eficiencia. Por lo tanto, se tiene que descartar el uso de datos RDF con el fin de garantizar un mínimo de usabilidad.

Para poder utilizar el modelo conceptual de la ontología, se propone traducir la ontología a un esquema de BBDD relacional. Las instancias de la ontología se insertarán como datos en las tablas de la BBDD. De esta manera se tiene la posibilidad de hacer consultas SQL para consultar y gestionar datos semánticos, que es mucho más eficiente que usar SPARQL. Como sistema de BBDD relacional se ha elegido SQLite, ya que es ligero y las librerías están integradas en el sistema Android.

Una vez encontrada la solución para disponer de la ontología en el móvil, en el próximo apartado se mostrará cómo llevar a cabo las operaciones espaciales sobre los datos almacenados en el móvil.

5 OPERACIONES ESPACIALES EN EL MÓVIL

En esta sección se analiza la forma de almacenar y gestionar datos espaciales en el dispositivo móvil. La primera solución que se propone es utilizar una BBDD espacial en el móvil, de la misma manera que utilizamos BBDD espaciales en los ordenadores. A continuación se verá qué problemas plantea esta solución y se mostrarán otras alternativas.

Prueba2A. Utilizar BBDD espaciales

El mejor método para almacenar y realizar operaciones espaciales de manera eficiente es el uso de BBDD espaciales. Este tipo de BBDD está especialmente diseñada para procesar datos espaciales y proporcionar índices espaciales y funciones geográficas. Una BBDD espacial se implementa generalmentecomo una extensión de una BBDD relacional. Esto es muy conveniente para la aplicación que se propone en este artículo porque se adapta a la arquitectura que se utiliza para administrar datos semánticos en el móvil (ver sección 4): en ambos casos se utiliza una BBDD relacional. Otra ventaja de usar una BBDD relacional es la facilidad y rapidez de integración con un SIG, ya que los SIG pueden utilizar BBDD espaciales como *backend* de base de datos.

Sin embargo, no hay BBDD espaciales diseñadas para Android. Por ejemplo, SQLite tiene una extensión espacial llamada SpatiaLite, pero no se encuentra disponible en

Android. Una solución a la falta de BBDD espaciales en Android es adaptar una desde cero. En particular, se podría adaptar SpatiaLite para Android, y así se dispondría de soporte completo para las funciones OpenGIS soportadas a través de GEOS (migración a C++ de la librería JTS Topology Suite), así como los índices espaciales R-Tree para mejorar la eficiencia de la consulta.

Problema: Adaptar la BBDD espacial a Android es complicado. Como SpatiaLite está escrito en C, es necesario compilar en código nativo utilizando el Native Development Kit (NDK) de Android. El Android NDK proporciona los *headers* y librerías que permiten construir aplicaciones programando en C o C++. Junto con SpatiaLite, las librerías espaciales como GEOS y Proj4 también se tienen que compilar porque SpatiaLite necesita estas librerías para funcionar correctamente.

Desarrollar en código nativo aumenta la complejidad de las aplicaciones, y Google sólo recomienda utilizar el NDK si es esencial para la aplicación y no hay otra forma de hacer lo mismo, por lo que es preferible buscar otra opción más sencilla.

Prueba2B. SQLite + Módulo R-Tree

La otra opción que se propone es usar SQLite y añadir índices R-Tree utilizando el módulo R-Tree que, al mismo tiempo, es una parte de los recursos de SpatiaLite. Para crear un índice R-Tree sólo se tiene que crear una tabla virtual en nuestra base de datos SQLite con un número de columnas determinado por la dimensión del R-Tree.

Problema: La adición de índices R-Tree a una BBDD relacional mejora el tiempo de consulta, pero no proporciona ningún tipo de soporte para gestionar datos espaciales. Operaciones como las medidas espaciales (distancia entre POIs, el área de una ciudad), funciones espaciales (intersección de dos calles, un POI se encuentra en esa zona) o predicados espaciales (saber si hay un hospital en un lugar determinado) no estan disponibles si se utiliza esta solución. Por lo tanto, la adición de índices espaciales no es suficiente si se necesitan funciones espaciales mínimamente complejas.

Prueba2C. Adaptar la librería JTS Topology Suite a Android

Para resolver la falta de funciones geográficas se puede adaptar a Android la librería JTS Topology Suite, que está escrita en Java, y obtener así todas las funciones y algoritmos espaciales. Esta adaptación es relativamente sencilla: se puede añadir la librería jts-1.11.jar al proyecto de Android y se pueden utilizar las clases que necesitamos.

Problema: Los recursos necesarios son demasiado grandes. JTS Topology Suite no es una librería ligera (tiene 694KB). Además, tiene un elevado número de métodos de las clases de JTS, que son innecesarios para la aplicación propuesta. En realidad, sólo se necesitan algunas funciones geográficas, como la proximidad de un POI, la distancia entre los POI y comprobar si hay POI dentro de un área determinada. Así, la solución es crear las funciones geográficas necesarias para la aplicación Android, lo que resulta en una librería más pequeña y ligera.

Solución elegida

Tras llevar a cabo las diversas pruebas, la solución elegida es: 1) utilizar una BBDD relacional (en particular SQLite); y 2) crear una librería con las funciones espaciales necesarias.

Teniendo en cuenta el volumen de datos y que el sistema sólo necesita funciones geográficas simples vale la pena preguntarse si es necesario utilizar los índices R-

Tree para mejorar las consultas espaciales. Evitar la utilización de índices R-Tree simplifica la implementación de la librería que realiza las operaciones espaciales necesarias y reduce la complejidad de la BBDD final.

Para verificar la necesidad o no de índices espaciales se realizan las siguientes pruebas: 1) se almacena la información de los POI turísticos (latitud, longitud, tipo de POI, horario de apertura, etc.) de España extraída de OpenStreetMap en una BBDD SQLite (102.529 POI); 2) se crea una función espacial para realizar las consultas sobre la BBDD con el fin de seleccionar los POI más cercanos a una localización determinada (longitud, latitud) dentro de una distancia definida (metros).

Los resultados que se obtienen son que, por ejemplo, para una localización dada, la función implementada necesita 2 segundos para devolver 596 POI dentro de un rango de 100 metros y 1 segundo para devolver los 73 POI más cercanos. De acuerdo con estas pruebas y dado que el usuario verá un máximo de 20-25 POI por ruta, se concluye que no es necesario utilizar índices R-Tree y, por lo tanto, se prescinde del módulo R-Tree de SQLite.

Sí que se utilizarán los índices tradicionales B-Tree de la BBDD SQLite, implementados en las coordenadas de los POI para optimizar las consultas, los cuales también están disponibles en Android.

6 DISCUSIÓN

Llegados a este punto, se ha mostrado que es posible usar datos semánticos y espaciales en el móvil, pero con algunas limitaciones. La Tabla1 resume las posibilidades y limitaciones de la gestión de datos espaciales y semánticos mediante la descripción de las pruebas que se han hecho y sus resultados. Para cada prueba, se indica si se ha concluido con éxito (columna “Sí”) y, si es así, las herramientas que han sido necesarias.

	SÍ	**NO**
Almacenar datos RDF en el móvil	Utilizar Androjena	
Consultas SPARQL en el móvil	Utilizar ARQoid	
Eficiencia de consultas SPARQL en el móvil		X
Usar el modelo conceptual de la ontología en el móvil	Como esquema de BBDD relacional y consultas SQL	
BBDD espaciales en el móvil		X
Índices espaciales en BBDD relacionales	Módulo R-Tree en SQLite	
Funciones geográficas en el móvil	Usar librería JTS Topology Suite en Android	
Necesidad de índices espaciales en el móvil		X (depende de las funciones espaciales necesarias)

Tabla1: Pruebas realizadas y resultados. Las cuatro primeras pruebas tratan la semántica en el móvil y los últimas cuatro pruebas el uso de datos espaciales.

Cuando se trata de datos semánticos, el problema es la ineficiencia de las consultas con el lenguaje SPARQL en el móvil. Con el fin de mejorar el tiempo de consulta algunas aplicaciones móviles que trabajan con datos semánticos almacenan y realizan las consultas en un servidor. En estos casos, las consultas se realizan en el

servidor y la aplicación sólo obtiene un archivo con los resultados. Ejemplos de ello son DBpedia mobile [14] y CsxPOI [2].

Cuando se usan datos espaciales el principal problema es que no hay BBDD espaciales disponibles para su uso en Android. A pesar de que no se ha usado, existe la posibilidad de utilizar la librería espacial JTS Topology Suite en Android e implementar los índices espaciales en una BBDD relacional como SQLite.

7 CONCLUSIONES

Los dispositivos móviles fueron diseñados originalmente para ser transparentes a las operaciones que hace una aplicación, es decir, para ser la capa que muestra la información. La limitación de este procedimiento es que el usuario siempre necesita conexión a Internet para obtener toda la información desde el servidor. Esto es un problema en aplicaciones, como las turísticas, que tengan que funcionar en zonas sin cobertura 3G, ya que hoy en día esta cobertura es aún muy pobre.

Con el fin de resolver ese problema, en este artículo se ha estudiado cómo crear una aplicación móvil que realice todas las operaciones necesarias a nivel local, es decir, sin conexión a Internet. Esta aplicación deberá generar, además, rutas turísticas siguiendo las preferencias del usuario, por lo que también será necesario almacenar las preferencias del usuario en el dispositivo móvil. Ello se lleva a cabo mediante tecnología de web semántica.

Almacenar la ontología en RDF obligaría a realizar consultas SPARQL. Se ha mostrado en el artículo que estas consultas son muy ineficientes en un dispositivo móvil tal, por lo que se evita esta aproximación. Así, para implementar la tecnología de Web Semántica se ha transformado la ontología turística que se ha creado a un esquema de base de datos SQLite. Esta base de datos SQLite se ha poblado con los datos de la ontología, lo que permite hacer consultas SQL.

Otro punto importante que se ha visto es que no hay BBDD espaciales para Android, aunque es posible conseguir funcionalidades similares. Se puede utilizar la librería espacial JTS Topology Suite adaptada a Android, aunque para las necesidades de la aplicación propuesta y el volumen de datos con que se trabajará, es preferible crear una aplicación “llave en mano”, más ligera que JTS. Además, se ha demostrado que en el caso particular tratado son prescindibles los índices espaciales R-Tree.

Así, la arquitectura final de la aplicación piloto estará compuesta por: una base de datos relacional que se usa para almacenar la información geográfica, turística y semántica; y un programa para Android que implementa las funciones geográficas básicas. La base de datos contiene la información de la vecindad próxima a la zona que va a visitar el usuario. Eso garantiza el poder realizar todos los cálculos en el dispositivo móvil, y evitar así tener que acceder a Internet [15].

Como trabajo futuro se evolucionará la aplicación piloto hacia una aplicación de producción. Ello implica hacer pruebas con diversos dispositivos en distintas zonas. Dado que la aplicación también se conecta a Internet cuando sea posible, será necesario realizar pruebas en áreas donde hay cobertura 3G y áreas donde no la hay y verificar su comportamiento en ambos entornos. También es necesario estudiar si es posible integrar más herramientas de la tecnología de Web Semántica u optimizar las consultas SPARQL para que se utilicen en un móvil.

8 AGRADECIMIENTOS

Este artículo ha estado desarrollado gracias al soporte del proyecto TSI-020110-2009-442 y al Instituto *Internet Interdisciplinary Institute* de la Universitat Oberta de Catalunya.

9 REFERENCIAS

- [1] MONTESINOS, M.; CARRASCO, J.; DEL REY, A. (2010), *gvSIG Mini y Phone Cache*, Girona, IV Jornadas de SIG Libre SIGTE 2010.
- [2] BRAUN, M. (2009), *Context-aware Collaborative Creation of Semantic Points of Interest as Linked Data*, University of Koblenz-Landau.
- [3] Cobertura Orange [web]. [última consulta, 8 de marzo de 2011]. <http://movil.orange.es/soporte-y-ayuda/herramientas/mapa-de-cobertura/>
- [4] Cobertura Movistar [web]. [última consulta, 8 de marzo de 2011]. <http://www.servicios.movistar.es/web/movistar/cobertura>
- [5] Cobertura Vodafone [web]. [última consulta, 28 de enero de 2011]. <http://www.vodafone.es/conocenos/cobertura/consulta-cobertura/>
- [6] DELLA VALLE, E.; DELL'AGLIO, D.; CELINO, I. (2010), "The Experience of Realizing a Semantic Web Urgan Computing Application", *TransactionsInGIS*, 14(2): p163-181.
- [7] ZIPF, A.; JÖST, M. (2006), "Implementing Adaptive Mobile GI Services based on Ontologies: Examples from pedestrian navigation support", *Computers, Environment and Urban Systems in Location* Based Services, Vol. 30, No. 6., pp. 784-798.
- [8] DESCAMPS-VILA, L.; CASAS, J.; CONESA, J.; PÉREZ-NAVARRO, T. (2011), Hacia la mejora de la creación de rutas turísticas a partir de información semántica, Girona, V Jornadas de SIG Libre SIGTE 2011.
- [9] RAMSEY, P. (2007), *The State of the Art of Open Source GIS*, Canada, FOSS4G 2007.
- [10] AUER, S.; LEHMANN, J.; HELLMANN, S. (2009), "LinkedGeoData - Adding a Spatial Dimension to the Web of Data", *8th International Semantic Web Conference* (ISWC2009)
- [11] Androjena [web page]. [última consulta, 8 de marzo de 2011]. <http://code.google.com/p/androjena/>
- [12] BIZER, C.; LEHMANN, J.; KOBILAROV, G.; AUER, S.; BECKER, C.; CYGANIAK, R.; HELLMANN, S. (2009), "DBpedia – A Crystallization Point for the Web of Data", *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, Issue 7, Pages 154–165.
- [13] LE-PHUOC, D.; PARREIRA, J.X.; REYNOLDS, V.; HAUSWIRTH, M.(2010), *RDF On the Go: An RDF Storage and Query Processor for Mobile Devices*, 9th International Semantic Web Conference (ISWC2010)
- [14] BECKER, C.; BIZER, C. (2008), *DBpedia Mobile: A Location-Enabled Linked Data Browser*. 1st Workshop about Linked Data on the Web (LDOW2008), Beijing, China, April 2008.
- [15] DESCAMPS-VILA, L.; CASAS, J.; CONESA, J.; PÉREZ-NAVARRO, T. (2011), Personalización de servicios basados en la localización: un caso práctico, Girona, V Jornadas de SIG Libre SIGTE 2011.