Software geoespacial por y para OpenStreetMap

I. Sánchez Ortega[(1)] *y J. Figueras i Jové* [(2)]

[(1)] Miembro de la Fundación OpenStreetMap, ivan@sanchezortega.es
[(2)] Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial (ESAII), Escola Tècnica Superior d'Enginyeries Industrial i Aeronàutica de Terrassa, Universitat Politècnica de Catalunya, Rambla Sant Nebridi, 10, 08222 Terrassa, jaume.figueras@upc.edu

RESUMEN

En este trabajo se presenta la arquitectura global, y de los diferentes componentes software que conforman el proyecto OpenStreetMap, una iniciativa mundial de proveer a la comunidad de un mapa realmente libre. El objetivo de esta ponencia es el análisis del sistema desde su punto de vista técnico-informático. Se describen desde su primer y vital componente, la base de datos geoespacial, el protocolo de acceso a esta, los editores de datos y mapas, los sistemas de dibujo (renderización) tanto locales como usando computación distribuida, los servidores de mapas rasterizados (tile servers), aplicaciones sobre estos, hasta llegar a los últimos proyectos innovadores propuestos desde la comunidad.

Key words: *OpenStreetMap, Open Source, Sistemas de Información Geográfica, Bases de Datos.*

INTRODUCCIÓN

Una característica de OpenStreetMap que lo hace poco atractivo para el mundo del GIS es el hecho de que no usa las primitivas geométricas habituales (puntos, líneas, polígonos), sino un modelo de datos orientado a topología.

La primitiva geométrica más sencilla que podemos encontrar es el *nodo*. Un nodo se define únicamente como un par de coordenadas y un identificador único. Por ejemplo, usando la representación XML común a casi todo el software de OpenStreetMap:

```
<node id="156804" lat="61.8083953857422" lon="10.8497076034546" />
```

Un nodo puede tener uno o varios *tags* (o *etiquetas*) para darle una semántica:

```
<node id='30894545' lat='41.9797918' lon='2.8201551'>
<tag k='place' v='town' />
 <tag k='name' v='Girona' />
 <tag k='name:es' v='Gerona' />
 <tag k='is_in' v='Geronès, Girona, Catalonia, Spain, EU' />
 <tag k='population' v='92186' />
```

```
 <tag k='source:population' v='BOE REAL DECRETO 1683/2007, de 14 de
diciembre' />
</node>
```

Varios nodos forman un *way* (o *camino*), dados los identificadores únicos de los nodos. Un *way* es un conjunto ordenado de nodos, una primitiva similar a un *linestring*:

```
<way id='23151177'>
  <nd ref='250101455' />
  <nd ref='250671042' />
  <nd ref='250101453' />
  <nd ref='250101446' />
  <nd ref='250101449' />
  <nd ref='250095866' />
</way>
```

También se pueden aplicar *tags* a los *ways*:

```
<way id='23151177'>
  <nd ref='250101455' />
  <nd ref='250671042' />
  <nd ref='250101453' />
  <nd ref='250101446' />
  <nd ref='250101449' />
  <nd ref='250095866' />
  <tag k='created_by' v='JOSM' />
  <tag k='highway' v='road' />
  <tag k='name' v='Carrer de Industria' />
</way>
```

En el modelo de datos de OSM no hay una diferencia explícita entre líneas cerradas y polígonos; esta diferencia ha de ser semántica. Por ejemplo, si tenemos un *way* cerrado, con una etiqueta de *landuse=farmland*, quiere decir que el polígono definido por ese *way* es una granja. Si, en cambio, tenemos un *way* cerrado, con una etiqueta de *barrier=fence*, quiere decir que hay una alambrada a lo largo de esa línea. Si tenemos un *way* que tiene tanto el tag de *landuse=farmland* como el de *barrier=fence*, quiere decir que hay una superficie dedicada a granja, y que su perímetro es una alambrada. Nótese que en este último ejemplo no hace falta definir por separado el polígono para la superficie de la granja y la línea para la alambrada del perímetro.

Al poder dotar de un sentido semántico tanto a los nodos como a los *ways*, se pueden relacionar fácilmente los unos con los otros. Es práctica habitual colocar las gasolineras (*amenity=fuel*) como nodos dentro de una carretera de servicio (*highway=service*), y las torres de apoyo (*power=tower*) como nodos dentro de una línea eléctrica de alta tensión (*power=line*).

Por último, existen las *relaciones*, que permiten enlazar entre sí nodos, ways, y otras relaciones. Actualmente se usan para definir multipolígonos (relacionando varios ways cerrados entre sí), y rutas (como líneas de autobús o metro, o rutas a pie como el Camino de Santiago). Por ejemplo, la ruta europea E-15 (Inverness – Londres – Paris – Barcelona – Valencia – Algeciras) se define como:

```
<way id='8500107'>
  <nd ref='33959879' />
  <nd ref='13851357' />
  <nd ref='13851388' />
```

```
  [...]
  <nd ref='331877131' />
  <nd ref='13852607' />
  <tag k='ref' v='AP-7' />
  <tag k='highway' v='motorway' />
  <tag k='name' v='Autopista del Mediterrani' />
  <tag k='oneway' v='yes' />
  <tag k='int_ref' v='E15' />
  <tag k='toll' v='yes' />
</way>

<relation id='48044'>
  <member type='way' ref='1216' role='' />
  <member type='way' ref='1655339' role='' />
  <member type='way' ref='1880986' role='' />
  [...]
  <member type='way' ref='8500107' role='' />
  [...]
  <member type='way' ref='30478509' role='' />
  <member type='way' ref='30478511' role='' />
  <tag k='ref' v='E 15' />
  <tag k='route' v='road' />
  <tag k='type' v='route' />
  <tag k='int_ref' v='E 15' />
</relation>
```

El *rol* de cada miembro de la relación sirve para determinar, por ejemplo, qué way es el contorno exterior o interior de un multipolígono o, para las relaciones de restricciones de giro, desde qué nodo está permitido llegar a qué nodo a través de qué nodo en una búsqueda de caminos.

Con respecto a las etiquetas, no existe una ontología estricta. Hay una lista de etiquetas comúnmente utilizadas, decidida por consenso, en la página Map Features [1] del wiki de OpenStreetMap. Cualquier persona es libre de usar el etiquetado que considere oportuno, aunque se recomienda encarecidamente que se intenten usar las mismas etiquetas que ya están en uso, y que son conocidas por el software existentes.

EL "RAILS PORT"

En el corazón de la arquitectura de OpenStreetMap, junto a la base de datos, está el denominado "**Rails Port**". En los principios del proyecto, la API para acceder a los datos (que se comunicada con la BDD) estaba escrita en Ruby. En un determinado momento, se decidió migrar todo ese código a Ruby on Rails (un framework de Ruby totalmente orientado a objetos y a modelo-vista-controlador). Precisamente, "Rails Port" se puede traducir como "migración a Rails".

Dada la naturaleza MVC de Ruby on Rails, es difícil diferenciar dónde empieza y dónde acaba cada uno de los elementos que implementa el Rails Port, a saber:

- Definición (y creación) de las tablas de la BDD.
- Interfaz web de www.openstreetmap.org (páginas de registro y gestión de usuarios, enlaces a Potlach, etc).
- API para acceso a los datos

Sobre la base de datos, hay que decir que se usa MySQL, con enteros de 32 bits[#] para representar la latitud y la longitud de los nodos. Todas las coordenadas están representadas en EPSG:4326, es decir, latitud-longitud referidas a WGS84. Se puede consultar el modelo de la base de datos en la página Database/Model [2] del wiki.

Esto proporciona una precisión de 3 diezmilésimas de segundo de grado, que equivalen a unos 9.3 milímetros en el ecuador).

Rails también maneja la interfaz web; de hecho, Rails funciona como un servidor web (es decir, no hace falta instalar un servidor web como Apache). La versión actual del Rails Port incluye las páginas web necesarias para que los usuarios se registren, cambien sus preferencias, y puedan acceder al editor Potlach desde su navegador web.

Por último, y lo más importante: el Rails Port proporciona la API para acceder a la información geográfica. Se podría definir la API de OSM como una interpretación propia (y eficiente) del protocolo WFS-T. Toda la API está basada en la filosofía REST, y trabaja con un XML muy similar al utilizado en los ejemplos de las primitivas de datos.

Mediante la API podemos, por ejemplo, solicitar información sobre un determinado nodo, way o relación, dada su ID:

```
http://api.openstreetmap.org/api/0.5/node/30894545

<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.5" generator="OpenStreetMap server">
  <node id="30894545" lat="41.9797918" lon="2.8201551"
user="Randbewohner" visible="true"
timestamp="2008-09-21T10:19:07+01:00">
    <tag k="name:es" v="Gerona"/>
    <tag k="name" v="Girona"/>
    <tag k="place" v="town"/>
    [...]
  </node>
</osm>
```

Se puede ver que la API añade ciertos atributos: el último usuario en modificar la entidad, la visibilidad (los objetos históricos no son visibles a priori), y la fecha de la última modificación. Este ejemplo es para la API 0.5 - la API 0.6, que todavía no ha sido puesta en funcionamiento, incluye soporte para transacciones. En lugar de especificar el usuario y fecha de la última modificación, se incluirá el identificador único de la última transacción que modificó el objeto.

Otra operación importante sobre la API es la petición de todos los datos dentro de un *bounding box* (especificado como oeste,sur,este,norte), por ejemplo:

```
http://api.openstreetmap.org/api/0.5/map?
bbox=2.815,41.977,2.823,41.983

<?xml version="1.0" encoding="UTF-8"?>
<osm version="0.5" generator="OpenStreetMap server">
  <bounds minlat="41.977" minlon="2.815" maxlat="41.983"
maxlon="2.823"/>
  <node id="250105526" lat="41.978049" lon="2.8176524"
```

```
user="gpesquero" visible="true"
timestamp="2009-01-29T23:30:33+00:00"/>
  <node id="250120078" lat="41.9783097" lon="2.8153464"
user="SteveC" visible="true"
timestamp="2008-03-02T17:13:02+00:00"/>
  <node id="30894545" lat="41.9797918" lon="2.8201551"
user="Randbewohner" visible="true"
timestamp="2008-09-21T10:19:07+01:00">
    <tag k="name:es" v="Gerona"/>
    <tag k="name" v="Girona"/>
    <tag k="place" v="town"/>
    [...]
  </node>
  [...]
  <way id="23153318" visible="true"
timestamp="2008-03-02T18:02:44+00:00" user="ivansanchez">
    <nd ref="250133843"/>
    <nd ref="250133844"/>
    <tag k="layer" v="-1"/>
    <tag k="oneway" v="true"/>
    <tag k="tunnel" v="true"/>
    <tag k="created_by" v="JOSM"/>
    <tag k="name" v="Carrer de Santa Eugènia"/>
    <tag k="highway" v="residential"/>
  </way>
  [...]
</osm>
```

Para editar datos a la API 0.5, se llama a la misma URL que para recibir la información de un objeto (pero usando el método HTTP PUT en vez de HTTP GET), enviando a la API el XML correspondiente a la nueva versión de un objeto. Para crear un nuevo objeto, se realiza la misma operación, pero sustituyendo la ID del objeto por "create" - la API nos devolverá el identificador asignado para ese nuevo objeto.

Para editar o añadir datos en la API 0.6, únicamente se encapsula el XML de todos los objetos a cambiar en una única petición, denominada un *changeset*.

Hay otras operaciones para pedir datos a la API, por ejemplo, solicitar el histórico de un objeto, solicitar los datos de todos los miembros de una relación, o borrar un objeto. Todo ello está documentado en las páginas OSM_Protocol_Version_0.5 [3] y OSM_Protocol_Version_0.6 [4] del wiki.

EL EDITOR: POTLACH

Un paso por encima de la API están los editores. El más accesible es *Potlach*, un editor basado en tecnología Flash, que se ejecuta directamente en el navegador web; sólo hay que hacer zoom al área a editar, y pulsar sobre la pestaña "edit" en la parte superior de la página. Potlach está diseñado para ser fácil de utilizar, y para que sus usuarios no se tengan que preocupar en exceso de conocer cómo funciona la topología o el modelo de datos de OSM. Para ayudar al etiquetado, muestra una serie de opciones que generan el conjunto de etiquetas más comunmente utilizado.

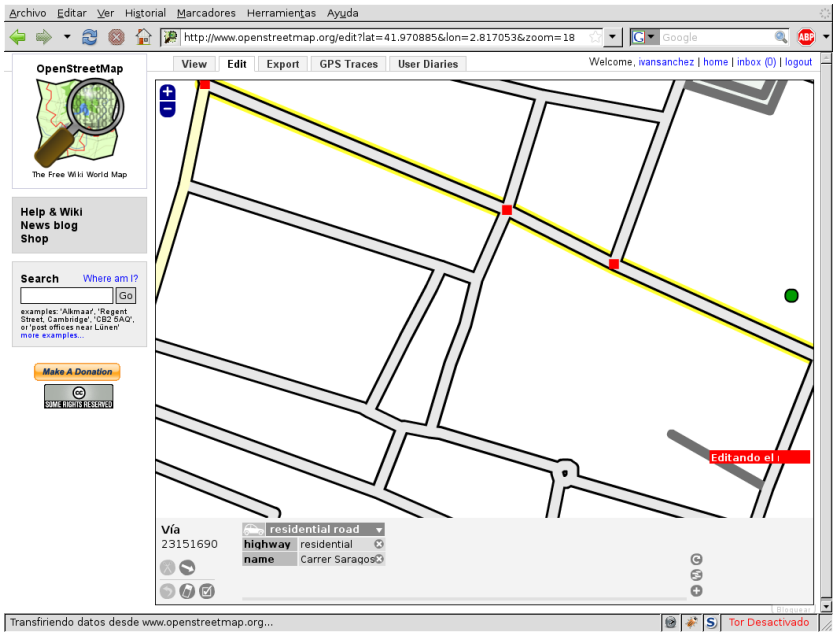

Figura 1: Editor Potlach

Potlach utiliza la API para transmitir al servidor los cambios que el usuario va realizando, de manera constante.

EL EDITOR: JOSM

Potlach está bien para hacer pequeños cambios, pero para operaciones más grandes se queda un poco corto. Otro de los editores más usados es JOSM, acrónimo de "Java OpenStreetMap editor". Es más complicado de usar que Potlach para la mayoría de los usuarios, pero es mucho más potente.

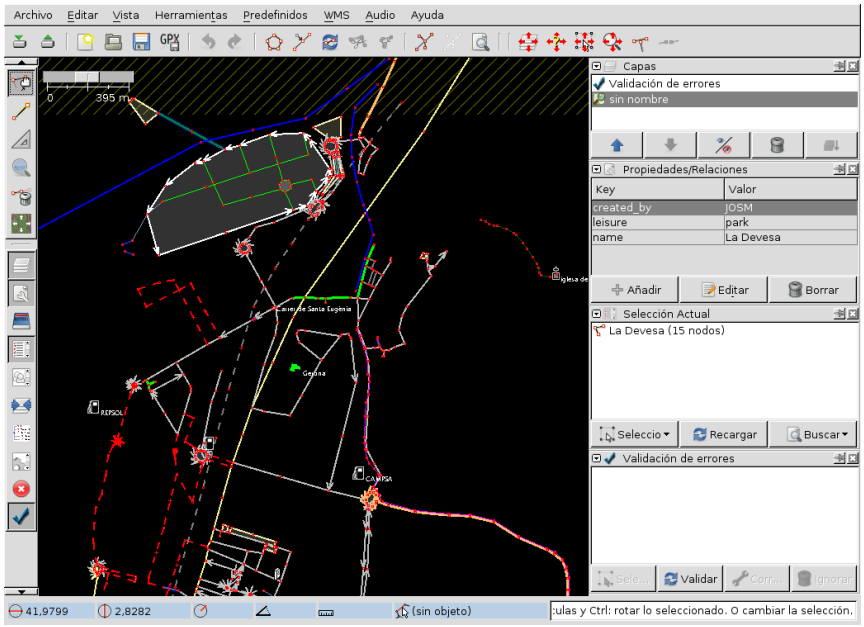

Figura 2: Editor JOSM

JOSM es un programa que muy bien podría considerarse un SIG de escritorio, dadas todas sus funcionalidades. Permite tener un control muy fino sobre las

primitivas geométricas, y puede trabajar con una gran cantidad de datos en un mismo momento; el único contrapunto es que el usuario tiene que indicar cuándo enviar los cambios. Soporta varias proyecciones, es cliente WMS (implementando una caché de teselas en local), soporta GPX con enlace a fotografías y/o audio georreferenciado, controla colisiones entre versiones de datos, y valida (e incluso corrige) diversos errores topológicos y semánticos.

LA EXPORTACIÓN: EL "PLANET"

Según los usuarios de OSM van editando, es necesario tomar toda la base de datos y procesarla para poder convertirla a gráficos, y poder generar los mapas propiamente dichos.

Con el objetivo de disminuir, en la medida de lo posible, la carga sobre el servidor de BDD, lo que se viene haciendo desde el año 2006 es extraer volcados completos de la BDD, para poder trabajar sobre copias (de sólo lectura) de la BDD, en vez de directamente sobre la BDD. Uno cualquiera de estos volcados se denomina *planet*, porque cubren el planeta por completo. Un *planet* típico es un fichero XML (con el formato visto anteriormente) de unos 100 GB de tamaño (comprimido a 4 GB mediante bzip2).

Habitualmente se realiza un volcado semanal todos los miércoles y, desde no hace demasiado tiempo, volcados diferenciales. Mientras que el volcado semanal es completo, un volcado diferencial sólo contiene los objetos que han cambiado cada día, cada hora o cada minuto.

Los volcados diferenciales no sólo sirven para poder obtener el estado de la BDD en un momento dado, sino para hacer diversos análisis sobre cuándo fueron editados los objetos de una determinada área.

Para elegir un sitio de donde descargar un planet, véase la página planet.osm del wiki.

EL RENDERIZADOR: MAPNIK

Mapnik [5] es un software desarrollado en Python y C++, diseñado para renderizar mapas de todo tipo. Dado que soporta Shapefiles, geoTIFFs, bases de datos de PostGIS y cualquier formato soportado por GDAL, es utilizado en muchos otros proyectos aparte de OSM. OSM lo utiliza para generar las teselas que después se verán en la portada de www.openstreetmap.org.

Las imágenes de Mapnik se actualizan cada semana, correspondiendo con cada volcado semanal del planet. El planet se importa a una base de datos PostGIS, que se procesa mediante mapnik y una hoja de estilos para mapnik, específica para OSM. OSM utiliza Mapnik para generar *quadtiles* proyectadas en EPSG:900913. Esto permite usar OpenLayers para mostrar estas teselas de una manera compatible con teselas de otros proveedores de mapas.

Hasta hace no mucho, las teselas ya renderizadas se almacenaban en disco, y se utilizaba Apache para servirlas a un OpenLayers, y una pequeña aplicación con MySQL para saber qué teselas han sido visitadas y deberían de ser actualizadas la siguiente semana. El método resultó demostrar bastante ineficiente cuando hay muchas peticiones, así que se terminó desarrollando un módulo de apache, mod_tile, que es capaz de marcar más rápidamente las teselas que hay que regenerar.

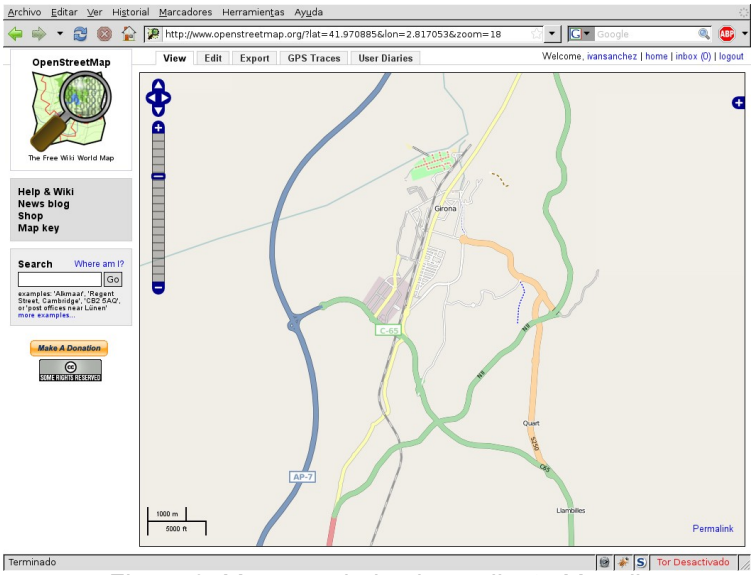

Figura 3: Mapa renderizado mediante Mapnik

EL RENDERIZADOR: OSMARENDER

Osmarender empezó siendo un conjunto de scripts en XSLT para transformar el XML de OSM en SVG, utilizando únicamente hojas de estilo de XML. La última versión, denominada OR/P u "OsmaRender/Perl", es una serie de scripts en Perl que realizan las transformaciones XSLT más eficientemente.

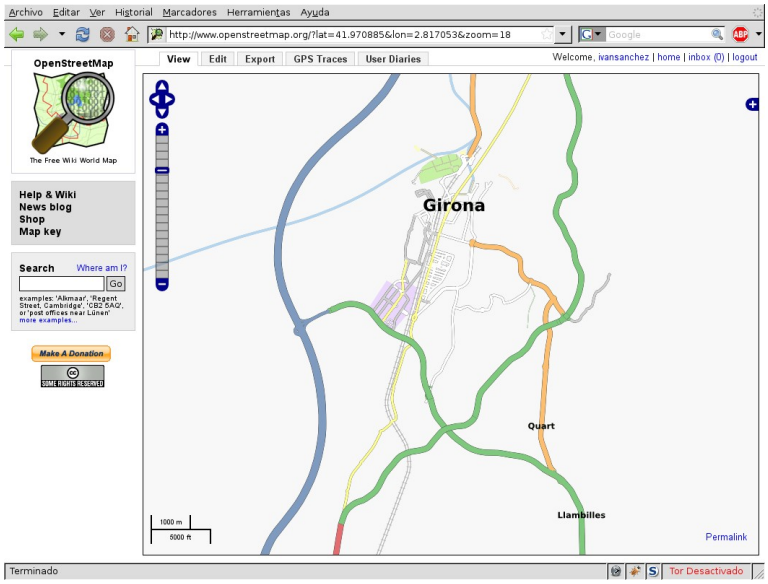

Figura 4: Mapa renderizado mediante Osmarender

Osmarender fue desarrollado con el objeto de poder en un mapa con Openlayers los cambios que se van realizando sobre los datos geográficos, sin tener que esperar a la siguiente semana. Osmarender ha sido el inspirador de varios desarrollos interesantes:

- Tiles@Home es una plataforma de renderizado distribuída - cientos de ordenadores ejecutan osmarender en paralelo, pidiendo datos a la API, y subiendo las teselas resultantes a un servidor centralizado.

- Como la carga sobre la base de datos era bastante grande, se han desarrollado réplicas de la base de datos, y tecnologías como XAPI ("OSM eXtended API"), ROMA ("Read-Only Map Api") y TRAPI ("Tile Read-only map API") para que los clientes de t@h puedan obtener la información geográfica necesaria para renderizar una determinada tesela.

Una característica curiosa de osmarender es que sólo renderiza teselas de zoom 12; las teselas de un zoom más amplio se generan a partir de unir y re-escalar las teselas de un nivel de zoom mayor. Esto hace que, al ver un zoom muy amplio, no se vean entidades características de ese nivel de zoom (fronteras, carreteras principales), sino la densidad de datos.

En el mapa de www.openstreetmap.org, se puede cambiar entre Mapnik y Osmarender pulsando sobre el "+" de la esquina superior derecha. La renderización de Osmarender ha sido también la usada para el mapa de asistentes a las III jornadas de SIG libre: http://www.sigte.udg.es/jornadassiglibre/index.php?page=participantes

ENRUTADO

Dado que los datos de OpenStreetMap tienen una topología implícita, y dado que cualquier persona puede descargarse un *planet*, una aplicación bastante obvia es el cálculo de rutas.

Actualmente existe una multitud de software diseñado para el cálculo de rutas en base a los datos de OSM - hay una lista completa en la página Routing del wiki. Sin embargo, caben destacar un par de aplicaciones:

OpenRouteService.org es un sitio web que implementa el estándar OpenLS para el cálculo de rutas, un gazeteer para la búsqueda de topónimos, es capaz de calcular rutas para distintos modos de transporte, y se pueden definir áreas de exclusión. Otro sitio web similar es YourNavigation.org, que permite obtener el perfil de elevación de la ruta integrando OSM y SRTM. YourNavigation.org usa Gosmore [6] como motor de búsqueda de rutas.

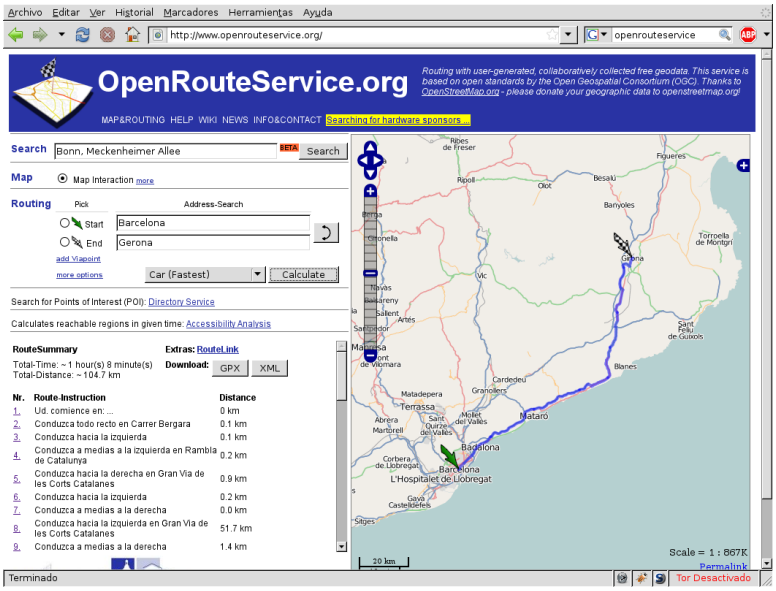

Figura 5: Entorno de cálculo de rutas OpenRouteService

Otro objetivo de OSM son los dispositivos móviles. Actualmente se puede usar la cartografía de OSM en prácticamente cualquier GPS de Garmin (véase OSM Map on Garmin [7] en el wiki). Una de las ventajas de OSM es que se pueden elegir qué parte de la información geográfica trasladar al dispositivo móvil; por ejemplo, hay mapas para Garmin específicos para ciclistas (indican carriles bici y obstáculos para ciclistas), o para trabajo de campo (indican dónde están los errores o datos incompletos en OSM).

Para otros dispositivos móviles se ha desarrollado software específico: MaemoMapper para las tabletas n800 de Nokia; AndNav2 para teléfonos móviles con Android; y se ha adaptado Navit para teléfonos móviles con OpenMoko.

Figura 6: GPS Garmin

Figura 7: Maemo Mapper

Por supuesto, existen herramientas de conversión a shapefile, PostGIS y un largo etcétera de formatos estándar, con lo que se podrían usar los datos con otros motores de búsqueda de caminos.

CONTROL DE CAMBIOS

A pesar de que la API 0.6 y sus transacciones no están todavía listas, el hecho de tener volcados diferenciales de la base de datos permite hacer un análisis y control de cambios bastante preciso. La idea más básica es tomar los volcados diferenciales, aplicar una conversión de formato, y visualizar los resultados. Ese es el modo de funcionamiento de OSM Aware, una de las primeras herramientas para la revisión de cambios.

Aparte de la funcionalidad de la API 0.5 y API 0.6, una herramienta desarrollada por ITO[1] llamada OSM Mapper ha ganado mucha popularidad. Esta herramienta permite ver gráficamente la "edad" de la cartografía de OSM, hacer diversos análisis sencillos sobre los datos, y ofrece la posibilidad de recibir, mediante un feed RSS, avisos sobre los cambios de la cartografía de cualquier zona específica.

1 ITO es una empresa británica que ofrece servicios de SIG a empresas de logística y transportes.

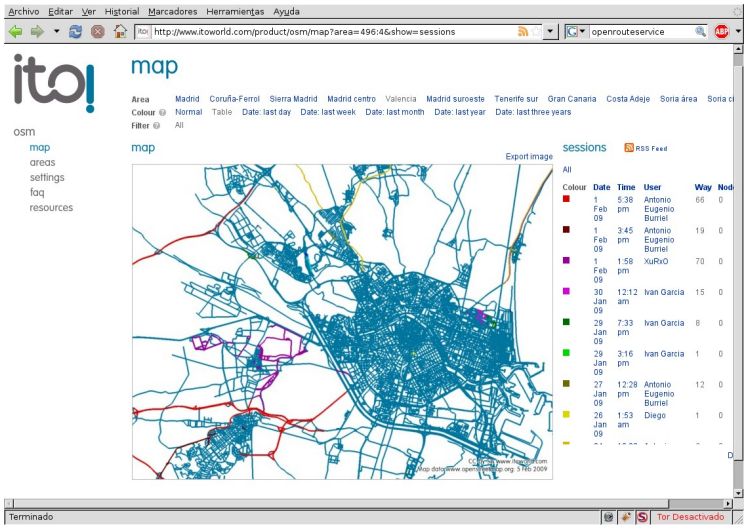

Figura 8: OSM Mapper

OPENSTREETBUGS

Ante las quejas de algunos usuarios sobre lo difícil que es hacer algunas correcciones avanzadas a la cartografía, o sencillamente la controversia sobre cual es la manera correcta de etiquetar datos incorrectos para corregirlos después, surgió la idea de OpenStreetBugs.

Lo único que permite hacer OpenStreetBugs es la posibilidad de añadir un bug[1] en cualquier lugar, siguiendo la idea de otros sistemas de "bug tracking" usados por la comunidad informática como son bugzilla o TRAC, permitiendo copiar las técnicas y métricas de control de calidad ya existentes. Una vez añadido el bug, cualquier usuario es capaz de ver qué zonas necesitan una revisión, puede corregir esos datos, y "cerrar" el bug una vez hechos los cambios necesarios.

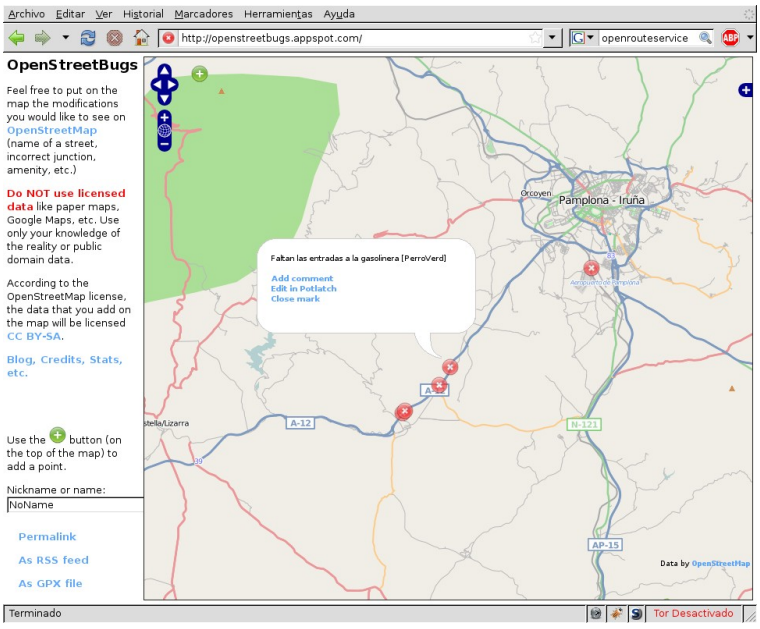

Figura 9: OpenStreetBugs

1 Bug es el nombre que se le da a un error informático, sobre todo a errores en el software.

PERSONALIZACIÓN

> *Innovations in web mapping are not going to be about finding new interesting places to stick pushpins. They're going to be about new ways to visualize the maps, new ways to drawing maps.*
> *Richard Fairhurst, durante el State of the Map 2007*

Se ha hablado mucho sobre los denominados "mashups": aplicaciones más o menos sencillas, que toman un mapa web como base. Sin embargo, llega un momento en el que el poder poner "chinchetas" (y otro tipo de marcadores) sobre un mapa web se queda corto para según qué aplicación.

Con los datos de OpenStreetMap y la tecnología de OpenLayers se pueden desarrollar estos "mashups" de manera igual de sencilla que con otros proveedores de mapas web. Sin embargo, dado que OpenStreetMap permite un acceso a los vectores que componen ese mapa web, el potencial para dibujar nuevas maneras de ver el mapa se multiplica.

Uno de los primeros ejemplos en aparecer fue el Cycle Map o mapa para ciclistas. No es más que una hoja de estilos de Mapnik, que da más peso a los carriles bici y rutas ciclistas. Esto no sería posible construirlo en base a los mapas web existentes, dado que no se pueden ocultar los datos que no interesan (en este caso, las carreteras para coches). Ya hay proyectos similares para mapas de vías de tren, autobuses y transporte público, rutas a pie o mapas para piragüistas.

Recientemente, CloudMade ha sacado a la luz una *beta* de su editor de estilos. El cambiar el estilo de dibujado de los mapas, que antes estaba limitado al software SIG profesional, ahora se puede hacer como parte de un mashup.

CloudMade es una empresa fundada por Steve Coast, creador de OpenStreetMap, y ofrece servicios de valor añadido sobre OpenStreetMap.

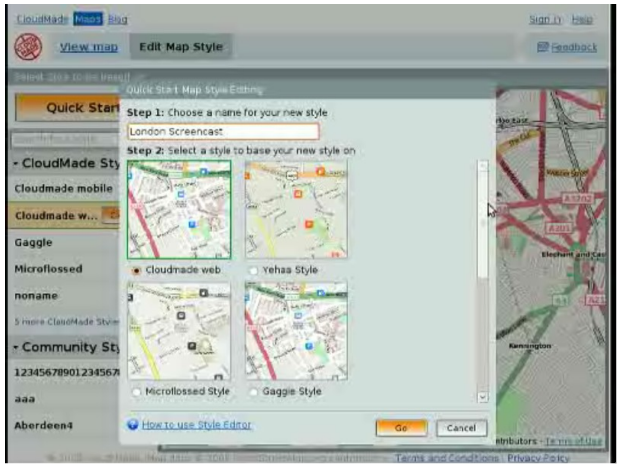

Figura 10: Editor de estilos de Cloudmade

Un ejemplo muy curioso de nuevas maneras de visualizar la información geográfica es City Murmur: un sistema que recolecta de muchos medios las noticias relativas a una ciudad, y dibuja el callejero dependiendo de la densidad o tipo de noticias que ocurren en cada calle.

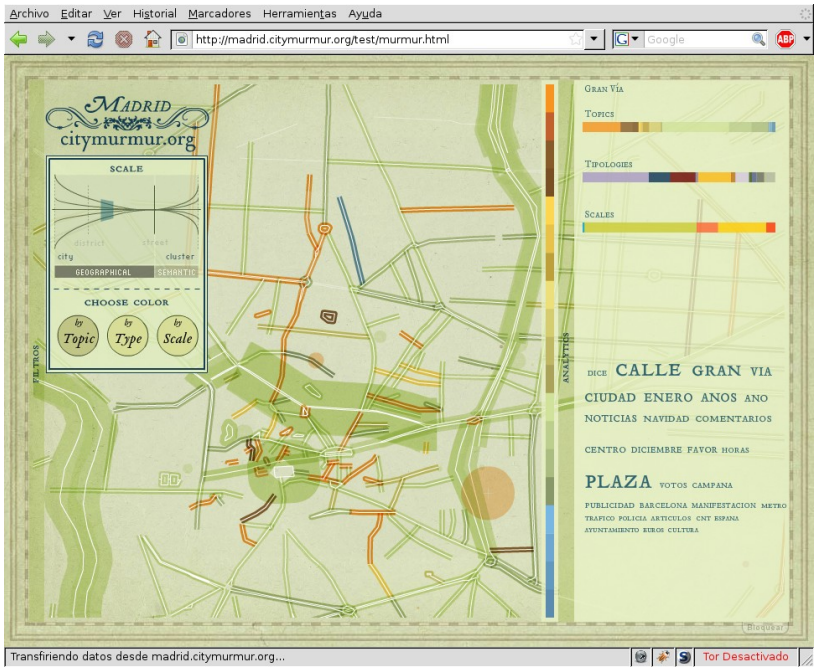

Figura 11: City Murmur

CONCLUSIÓN

Aquí se han visto los principales componentes software de la plataforma de OpenStreetMap y, de forma no exhaustiva, algunas de las aplicaciones construidas a partir de sus datos. A pesar de que apenas se usan estándares OGC, OpenStreetMap ha conseguido formar todo un ecosistema de software a su alrededor. Esto es, entre otras razones, gracias a la naturaleza open-source de su información y metainformación geográfica, y gracias a la simplicidad del formato de datos.

REFERENCIAS

- [1] Diversos Autores. *OSM Map Features* [en línia], <http://wiki.openstreetmap.org/wiki/Map_Features>, enero de 2009 [consulta febrero de 2009]
- [2] Diversos Autores. Database Model [en línia] <http://wiki.openstreetmap.org/wiki/Database/Model>, junio de 2008 [consulta febrero de 2009]
- [3] Diversos Autores. OSM Protocol Version 0.5 [en línia], <http://wiki.openstreetmap.org/wiki/OSM_Protocol_Version_0.5>, febrero de 2009 [consulta febrero de 2009]
- [4] Diversos Autores. OSM Protocol Version 0.6 [en línia], <http://wiki.openstreetmap.org/wiki/OSM_Protocol_Version_0.6>, febrero de 2009 [consulta febrero de 2009]
- [5] Mapnik [en linia], versión 0.5.1, Programa iformático <http://www.mapnik.org/>
- [6] Diversos Autores. Gosmore, [en línia], <http://wiki.openstreetmap.org/wiki/Gosmore>, febrero de 2009 [consulta febrero de 2009]
- [7] Diversos Autores. OSM Map On Garmin, [en línia], <http://wiki.openstreetmap.org/wiki/OSM_Map_On_Garmin>, febrero de 2009 [consulta febrero de 2009]