Panorama SIG Libre Documentation

Publicación 1.0

OSGeo-es

04 de February de 2017

Índice general

CAPÍTULO 1

Introducción

A la hora de abordar este proyecto lo primero que nos planteamos es cómo lo queríamos hacer, es decir, qué requerimientos tendríamos que cumplir. De ahí salió la lista siguiente:

1.1 ¿Qué requerimientos tenemos?

- Toda la discusión debía ser en abierto
- Todo el contenido en abierto y versionado
- A ser posible autopublicado
- Tener un seguimiento semanal para tener claras las tareas y quién está haciendo qué.

1.2 ¿Cómo lo hacemos?

Tras esto la forma en la que empezamos a montar toda esta vorágine, fue mediante la lista de correo y la wiki de OSGeo. La lista fue el punto de partida, fue el sitio donde nos comprometimos a colaborar y a realizar esta presentación.

1.3 ¿Cómo nos organizamos?

A partir de aquí surgió la forma en la que nos debíamos organizar, mediante una reunión semanal a través de irc, para evaluar el estado de consecución de los objetivos marcados. Teniendo en cuenta que cada una de estas reuniones tienen su acta en la wiki de OSGeo, además del *log* correspondiente para aquél que no haya podido asistir.

Todas las discusiones relacionadas con el contenido de cada sección las llevamos a cabo mediante la generación de tickets (issues) en GitHub, de forma que se creasen hilos de conversación en paralelo sobre los principales temas a discutir y no enviáramos ruido a la lista. Aunque a pesar de tener estos hilos, en la lista se seguían notificando las cosas más importantes.

1.4 ¿Cómo contribuimos los contenidos?

Disponemos de un repositorio en el que se alojan todos los archivos fuentes del artículo, así como todos los recursos relacionados con la presentación.

Para ello tenemos habilitadas en el repositorio dos ramas:

- `gh-pages`: Rama para el desarrollo de la presentación

- `paper`: Rama para el desarrollo del artículo

La rama gh-pages, que empleamos para el desarrollo de la presentación, se ha fabricado mediante RevealJS, un framework javascript para el desarrollo de presentaciones dinámicas basadas en HTML5 Y CSS3.

La rama paper, que empleamos para el desarrollo del artículo, se ha fabricado mediante Sphinx, un generador de documentación escrito en Python, que hace que se genere la estructura por defecto de un proyecto base para la generación de un artículo, tal y como el que aquí se presenta.

Para contribuir y hacer crecer esta documentación únicamente tendremos que realizar nuestro *fork* del repositorio y dependiendo de lo que queramos modificar, tendremos que seleccionar una u otra rama. Para trabajar con la presentación, simplemente tendremos que modificar el fichero `index.html` que se encuentra en la rama gh-pages. En cambio para trabajar con el artículo, tendremos que modificar el archivo `index.rst` dentro de la carpeta de la sección con la que queremos colaborar.

En cuanto tengamos los cambios listos para subir, debemos realizar el correspondiente *commit* y sucesivo *pull request*. Una vez éste esté aceptado, se procederá a la actualización automática de la documentación alojada en la *web*.

1.5 ¿Cómo publicamos los contenidos?

La publicación del contenido de la presentación se realiza de manera automática mediante el uso de GitHub Pages.

La publicación del contenido del artículo se realiza de manera automática mediante el uso de una herramienta denominada Read the Docs, haciéndola fácil de encontrar y ofreciendo opción de búsqueda. Esta herramienta nos permite subir la documentación generada con Sphinx mediante la dirección al repositorio de Git. La documentación será compilada cada vez que se realice un commit, de forma que tendremos siempre la última versión de nuestra documentación disponible en la web.

1.6 Sobre la tabla de información de productos

En todas las secciones de este trabajo se utiliza una tabla de descripción de productos que utiliza un juego de campos común. A continuación se describen qué significan esos campos.

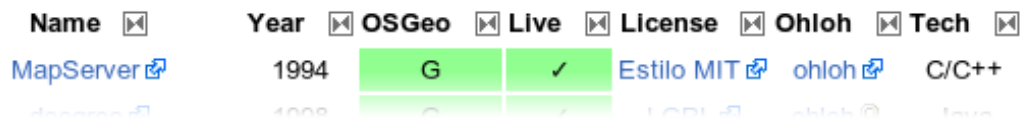

Figura 1.1: Encabezados de las tablas de productos

name: Nombre del producto

year: Año de aparición del producto como *Software* Libre

OSGeo: Indica si el producto forma parte de la fundación OSGeo, especificando si el producto está Graduado o en Incubación.

Live: Indica si el producto forma parte del *Live DVD* que empaqueta el proyecto OSGeo Live.

License: Se especifica la licencia con la que se distribuye el producto

Ohloh: Ofrece un enlace, si existe, a la página del producto en la web de estadísticas de proyecso de *Software* Libre Ohloh.net

Tech: Indica la tecnología principal con la que se ha desarrollado el producto.

CAPÍTULO 2

Servidores

2.1 Autores

- Alejandro Díaz @alediator
- María Arias de Reyna @delawen
- Jorge Sanz @xurxosanz

2.2 Introducción

En esta sección se va a abordar el amplio conjunto de proyectos correspondientes a la capa intermedia de cualquier sistema de información, también conocida como *middleware*. En esta sección se cubre por tanto cualquier aplicación que se ejecuta en un servidor y que tiene como objetivo proporcionar uno o más servicios que serán consumidos por clientes, independientemente de si estos son otras aplicaciones de servidor, clientes *web*, clientes de escritorio o dispositivos móviles.

¿Qué tipos de servicios actualmente se ofrecen relacionados con la Información Geográfica? Una agrupación funcional podría ofrecernos las siguientes categorías:

- **Servidores de mapas**: Encargados de renderizar datos tanto vectoriales como *raster* en diferentes estilos y proyecciones cartográficas, en general cumpliendo el estándar WMS de OGC
- **Servidores de teselas**: Una variante del anterior es aquellos servidores que ofrecen la cartografía renderizada únicamente en juegos de teselas con un número limitado de resoluciones, casi siempre además empleando sistemas de almacenamiento intermedio (*caches*). En este caso hay dos estándares en el sector, el estándar TMS y el WMTS de OGC
- **Servidores de datos brutos**: Al contrario que las categorías anteriores, estos servidores ofrecen la cartografía vectorial o *raster* en formatos que deben luego ser procesados por los clientes para la tarea para la que estén desarrollados, tanto si es para su análisis como su visualización. Los estándares de comunicación son WFS y WCS respectivamente para datos vectoriales y *raster*. En esta categoría se podrían incluir también a los servidores relacionados con la publicación de datos de sensores, en todo el abanico de estándares OGC englobados en lo que se conoce como Sensor Web Enablement
- **Servidores de metadatos**: Estos servidores implementan estándares de descubrimiento de datos como CSW. Mediante estos servidores usuarios y otros componentes pueden encontrar juegos de datos y otros servicios mediante protocolos estandarizados. Estos servidores son el corazón de las Infraestructuras de Datos Espaciales.
- **Servidores de geoprocesos**: Estos servidores exponen operaciones de análisis, que pueden partir de datos directamente disponibles en el servidor o bien acceder a otros servidores de datos brutos para encadenar servicios que realicen flujos de geoprocesamiento de todo tipo. El estándar de OGC para geoprocesamiento es el WPS.

Es habitual que un producto de *Software* Libre cubra más de una funcionalidad de las definidas en la categorización anterior, especialmente con los proyectos más veteranos. La interoperabilidad es otra de las características del *Software* Libre geoespacial y es por ello que la mayoría de los productos suelen intentar implementar aquellos estándares que afectan a su área de interés.

En las siguiente tablas se presentan las características principales de los productos revisados en el contexto de este trabajo así como los principales estándares que implementan. Se puede consultar más información sobre la estructura de la primera tabla en la *introducción*.

Name	Year	OSGeo	Live	License	Ohloh	Tech
MapServer	1994	G	✓	Estilo MIT	ohloh	C/C++
deegree	1998	G	✓	LGPL	ohloh	Java
GeoServer	2001	G	✓	GPL2	ohloh	Java
GeoNetwork	2003	G	✓	GPL 2	ohloh	Java
52°north SOS	2004?	☹	✓	GPL	ohloh	Java
MapGuide OpenSource	2005	G	☹	LPGL	ohloh	C++
PyWPS	2006	☹	☹	GPL 2	ohloh [1]	Python
GeoWebCache	2007	☹	✓	LGPL	ohloh	Java
TileCache	2007	☹	☹	BSD	ohloh	Python
52°north WPS	2008?	☹	✓	GPL 2	ohloh	Java
MapProxy	2010	☹	✓	Apache	ohloh	Python
PyCSW	2010	I	✓	MIT	ohloh	Python
QGIS Server	2010	G[2]	✓	GPL	ohloh	C++
TileStache	2010	☹	☹	BSD	ohloh	Python
ZOO Project	2010	I	✓	MIT/X11	ohloh	C/C++
EOxServer	2011	☹	✓	MIT Style	ohloh	Python
TileStream	2011	☹	☹	BSD?	☹	NodeJS

Figura 2.1: Información general sobre servidores

Importante: Se puede consultar la versión más reciente, así como los enlaces asociados y anotaciones en la sección de Servidores del wiki de OSGeo.

2.3 Software

En esta sección se describirá brevemente cada uno de los productos evaluados en esta comparativa. El orden de aparición es cronológico, empezando por los proyectos más veteranos.

MapServer: Probablemente el servidor de mapas más usado en el mundo sigue desarrollándose sin pausa, incorporando en los últimos años nuevos subproyectos como MapCache y TinyOWS. Su configuración sigue siendo mediante ficheros de texto aunque aparecen nuevas interfaces para facilitar su edición como MapManager. Sigue destacando de este servidor su ligereza y facilidad de despliegue (si no se necesita soporte para formatos privativos).

deegree: Este servidor de mapas con una comunidad principalmente alemana soporta una gran cantidad de estándares. Este servidor se caracterizó durante mucho tiempo por su ardua configuración, cosa que está cam-

Nombre	WMS	WFS	WFS-T	WCS	WMTS	TMS	WPS	SOS	CSW
MapServer	✓	✓	✓[3]	✓	✓[4]	✓[4]	☹	✓	☹
deegree	✓	✓	✓	✓	✓	☹	✓	☹	✓
GeoServer	✓	✓	✓	✓	✓ [5]	✓ [5]	✓	☹	✓
GeoNetworK	☹	☹	☹	☹	☹	☹	☹	☹	✓
52north SOS	☹	☹	☹	☹	☹	☹	☹	✓	☹
MapGuide OpenSource	✓	✓	☹	☹	☹	☹	☹	☹	☹
PyWPS	☹	☹	☹	☹	☹	☹	✓	☹	☹
GeoWebCache	✓	☹	☹	☹	✓	✓	☹	☹	☹
TileCache	☹	☹	☹	☹	✓	✓	☹	☹	☹
52north WPS	☹	☹	☹	☹	☹	☹	✓	☹	☹
MapProxy	✓	☹	☹	☹	✓	✓	☹	☹	☹
PyCSW	☹	☹	☹	☹	☹	☹	☹	☹	✓
QGIS Server	✓	✓	☹	☹	☹	☹	☹	☹	☹
TileStache	☹	☹	☹	☹	☹	✓	☹	☹	☹
Zoo Project	☹	☹	☹	☹	☹	☹	✓	☹	☹
EOxServer	✓[6]	☹	☹	✓[6]	☹	☹	☹	☹	☹
TileStream	☹	☹	☹	☹	☹	✓	☹	☹	☹

Figura 2.2: Implementación de estándares OGC

biando gracias al esfuerzo puesto en desarrollar una consola *web* para los servicios y una documentación mucho más detallada.

GeoServer: GeoServer es un proyecto ejemplar por su activa comunidad, frecuentes actualizaciones e innovaciones más allá de la implementación de gran parte de los estándares OGC existentes. Sus extensiones al estándar SLD, el soporte de estilos CSS, el buen soporte para multitud de orígenes de datos y su amigable interfaz lo convierten en un servidor de mapas ampliamente utilizado en todo tipo de contextos, especialmente en grandes instalaciones.

GeoNetwork: El servidor de metadatos más utilizado en el sector sigue ofreciendo actualizaciones y mejoras constantes gracias al apoyo tanto de las empresas que lo desarrollan como de las administraciones públicas que lo utilizan, haciendo de este servidor la primera opción a la hora e implementar un nodo de una Infraestructura de Datos Espaciales, con múltiples referencias y casos de éxito.

52ºnorth SOS: Probablemente la implementación más relevante del principal estándar OGC sobre sensores. No dispone de una interfaz de usuario salvo una sencilla consola para ejecutar consultas de ejemplo. La última versión (4.0) implementa la versión más reciente del estándar SOS, la 2.0.

MapGuide: El servidor liberado por AutoDesk siempre se ha vinculado a entornos Windows (aunque aparentemente puede funcionar en Linux) y su falta de soporte para estándares OGC parece que le ha dejado atrás en el panorama de *Software* Libre. Pese a todo, es el único servidor de mapas que integra completos visores y se puede considerar una solución completa, aunque en general se prefieran entornos desacoplados donde el uso de estándares facilite desarrollar clientes personalizados.

PyWPS: Este pequeño servidor WPS se ha reescrito durante el último año y pretende ofrecer una interfaz ligera en la que desarrollar geoprocesos en el popular lenguaje de programación Python. La creciente comunidad geoespacial alrededor de este lenguaje de programación y por lo tanto, una cada vez mayor oferta de componentes que ofrecen funcionalidad pueden hacer este producto interesante.

GeoWebCache: GeoWebCache nació como un proyecto dentro del programa de becas de verano de Google para estudiantes (*Google Summer of Code*). El objetivo era cubrir la necesidad en GeoServer de un servidor de teselas que permita pregenerar y acelerar la cartografía servida por este producto. Con el tiempo ha ido creciendo en funcionalidad y se puede considerar un producto independiente, aunque en general se

utilice conjuntamente con GeoServer. Al igual que GeoServer, destaca por su cómoda interfaz de usuario, capacidad para limitar en disco las *caches*, generación y borrado de las mismas, etc.

TileCache: Durante mucho tiempo TileCache fue la implementación de referencia del estándar *de facto* TMS. El proyecto lleva ya bastante tiempo estancado sin prácticamente actividad por lo que se puede considerar en periodo de *senectud* y no sería aconsejable instalarlo en nuevos proyectos.

52ºnorth WPS: Este servidor de procesos dispone de una sencilla interfaz y conectores para implementar geoprocesos en el lenguaje de programación R, usar procesos del GIS de escritorio GRASS, o de SEXTANTE.

MapProxy: MapProxy es un servidor de teselas que dispone de algunas características interesantes. Al igual que GeoWebCache permite responder a cualquier petición WMS, no solo a las correspondientes con teselas de las *caches* definidas. Además dispone de varios mecanismos de almacenamiento de teselas más allá del uso del sistema de ficheros. Además puede usarse como un servidor de mapas estándar para ficheros de configuración de la biblioteca de renderización Mapnik. Su configuración mediante sencillos ficheros de texto y las herramientas de ayuda a configuración y despliegue de las mismas lo convierten en un producto ciertamente versátil.

PyCSW: Servidor de metadatos desarollado en el lenguaje Python, diseñado para ser tanto una librería como un servidor independiente, por lo que se puede integrar en otros desarrollos o desplegar como un servidor. Es un proyecto pequeño, reciente y sin interfaz de usuario. Se puede arrancar el servidor importando una carpeta de ficheros XML de metadatos y utiliza una base de datos, implementando el estándar CSW (entre otros) para acceso a los metadatos y los estándares OGC más importantes para recopilar metadatos de servicios existentes. Este servidor se ha integrado en productos más grandes como son el servidor de *Open Data* CKAN y el portal de datos geográficos GeoNode.

QGIS Server: Este servidor nace en el contexto del proyecto QGIS como una forma sencilla de exponer proyectos de este *software* de escritorio a través de la red y mediante los estándares WMS y WFS. Técnicamente es un producto escrito en C++ que se despliega como un servidor FastCGI/CGI (como MapServer) y que proporciona un plugin para el software de escritorio de tal forma que se facilita enormemente la publicación de datos geográficos desde este producto.

TileStache: Servidor de teselas heredero del veterano TileCache, desarrollado para mejorar las funcionalidades de éste, aunque no ha tenido mucha actividad en los últimos meses. Sus principales características son que está orientado a renderizar mapas a partir de ficheros de configuración de Mapnik, generar teselas vectoriales en formato GeoJSON, puede almacenar las *caches* no solo en disco sino también en memoria o en Amazon S3.

ZOO Project: Este proyecto consiste en un núcleo escrito en C++ que permite ejecutar los procesos en diferentes lenguajes y para cualquier propósito, un conjunto de servicios a modo de ejemplo que escritos en diferentes lenguajes de programación y utilizando librerías bien conocidas realizan diferentes procesos como el cálculo de rutas, conversión de datos, etc. Finalmente el proyecto proporciona una interfaz de programación (API) escrita en JavaScript para ejecutar procesos WMS desde el servidor. Este proyecto se integra muy bien con MapServer para generar servicios WMS dinámicos a partir del resultado de procesos WPS de forma que el *software* ofrece directamente un método de visualización de los resultados.

EOxServer: Este *software* es básicamente una aplicación *web* escrita en el *framework* Django, que permite modelar juegos de datos de observación de la Tierra, utilizando y extendiendo MapServer para ofrecer servicios WMS y WCS de estos datos así como una interfaz de visualización, consulta y administración en un entorno *web*. A partir de un conjunto de imágenes *raster* de diferentes áreas e instantes temporales, EOxServer además de exponerlas por WMS y WCS usando el perfil específico para datos de observación de la Tierra, ofrece servicios para generar mosaicos y previsualizaciones, mediante un entorno web (usando OpenLayers) para navegar por las diferentes fechas y regiones.

TileStream: TileStream es un servidor de teselas pensado para servir archivos MBTiles. Es decir no genera ningún tipo de *cache* ni se conecta con servicios para solicitar las imágenes como el resto de servidores. Se trata de un *software* limitado únicamente a servir teselas almacenadas en este tipo de bases de datos. El *hosting* de teselas de Mapbox usa una variante de este servidor para su servicio.

2.4 Puntos calientes

Este área del *Software* Libre geoespacial está en continua evolución, pese a que la complejidad inherente a desarrollar este tipo de productos es elevada y hace algún tiempo que no aparece ningún producto relevante. Así y todo los proyectos existentes en general gozan de buena salud y no dejan de actualizarse e innovar.

En el área de los servidores de mapas la mejora del rendimiento y en especial de las capacidades de simbolización han facilitado la aplicación de estos productos en proyectos cada vez más complejos, como en el caso del uso de GeoServer en el Instituto Geográfico Francés *[GeoServerIGN]*. La innovación de aplicar al área de la cartografía un lenguaje de definición de simbologías análogo a las hojas de estilo de las páginas *web* (el estándar CSS) va a facilitar a los especialistas diseñar y mantener los estilos de sus mapas de una forma mucho más sencilla y a la vez expresiva.

En cuanto a los servidores de geoprocesamiento, la capacidad para definir procesos de análisis geográfico utilizando lenguajes de programación de alto nivel, como Python o JavaScript entre otros, va a facilitar la inevitable transición de este tipo de procesos desde los clientes de escritorio a los servidores. Así, proyectos como ZOO Project o el uso de GeoScript en GeoServer ponen a disposición de los analistas un entorno de trabajo que soporta varios lenguajes con un rendimiento elevado y en un entorno distribuido.

Los servidores de teselas siguen siendo de momento un *mal necesario* para ofrecer un rendimiento adecuado en proyectos con cartografía que no sufre actualizaciones frecuentes, imágenes *raster* o simplemente que no necesitan estilos dinámicos. Esta situación en cualquier caso para la información vectorial se percibe como transitoria ya que están empezando a aparecer productos y servicios que optimizan la presentación de cartografía sirviéndola en un formato vectorial junto con los estilos, siendo responsabilidad del cliente la renderización de la misma. Esta variante permite ofrecer cartografía mucho más dinámica tanto en su componente temporal como en la de la simbolización.

Finalmente en el área de los servicios de descubrimiento el desarrollo de GeoNetwork, el principal servidor de metadatos libre, sigue activo y van apareciendo nuevas alternativas y variantes como el soporte del protocolo CSW por parte de GeoServer, así como la aparición del proyecto PyCSW y su integración en otros productos.

2.5 Curva de aprendizaje y conocimientos previos

Principalmente existen dos perfiles de técnicos a la hora de trabajar con este tipo de productos. Esto se debe a que la implantación de un servicio en la red en primer lugar necesita de especialistas en sistemas que se encarguen de una correcta instalación del producto, adaptación del sistema operativo y de la red a la que se conecta, otros sistemas que puedan afectar al mismo, configuración y acceso a bases de datos, etc. Por otro lado igualmente en general es necesaria la intervención de un técnico especializado en el área geoespacial para la configuración avanzada del producto, preparar la cartografía o bases de datos a ofrecer, generar metadatos, etc. Es habitual que técnicos de un único perfil hagan todo el trabajo pero como en cualquier proyecto geoespacial, es en la multidisciplinariedad de los equipos de trabajo donde mejores resultados se van a conseguir.

Por lo tanto los conocimientos necesarios para este amplio conjunto de productos y desde el punto de vista tanto de la administración de sistemas como de la Información Geográfica es elevado, aunque puede depender también del producto. Nombrando los más importantes:

Desde el punto de vista de la administración de sistemas:

- Instalación y configuración de servidores *web* y de aplicaciones.
- Configuración de aplicaciones FastCGI/CGI, WSGI, NodeJS, JEE, etc. en función del producto.
- Creación y configuración de *Bases de datos*, esquemas de datos, usuarios y roles, etc.
- Configurar *caches*, *proxies* inversos y reescritura de direcciones *web* para integrar varios servidores. Por ejemplo es habitual exponer un servidor de aplicaciones JEE (por ejemplo GeoNetwork ejecutándose en el contenedor de *servlets* Tomcat) detrás de un servidor web, e incluso éste último detrás de un acelerador web como Varnish.

Desde el punto de vista del técnico en tecnologías geoespaciales:

- Conversión de formatos de datos geográficos (*raster* y vectorial).

- Manejo y carga de bases de datos espaciales.
- Conocer los diferentes especificaciones OGC, tanto en protocolos como en formatos (en función del producto).
- Comprender las principales proyecciones cartográficas y sistemas de referencia.
- Comprender las diferencias entre los diferentes formatos de imagen soportados por los navegadores, el concepto de *cache*, etc.

2.6 Documentación

A continuación se ofrecen enlaces a las principales páginas de documentación, tutoriales o ejemplos que pueden ayudar a empezar a trabajar con cada uno de los productos revisados.

Tabla 2.1: Documentación de proyectos

Proyecto	Documentación	OSGeo Live	Otros
MapServer	MapServer docs	MapServer qs	
deegree	deegree docs	deegree qs	
GeoServer	GeoServer docs	GeoServer qs	taller de introducción
GeoNetwork	GeoNetwork docs	GeoNetwork qs	Geonetwork workshop
52ºnorth SOS	52ºnorth SOS docs	52ºnorth SOS qs	
MapGuide	MapGuide docs	MapGuide qs	
PyWPS	PyWPS docs		PyWPS tutorial
GeoWebCache	GeoWebCache docs		taller de Boundless
TileCache	TileCache README		
52ºnorth WPS		52ºnorth WPS qs	tutoriales de 52ºnorth WPS
MapProxy	MapProxy docs	MapProxy qs	
PyCSW	PyCSW docs	PyCSW qs	taller de PyCSW
QGIS Server		QGIS Server qs	tutorial de QGIS Server
TileStache	TileStache docs		
ZOO Project:	ZOO Project docs	ZOO Project qs	taller del FOSS4G2013
EOxServer	EOxServer docs	EOxServer qs	
TileStream	notas de instalación y uso		

CAPÍTULO 3

Bases de datos

3.1 Autores

- María Arias de Reyna @delawen

3.2 Introducción

Una base de datos espacial es una base de datos que se ha optimizado para almacenar y consultar datos que representa los objetos definidos en un espacio geométrico. La mayoría de las bases de datos espaciales permiten representar objetos geométricos simples, tales como puntos, líneas y polígonos. Algunas bases de datos espaciales manejan estructuras más complejas, tales como objetos en tres dimensiones, coberturas topológicas, redes lineales, y TIN.

Aunque a día de hoy existen bases de datos NoSQL que también empiezan a soportar funcionalidades geométricas, en este artículo vamos a centrarnos únicamente en las bases de datos clásicas, dado que son las que tienen, con mucha diferencia, la implementación más madura.

En la siguiente tabla se presentan las características principales de los productos revisados en el contexto de este trabajo. Se puede consultar más información sobre la estructura de la tabla en la *introducción*.

Name	Year	OSGeo	Live	License	Ohloh	Tech
MySQL Spatial	2000	☹	☹	Oracle	ohloh	C/C++
PostGIS	2005	G	✓	GPL v2	ohloh	C/C++
Spatialite	2008	☹	☹	MPL tri-license	ohloh	C/C++
H2GIS	2008	☹	☹	GPL3	☹	Java

Figura 3.1: Información general sobre servidores

Importante: Se puede consultar la versión más reciente, así como los enlaces asociados y anotaciones en la sección de Bases_de_datos del wiki de OSGeo.

3.3 Software

En esta sección se describirá brevemente cada uno de los productos evaluados en esta comparativa. El orden de aparición es cronológico, empezando por los proyectos más veteranos.

Name	Binary Geometry	Normalized Geometry	Types and Functions
PostGIS	✓	✓	✓
MySQL Spatial	☹	☹	☹[1]
Spatialite	✓	✓	✓
H2GIS	✓	✓	✓

1. ↑ Partially supported

Figura 3.2: Implementación de estándares OGC

MySQLSpatial: Siendo la extensión para la base de datos más extendida de la lista, su instalación y manejo es muy sencillo. Es una base de datos que se encuentra por defecto en la mayoría de servicios de hosting y por tanto suele ser la puerta de entrada de muchos desarrolladores. Sin embargo, su falta de compatibilidad con los estándares hace que el código no sea fácilmente portable a otras plataformas.

PostGIS: Basado en postgres, esta potente base de datos multiplataforma es totalmente compatible con OGC. Aunque su uso a nivel general no está tan extendido como MySQL, dentro del sector GIS su uso es casi canónico.

Spatialite: Al estar basada en SQLite, es una base de datos basada en ficheros, lo que simplifica en gran medida su uso y distribución. Esta extensión es candidata para formar parte de Geo Package, un nuevo formato abierto de OGC para almacenar y transferir datos geográficos.

H2GIS: H2 es una base de datos Java ligera, con gran facilidad de instalación y distribución. Basada también en ficheros, se distingue de Spatialite porque contiene un servidor que permite varias conexiones concurrentes a la misma.

3.4 Puntos calientes

El Open Geospatial Consortium (OGC) ha desarrollado el estándar Simple Feature y establece normas para las funcionalidades espaciales de los sistemas de bases de datos. También llamada norma ISO 19125, se presenta en dos partes:

La primera parte, la norma ISO 19125-1 (SFA-CA para la "arquitectura común"), define un modelo de características simples de dos dimensiones, con interpolación lineal entre vértices. El modelo de datos definido en SFA-CA es una jerarquía de clases. Esta parte también define la representación utilizando WKT (well known text) y WKB (well known binary). Este tipo de datos contiene no sólo las coordenadas del objeto a representar, sino que también contiene referencias a la proyección utilizada y otras características que pudieran resultar de interés a la hora de operar con dichas geometrías.

La segunda parte de la norma ISO 19125-2 (SFA-SQL), define una serie de funcionalidades utilizando SQL. Estas funcionalidades abarcan la mayoría de las operaciones típicas que pueden llevarse a cabo con datos geográficos:

- Mediciones Espaciales: Para calcular la longitud de una línea, el área de un polígono, la distancia entre geometrías,...
- Funciones Espaciales: Modificar geometrías existentes para crear nuevas, es decir, operar con geometrías para obtener nuevas geometrías (intersección, buffering,...)
- Predicados Espaciales: Devuelven un booleano (verdadero/falso) acerca de la condición de una o más geometrías espaciales. Por ejemplo, si dos geometrías interseccionan o si están dentro de un buffer.
- Creación de Geometrías: Normalmente en base a una lista de coordenadas, se genera una nueva geometrías.
- Descripción de Geometrías: Devuelven información específica acerca de una geometría, por ejemplo qué punto es el centro de un círculo.

3.5 Curva de aprendizaje y conocimientos previos

Dado que la mayoría de las bases de datos con extensiones espaciales siguen un mismo estándar, la programación y uso de dichas extensiones es muy similar de una plataforma a otra.

3.6 Documentación

A continuación se ofrecen enlaces a las principales páginas de documentación, tutoriales o ejemplos que pueden ayudar a empezar a trabajar con cada uno de los productos revisados.

Tabla 3.1: Documentación de proyectos

Proyecto	Documentación	OSGeo Live	Otros
MySQL Spatial	MySQL Spatial docs		introducción a MySQL Spatial
postGIS	postGIS docs	postGIS qs	postGIS introduction
H2GIS	H2GIS docs		quickstart
Spatialite	Spatialite docs		KISS spatialite in 5 minutes

CAPÍTULO 4

Librerías

4.1 Autores

- Alejandro Díaz @alediator
- Roberto Antolín @Tolanss
- Santiago Higuera @santiagohiguera
- María Arias de Reyna @delawen

4.2 Introducción

En esta sección se va a cubrir todo el conjunto de paquetes que ofrecen funcionalidades avanzadas para cualquier sistema de información en la forma de librerías. Como se verá, algunos de estos proyectos han cogido protagonismo a lo largo del tiempo ofreciendo no sólo una interfaz de programación (o API en inglés), si no también un conjunto de herramientas propias. Dado el gran abanico que representan, se dividirá esta sección en otras cuatro secciones en las que se agrupan librerías con características y funcionalidades similares. Así, tendremos la siguiente división:

- **Geoprocesamiento**: Aquí se incluyen todas las herramientas y librerías que proporcionan métodos para la manipulación de información geoespacial como GDAL/OGR, GEOS, JTS/GeoTools, Geoscript o Shapely.
- **Routing**: Estas librerías ofrecen enrutamiento geoespacial y funcionalidad de análisis de redes. Comprendemos aquí las librerías OSRM, pgRouting y OpenTripPlanner
- **LiDAR**: Aquí se hablará sobre aquellas librerías y conjunto de herramientas capaces de trabajar con datos LiDAR. Destacan en esta categoría las librerías libLAS, LASlib/LASzip, SPDlib, PDAL y LASpy.
- **Varios**: Por último, consideramos dos librerías aisladas, una para el renderizado como Mapnik y otra para la automatización procesos para la publicación de información como GeoBatch.

La siguiente tabla muestra las principales características las librerías tratadas en este artículo. Información detallada sobre la estructura de la tabla se encuentra en la *introducción*.

Importante: Se puede consultar la versión más reciente, así como los enlaces asociados y anotaciones en la sección de Librerías del wiki de OSGeo.

4.3 Software

Cualquier librería LiDAR presentada en este documento se puede utilizar para la transformación de formatos láser, pero quizás la más adecuada para ello es PDAL, ya que está pensada especialmente para ello. Es más, PDAL trata de mejorar la librería libLAS cuyo desarrollo lleva parado algún tiempo. A su vez, libLAS se generó a partir de

Name	Year	OSGeo	Live	License	Ohloh	Tech
GDAL/OGR	2000	G	✓	Estilo MIT	ohloh	C/C++
Mapnik	2005	☹	✓	LGPLv2	ohloh	C++, Python
GeoTools	2006	G	✓	LGPL	ohloh	Java
Shapely	2007	☹	☹	BSD	ohloh	Python
libLAS	2007	☹	✓	BSD	ohloh	C++
LASzip/LASlib	2007	☹	☹	LGPL	☹	C++
GEOS	2008	G	✓	LGPL	ohloh	C/C++
GeoScript	2009	☹	☹	MIT	ohloh	JavaScript, Python, Scala & Groovy
OpenTripPlanner	2009	☹	☹	LGPLv3	ohloh	Java, Python
GeoBatch	2009	☹	☹	GPLv3	ohloh	Java
JTS Topology Suite	2010	-	-	LGPL	ohloh	Java
OSRM	2010	☹	☹	BSD	ohloh	C++
pgRouting	2010	☹	✓	GPLv2	ohloh	C/C++
PDAL	2011	☹	☹	BSD	ohloh	C++
SPDlib	2011	☹	☹	GPLv3 and MIT	ohloh	C++
Laspy	2012	☹	☹	BSD	ohloh	Python

una versión anterior de LASlib, aunque estas últimas han seguido evolucionando hacia una potente herramienta de procesado. Cabe notar, para evitar equívocos, que la librería LASlib es completamente libre, mientras que las herramientas (LASTools) tienen el código cerrado pero su utilización está autorizada para usos no comerciales. Además de lectura y escritura de datos y la transformación entre formatos, SPDlib incorpora herramientas para procesar y analizar datos y para generar modelos digitales de elevación. Por último, LASzip es la única librería capaz de leer y escribir datos LiDAR en formato LAZ. Éste es un tipo de formato comprimido pero con las mismas especificaciones que el formato LAS.

El procesamiento de datos es, en general, *la recogida y manipulación de elementos de datos para producir información significativa*. Así pues, a través del **geoprocesamiento**, se manipulan los datos espaciales con el objetivo de presentarlos en un contexto específico. En cuanto al procesamiento de datos vectoriales aparecen dos tendencias claras: una serie de librerías orientadas a la generación de scriptlets de forma intuitiva y sencilla (GeoScript/Shapely); y librerías clásicas de geoprocesamiento a nivel servidor (GEOS/JTS/GeoTools).

Shapely es un paquete de algoritmos para procesamiento espacial 2D escrito en Python.

GeoScript añade capacidades espaciales a los lenguajes dinámicos como JavaScript, Python, Scala o Groovy.

JTS Topology Suite es una librería escrita 100 % en Java que implementa la SFA y permite operar con geometrías incorporando una buena colección de algoritmos espaciales 2D. El rendimiento es elevado permitiendo la utilización de sus algoritmos en entornos de producción.

Geotools es una librería 100 % Java con todo tipo de herramientas para procesamiento y visualización de contenidos geoespaciales. Implementa la mayoría de estándares del OGC. El procesamiento espacial lo realiza embebiendo JTS. Añade la posibilidad de trabajar con Sistemas de Referencia y acceder a bases de datos. Es parte del backend utilizado por Geoserver.

GEOS es una versión C++ de la librería *JTS Topology Suite*

GDAL se suele utilizar como complemento de todas las demás herramientas para la preparación de imágenes ráster de forma que mejoren el rendimiento al ser servidas.

GeoBatch da un paso más allá en el geoprocesamiento, permitiendo la definición de ciertos flujos de procesamiento y permitiendo la ejecución de los mismos en background a través de distintos roles y usuarios (así como la ejecución programada de los mismos).

Mapnik es una herramienta para el renderizado de mapas atractivos, con bordes de geometrías limpios y suaves, provisto de un sistema gráfico con anti-aliasing de calidad, posicionamiento inteligente de etiquetas, y simbolización SVG escalable. La mayor fama de mapnik viene por ser utilizado como render de la capa principal de Open

Street Map.

En relación con el routing o cálculo de rutas, mencionamos en este grupo las librerías **pgRouting**, **OSRM** y **OpenTripPlanner**.

pgRouting es una extensión de la base de datos espacial Potgres-PostGIS que añade funcionalidades de routing, esto es, de cálculo de rutas y caminos mínimos a través de los distintos algoritmos. Trabaja con los datos en crudo, sin necesidad de pre-procesamiento.

OSRM Es una librería escrita en C++ que proporciona algoritmos de camino mínimo y herramientas de routing utilizando la cartografía de OpenStreetMap como red de caminos.

OpenTripPlanner ofrece una API REST que permite el cálculo de rutas basadas el distintos métodos de transporte (incluyendo parámetros como el alquiler de bicicletas o el transporte público).

4.4 Puntos calientes

Últimamente, la tecnología LiDAR está en auje y cada vez aparecen más librerías que ofrecen la posibilidad de trabajar con datos LiDAR en el formato LAS. Algunas también incorporan conjuntos herramientas para el procesado y análisis de datos. Estas herramientas suelen aparecer como comandos de consola para favorecer el desarrollos de *scripts*. Con el objetivo de favorecer la creación de sencillos programas, los esfuerzos se están centrando en incorporan *bindings* en python. Para permitir un uso más sencillo y cómodo la comunidad está haciendo esfuerzos en la creación interfaces gráficas y en la incorporación de forma nativa de lectura y escritura de datos láser en clientes de escritorio como QGIS o GRASS.

En relación al **cálculo de rutas**, existe una clara diferenciación entre las librerías que se basan en datos pre-procesados y las librerías que pueden trabajar sin pre-procesar los datos. Mientras que las librerías con datos procesados hoy en día devuelven respuestas instantáneas a rutas complejas, las librerías con datos en crudo permiten una mayor flexibilidad a la hora de incorporar datos en tiempo real, como puede ser el tráfico o el clima. También es relevante mencionar que OpenTripPlanner permite el cálculo de rutas multimodal; esto es, cálculo de rutas combinando diferentes medios de transporte.

Respecto al **geoprocesamiento**, probablemente, los avances más significativos se están realizando en la abstración de las librerías clásicas con el objetivo de facilitar su uso de cara a los desarrolladores. Dentro de este grupo se enmcuentran GeoScript, Shapely o GeoBatch.

Mapnik se suele embeber típicamente en aplicaciones python que publican mapas en Internet, aunque las últimas mejoras incorporadas han permitido que Mapnik también se utilice para crear mapas de alta resolución en papel.

4.5 Curva de aprendizaje y conocimientos previos

Para trabajar con las librerías LiDAR es conveniente tener conocimientos de C++, ya que es el lenguaje común a todas ellas. Aunque existen ya interfaces gráficas para trabajar con estas herramientas, el módo más rápido y versátil es la línea de comando y la utilización de *scripts* o incluso la programación en Python. Por tanto, tener experiencia en estos campos facilitaría su utilización. Escribir en la consola nunca es agradable para cualquier persona que empieza, pero la mayor ventajan que presentan es que todas las herramientas tienen las mismas funcionalidades y es muy intuitivo aprender el lenguaje utilizado en cualquiera de ellas si se adquieren conocmientos previos en alguna otra librería.

A la hora de trabajar con cálculo de rutas, es conveniente entender la terminología básica de la teoría de grafos, como qué es un nodo y cómo se interconectan los nodos. Una vez comprendida la teoría básica de grafos, lo que va a marcar qué algoritmo elegimos y cómo vamos a usarla será nuestra fuente de datos y nuestras necesidades o no de tener rutas multimodales en tiempo real.

En cuanto al **geoprocesamiento**, los conocimientos necesarios son distintos según la(s) librería(s) que quieras usar. Para todas necesitarás conocimentos acerca del modelo SFA. A no ser que necesites una funcionalidad específica de una de las librerías, podrás elegir aquella que se adecúe más a tus conocimientos. Los programadores en C++ pueden utilizar GEOS, mientras que los programadores Java encontrarán en JTS y Geotools las herramientas de geoprocesamiento necesario. Python dispone de GeoScript, Shapely como herramientas específicas del lenguaje.

4.6 Documentación

A continuación se enumeran una serie de enlaces a páginas de documentación, tutoriales o ejemplos que pueden ser de ayuda para trabajar con estas librerías:

Tabla 4.1: Documentación de proyectos

Proyecto	Documentación	OSGeo Live	Otros
GDAL/OGR	GDAL docs; OGR docs	GDAL qs	GDAL wiki
JTS Topology Suite	JTS Topology Suite docs		
GEOS	GEOS API docs		Tutorial de la documentación de Django
GeoTools	GeoTools docs		Soporte y comunidad
Shapely	Shapely docs		
GeoScript *	GeoScript docs		**‘Tutoriales oficiales <http://geoscript.org/tutorials/index.html>>‘_**
libLAS	libLAS docs	libLAS qs	
LAS-lib/LASzip	Tutoriales; Artículo sobre LASzip		Manual de *Minnesota Department of Natural Resources*
PDAL	PDAL docs		
SPDlib	SPDlib docs		Tutoriales
Laspy	LASpy docs		
OSRM	General OSRM instructions		OSRM Server API
pgRouting	pgRouting docs	pgRouting qs	Guía para principiantes
OpenTripPlanner	OpenTripPlanner docs		Guía de desarrollador;
GeoBatch	GeoBatch docs		
Mapnik	Mapnik docs	Mapnik qs	Tutoriales

CAPÍTULO 5

Web Mapping

Por hacer

Sección por completar. Discusión del contenido en la issue 3

5.1 Autores

- Alejandro Díaz @alediator
- María Arias de Reyna @delawen
- Moisés Arcos @moiarcsan

5.2 Introducción

Resulta evidente la importancia del FOSS en el ámbito de la geomática. No hay más que ver la cantidad de proyectos, comunidades, blogs, congresos y demás eventos (como el que ocupa este artículo) que se organizan con cada vez mayor éxito. Esto puede provocar al recién llegado cierta confusión ya que el inherente carácter modular del software libre hace que muchos proyectos dependan de otros y por tanto las interconexiones son múltiples y a todos los niveles.

Con la proliferación de Internet, la aparición de los servidores de mapas se produjo de forma conjunta a la de aplicaciones web que exponían los contenidos servidos por estos productos. Al principio la mayor parte de ellas se materializaban como desarrollos ex profeso y por tanto se resolvían los mismos problemas una y otra vez. Esta situación derivó como es natural hacia proyectos que intentan proporcionar un conjunto de componentes comunes en general en forma de documentos HTML y aplicaciones escritas en JavaScript que proporcionan al desarrollador una base sobre la que realizar su aplicación específica. También han ido apareciendo proyectos que se basan en mayor o menor medida en código de servidor, básicamente PHP o Java.

La motivación de este texto por tanto es la presentación ante los lectores, desde un punto de vista lo más generalista posible, del estado del arte en este ámbito de la ciencia para dar al lector pistas que le acerquen a aquellos proyectos/productos que le puedan ser de interés para realizar cualquier tipo de proyecto.

En esta sección se tratarán los diferentes proyectos relacionados con la representación de la información geográfica en un cliente web y que tiene como objetivo la visualización y manipulación de la misma.

En la siguiente tabla se presentan las características principales de los productos revisados en el contexto de este trabajo. Se puede consultar más información sobre la estructura de la tabla en la *introducción*.

Importante: Se puede consultar la versión más reciente, así como los enlaces asociados y anotaciones en la sección de Webmapping del wiki de OSGeo.

Name	Year	OSGeo	Live	License	Ohloh	Tech
OpenLayers	2006	G	✓	BSD	ohloh	Javascript
Leaflet	2010	☹	✓	BSD	ohloh	Javascript
GeoExt/GXP	2009	☹	☹	BSD	ohloh	Javascript
MapStore	2012	☹	☹	GPL	ohloh	Javascript
Mapbender	2003	G	✓	GPL and BSD	ohloh	PHP, JavaScript and XML
Cartaro	2012	☹	☹	[GPL versión 2]	-	PHP and Javascript
GeoMoose	2009	G	✓	MIT	ohloh	PHP and Javascript

Figura 5.1: Información general sobre tecnologías

Nombre	WMS	WFS	WFS-T	WCS	WMTS	TMS	WPS	SOS	CSW
OpenLayers	✓	✓	✓	✓	✓	✓	✓	✓	✓
Leaflet	✓	✓	✓[1]	☹	✓[2]	✓	☹	☹	☹
GeoExt/GXP	✓	✓	✓	✓[3]	✓[3]	✓[3]	✓[3]	✓[3]	✓[3]
MapStore	✓	✓	✓	✓[3]	✓[3]	✓	✓	✓[3]	✓
Mapbender	✓	✓	✓	☹	✓[3]	✓[3]	✓[3]	✓[3]	✓[3]
Cartaro	✓	✓	✓[4]	✓	✓[3]	✓[3]	✓[3]	✓[3]	✓[3]
GeoMoose	✓	✓	✓	✓[3]	✓[3]	✓[3]	✓[3]	✓[3]	✓[3]

Figura 5.2: Implementación de estándares OGC

5.3 Software

En esta sección se describirá brevemente cada uno de los productos evaluados en esta comparativa. El orden de aparición es cronológico, empezando por los proyectos más veteranos.

Mapbender:

Cliente Web-GIS construido con Javascript, que ofrece un interfaz de usuario configurable no dependiente de ningún servidor de mapas concreto. Su orientación es la de un geoportal cliente de servicios OGC. Incluye un soporte bastante completo de usuarios, grupos y servicios OGC (OWS). Una característica diferenciadora de Mapbender es la capacidad de edición en cliente sobre navegador, utilizando WFS-T. MapBender es un proyecto graduado de OSGeo.

OpenLayers:

OpenLayers es un cliente Web-GIS ligero construido con clases Javascript, sin dependencia de servidores de mapas concretos. Ofrece una interfaz de usuario simplificada que ataca a servicios WMS y WFS de forma transparente para el usuario y desarrollador. Las características a destacar de este producto es la cantidad de herramientas ya implementadas, que hacen que el desarrollar con esta librearía sea mucho más fácil para funcionalidades más complejas. Actualmente es uno de los proyectos de SIG libre cuya comunidad es de las más activas que existen, a pesar de contar con un grupo de desarrolladores no muy numeroso. OpenLayers es un proyecto graduado de OSGeo. Destaca la integración de OpenLayers con otros proyectos como GeoExt o Mapstore.

GeoExt GXP:

Es una biblioteca basada en Javascript para el desarrollo de aplicaciones web interactivas integrando la tecnología de OpenLayers y ExtJS. Ofrece la interfaz de usuario propia de ExtJS, con todas las herramientas adaptadas al visor de mapas OpenLayers.

GeoMoose:

GeoMOOSE es un framework de navegación de mapas para la visualización distribuida de datos cartográficos. Es particularmente útil para gestionar datos geoespaciales y no geoespaciales en oficinas regionales, urbanas y municipales (GeoMOOSE se originó en éstas últimas). Extiende la funcionalidad de MapServer y OpenLayers para proporcionar servicios de serie, como la identificación 'drill-down' para ver y organizar muchas capas, operaciones de selección y búsquedas en los juegos de datos.

GeoMOOSE es rápido, con buen rendimiento con cientos de capas y/o servicios al mismo tiempo. Los datos provenientes de diferentes orígenes se pueden mantener usando diferentes herramientas y con diferentes planificaciones ya que cada capa del mapa tiene su propio juego de ficheros de configuración para la publicación, simbología, plantillas así como datos de origen.

La interfaz de usuario es fácilmente configurable, y gracias a su arquitectura modular se pueden agregar servicios adicionales.

Leaflet:

Leaflet está diseñado con la mente puesta en la simplicidad, el rendimiento y la facilidad de uso. Funciona de manera eficiente en las principales plataformas de escritorio y móviles, aprovechando las ventajas del HTML5 y CSS3 en los navegadores modernos, sin dejar de ser accesibles a los más antiguos. Se puede ampliar su funcionalidad con una gran cantidad de plugins, tiene una API bonita, fácil de usar y bien documentada, así como un código fuente simple y legible, que es una fácilidad para los desarrolladores que quieran contribuir.

Mapstore:

MapStore ha sido desarrollado para crear, guardar, buscar y compartir de una manera sencilla e intuitiva mashups creados con contenido de fuentes del servidor como Google Maps, OpenStreetMap, MapQuest o servidores específicos proporcionados por la organización o cualquier otra persona. MapStore consta de dos componentes principales como MapManager y GeoStore, respectivamente front-end y back-end.

MapManager, utilizando una sola interfaz, permite al usuario crear, borrar y buscar mapas, generar un vínculo de inserción para poner un mapa en un sitio web, compartir tus propios mapas con otros usuarios. Además lleva a cabo la interacción con GeoStore. MapManager soporta la autenticación y la definición de políticas de acceso para proteger los mapas gestionados por GeoStore.

GeoStore es una aplicación JEE de código abierto cuyo objetivo es el almacenamiento, la búsqueda y la recuperación de datos sobre la marcha. GeoStore implementa una infraestructura flexible y modular desarrollado por encima de la tecnología de Java Enterprise con el fin de crear, gestionar, navegar y buscar las definiciones del mapa. GeoStore integra la autenticación y gestión de autorizaciones según el paradigma de Role Based Access Control (RBAC). Esto protege a los mapas de accesos no autorizados. El mecanismo de almacenamiento estándar de GeoStore consta de un DBMS: Oracle y PostgreSQL son compatibles.

Cartaro:

Cartaro es la plataforma de cartografía web que proporciona los mejores componentes geoespaciales de código abierto en un sistema de gestión de contenidos. Con Cartaro usted es capaz de instalar y ejecutar su propio sitio web geográfico y compatible con los estándares de la OGC, con no más de unos pocos clics. Los componentes geoespaciales utilizados en Cartaro son PostGIS, GeoServer, GeoWebCache y OpenLayers. Todos los que se gestionan desde el potente CMS Drupal.

Cartaro es para las organizaciones e individuos que necesitan ejecutar una infraestructura de datos espaciales ligera (SDI), sin necesidad de extensas configuraciones y mucha programación individual.

Cartaro sirve también para montar un sitio web con los beneficios de cualquier CMS pero con la ventaja de poder tratar la información espacial.

5.4 Puntos calientes

Últimamente todo lo relacionado con Leaflet se convierte en tendencia, ya que están apareciendo distintas tecnologías que hacen uso de esta librería en sus desarrollos, como por ejemplo Mapbox o CartoDB, cuyo aspecto visual tan aparente y resultón hacen que su uso prolifere.

Otro de los puntos en los que se está empleando mayor esfuerzo tecnológico es en los renderizadores 3D basados en WebGL tales como WebGL Earth o F4 Map, que le dan otra dimensión a la forma de representar los datos espaciales en la web.

El futuro de los mapas podría pasar por mejorar las versiones móviles que hagan que su interacción con el entorno los enriquezca, de forma que pasen de ser consultores o indicadores de direcciones a compañeros imprescindibles en la vida cotidiana, ya que pueden convertirse en planificadores de jornadas, de escapadas de fin de semana e incluso de guías turísticos.

5.5 Curva de aprendizaje y conocimientos previos

Para poder trabajar con la mayoría de las librerías que aquí se presentan son imprescindibles conocimientos en Javascript, ya que la mayoría de éstas están desarrolladas bajo este lenguaje de programación. Aunque hay algunos como Cartaro que está basado en Drupal, para lo que hay que tener conocimientos básicos en PHP.

Es bueno tener conocimientos de servicios web, así como de XML, ya que muchas de las respuestas de los principales protocolos de transferencia de información geográfica usan éste lenguaje de respuesta. También hay que tener conocimientos de los estándares OGC que consumiran la parte cliente.

Hay librerías como OpenLayers o Leaflet donde esta curva de aprendizaje es muy poco pronunciada ya que su facilidad de comprensión y de uso, así como la documentación que poseen, hacen que la experiencia del desarrollador no sea determinante a la hora de elegirlas como posible base para el desarrollo de cliente web geográficos. De hecho otras librerías que aquí se mencionan usan OpenLayers como base cartográfica con la que interactuar y a la que integrarse.

5.6 Documentación

Toda la documentación necesaria para comenzar a usar cualquiera de las tecnologías que aquí se encuentran, la puedes encontrar en las guías de inicio rápido que se adjuntan:

Tabla 5.1: Documentación de proyectos

Proyecto	Documentación	OSGeo Live	Otros
Mapbender	Mapbender user doc	Mapbender qs	MapBender tutorials
OpenLayers	OpenLayers library doc	OpenLayers qs	OpenLayers API doc, OpenLayers Examples
GeoExt GXP	GeoExt doc, GXP doc		GeoExt qs, OpenGeo Suite ws
GeoMoose	GeoMoose doc	GeoMoose qs	
Leaflet	Leaflet reference		Leaflet qs
MapStore	MapStore docs		MapStore qs, MapStore training docs, MapStore demo
Cartaro	Cartaro docs	Cartaro qs	

5.7 Referencias

- Panorama SIG Libre, M. Montesinos y J. Sanz Artículo v2, Diapos v4, Wiki Prodevelop

CAPÍTULO 6

Clientes

6.1 Autores

- Josep Sitjar @JosepSitjar
- Roberto Antolín @Tolanss

6.2 Introducción

Entendemos por Clientes al conjunto de aplicaciones SIG de Escritorio, es decir, aplicaciones en las que se implementan herramientas para llevar a cabo las tareas básicas del trabajo con datos geográficos: creación o edición, manejo y análisis. Con esta filosofía fueron desarrollados los primeros programas SIG, especialmente para el tratamiento y análisis de datos geográficos, y posteriormente, para dotar a estos de mayor versatilidad, incorporando otras funciones adicionales que facilitaran el trabajo con esos mismos datos.

Los SIG de escritorio siguen manteniendo su posición como aplicaciones fundamentales, y hablar genéricamente de un SIG implica por lo general hacerlo de una aplicación de escritorio antes que de otros tipos de aplicaciones. Por otra parte, las herramientas de escritorio son soluciones en general completas que cubren la totalidad de necesidades que se presentan en el desarrollo de proyectos SIG, y por ello constituyen las herramientas primordiales para llevar estos a cabo. Ofrecen un gran numero de herramientas para gran diversidad de usuarios en diversidad de campos.

Veamos con un poco más de detalle las principales funcionalidades de los SIG de Escritorio:

- **Entrada y salida de datos**: Todas las aplicaciones SIG de escritorio deben obligatoriamente implmentar capacidades para leer datos y, opcionalmente, guardarlos. Pese a ser de tal importancia, la implementación de las capacidades de entrada y salida es muy variable en unos u otros SIG. Una razón por la que esto sucede es el gran número de formatos de fichero distintos. Así, cada SIG de escritorio es capaz de abrir unos u otros formatos de archivo, y mientras que algunas tratan a todos ellos por igual, ciertas aplicaciones trabajan en un formato propio con carácter nativo y son capaces de incorporar datos en otros formatos a través de extensiones o funciones de conversión entre estos y el formato particular del programa. Cabe destacar también la capacidad de conexión a bases de datos o servicios remotos que ofrecen algunos softwares (ahora la mayoría).
- **Visualización**: La visualización es una función fundamental dentro de los SIG y del trabajo con cartografía en general. La gran mayoría de las herramientas de escritorio incluyen un gran número de elementos para representar los datos geográficos con los que se trabaja. En ocasiones, interesa únicamente crear una representación de los datos, pero incluso cuando el trabajo con una herramienta SIG está enfocado a la realización de un análisis, la visualización y exploración visual de los datos de partida es un paso previo. En general, la forma de operar con los elementos de visualización es muy similar entre soluciones SIG distintas y, a diferencia de lo que sucede con la implementación de otras funcionalidades, el manejo es prácticamente igual.
- **Análisis**: Posiblemente, una de las funcionalidades más destacadas y significativas de un SIG de Escritorio. La tendencia actual es considerar las capacidades de análisis como herramientas modulares que se ejecutan

sobre una plataforma base, la cual comprende las capacidades de visualización y entrada y salidad de datos. Todas estas capacidades de análisis son independientes entre sí, auqnué pueden coordinarse y emplearse en conjunto para alcanzar un resultado concreto.

- **Edición**: Funcionalidades que permiten modificar y corregir los datos geográficos con los que se trabaja en un SIG. Las operaciones de edición pueden emplearse para la actualización de cartografía, pero también para la creación de nuevas capas, que pueden crearse a partir de la digitalización de imágenes o a partir de cualquier otra capa de la que se disponga. Puede distinguirse entre diversas formas de edición: Edición de geometrías de una capa vectorial, edición de atribuos en una capa vectorial, edición de valores en una capa raster. Fundamentalmente, estas capacidades permiten la composición de documentos cartográficos de acuerdo con un diseño dado. En la elaboración del diseño, pueden emplearse todos los elementos que habitualmente podemos encontrar en un mapa: el propio mapa en sí, leyenda, título, escala, orientación, etc.
- **Generación de cartografía**: Capacidades de creación de cartografía impresa, para generar documentos que puedan posteriormente imprimirse y emplearse como una mapa cásico. Las razones para la existencia de tales funcionalidades son muchas, pero la principal sigue siendo la necesidad que aún existe de apoyarse en este tipo de documentos cartográficos para poder incorporarlos a proyectos o estudios como parte de anexos cartográficos.

(Fuente: Olaya, V. 2012. Libro Libre SIG.)

Los clientes de escritorio ofrecen un amplio rango de aplicaciones, desde simples visualizadores a software de creación de mapas y análisis y tecnología punta en sistemas de edición y análisis profesional.

En la siguiente tabla se presentan las características principales de los productos revisados en el contexto de este trabajo.

Name	Year	OSGeo	Live	License	Ohlohno	Tech
gvSIG	2004	I	✓	GPL 2.0+	ohloh	Java
QGis	2002	G	✓	GPL 3.0	ohloh	C++, Python
GRASS GIS	1982	G	✓	GPL 2.0+	ohloh	C/C++, Python
UDig	1994	☹	✓	BSD-3-Clause and EPL 1.0	ohloh	Java
openJUMP	2002	☹	✓	GPLv2	ohloh	Java
SAGA GIS	2004	☹	✓	GPLv3 and LGPLV3	ohloh	C++
OPTICKS	2000	☹	☹	LGPLv2	ohloh	C++, Python
GeoDa	2003	☹	☹	GPLv3	☹	C++

Figura 6.1: Información general sobre tecnologías

Nombre	WMS	WFS	WFS-T	WCS	WMTS	TMS	WPS	SOS	CSW
gvSIG	✓	✓	✓	✓	✓	☹	☹	☹	✓
QGis	✓	✓	✓	✓	✓	☹	✓[1]	☹	✓[2]
GRASS GIS	✓	✓	☹	☹	☹	☹	✓	☹	☹
UDig	✓	✓	✓	✓	✓	✓[3]	✓[4]	✓[5]	☹
openJUMP	✓	✓	✓[6]	☹	☹	☹	✓[7]	☹	☹
SAGA GIS	☹	☹	☹	☹	☹	☹	☹	☹	☹
OPTICKS	✓	☹	☹	✓	☹	☹	☹	☹	☹
GeoDa	☹	☹	☹	☹	☹	☹	☹	☹	☹

Figura 6.2: Implementación de estándares OGC

Breve descripción de la sección con retrospectiva y evolución incluyendo la tabla de la sección.

6.3 Software

En esta sección se describirá brevemente cada uno de los productos evaluados en esta comparativa.

gvSIG:

GvSIG es un proyecto de desarrollo de Sistemas de Información Geográfica en software libre, que incluye principalmente las aplicaciones gvSIG Desktop y gvSIG Mobile. La aplicación gvSIG Desktop fue la primera que se desarrolló dentro del proyecto gvSIG, por lo que también se conoce abreviadamente como gvSIG. Este proyecto fue desarrollado por el gobierno local de la Comunidad Valenciana (Generalidad Valenciana) de España, con el objetivo inicial de realizar la gestión de datos geográficos de esa colectividad; precisamente la sigla gvSIG abrevia la denominación Generalitat Valenciana Sistema de Información Geográfica.

QGIS

Es un Sistema de Información Geográfica que nació en mayo de 2002 y se estableció como proyecto en SourceForge en junio del mismo año. Fue además uno de los ocho primeros proyectos de la fundación OSGeo. Se trata de una aplicación de escritorio que pretende ofrecer a usuarios con necesidades básicas un entorno sencillo y agradable.

GRASS

Proyecto ya muy veterano, anterior al nacimiento del FOSS, que el **CERL** (*Construction Engineering Research Laboratory*) comenzó a desarrollar ante la necesidad de gestionar la gran cantidad de recursos naturales a cargo del ejército en los Estados Unidos. Actualmente, la infraestructura principal se gestiona entre el Instituto de Cultura de Trento y el *Geselleschaft für Datenanalyse und Fernerkundung* (GDF) de Hannover. La principal característica de GRASS es su gran número de funcionalidades, derivadas de todos los años de desarrollo y de la estructura modular del programa, que favorece que los desarrolladores aporten al proyecto contribuciones individuales. Por otro lado, el mayor problema de cara a su difusión y adopción es su complejidad y su pronunciada curva de aprendizaje. Aun siendo un software muy potente, carece de una interfaz gráfica amigable.

UDig

Este proyecto está desarrollado por la empresa canadiense Refractions Research Inc, y tiene como principal objetivo ofrecer un cliente de escritorio que soporte el mayor número de fuentes de datos tanto locales como remotas, y especialmente las basadas en protocolos OGC. Aunque presenta únicamente capacidades de análisis y edición vectorial, el proyecto JGrass añade capacidades ráster adaptadas desde GRASS (con especial énfasis en las relacionadas con análisis del terreno y similares).

OpenJUMP

JUMP (*Java Unified Mapping Platform*) fue uno de los primeros proyectos de cliente GIS de escritorio en el lenguaje Java. Destaca por hacer uso de la biblioteca JTS para poder realizar algunas operaciones de análisis espacial, así como el soporte del formato GML y el protocolo WMS desde sus aparición. Este proyecto fue liderado por Vivid Solutions1 pero dada la política de aceptación de contribuciones externas por parte de la empresa motivó la aparición de un nuevo proyecto derivado (llamado fork en el ámbito FOSS) conocido como The JUMP Pilot Project (JPP) que pretende coordinar de forma más democrática las contribuciones de diferentes equipos de desarrollo para evitar duplicidad de esfuerzos. Esto último es especialmente importante, ya que es destacable la cantidad de proyectos derivados que han surgido a partir de él:

- Open JUMP
- Open JUMP *Viatoris*
- DeeJUMP
- SkyJUMP
- PirolJUMP
- Kosmo

Kosmo

En España destaca el proyecto Kosmo, desarrollado por la empresa SAGE, que pretende incorporar a la plataforma JUMP otros desarrollos de interés realizados en otros proyectos. El cliente de escritorio Kosmo permite

explorar, editar y analizar datos espaciales desde gran variedad de bases de datos, formatos vectoriales y formatos raster. Además, cumple los estándares OGC y proporciona una excelente integridad topológica. Su arquitectura de extensiones permite personalizarlo fácilmente para fines específicos.

SAGA GIS:

Esta herramienta se ha desarrollado sobre todo en Gottingen, Alemania. Se trata de un GIS de escritorio para Windows con una clara separación entre su interfaz de programación (API) y su interfaz de usuario. De hecho la primera tiene una licencia LGPL y la segunda es GPL. Esto permite realizar módulos *cerrados* sin incumplir ninguna licencia. Este software destaca especialmente por su orientación a la realización de análisis de imágenes y modelos digitales del terreno.

OPTICKS: Opticks es una herramienta para el análisis de imágenes y datos provenientes de la teledetección. Es muy similar en funcionalidad y propósito a las herramientas comerciales *ERDAS Imagine*, *RemoteView*, *ENVI* o *SOCET GXP*. Opticks tiene funcionalidades del tipo GIS (como la utilización de shapefiles), pero principalmente está pensado para el anánisis de imágenes y vídeo o, de manera más general, el análisis de datos raster.

GEODA:

GeoDa es el principal programa desarrollado dentro de Centro GeoDa. Está diseñado para implementar técnicas para el análisis exploratorio de datos espaciales. Proporciona una interfaz gráfica amigable para el uso de métodos de análisis de datos espaciales descriptivos, realización de estadísticas de autocorrelación espacial y regresiones espaciales básicas, análisis de datos espacio-temporales de datos, y visualización 3D.

TILEMILL: TileMill es una herramienta pensada para diseñar y crear mapas para su visualización via web de una manera rápida y fácil. Está construido sobre la biblioteca de renderizado, Mapnik, la misma que utilizan tanto OpenStreetMap como MapQuest. **TileMill** no pretende ser una herramienta de cartografía de uso general, sino que se centra en la racionalización y simplificación de un conjunto limitado de casos de uso.

6.4 Puntos calientes

En el ámbito geoespacial, el panorama de clientes SIG opensource goza de buena salud, y a pesar que se ha alcanzado un buen nivel tanto en relación a la diversidad de productos disponibles, com en las prestaciones de estos, la mejora es continua y el desarrollo nuevas funcionalidades parece avanzar favorablemente.

Una de las limitaciones existentes en los SIG de escritorio actuales es la capacidad para la gestión y visualización de datos en 3D. Algunos paquetes contienen estructuras híbridas para la visualización en 2.5D, en las que se representan los datos como una falsa tridimensionalidad mediante proyecciones gráficas en 2D. Sin embargo, se requiere funcionalidades más avanzadas para el manejo de estos datos más complejos con más de dos dimensiones. Este paradigma se repite con los datos temporales. Añadir la tercera componente espacial, es decir tener puntos (X,Y,Z), más una componente tiempo asociada, se asemejaría más a la realidad, pudiendo además, analizar procesos dinámicos de los elementos representados.

Un ejemplo de datos en los que es necesario una visualización 3D son los datos LiDAR. El uso de datos LiDAR se está extendiendo cada vez más dentro de los usuarios de herramientas SIG. Ya existen librerías capaces de no sólo leer y escribir sino también de analizar y procesar datos láser en formato LAS *Librerías*. Hasta la fecha, ninguno de los paquetes presetados son capaces de trabajar con datos laser de forma nativa. Sólo GRASS es capaz de importar datos en formato LAS, pero la forma en que gestiona y almacena los mismos no es óptima.

En los últimos años, todos los SIG de escritorio están trabajando para incorporar la posibilidad de *scripting* dentro de sus funcionalidades. La principal ventaja que representa esto es la automatización de procesos, que de otra manera se tendrían que ejecutar una a una de manera manual, permitiendo la gestión de grandes volúmenes de datos con la menor interacción del usuario. En la mayoría de los casos, el lenguaje utilizado para ello es Python (o alguno de sus derivados como Jython en el caso de gvSIG)

El análisis y procesado de datos, tanto raster como vectorial, ha sido uno de los puntos fuertes de los clientes SIG de código abierto. Sin embargo, muchos de los nuevos paquetes de software incluyen cada vez más mejoras en las herramientas disponibles para la edición de datos, su visualización y representación. También en cuanto al entorno para el diseño de los mapas, aunque este aspecto resulta cada vez menos relevante al publicarse frecuentemente los datos analizados en entornos web o móviles.

La integración de bibliotecas de contrastado prestigio en los clientes SIG (ej, GDAL, JTS, Geotools...) es también uno de los aspectos a destacar. Ello dota de robustez y versatilidad a muchos de estos proyectos, y permite a los usuarios tener garantías en los procesos ejecutados. Asimismo, la arquitectura modular de muchos clientes SIG permite incorporar multitud de complementos que los dotan de más funcionalidades.

Actualmente la interrelación de librerías, complementos, algoritmos... entre diversos clientes, facilita la labor del usuario, que puede disponer de todo este ecosistema en un mismo entorno de trabajo -sea cual sea-. Por ejemplo, podemos trabajar con SEXTANTE desde gvSIG, QGIS, Kosmo..., o utilizar GRASS como plugin desde QGIS, por citar algunos ejemplos.

Durante los últimos años el software SIG de escritorio ha dominado pero se prevé que, debido a la continua mejora de las conexiones a internet, los servidores SIG sean el producto dominante en la próxima década. La razón de ser de los clientes de escritorio en el mundo GIS se apoya en la necesidad de realizar complejos y pesados análisis espaciales, sin embargo, es tendencia que los servidores SIG incorporen capacidades de procesado de datos. Es más, el volumen de captura de datos espaciales está yendo en aumento debido al abaratamiento de los sensores y su accesibilidad, lo que provoca una necesidad de equipos cada vez más potentes. Actualmente, los servidores web ofrezen la posibilidad de utilizar grandes *clústers* de procesadores están siendo la respuesta del mercado en ese sentido, y la tendencia es la de recurrir a estos servidores para gestión y procesado de tanta información. Sin embargo, hasta que el uso de los servidores web no se estabilice y tenga un abanico mucho más extenso de herramientas de análisis, el uso de clientes de escritorio seguirá siendo necesario.

6.5 Curva de aprendizaje y conocimientos previos

Al estar todos los clientes de escritorio basados en interfaces gráficas, su uso suele ser bastante sencillo. Todos poseen a rasgos generales las mismas características siendo su curva de aprendizaje muy pronunciada, esto es, se aprende muy rápido en poco tiempo. Pero como siempre, hay excepciones. La interfaz gráfica de GRASS no es muy intuitiva y no es fácil dónde buscar los diferentes módulos de análisis. La situación se agrava cuando se trabaja a través de la consola, aunque realmente aquí es donde radica su potencial debido a su versatilidad. Por tanto, es recomendable utilizar GRASS sólo si se tiene alguna experiencia previa en SIG y con línea de comando. Por el contrario, paquetes como QGIS, uDig, gvSIG u openJUMP, podrían estar especialmente recomendados para principiantes.

Los conocimientos previos necesarios para trabajar con este tipo de software coinciden con los conocimientos en tecnologías geoespaciales. Esto incluye compresión de los distintos formatos *raster* y vectorial, y el modo de conversión entre ellos, conocimiento de distintos algorítmos de análisis y gestión de datos geoespaciales, manejo de bases de datos, compresión de proyecciones cartográficas y sistemas de referencia. Además, existen conocimientos específicos para distintos paquetes. Así, es necesario cierta competencia en CSS si se quiere trabajar con Tilemill, o experiencia en lenguajes del tipo *scripting* como **shell** o **Python** si se quiere profundizar en la utilización de GRASS o QGIS, respectivamente.

6.6 Documentación

Empezar a trabajar con cualquier software siempre es un reto y, como hemos visto, en algunas ocasiones puede resultar incluso una tarea árdua. En algunas ocasiones esto se debe porque tampoco sabemos dónde encontrar una guía de inicio amena o tan siquiera la documentación. Este apartado intenta cubrir este hueco, pretende ser un conjunto de enlaces tanto a la documentación oficial como a una serie de tutoriales y ejemplos de los distintos proyectos.

Tabla 6.1: Documentación de proyectos

Proyecto	Documentación	OSGeo Live	Otros
GRASS GIS	GRASS docs	GRASS qs	Tutoriales; Primera vez con GRASS
uDig	uDig docs	MapServer qs	Canal de vídeos en YouTube
OPTIKS	Notas de instalación y uso		
openJUMP	openJUMP docs	openJUMP qs	
QGIS	QGIS docs	QGIS qs	Tutorial (español); Tutorial oficial (inglés)
GeoDa	GeoDa docs		Tutoriales y vídeos demostrativos
gvSIG	gvSIG docs	gvSIG qs	Videotutoriales
SAGA GIS	SAGA GIS docs	SAGA GIS qs	Tutoriales
TileMill	TileMill docs	TileMill qs	Guía de la interfaz

CAPÍTULO 7

Dispositivos móviles

7.1 Autores

- Santiago Higuera @santiagohiguera

7.2 Introducción

Incluiremos en la categoría de dispositivos móviles los paquetes de software que se pueden utilizar desde dispositivos móviles del tipo de los *smartphones* y tabletas.

La posibilidad de disponer de GPS en los dispositivos móviles, ha convertido a estos aparatos en herramientas muy valiosas para su utilización como navegadores, plataformas de geomarketing, sensores móviles, y en general como plataforma para todo tipo de aplicaciones en los que la geolocalización sea un componente de valor.

Actualmente existen dos sistemas operativos para dispositivos móviles que abarcan la gran mayoría de los dispositivos funcionando: Android e iOS. El primero, Android, es el más abierto de los dos, y dispone de un buen número de aplicaciones y librerías que sí se ofrecen en modalidad Open Source permitiendo su utilización en desarrollos ulteriores.

En ambos casos el sistema operativo subyacente es Linux. En los dispositivos iOS se utiliza una variante del lenguaje C para programarlos: el Objective C. En los dispositivos Android se utiliza una variante del lenguaje Java para programarlos.

Vamos a centrar este artículo en el software y librerías disponible para dispositivos basados en Android, por ser a día de hoy el que dispone de suficientes herramientas con licencias libres para ser utilizadas en el desarrollo de aplicaciones para móviles.

7.3 Software

El sistema operativo Android permite utilizar una variante del lenguaje Java para programar los dispositivos móviles. En este sentido, muchas de las librerías Java disponibles para ordenadores de escritorio se pueden utilizar bajo condiciones de operación Android. Sería el caso, por ejemplo, de la librería *Java Topology Suite (JTS)*, que ha sido tratada en la sección correspondiente a librerías.

Existen por otra parte librerías específicas desarrolladas para Android, como es el caso de *Mapsforge*, pensada para visualizar información cartográfica en dispositivos Android.

La utilización de Java permite también que librerías inicialmente pensadas para ser utilizadas en dispositivos móviles puedan ser reutilizadas en ordenadores de escritorio bajo entorno Java. En general la diferencia está en la parte gráfica de Java: En el caso de ordenadores de escritorio es usual utilizar el entorno Swing que no es compatible con Android.

Name	Year	OSGeo	Live	License	Ohloh	Repository	Tech
OpenLayers	2007	Yes	Yes	2-clause BSD	ohloh	Github	HTML/JS
OSMDroid	2008	-	-	CC3.0BySA	ohloh	Google-code	Android
Mapsforge	2010	-	-	LGPL3	ohloh	Google-code	Android/Java
Leaflet	2010	-	-	2-clause BSD	ohloh	Github	HTML/JS
OsmAnd	2010	-	-	CC ByNcSa3 and GPL-3.0	ohloh	Google-code	Android
GeoPaparazzi	2010	-	-	GPL-3.0+	ohloh	Google-code	Android
gvSIG-Mini	2010	-	-	GPL-2.0+	ohloh	Prodevelop	Android
Graphhopper	2012	-	-	Apache-2.0	ohloh	Github	Android
OsmSharp	2012	-	-	CC ByNcSa3 and GPL-3.0	ohloh	Github	All
Glob3 Mobile	-	-	-	BSD	-	Github	HTML5, Android, iOS -

Importante: Se puede consultar la versión más reciente, así como los enlaces asociados y anotaciones en la sección de Moviles del wiki de OSGeo.

Las principales librerías y programas disponibles en código abierto son:

OpenLayers y LeafLet:

En ambos casos se trata de librerías Javascript cuyo objetivo es la visualización de información cartográfica en entornos de Web-mapping. Los dispositivos móviles disponen de navegadores que interpretan el Javascript, por lo que, en general, es posible utilizar las librerías Javascript para visualizaciones desde dispositivos móviles, sean estos Android, iOS u otros. Las páginas web basadas en OpenLayers o Leaflet permiten su visualización en dispositivos móviles a través del navegador. En el caso de OpenLayers versión 2 hay que tener algunas precauciones al programar las páginas. La futura versión 3 de OpenLayers superará estas limitaciones. El funcionamiento de estas librerías se ha tratado en la sección correspondiente a Librerías.

OSMDroid:

El objetivo de OSMDroid es proporcionar una librería Java-Android para visualizar mapas en dispositivos Android. Ofrece una clase MapView para sustituir a la que viene de serie en Android que permite la visualización de tiles de OpenStreetMap. Se pueden visualizar tiles en modo on-line y en modo off-line. También proporciona clases para visualizar e interactuar con overlays, marcadores y otros.

MapsForge:

Mapsforge es una librería Open Source que permite gestionar la visualización de mapas de OpenStreetMap en los dispositivos Android. Es ligera, ocupa unos 300 Kb. Los mapas tienen un formato vectorial binario que los hace también ligeros. Utiliza un elemento MapView similar al del API de Google. Tiene una buena API para overlays (capas vectoriales superpuestas). Se pueden personalizar los estilos de renderización de los mapas. Tiene una herramienta para crear mapas a medida a través de Osmosis. Algunos puntos débiles serían que no dispone de API para Bubbles y no limita la extensión del mapa.

OsmAnd:

Es una herramienta de navegación y routing para trabajar desde dispositivos Android con cartografía procedente de OpenStreetMap. Puede trabajar online y offline, previa descarga de los mapas. Proporciona instrucciones de navegación de forma visual y mediante voz sintetizada. La navegación y el routing es adaptable a vehículos automóviles, bicicletas o peatones. Permite mostrar la posición GPS del dispositivo y su orientación sobre la cartografía. Se puede mostrar el mapa orientado con el Norte arriba o en modo *head-up* o *proa-arriba*.

GeoPaparazzi:

Permite tomar fotografías y notas georeferenciadas desde dispositivos Android, para poder ser visualizadas posteriormente con otras herramientas GIS. También proporciona una herramienta de tracking durante los recorridos.

gvSIGMini:

Es un cliente visualizador de cartografía para Android. Proporciona clientes WMS y WMS-C. Permite la búsqueda de direcciones y el cálculo de rutas. Trabaja en modo on-line y off-line. Se pueden superponer varias capas.

Proporciona funciones de navegación GPS y posicionamiento por GPS o por red telefónica. Se puede compartir la posición a través de las redes sociales (Twitter, Facebook) o por SMS y eMail. Permite la integración de *Street View*.

Graphhopper:

Es una herramienta de routing que trabaja con datos de OpenStreetMap. Se puede utilizar desde dispositivos Android, a través de su integración con MapsForge. También es posible utilizarlo desde páginas web como un servicio mediante llamadas HTTP desde Java o Javascript. Desde aplicaciones Java de escritorio es posible trabajar con Graphhopper en modo off-line. Se pueden calcular rutas para automóviles, bicicletas o paseos andando. También se pueden crear vehículos personalizados.

OsmSharp:

Es una herramienta para trabajar con cartografía de OpenStreetMap. Permite la visualización de información vectorial y el cálculo de rutas. Se puede utilizar en Android, iOS y WindowsPhone, así como en Linux, Windows y OSX.

Glob3Mobile:

El proyecto Glob3 Mobile, desarrollado por IGO SOFTWARE y ULPGC con el respaldo de organismos públicos y privados españoles. Se trata de un componente para desarrollar mapas en dispositivos móviles, nativo y multiplataforma; permite aplicaciones 2D, 2.5D y 3D, tanto Android como iPhone y en general en cualquier entorno HTML5.

7.4 Puntos calientes

Actualmente hay dos sistemas operativos para móviles que acaparan la mayoría de los dispositivos utilizados. Se trata del sistema iOS de Apple y el sistema Android de Google. El más '*abierto*' es el sistema operativo Android, si bien ninguno de los dos es realmente abierto.

Es importante destacar que, cuando el dispositivo móvil utiliza un navegador web para visualizar información en la red, la distinción entre dispositivos móviles y ordenadores de escritorio se hace más difusa, pues las mismas herramientas y librerías nos sirven para visualizar información en un dispositivo móvil u otro tipo de elemento de computación. Es el caso de las librerías OpenLayers o LeafLet, que nos permiten, con pequeñas diferencias, acceder a la visualización de información cartográfica desde cualquier navegador, sea este basado en móviles o en ordenadores de escritorio.

Otro frente actual es el de los sistemas operativos que tratan de ser funcionales tanto en dispositivos móviles como en ordenadores de escritorio. En esta categoría se incuyen el sistema operativo Ubuntu móvil o el Windows 8 que permiten ser ejecutados en todo tipo de ordenadores. Es previsible que el aumento de potencia de los dispositivos móviles unido a la conveniencia de compatibilidad entre dispositivos, permita en un futuro cercano que la distinción entre sistemas operativos para dispositivos móviles y para ordenadores de escritorio se haga cada vez más difusa.

Es previsible que la visualización 3D vaya ganando protagonismo en los próximos años.

7.5 Curva de aprendizaje y conocimientos previos

La programación de los dispositivos Android se realiza a través de un lenguaje Java propio de Android. Muchas de las librerías disponibles para Java funcionan también en dispositivos Android. Es necesario conocer los fundamentos básicos de la programación Java y, además, conocer la forma de utilizar Java en Android. No es un lenguaje sencillo y además la variedad de dispositivos y de tamaños de pantallas hace un poco más complicada la programación.

Conocidos los fundamentos de la programación en Android, es posible utilizar las herramientas descritas anteriormente con un pequeño esfuerzo adicional que permita conocer cada una de ellas.

7.6 Documentación

Tabla 7.1: Documentación de proyectos

Proyecto	Home	Documentación	Otros
OpenLayers	OpenLayers Home	Documentation	OL OSGeoLive Quickstart
LeafLet	Leaflet Home	Tutorials	OSGeoLive Quickstart
OSMDroid	Repository	How to get...	[Article: AndroCode]
MapsForge	Mapsforge Home	Mapsforge 0.3.0 Javadoc	Taller Mapsforge Sigte 2013
OsmAnd	OsmAnd Home	OsmAnd repository	...
Geopaparazzi	Geopaparazzi Home	Geopaparazzi Documentation	...
gvSIG-Mini	gvSIG-Mini Home	gvSIG-Mini Documentation	...
Graphhopper	Graphhopper Home	Quickstart users	Quickstart developpers
OsmSharp	OsmSharp Home	OsmSharp documentation	...
Glob3 Mobile	Glob3 Home	Glob3 documentation	About

CAPÍTULO 8

Conclusiones

En este artículo hemos tratado de recoger cuál es el estado actual de las herramientas libres para el procesado de datos geoespaciales. Para ello, hemos agrupado en seis categorías diferentes (**Servidores**, **Webmapping**, **Librerías**, **Clientes**, **Móviles** y **Bases de Datos**) los proyectos que ofrecen soluciones para el almacenamiento, procesado, análisis, publicación, visualiación y, en general, cualquier actividad relacionada con datos geoespaciales.

Tanto los servidores, como los clientes de escritorio y las bases de datos han alcanzado ya una madurez plena y ofrecen una muy alta calidad de productos capaces de competir y superar conocidos paquetes comerciales. Sin embargo, en el caso de los clientes de escritorio se observa que van perdiendo relevancia en favor de proyectos dedicados al Webmapping. Por su parte, las nuevas tecnologías móviles, con la incorporación de sistemas GPS están, incentivando la aparición de software dedicado a geolocalización. También son numerosos los nuevos proyectos enfocados a la manipulación de datos LiDAR.

Python se está convirtiendo en el lenguaje más extendido en el mundo SIG libre. De hecho, existe una tendencia en casi todas las categorías a la utilización de Python, bien sea para el propio desarrollo de los proyectos o bien para dotarlos de una herramienta de *scripting*.

A continuación se incorporan unas conclusiones más detalladas de cada una de las categorías de proyectos tratados en este escrito.

8.1 Servidores

Tras revisar el estado de los principales proyectos encargados de ofrecer servicios geoespaciales, se aprecia una elevada madurez de la mayoría de productos: no han aparecido proyectos nuevos relevantes en los últimos dos años y los proyectos más veteranos siguen en pleno desarrollo, sin dejar de ofrecer nuevas funcionalidades y mejoras. Cabe destacar también la relevancia que van adquiriendo proyectos desarrollados en Python, uniéndose como base tecnológica a la de proyectos más veteranos escritos en C/C++ y Java.

8.2 Webmapping

Tras revisar el estado de los principales proyectos encargados de la visualización interactiva de mapas en el navegador, se aprecia una elevada aparición de distintos proyectos que ofrecen la integración de componentes variados integrándolos en una aplicación mucho más completa y robusta. Además hay que destacar la aparición de frameworks que hacen que la interacción con el usuario sea más interactiva, aprovechando los beneficios de HTML5 y CSS3.

PD: Opinión personal Para mí creo que la evolución de las librerías destinadas al webmapping han avanzado gracias a la integración de componentes externos haciéndolas evolucionar a portales muchos más completos y añadiéndole funcionalidades que a primera instancia no se incluían, dándole a las herramientas otra dimensión más.

8.3 Librerías

La aparición de librerías en Python y soluciones de scripting sencillas aplicables a diversas plataformas y lenguajes de programación, es lo más destacable en los últimos tiempos. En ese ámbito estarían Shapely o GeoScript. Cada vez más son las librerías que intentan ofrecer soluciones para trabajar con datos LiDAR y, aunque todas ellas ofrecen lectura y escritura de datos, sólo algunas aportan capacidad de procesado y análisis. La librería Geotools es una librería Java consolidada y con un desarrollo muy activo que sigue añadiendo componentes, tanto para procesamiento geoespacial como para visualización. Las librerías para el cálculo de rutas están congiendo más relevancia debido a la oferta de datos públicos.

8.4 Clientes

Debido a la larga trayectoria de todos los clientes de escritorio, es destacable su gran madurez en todos sus aspectos. De hecho no ha aparecido ningún nuevo proyecto en los últimos años y todos han seguido un desarrollo constante. Cabe destacar, quizás, el caso de QGIS cuya popularidad ha ido en aumento convirtiéndose en el SIG de Escritorio libre más popular. En gran parte, esto se ha sido debido a la integración de Python tanto para la creación de *script* como para la implementación de nuevos módulos, que le confieren una gran versatilidad.

8.5 Móviles

La disponibilidad del GPS en los dispositivos móviles, hace que se prevea un crecimiento del mercado de aplicaciones móviles basadas en la geolocalización. La utilización de aplicaciones Web a través del navegador del dispositivo móvil se convierte en una solución muy eficaz al problema de la multiplataforma. En cuanto a soluciones nativas en Java-Android para aplicaciones de visualización cartográfica y geolocalización, tanto OSMDroid (mapas de tiles) como Mapsforge (mapas vectoriales) tienen un desarrollo activo y evolucionan a buena velocidad. La reciente aparición de Graphhopper con soluciones móviles de routing apoyándose en Mapsforge promete dar un impulso a ambos proyectos.

8.6 Bases de Datos

El ecosistema de bases de datos geográficas está bastante maduro, ofreciendo tanto una gran calidad como una variedad de tipos; casi todos ellos muy alineados con la compatibilidad OGC, haciendo que el cambio de una base de datos a otra sea prácticamente transparente. En cuanto a las bases de datos NoSQL, están experimentando una interesante evolución que habrá que seguir de cerca los próximos años.

CAPÍTULO 9

Acerca de este documento

En 2007, durante las primeras Jornadas de SIG Libre, Miguel Montesinos y Jorge Sanz presentaron una comunicación titulada *Panorama actual del ecosistema de software libre para SIG* [Panorama07]. En esta presentación se hacía una rápida revisión a los principales proyectos además de presentar la fundación OSGeo, la distribución geográfica de los principales grupos de desarrollo y unas ichas que resumían los datos más relevantes de los proyectos evaluados. Un año más tarde esta misma comunicación se volvió a realizar (actualizada) como sesión plenaria [Panorama08]. El trabajó siguió actualizándose durante un tiempo en forma de artículo que se publicó en un par de revistas (como [Novatica09]) y como una web en el wiki de Prodevelop. Han pasado algunos años y el panorama ha cambiado, hay nuevos proyectos, otros han ido desapareciendo y la mayoría se han ido actualizando y madurando. Es momento por tanto de retomar aquel trabajo y darle un enfoque un poco más colaborativo, pasando de un esfuerzo de dos personas que trabajan n la misma organización a un grupo más amplio, con perfiles más variados y por tanto, ofreciendo diferentes visiones y sensibiliades.

Este trabajo se enmarca en las octavas Jornadas de SIG Libre, yendo más allá de la presentación de un artículo y una charla durante el evento para intentar tener un mayor recorrido, intentando ser un verdadero proyecto de documentación, abierto a cualuier aportación y utilizando prácticas habituales en proyectos de *Software* Libre.

La sección de *introducción* explica en detalle la metodología con la que hemos abordado el trabajo, con qué herramientas y canales de comunicación se trabaja, cómo se coordina el grupo y se toman las decisiones, etc. Después sucesivamente se irán presentando los proyectos agrupados funcionalmente para finalmente acabar con unas *conclusiones*.

Este documento estará siempre accesible vía web en http://panorama-sig-libre.rtfd.org/, y gracias al soporte de Read the Docs está disponible también en otros formatos para su descarga (PDF y EPUB entro otros). Nuestra intención es mantenerlo vivo, mejorarlo y completarlo en la medida de nuestras posibilidades y hacer de él un punto de entrada para cualquier persona que se quiera iniciar en el ámbito de las tecnologías geoespaciales, a veces complejo por la variedad de productos, estándares y formatos. Por supuesto el proyecto está abierto a nuevas colaboraciones a través de los mecanismos expuestos en la *introducción*.

Marzo de 2014

Los autores

9.1 Autores

- Alejandro Díaz @alediator · Ingeniero de Software en GeoSolutions S.A.S · http://about.me/alediator
- Jorge Sanz @xurxosanz · Analista en el equipo de tecnologías espaciales de Prodevelop S.L. · http://jorgesanz.net
- Josep Sitjar @JosepSitjar · Técnico en SIG en el SIGTE y profesor del máster UNIGIS
- María Arias de Reyna @delawen· Ingeniera especialista en Spatial IT de GeoCat · http://delawen.com
- Moisés Arcos @moiarcsan · Ingeniero Técnico Informático · Desarrollador GIS en Emergya · http://moisesarcos.wordpress.com/
- Roberto Antolín @Tolanss · Investigador y desarrollador de aplicaciones LiDAR en Forestry Commission

- Santiago Higuera @santiagohiguera . Ingeniero de caminos y master en Sistemas de Ingeniería Civil, CEO en MercatorLab S.L.

9.2 Licencia

[GeoServerIGN] Using GeoServer at IGN (the French National Mapping Agency) to create new digital maps

[Panorama07] Panorama actual del ecosistema de software libre para SIG.

[Panorama08] Panorama actual del ecosistema de software libre para SIG (2a edición)

[Novatica09] Geographic Information Systems Cepis UPGRADE - Volume 2009 - Issue II