Definición e implementación de soluciones basadas en APIs Universales para la integración de estándares OGC.

P. López Escobés, J.P. de Castro Fernández, R. García Martín, M. J. Verdú Pérez, L. Regueras Santos y E. Verdú Pérez

Laboratorio de Infraestructuras de Datos Espaciales (IDELab), Escuela Técnica Superior de Ingenieros de Telecomunicación, Campus Miguel Delibes, Universidad de Valladolid, Camino del Cementerio s/n, 47011 Valladolid, plopesc@ribera.tel.uva.es, {juacas, ricgar, marver, luireg, elever}@tel.uva.es.

RESUMEN

Debido al atractivo que representan los clientes ligeros de mapas a la hora de poder representar información geográfica en cualquier sitio web de forma sencilla, han surgido gran cantidad de alternativas tecnológicas que puede llegar a desorientar al desarrollador, ya que cada una de ellas ofrece una interfaz diferente y resulta complicado conocer varias de ellas para poder comparar los resultados que se obtendrían. Ante este escenario, los desarrolladores se decantan en muchas ocasiones por la opción más extendida sin tener en cuenta al resto.
Por ello surge la idea de desarrollar un api universal de clientes de mapas con la que se puedan integrar mapas de diferentes clientes utilizando la misma interfaz. Ya existe una librería que pretende lograr este objetivo, Mapstraction. Sin embargo, la filosofía de esta librería es de "mínimo común múltiplo". Ofrece una interfaz única sólo para las funcionalidades más genéricas y que se incluyen en la mayoría de los clientes. De esta forma, los desarrollos que se pueden llevar a cabo son bastante limitados.
Dada esta limitación, este trabajo pretende, a partir de la interfaz de Mapstraction, modificar la filosofía de la librería. Se desea que ofrezca la posibilidad de llevar a cabo desarrollos más ambiciosos, sobre todo basados en el soporte a los estándares del Open Geospatial Consortium. Estos objetivos se logran principalmente haciendo que la propia librería sea la encargada de asumir algunas de las funcionalidades que algunos de los clientes no implementan de forma nativa. De esta forma se ha conseguido que desde la librería se puedan ofrecer nuevas funcionalidades, como la carga de capas WMS y WFS, consiguiendo que, este tipo de APIs universales puedan resultar mucho más atractivas a los posibles usuarios.

Palabras clave: cliente ligero, estándar, *Javascript, Mapstraction, Open Geospatial Consoritum.*

ABSTRACT

Due to the attractive that represent thin map clients represent when show geographic information in any web site easily, there have emerged many alternatives. Developers can became disoriented in time to decide between different clients. Each one has a different interface and it is difficult to know several of them to compare the results to be obtained. Therefore, developers usually adopt the most extended client regardless of the rest.
Thus arises the idea of developing a map clients universal API that allows to develop maps with different clients using the same interface. There is already a library that seeks to achieve this goal, Mapstraction. However, the philosophy of this library is "least common multiple". This interface only allows the most generic functionalities that are implemented by all the clients. In this way, developers are fairly limited.
Due to this limitation, this work pretends, from Mapstraction interface, change the philosophy of the library. It should offer the possibility to carry out more ambitious developments, mainly giving support to the Open Geospatial Consortium standards.
These objectives are achieved mainly by the library itself, which assumes some of the functionalities that clients can't offer. In this way, library now can load WMS or WFS layers, noting that with this new philosophy, this kind of universal APIs can be much more attractive to potential users.

Key words: *Javascript, Mapstraction, Open Geospatial Consortium, thin client.*

INTRODUCCIÓN

El acceso masivo a Internet, la abundante oferta de servicios públicos de productos de información, la popularización del uso de dispositivos portátiles de comunicación (tales como PDA y GPS), las nuevas comunidades digitales que se forman bajo algún tópico de interés particular y los servicios geográficos básicos brindados tanto por grandes empresas (tales como Google, Yahoo y Microsoft) como por parte de otras iniciativas libres como OpenLayers u OpenStreetMap son algunos de los factores que en conjunción dan el marco para el desarrollo de la neogeografía [1].

Neogeografía es el término acuñado para referirse a las aplicaciones sociales derivadas del movimiento Web 2.0 que se basan en el uso de mapas. Para Turner [2], la definición precisa es "técnicas y herramientas geográficas utilizadas para actividades personales o utilizadas por personas o grupos de no expertos, de manera informal". De alguna manera se está hablando de la socialización de la geografía, a partir del hecho de que la construcción de mapas o productos geográficos básicos está al alcance de casi cualquier usuario con mínimo entrenamiento en herramientas de publicación digital y que potencialmente millones de personas pueden acceder a tales productos de información y servicios.

El problema que históricamente ha perjudicado el desarrollo y evolución de las tecnologías incipientes, y esta no es una excepción, es la competencia descoordinada entre diferentes alternativas tecnológicas. Actualmente hay una serie de APIs (*Application Programming Interface*) para la elaboración de aplicaciones de neogeografía [3] que obligan a los desarrolladores a la adopción de un estilo de programación determinado y una serie de decisiones de diseño que hacen que el resultado esté fuertemente ligado al proveedor original. De esta forma se produce una

competición despiadada por imponer un determinado proveedor de *widgets* que a la larga genera usuarios cautivos [4].

Por este motivo surgió la idea de la creación de un API "universal y políglota", que sea la encargada de introducir una capa de abstracción para poder controlar distintos clientes de mapas con un mismo API. De esta forma se permite el intercambio entre los clientes de mapas para poder utilizar el más apropiado en cada caso, sin necesidad de aprender un nuevo API.

Este tipo de APIs también protegen al desarrollador de estos *mashups* frente a los arbitrarios cambios que puedan aparecer en los clientes nativos, ya que permite soportar diferentes versiones de un mismo cliente. Sirva como ejemplo la reciente aparición de la nueva versión de Google Maps, Google Maps Javascript v3, ha hecho que un gran número de desarrolladores esté migrando paulatinamente a la nueva versión. Con la utilización de un API universal, estos cambios no serían necesarios, sino que una vez que la comunidad dé soporte a la nueva versión, solamente con cambiar una línea de código dentro de los sitios web serían compatibles con la nueva versión de Google Maps.

Ya existe un proyecto orientado a obtener ese API universal y políglota, denominado Mapstraction [5,6], y es el utilizado como base para llevar a cabo este trabajo, cuyo concepto se puede observar en la Figura 1. Sin embargo el diseño de este API todavía no está definido y todavía son necesarios más esfuerzos de investigación en este sentido.

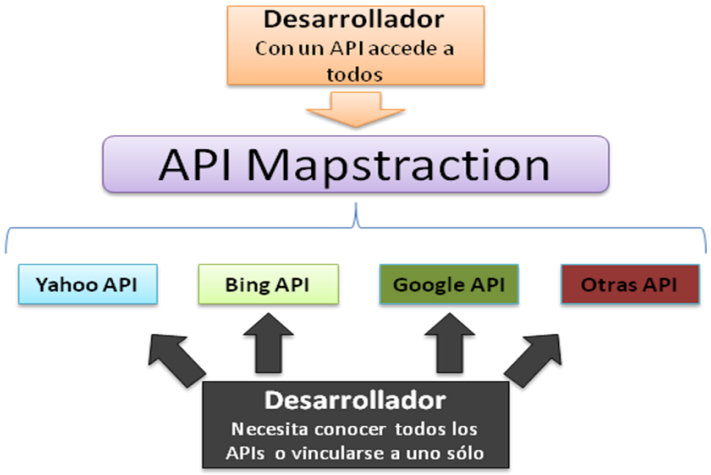

Figura 1 Esquema del concepto de abstracción que plantea Mapstraction

A la hora de diseñar un API de este tipo, se pueden seguir dos tipos de alternativas, la primera de ellas consiste en realizar un API de "mínimo común múltiplo", que sólo ofrezca las funcionalidades genéricas comunes a todos los clientes a integrar, y otra más ambiciosa que pretenda añadir nuevos servicios a las distintas APIs nativas para conseguir que los resultados obtenidos por la unión de todos sean mejores que los de cada uno por separado. En la Figura 2 se puede observar un esquema de ambas aproximaciones. En primer lugar se observa cómo se obtiene un API pequeña a partir de las partes comunes a todos, mientras que en el segundo caso, añadiendo pequeñas partes a todos ellos se puede obtener un API mucho mayor, lo que repercute en un mejor servicio.

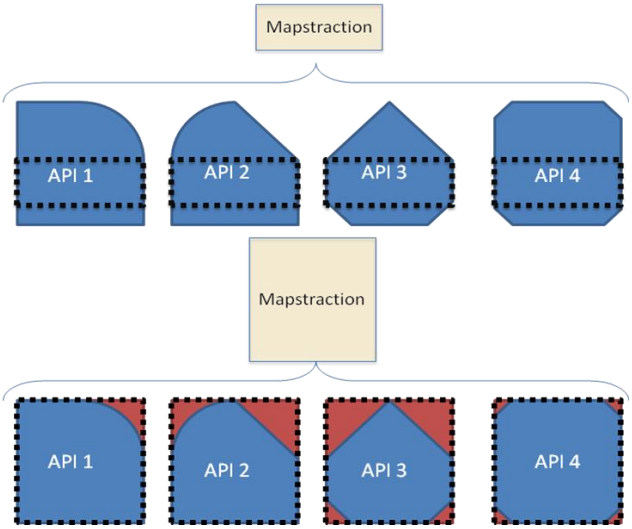

Figura 2 Esquema de las distintas estrategias a seguir a la hora de realizar un API universal. Este trabajo pretende demostrar que la segunda de ellas resulta más útil y eficiente

En Mapstraction, hasta el momento se había optado por la primera de las estrategias, ya que en el caso de las funcionalidades básicas de los mapas, la inmensa mayoría de los proveedores las implementan sin problemas. En este trabajo se propone dar un paso más, haciendo que este API de abstracción también sea la encargada de mejorar los puntos débiles de algunas de las APIs integradas, consiguiendo de esta forma que utilizando la librería se puedan realizar aplicaciones que no se puedan llevar a cabo utilizando su API nativa.

Dado que muchos de los clientes de mapas existentes están orientados al desarrollo de aplicaciones de consumo general con pocos requisitos técnicos geográficos, no dan soporte a algunos de los estándares propuestos por la OGC[1] (*Open Geospatial Consortium*). Se pretende lograr que la librería sea la encargada de conseguir que estos clientes de mapas sean capaces de aceptar los estándares establecidos, como por ejemplo el WMS (*Web Map Service*) o el WFS (*Web Feature Service*), siempre que sea posible. De esta forma, se podrían utilizar todos los clientes integrados de forma intercambiable y ofreciendo mejores prestaciones que en su versión nativa.

En este trabajo se pretende mejorar la extensión IDELabMapstraction Interactive [7] para que aprovechándose de las características de los clientes ligeros de mapas incrustables en páginas web se puedan desarrollar aplicaciones profesionales que hasta ahora tan sólo se podían desarrollar con sistemas GIS específicos.

Al mismo tiempo se pretende que Mapstraction se convierta en un referente por su orientación no sólo a conseguir la transparencia en cuanto al cliente de mapas utilizado por el desarrollador, sino en cuanto a que busca lograr la transparencia entre los formatos de las fuentes de datos espaciales estandarizadas y los propios de cada uno de los proveedores.

El resto de este trabajo se estructura de la siguiente forma. En primer lugar se describe el estado actual del campo de los *mashups* de mapas y con los diferentes actores existentes y sus respectivas funciones. A continuación se describe el trabajo desarrollado, indicando las diferentes fuentes de datos que se han integrado en el IDELabMapstraction a lo largo de este trabajo. Finalmente se exponen las principales conclusiones de este trabajo, así como sus posibles líneas futuras.

[1] http://www.opengeospatial.org

ESTADO DEL ARTE

Al albor de esta tendencia de generación de *mashups* basados en mapas, han surgido gran cantidad de clientes ligeros incrustables en cualquier Web. Esta proliferación de clientes ha desencadenado una feroz batalla por hacerse con un hueco en el mercado, en función de los servicios que deseen ofrecer. Esta competición entre los diferentes proveedores de información geográfica ha traído consigo una simplificación a la hora de ofrecer sus diferentes APIs

Sin embargo, esta paridad entre las características de unos y otros no se refleja a la hora de su utilización, donde Google Maps es el más fuerte con mucha diferencia. Este hecho se produce porque hay gran cantidad de desarrolladores que no se ocupan de estudiar las diferentes alternativas existentes antes de llevar a cabo una aplicación y se inclinan directamente por la opción mayoritaria o con la que ya hayan tenido una experiencia anterior.

En [8] se hace un análisis del potencial que tienen los clientes de mapas, dada la gran popularidad que han adquirido en los últimos años. Una de las labores que se llevan a cabo en ese trabajo es realizar una comparativa entre el cliente más utilizado globalmente, Google Maps y 11 alternativas de código abierto, mucho más minoritarias. En esta comparativa se demuestra que todos ellos tienen sus puntos fuertes y débiles, incluso alguno de ellos obtiene mejores resultados que Google Maps.

Sin embargo, esta paridad entre las características de unos y otros no se refleja a la hora de su utilización, donde Google Maps es el más fuerte con diferencia. Este hecho se produce porque hay gran cantidad de desarrolladores que no se ocupan de estudiar las diferentes alternativas existentes antes de llevar a cabo una aplicación y se inclinan directamente por la opción mayoritaria o con la que ya hayan tenido una experiencia anterior.

Mapstraction es una librería en lenguaje Javascript de código abierto que ofrece una capa de abstracción entre distintos clientes de mapas, con el fin de ofrecer para todos ellos el mismo API. Dado que no todos los clientes ofrecen las mismas funcionalidades, también cubre algunas de las deficiencias existentes en el API de algunos de los clientes integrados en la librería.

Hasta el momento, Mapstraction ha apostado por una filosofía de “mínimo común múltiplo” en la que resulta sencillo realizar aplicaciones básicas con los clientes de mapas integrados en la librería, sin embargo, para realizar tareas más complejas y profesionales, era necesario que ofreciera más y mejores servicios para poder

Sin embargo, ya en [9] se observó que para llevar a cabo tareas más específicas iba a ser necesario optar por la segunda de las estrategias, ya que sino el API iba a quedar estancada, o con diferencias notables entre unos clientes y otros. Existen funcionalidades más complejas que sólo las permiten algunos de los clientes, lo que supone un reto para la librería, ya que ella debe ser la encargada de cubrir los huecos que dejan las APIs de los diferentes clientes para conseguir que todos ellos ofrezcan desde la librería comportamientos similares.

Los estándares OGC han llevado la interoperabilidad a un ámbito en el que cada proveedor tecnológico ofrecía su propio interfaz de acceso a la publicación de cartografía provocando el conocido como *vendor lock-in*. Mediante los diferentes estándares se permite desacoplar la aplicación que produce la información de los clientes que la consumen multiplicando las aplicaciones y las posibilidades de explotación de dicha información en todo tipo de sistemas, pertenezcan o no a una Infraestructura de Datos Espaciales.

METODOLOGÍA

Tomando como punto de partida el código existente en la librería tras la integración de IDELabMapstraction Interactive [7], se ha procedido a la ampliación de esta extensión para que los clientes integrados en esta primera aproximación vean aumentadas las posibilidades que ofrecen dentro de este API. Tras esta primera versión de la extensión se detectaron varias deficiencias dentro de Mapstraction a la hora de desarrollar aplicaciones de mayor complejidad y que lleven consigo la utilización de servicios ofrecidos por servidores de ámbito más profesional.

En este trabajo se pretende mejorar la extensión IDELabMapstraction Interactive para que aprovechándose de las características de los clientes ligeros de mapas incrustables en páginas web se puedan desarrollar aplicaciones profesionales que hasta ahora tan sólo se podían desarrollar con sistemas GIS específicos. En esta sección se describirá el proceso seguido para conseguir este objetivo tomando como punto de partida el final del desarrollo de la primera versión de IDELabMapstraction Interactive.

El trabajo desarrollado se organiza en torno a cuatro líneas de actuación principales:

- En primer lugar se trata la integración de los formatos KML y GeoRSS[2], que dada su similitud en cuanto a formato se mapean mediante la clase *XMLLayer*.
- El segundo formato a integrar es el WFS, que se integra a través de la clase *WFSLayer*, cuyo proceso de integración ha supuesto un interesante reto.
- El tercer formato a integrar es el estándar WMS, mapeado a través de la clase *WMSLayer*. Ha resultado un proceso complejo, dado que varios de los clientes no lo soportaban de forma nativa.
- Soporte de capas teseladas adicionales: Finalmente se trata la integración del soporte para capas teseladas, como las que se pueden obtener de cachés espaciales, a través de la clase *TileLayer*.

La labor de diseño de este tipo de APIs debe ser muy cuidadosa [10,11], de manera que ofrezca los servicios necesarios para cualquier desarrollador interesado en ese campo de forma sencilla, ya que si el API ofrecida no cumple con las expectativas de los desarrolladores que van a hacer uso de ellas, su utilización se verá muy reducida y no se llegará a la masa crítica [12] de usuarios necesaria para lograr que el API Universal se convierta en una alternativa seria.

Integración de servicios GeoRSS y KML

Dentro de la larga lista de retos que ofrecía la ampliación de las funcionalidades que ofrece la librería, el siguiente que se planteó fue el de la integración de servicios GeoRSS y KML [13]. En la versión inicial de la librería ya se optaba por una básica integración de estos formatos de datos, pero en una versión muy básica en la que sólo se indicaba la URL de la que obtener el fichero XML a mostrar, sin poder mantener el control del mismo a la hora de por ejemplo, dejar de mostrarlo en un determinado momento. Este tipo de integración era un tanto deficiente, por lo que se decidió dar un soporte basado en una clase denominada *XMLLayer*.

La clase *XMLLayer* guarda la referencia a una capa KML o GeoRSS y permite mostrarla y ocultarla sobre el mapa de forma intuitiva con simples métodos *addXMLLayer* y *removeXMLLayer.* Con esta implementación, se refuerza para el

[2] http://georss.org

desarrollador el concepto de capa que se muestra sobre el mapa y que se puede ocultar o mostrar, como se puede observar en la Figura 3.

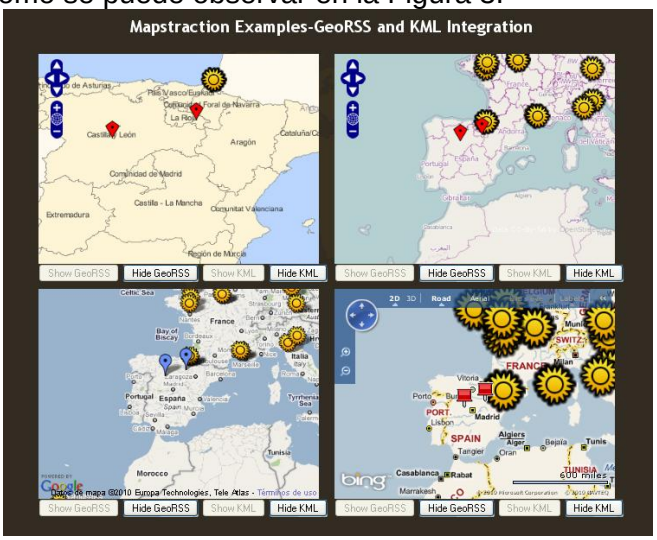

Figura 3 Demo de la integración de servicios KML y GeoRSS en IDELabMapstraction de forma transparente para cada uno de los clientes integrados

Integración de servicios WFS

El siguiente reto dentro de los desarrollos planeados para este trabajo es el de proporcionar el soporte necesario para que los clientes puedan implementar los protocolos WFS y WFS-T [14]. Este protocolo es el encargado de ofrecer una capa vectorial que puede representar puntos líneas o superficies como geometrías, denominadas *features*.

De entre todos los clientes de mapas que se engloban dentro de este trabajo, tan sólo OpenLayers es capaz de ofrecer este soporte, lo que hace que a esta parte del trabajo se hayan dedicado grandes esfuerzos por las dificultades en cuanto a diseño y arquitectura que se han ido presentando. Dada esta situación, la solución más sencilla y factible consiste en utilizar para el caso de OpenLayers su soporte nativo y para el resto de clientes diseñar un sistema que sea capaz de ofrecer los resultados esperados de forma programática, aunque ofreciendo al desarrollador una misma interfaz.

OpenLayers no sólo es un visor de mapas, sino que también ofrece un soporte muy completo para interacciones con servidores de mapas. Por lo tanto, la primera aproximación consiste en utilizar el soporte de OpenLayers para poder ofrecer el servicio al resto de clientes, ocultando al desarrollador que internamente se esté utilizando esta implementación. Después de un estudio de los diferentes patrones de diseño que se pueden utilizar en este tipo de situaciones [15,16], se decidió que la mejor opción consistía en crear una clase adaptadora para poder utilizar el soporte WFS desde otros clientes.

En este caso, el diseño se hará de la siguiente manera, aprovechando la clase OpenLayers.Layer.Vector, que es la encargada de implementar el protocolo WFS, y que se comunica con la clase OpenLayers.Map para obtener los datos necesarios del mapa, como el *bounding box*, la proyección o la resolución. Esta clase será el cliente en el diseño que se pretende. El papel del *adaptor* consiste en hacer que en lugar de un objeto OpenLayers. Map, se puedan incluir otros clientes de mapas, obtener sus datos y pasárselo a la capa, de esta forma, esta no distingue si el cliente utilizado es OpenLayers u otro. En la Figura 4 se puede observar un esquema de este diseño.

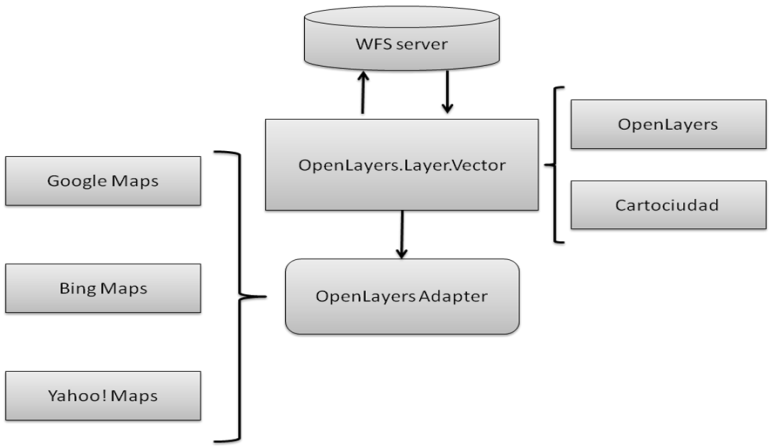

Figura 4 Esquema del diseño del patrón *adaptor* en IDELabMapstraction para la integración de capas WFS en los clientes que no las integran de forma nativa

Una vez desarrollada e implementada esta clase adaptadora, es posible asociar una capa WFS auxiliar a cualquiera de los proveedores que se incluyen en Mapstraction, de esta forma, será esta capa la encargada de comunicarse con el servidor para obtener las *features*, así como para informarle de los cambios que se hayan dado para guardarlos en el servidor. Con esta implementación se reutiliza el código de la clase de la librería OpenLayers y no es necesario desarrollar una clase similar que se encargara de mantener el contacto con el servidor, por lo que se ahorra tiempo de desarrollo que se puede dedicar al siguiente reto que se plantea, comunicar a los distintos clientes de Mapstraction con el servidor de *features* a través de la capa auxiliar a través de un API. En la Figura 5 se pueden ver los resultados obtenidos de esta integración.

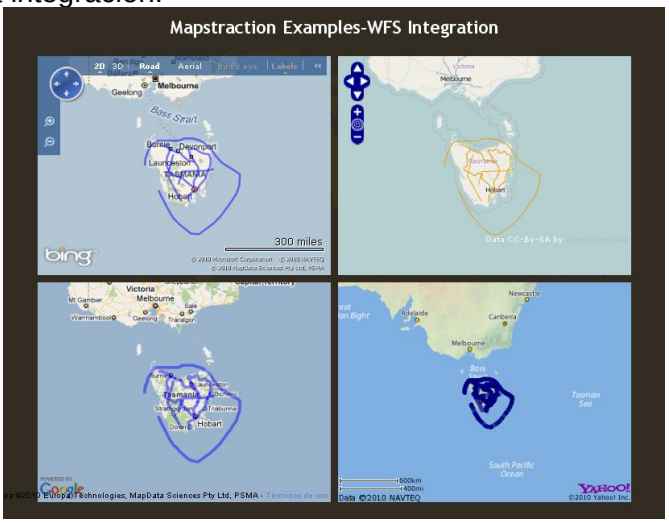

Figura 5 Integración de capas WFS dentro de diferentes clientes de mapas de forma transparente gracias a IDELabMapstraction

Integración de servicios WMS

La siguiente fuente de datos geográficos a integrar dentro de Mapstraction es la ofrecida por los servidores WMS [17]. Este servicio permite el acceso a imágenes generadas dinámicamente a través de una simple interfaz (*getMap*). Estas imágenes están definidas por su tamaño y por el formato de la imagen. Las imágenes son generadas a partir de la información geográfica almacenada en un servidor de mapas.

Este servicio también puede proporcionar la descripción de la estructura y tipo de los datos asociados a las *features* (*describeFeature*). También permite obtener la información asociada a cada una de las *features* mostradas en la capa (*getFeatureInfo*) [18].

Pese a que este servicio es básico para poder mostrar distintas capas en cualquier cliente de mapas, las diferentes políticas de distribución de los clientes integrados hacen que este proceso no sea tan sencillo como aparentemente pueda parecer. La principal diferencia radica en que existen clientes *OpenSource* que no están vinculados a ningún proveedor de mapas, mientras que los clientes propietarios se vinculan a un determinado proveedor.

El primero de los casos es el que presenta OpenLayers, que no presenta ningún tipo de problema para poder mostrar cualquier tipo de capa WMS. Al no estar vinculado a ningún proveedor ofrece un completo soporte para poder integrar cualquier capa WMS sin ningún tipo de inconveniente de forma sencilla.

El segundo tipo de clientes es al que pertenecen Yahoo, Google o Bing. Estos clientes propietarios por defecto muestran sus propias capas asociadas. Al mismo tiempo, no les interesa que se superpongan capas WMS a las que ofrecen. Por ello, parte de ellos nos soportan directamente ningún tipo de capa que se superponga sobre las propias (Yahoo), otros no ofrecen esta posibilidad de forma nativa, pero al ofrecer soporte para otro tipo de capas, se pueden implementar diferentes estrategias que permiten mostrar estas capas (Google y Microsoft).

Es por esto que la integración de este tipo de capas haya supuesto un nuevo reto a la hora de integrar nuevas funcionalidades en Mapstraction. Una vez solventados los obstáculos que algunos clientes ponen para la adición de este tipo de capas, será necesario definir una interfaz lo suficientemente flexible para todos ellos. En la Figura 6 se pueden ver los resultados de esta integración.

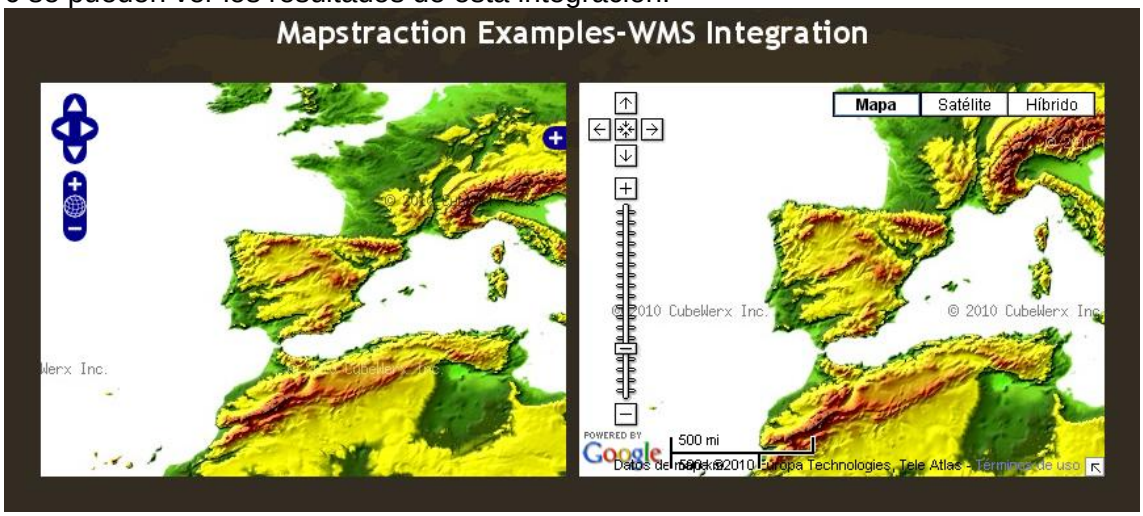

Figura 6 Integración de la misma capa WMS en diferentes clientes de mapas de forma transparente gracias a IDELabMapstraction

La clase *WMSLayer* guarda la referencia a una capa WMS y permite añadirla al mapa con los métodos *addWMSLayer* y *removeWMSLayer.* De esta forma se pueden añadir este tipo de capas de forma sencilla a los mapas de los clientes que dan esta posibilidad. En este apartado se han visto las limitaciones que varios clientes de mapas ofrecen por el hecho de no implementar los estándares de la OGC, sin embargo, se han encontrado distintas alternativas para poder integrarlos de la mejor forma posible y de esta forma aumentar sus posibilidades.

Integración de capas teseladas

La última de las fuentes de datos geográficos a integrar en Mapstraction dentro de este trabajo son las capas teseladas. Mediante la inclusión de este tipo de capas se ofrece la posibilidad de utilizar estos servicios teselados, como el WMTS (Web Map Tile Service) [19] de OGC o el WMS-C (Web Map Service Tile Caching) [20] de OSGeo[3] (*Open Source Geospatial Foundation*) así como capas obtenidas a través de *cachés* que se encargan de optimizar los resultados de estos servicios.

Este tipo de fuente de datos ya se podía utilizar en las primeras versiones de la librería, pero de una forma muy rudimentaria, en la que era difícil gestionar las capas añadidas. Para poder ofrecer un mejor servicio y favorecer la arquitectura de la librería, se optó por integrar esta funcionalidad dentro de la clase *TileLayer*.

La posibilidad de superponer capas teseladas ya está incluida en la mayoría de los clientes de mapas que se tratan en este trabajo, dado que su utilización está bastante extendida. La única excepción es la de Yahoo! Maps, que no incluye este soporte, pero al ser este un comportamiento intrínseco a cada uno de los clientes, no se ha considerado la posibilidad de dar un soporte alternativo para estas capas, por lo que no se podrá conseguir una interfaz totalmente uniforme en este aspecto.

Las diferencias existentes entre las estrategias que cada cliente de mapas utiliza respecto a este tipo de capas ha dificultado el proceso, habiendo tenido que renunciar a alguna de las opciones que ofrece alguno de ellos. Este sacrificio se ha llevado a cabo teniendo en cuenta que la solución de compromiso obtenida es beneficiosa, ya que se permite que todos ellos ofrezcan el mismo API, uno de los objetivos primordiales de Mapstraction. En la **¡Error! No se encuentra el origen de la referencia.** se pueden observar los resultados de esta integración.

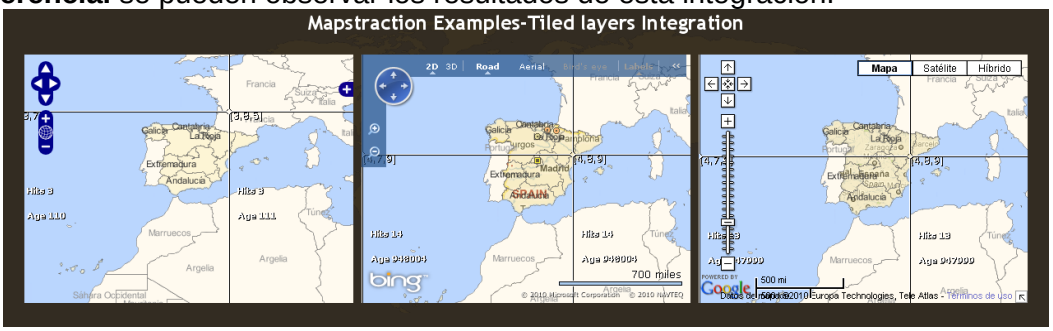

Figura 7 Captura de la demo de integración de servicios teselados. Se muestra una capa de Cartociudad cacheada en tres clientes diferentes

CONCLUSIONES

En este trabajo se ha expuesto la estrategia que se ha seguido para poder conseguir los objetivos deseados, en concreto la mejora de las funcionalidades que ofrece Mapstraction, en especial haciendo que sea capaz de integrar algunos de los principales estándares de la OGC relacionados con este campo. Para llevar a cabo la integración de estas nuevas funcionalidades, ha sido necesaria una importante labor de diseño para poder desarrollar la extensión del API original, así como para conseguir que los clientes que de forma nativa no soportan algunas de estas

[3] http://www.osgeo.org

funcionalidades sean capaces de ofrecer la misma interfaz gracias al diseño desarrollado para solventar estas deficiencias en su API.

Además, se ha trabajado en mejorar la arquitectura de la librería, diseñando clases específicas para abstraer algunos elementos, como las capas GeoRSS y KML, que antes no estaban correctamente integradas dentro de la librería. De esta forma, la arquitectura queda mucho más clara y es más sencillo tener referencias a las distintas capas integradas dentro de los mapas.

Con todas estas funcionalidades integradas dentro del API de Mapstraction, se ha dado un nuevo paso en la consecución del objetivo principal de este trabajo, aumentar el número de funcionalidades integradas dentro de la librería para que fuera más atractiva para los desarrolladores interesados en el campo de los *mashups* de mapas o *Web Mapping.* Con estas nuevas funcionalidades, Mapstraction es capaz de soportar algunos de los estándares de la OGC más importantes, con lo que se abre el abanico de aplicaciones que se pueden crear con la utilización de esta librería, además de aumentar la entidad de las mismas, ya que se pueden conseguir aplicaciones más complejas y profesionales. En definitiva, este trabajo ha servido para realizar un pequeño avance en el estado de la técnica de las APIs políglotas para las Infraestructuras de Datos Espaciales. Se ha demostrando que con ellas se pueden realizar aplicaciones web más complejas de las que hasta el momento se estaban realizando, por lo que se pueden considerar una alternativa más a la hora de llevar a cabo una aplicación GIS online.

Dado que este trabajo pretendía mejorar el API existente y demostrar que cambiando el enfoque de "mínimo común múltiplo" que había hasta ahora se podrían aumentar las posibilidades de la librería, no se ha seguido ningún estudio del rendimiento de la librería o de las pequeñas diferencias que se observan entre los comportamientos de unos clientes y otros. Estas pequeñas diferencias en la manejabilidad de unos clientes y otros son totalmente nativas a los mismos, y tampoco es objetivo de este trabajo hacer que todos los clientes se comporten de la misma manera, sino conseguir un API uniforme para todos ellos que respete las especificaciones funcionales de las diferentes operaciones. Pese a que se ha logrado conseguir en la mayoría de los casos, ha habido otros, como la integración de las capas WMS en Bing Maps en los que no ha sido posible por intentar mantener la simplicidad y portabilidad de la librería.

Durante el desarrollo se han seguido las directrices que se marcan desde la comunidad de Mapstraction, para que la extensión desarrollada se pueda integrar de forma sencilla [9]. Asimismo, los resultados de este proyecto son accesibles para todos los usuarios interesados en su sitio de desarrollo[4], donde además se pueden visitar diversos ejemplos de las mejoras introducidas en la librería.

Haciendo que el uso de Mapstraction sea cada vez más generalizado, será más sencillo que la comunidad adopte esta librería como la básica para la creación de estas aplicaciones. Los desarrolladores que utilicen clientes que todavía no estén incluidos en la librería tratarán de incluirlos para no quedarse fuera de esta iniciativa y la librería aumentará su universalidad.

El proceso de diseño de un API universal y políglota es una tarea de investigación no resuelta como puede ser el desarrollo de ontologías en otras áreas científicas o tecnológicas, lo que hace que sea una vía de investigación atractiva. El hecho de que esta iniciativa esté incluida dentro de un proyecto *OpenSource* invita a que muchos desarrolladores interesados la puedan probar y dar realimentación para los

[4] http://mvn.idelab.uva.es/idelabmapstraction

desarrolladores, así como unirse y colaborar incluyendo nuevos proveedores o funcionalidades.

AGRADECIMIENTOS

El desarrollo de este trabajo ha sido posible gracias a la financiación por parte del Instituto Geográfico Nacional en el marco del Proyecto Conjunto al amparo del convenio de colaboración entre la dirección general del Instituto Geográfico Nacional y la Universidad de Valladolid. Este trabajo ha sido realizado como parte del proyecto CENIT España Virtual (ref. CENIT 2008-1030), cofinanciado por el CDTI, dentro del programa Ingenio 2010 y por el CNIG.

REFERENCIAS

[1] A. Turner, *Introduction to Neogeography*, O'Reilly, 2006.
[2] A. Turner, "Neogeography – towards a definition," *High Earth Orbit* Available: http://highearthorbit.com/neogeography-towards-a-definition/.
[3] M. Haklay, A. Singleton, y C. Parker, "Web Mapping 2.0: The Neogeography of the GeoWeb," *Geography Compass*, vol. 2, 2008, págs. 2011-2039.
[4] W. Cartwright, "Google Maps and mobile devices: Can just one generic design work?," *Revista Brasileira de Cartografia*, vol. 3, Ago. 2008, págs. 215-222.
[5] A. DuVander, *Map Scripting 101: An Example-Driven Guide to Building Interactive Maps with Bing, Yahoo!, and Google Maps*, No Starch Press, 2010.
[6] A. Turner, "Mapstraction - a javascript library to hide differences between mapping APIs." Available: http://mapstraction.com/.
[7] P. López Escobés, R. García Martín, y J.P. de Castro Fernández, "IDELab MapstractionInteractive: API Universal y Políglota," Girona: 2010.
[8] Emanuel Schütze, "Current state of technology and potential of Smart Map Browsing in web browsers," 2007.
[9] P. López Escobés, *Aplicación de técnicas de neogeografía en un sistema de gestión de contenidos web*, Valladolid (Spain): Universidad de Valladolid, 2009.
[10] J. Bloch, "How to design a good API and why it matters," *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, Portland, Oregon, USA: ACM, 2006, págs. 506-507.
[11] M. Henning, "API Design Matters," *Queue*, vol. 5, 2007, págs. 24-36.
[12] P. Ball, *Critical Mass: How One Thing Leads to Another*, Farrar, Straus and Giroux, 2006.
[13] OGC, "Keyhole Markup Language (KML) Implementation Specification," 2009.
[14] OGC, "Web Feature Service (WFS) Implementation Specification."
[15] E. Gamma, R. Helm, R. Johnson, y J. Vlissides, *Design patterns: elements of reusable object-oriented software*, Addison-Wesley Professional, 1994.
[16] J. Resig, *Pro Javascript Techniques*, Apress, 2006.
[17] OGC, "OpenGIS Web Map Service (WMS) Implementation Specification," 2009.
[18] M.Á. Manso Callejo y M.Á. Bernabé Poveda, "The components of an open source-based geoportal," *International Cartographic Conference*, La Coruña: 2005.
[19] Joan Masó, Keith Pomakis, y Núria Julià, "OpenGIS Web Map Tile Service Implementation Standard | OGC®" Available: http://www.opengeospatial.org/standards/wmts.
[20] OsGeo, "WMS Tile Caching," *WMS Tile Caching - OSGeo Wiki* Available: http://wiki.osgeo.org/wiki/WMS_Tile_Caching.