# WMSCWrapper: caché de teselas OpenSource para la aceleración de servicios de mapas teselados.

*R. García Martín, J.P. de Castro Fernández, P. López Escobés, 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, {ricgar, juacas, marver, luireg, elever}@tel.uva.es, plopesc@ribera.tel.uva.es.

**RESUMEN**

*La gran proliferación en el uso de servicios de mapas a través de la Web ha motivado la necesidad de disponer de servicios cada vez más escalables en las Infraestructuras de Datos Espaciales (IDE). Los servicios de mapas basados en teselado se han perfilado como una alternativa escalable frente a los servicios de mapas tradicionales, permitiendo la actuación de mecanismos de cache o incluso la prestación del servicio mediante una colección de imágenes pregeneradas.*

*En este trabajo se presentan los avances realizados en el proyecto Open Source WMSCWrapper [1]: una implementación de una caché de teselas que permite transformar cualquier servidor de mapas estándar en un servidor de teselas capaz de satisfacer la demanda de un elevado número de usuarios simultáneos.*

*Entre las novedades de esta versión cabe destacar la disponibilidad de una arquitectura más modular y flexible, tras la adopción del framework Spring, que facilita en gran medida la incorporación de nuevos componentes para la gestión de la caché.*

*Se han desarrollado diversos módulos para el almacenamiento de las teselas en base de datos, aprovechando el indexado proporcionado por éstas.*

*A las numerosas interfaces de acceso ya disponibles (recomendación WMS de OGC, perfil WMS-C de OSGeo, acceso REST de Google Maps y Microsot Bing Maps, y pasarela KML para Google Earth) se ha añadido soporte para el reciente estándar WMTS (Web Map Tile Service) de OGC.*

*Se han incluido sendos modelos, tanto estadísticos como heurísticos, que tienen en cuenta los patrones de acceso de los usuarios para la realización de una precarga de teselas y mejorar así la Calidad de Servicio (QoS) experimentada por los usuarios.*

***Palabras clave:*** WMSCWrapper, WMS-C, WMTS, caché, software libre.

## INTRODUCCIÓN

Las diversas especificaciones WMS del OGC [2] ofrecen una gran flexibilidad en el servicio. Los parámetros espaciales de las peticiones no están restringidos, lo que hace que cada petición de mapa deba ser atendida en tiempo real mediante un procedimiento, generalmente costoso, que implica acceso a datos de origen, aplicación de estilos, composición de capas y codificación de la imagen comprimida.

Este procedimiento se ha demostrado ineficaz para satisfacer la demanda de algunas aplicaciones de difusión masiva como se expone en [3] tras la experiencia del servidor OnEarth de la NASA. Por este motivo, generalmente los servicios comerciales más populares se prestan con servidores no OGC en los que el espacio geográfico está teselado de acuerdo a una rejilla predefinida y cuyo contenido está frecuentemente pregenerado [4,5].

Cuando un servicio OGC se va a utilizar en un escenario exigente con una información poco parametrizable y estacionaria, se puede utilizar el patrón *proxy web cache* para conseguir una mejoría en la calidad de servicio. El *proxy* es un dispositivo que se interpone de forma preferiblemente *transparente* entre el cliente y el servicio final, como se observa en la Figura 1.

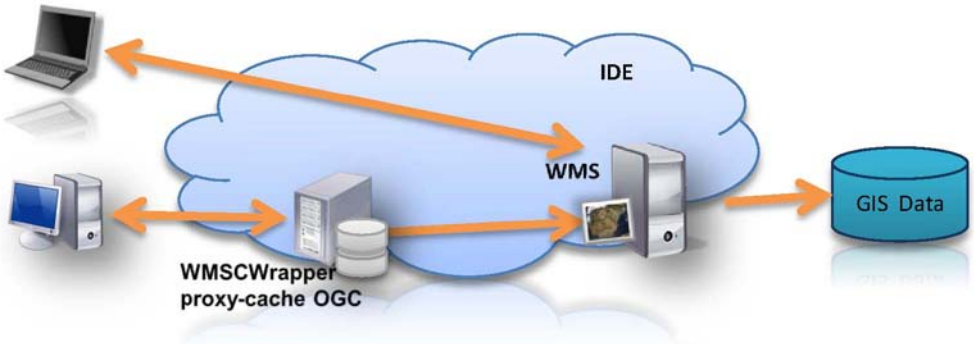


Figura 1: Proxy Web Cache de teselas en la IDE.

En [1] se presentó el prototipo de *caché* de teselas WMSCWrapper, desarrollado en el laboratorio IDELab (**Lab**oratorio de **I**nfraestructuras de **D**atos **E**spaciales) de la Universidad de Valladolid. En el presente documento se recogen los principales avances realizados sobre dicho sistema, el cual ha experimentado reformas significativas. Para ello, en la siguiente sección se presenta la arquitectura de la nueva implementación, para posteriormente pasar a describir los componentes principales del sistema. Finalmente se recogen las principales conclusiones de este trabajo.

## ARQUITECTURA

La aplicación ha sido desarrollada utilizando la herramienta de *software* para la gestión de proyectos Maven de Apache [6]. Esta herramienta facilita la tarea de desarrollo en todas sus fases, desde el proceso de compilación, descarga automática de las dependencias, despliegue de la aplicación o incluso documentación de la misma. Para mejorar la modularidad de la aplicación, ésta ha sido construida como un proyecto multi-módulo de Maven, con un módulo “padre” y varios “hijos” como se muestra en la Figura 2.

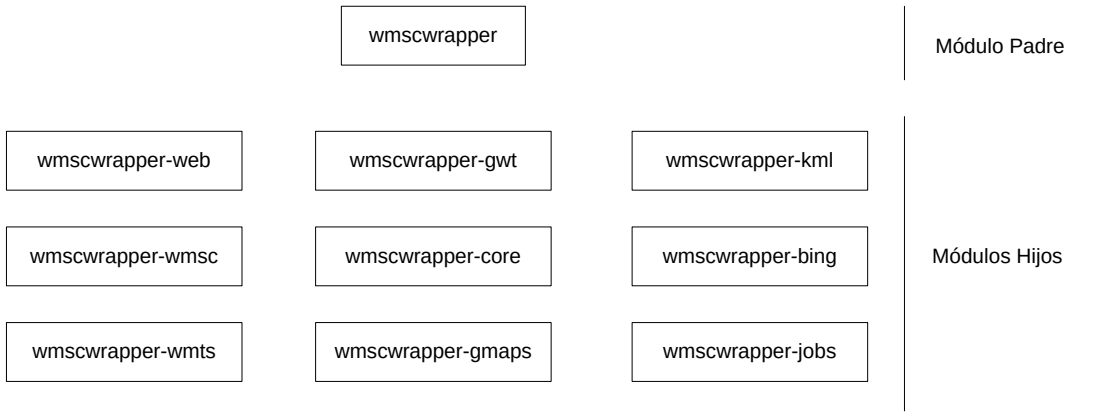


Figura 2: Configuración multi-módulo del proyecto Maven

De esta forma se aíslan los diferentes módulos para su posible reutilización en otros desarrollos, y se facilita la incorporación de nuevos módulos al sistema. Cada módulo recoge una funcionalidad concreta:

- wmscwrapper-core: núcleo del sistema de *caché*, utilizado por el resto de los módulos.
- wmscwrapper-web: módulo web que contiene el servlet principal de la aplicación (DispatcherServlet) y el HandlerMapping para mapear las peticiones hacia los servicios correspondientes.
- wmscwrapper-wmsc: interfaz compatible con la especificación WMS-C de OSGeo.
- wmscwrapper-wmts: interfaz compatible con el estándar WMTS de OGC.
- wmscwrapper-gmaps: interfaz para el cliente Google Maps.
- wmscwrapper-bing: interfaz para el cliente de Microsoft Bing Maps.
- wmscwrapper-kml: pasarela KML para Google Earth.
- wmscwrapper-gwt: interfaz de usuario AJAX desarrollada en Google Web Toolkit.
- wmscwrapper-jobs: tareas de mantenimiento de la *caché*.

Asimismo, se ha hecho uso del *framework* de código abierto Spring [7]. Este *framework* permite el desarrollo de componentes intercambiables, llamados "*beans*", que son orquestados mediante un fichero XML, pudiendo intercambiar componentes sin necesidad de recompilar el proyecto.

**Componentes del sistema**

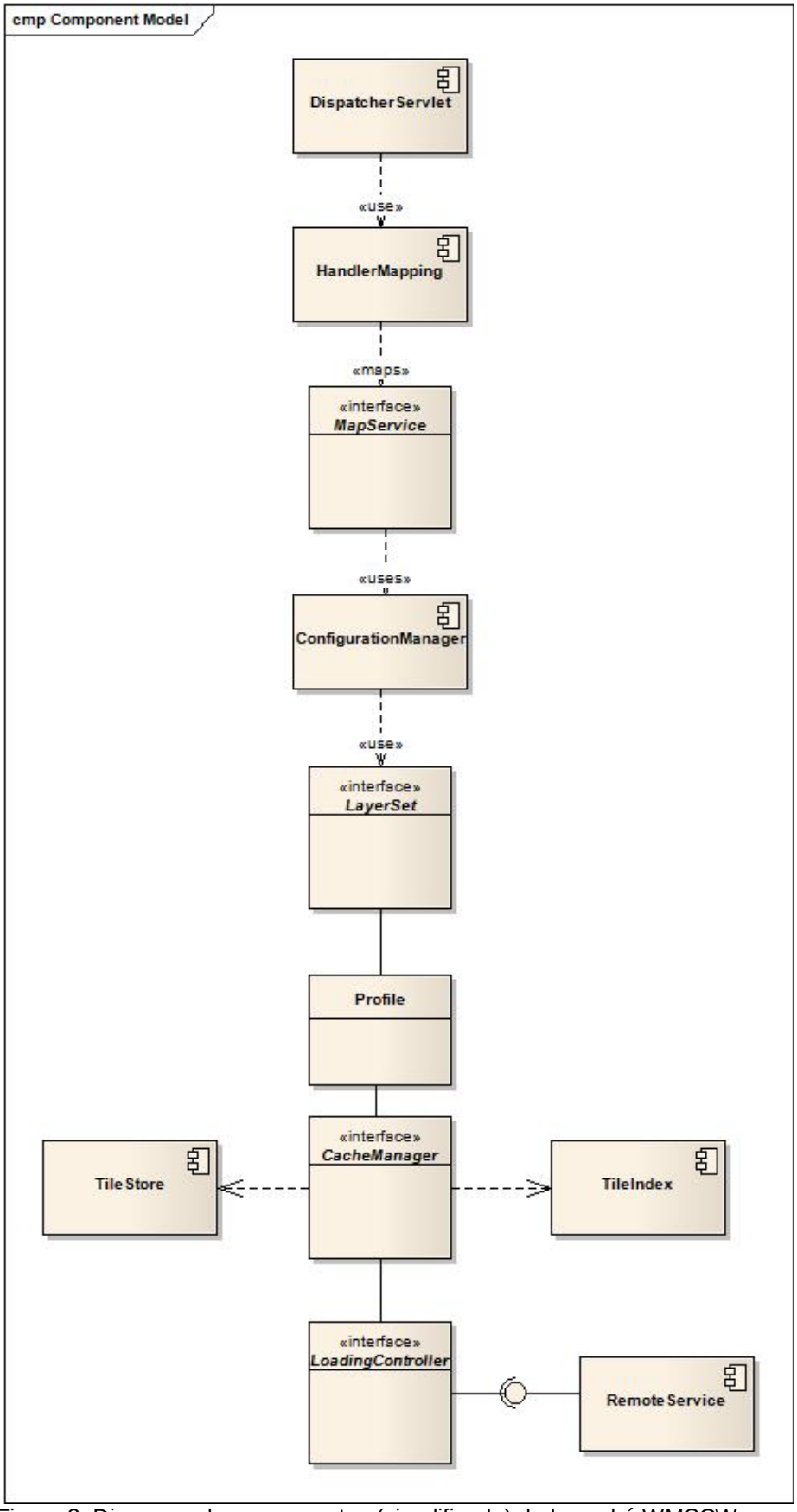


Figura 3: Diagrama de componentes (simplificado) de la caché WMSCWrapper

El DispatcherServlet actúa como punto de entrada de la aplicación, despachando las peticiones hacia la interfaz correspondiente mediante un HandlerMapping, que realiza un mapeo de la URL de la petición con el controlador que debe procesarla.

## Interfaces

WMSCWrapper es capaz de ofrecer el servicio a través de múltiples interfaces de acceso: actualmente como WMS *proxy* según la especificación WMS-C, a través de una interfaz estándar WMTS, y otras especificaciones no estándar como Google Maps y Microsoft Bing Maps, ó actuando como pasarela KML para Google Earth.

### *WMS-C*

El *proxy caché* desarrollado es compatible con la especificación WMS-C (*WMS-Cached*) publicada por OSGeo [8]. Se trata de una estrategia de extensión de los metadatos de servicio del estándar WMS 1.1.1 que permite informar a los clientes de que el servidor ofrece un patrón concreto de teselado bien definido y disponible para su consulta. Se han implementado las operaciones GetCapabilities y GetMap, para la obtención de los metadatos de servicio y la devolución de una imagen de mapa, respectivamente.

Con esta implementación, la operación GetCapabilities del WMS incluye adicionalmente un elemento VendorSpecificCapabilities (ver Listado 1) para informar a los clientes de la disponibilidad opcional de ciertas combinaciones de parámetros en forma de teselas pregeneradas.

```
<WMT_MS_Capabilities>
…
<VendorSpecificCapabilities>
   …
  <TileSet name="CARTOCIUDAD">
    <SRS>EPSG:4326</SRS>
    <BoundingBox SRS="EPSG:4326" minx="0" miny="0" maxx="22.5" maxy="22.5"/>
    <Resolutions>
      1.40625 0.703125 0.494384765625 0.3515625 0.17578125 0.087890625
      0.0439453125 0.0219726563 0.010986328125 0.0054931640625
    </Resolutions>
    <Width>256</Width>
    <Height>256</Height>
    <Format>png</Format>
    <Layers>
      Vial,Toponimo,SeccionCensal,FondoUrbano,CodigoPostal,Municipio
    </Layers>
    <Styles>default,style1</Styles>
  </TileSet>
  …
</VendorSpecificCapabilities>
…
</WMT_MS_Capabilities>
```

Listado 1: Fragmento de documento de *capabilities* producido por el servicio wms-c implementado.

### *WMTS*

A pesar de que la propuesta WMS-C de OSGeo ha sido la solución teselada no-propietaria más adoptada hasta el momento, con la reciente adopción como estándar del servicio *Web Map Tile Service* (WMTS) [9] de OGC resulta previsible una mayor proliferación de este servicio.

WMSCWrapper implementa las operaciones GetCapabilities y GetTile definidas en este estándar. Estas peticiones pueden codificarse tanto en KVP (Key-Value Pair) como en REST.

***KML***

El *proxy caché* implementado puede funcionar como pasarela KML para el cliente Google Earth. Al solicitar una capa en este formato, como respuesta se devuelve un documento KMZ (KML comprimido) interpretable por el cliente Google Earth. El documento KML contiene un elemento <NetworkLink>, que contiene, a su vez, un elemento <Link> mediante el cual se solicita el contenido que se desea visualizar, indicado a través del elemento <viewFormat>. El contenido se actualiza periódicamente, o cada vez que se detiene la navegación. Al recibir las peticiones de los <Link> se devuelve un fichero KMZ generado dinámicamente, que contiene una matriz de elementos <GroundOverlay>, cada uno de los cuales contiene una única petición conforme al estándar WMS-C.

***Google Maps***

Google Maps utiliza una proyección esférica mercator, por lo que la capa a visualizar debe estar disponible en el sistema de coordenadas denominado EPSG:900913 (muy similar al EPSG:3857). En la Tabla 1 se muestra el código JavaScript necesario para superponer una capa procedente de la caché WMSCWrapper[1].

Tabla 1: Código JavaScript para superponer en Google Maps una capa accesible a través de WMSCWrapper

```
var tilelayer = new GTileLayer(<copyrights>, <minResolution>, <maxResolution>,
{
  tileUrlTemplate: 'http://<host>:<port>/WMS_C_wrapper/gmaps?
                    layers=<layer_name>&zoom={Z}&x={X}&y={Y}&format=<format>',
  isPng:<{true/false}>,
  opacity:<opacity>
} );
```

Los índices X, Y, Z anteriores son sustituidos por el cliente de Google por los valores apropiados de coordenadas latitudinal, longitudinal, y nivel de zoom, respectivamente, según la localización a visualizar en el mapa.

El esquema de teselado utilizado por Google Maps tan sólo difiere del utilizado internamente por WMSCWrapper en la inversión del índice Y, por lo que la traducción es inmediata: $Y_{WMSCWrapper} = MaxY - Y_{GoogleMaps}$.

***Bing Maps***

Bing Maps utiliza la misma proyección que Google Maps por lo que se requiere que, al igual que para este último, la capa esté disponible en esta proyección.

Tabla 2: Código JavaScript para superponer en Bing Maps una capa accesible a través de WMSCwrapper

```
var map = new VEMap('myMap');
var tileSourceSpec =
    new VETileSourceSpecification(
```

[1] Información adicional en la documentación del API de Google Maps: http://code.google.com/intl/es-ES/apis/maps/documentation/reference.html#GTileLayer

```
      'TITLE_OF_LAYER',

'http://<host>:<port>/WMS_C_wrapper/bing?quadkey=%4&format=image/png&layers=<l
ayer_name>'
    );
tileSourceSpec.Opacity = 0.5;
map.AddTileLayer(tileSourceSpec, true);

...

<body onload="GetMap();">
```

Para optimizar el indexado y almacenamiento de las teselas, las coordenadas XY de una tesela se combinan en una cadena uni-dimensional denominada "*quadkey*". Cada *quadkey* identifica una tesela en un determinado nivel de resolución. Al recibir una petición procedente de Bing Maps, el *proxy* WMSCWrapper traduce el *quadkey* recibido en las coordenadas XY correspondientes.

**LayerSets**

En esta última versión del prototipo se ha añadido la posibilidad de visualizar otras capas a mayores de la propia capa de mapa (*WMSLayerSet*) configurada. Estas capas son *CachedLayerSet* y *StatsLayerSet*. La primera de ellas permite ver sobre el mapa qué teselas están cacheadas y cuáles no. La segunda representa en forma de *heatmap* o mapa de calor las estadísticas de las peticiones recibidas. Estos *heatmaps* permiten detectar la existencia de regiones de interés, como se muestra en la Figura 4, donde se recogen las estadísticas de peticiones realizadas al servicio de mapas del PNOA (Plan Nacional de Ortofotografía Aérea) .

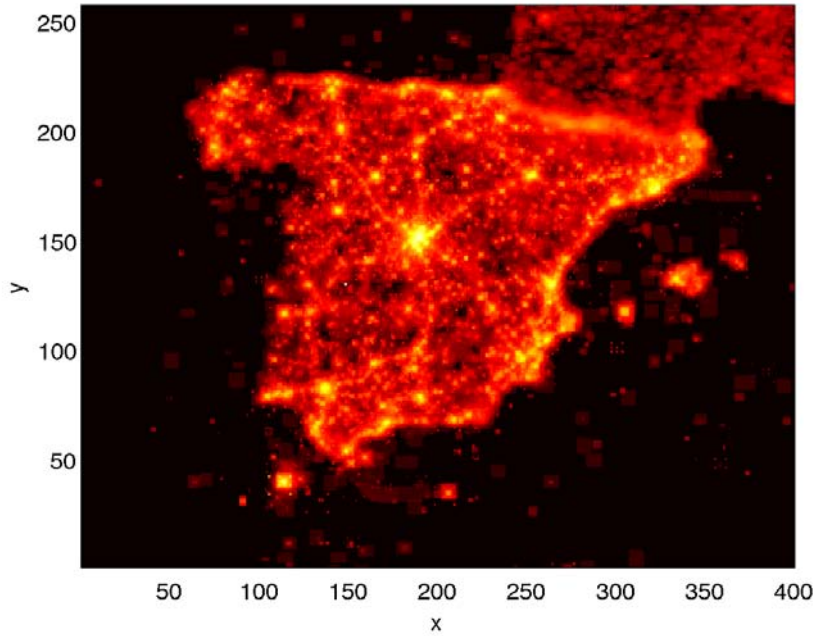


Figura 4 - *Heatmap* de los registros de peticiones realizadas al servicio WMS-C PNOA.

**Almacenes de teselas (*TileStore*)**

Frente a la anterior versión de esta *caché*, el almacenamiento de las teselas es ahora más flexible, permitiendo que éstas sean almacenadas tanto en disco como en múltiples Sistemas Gestores de Bases de Datos (SGBD).

*DiskTileStore*

Este componente permite almacenar las teselas con persistencia en disco siguiendo la siguiente estructura de directorios: raíz/layer/resolución/x/y.<extension> (ver Figura 5).

A partir de este módulo pueden extenderse, de forma sencilla, otros componentes que utilicen otras estructuras de directorios. Esto permite realizar una optimización en función del sistema de ficheros utilizado.

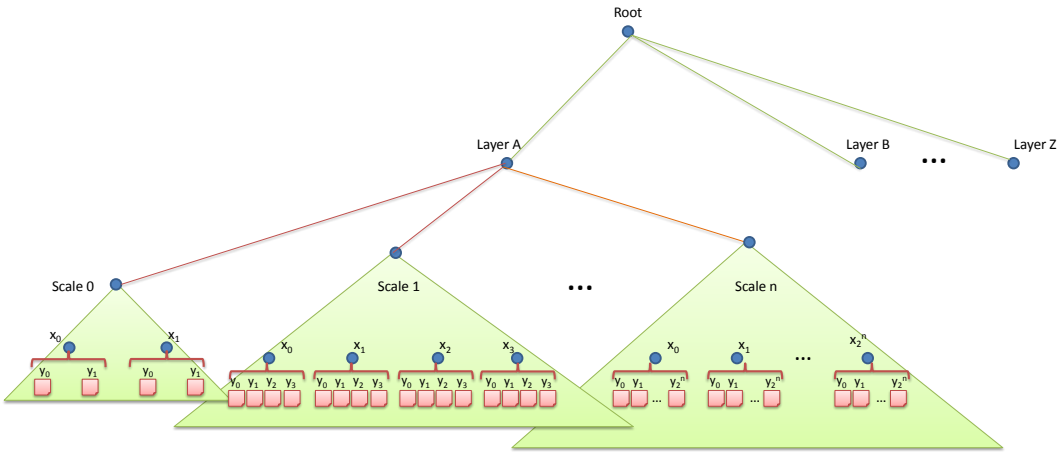


Figura 5: Estructura de directorios para el almacenamiento persistente de las teselas en WMSCwrapper

*Base de datos*

Se han implementado diversos componentes para el almacenamiento de las teselas en distintos SGBD como MySQL, PostgreSQL, HSQLDB o Derby. El almacenamiento de los objetos en base de datos permite aprovechar el indexado ofrecido por estas. En algunos escenarios, con un reducido número de teselas a *cachear*, puede hacerse uso de bases de datos en memoria (IMDB – *In-memory database*) como Derby o HSQLDB. Iniciativas como Oracle GeoRaster o PostgisRaster (WKTRaster) respaldan el uso de bases de datos para el almacenamiento de datos ráster.

***Índices de teselas (TileIndex)***

Se han implementado diversos índices para mantener un registro espacial en tiempo real de la actividad de la caché. Toda la actividad de ésta queda registrada en estos índices mediante el cual se pueden realizar búsquedas espaciales de gran eficacia, especialmente los recorridos en profundidad. Dada la naturaleza exponencial de la estructura de la pirámide, resulta impráctico albergar un índice del mismo tamaño que la *caché* que se quiere representar, por lo que se ofrece la posibilidad de truncar el índice en un nivel concreto configurable a través de un parámetro. Es evidente que ajustando este parámetro se puede transformar el índice para que cubra completamente la pirámide de teselas.

El antiguo prototipo tan solo permitía utilizar índices de tipo *quadtree* que se cargaban en memoria y se persistían periódicamente en disco.

En esta nueva versión se ha incorporado la posibilidad de recoger las estadísticas de la actividad de la caché en una base de datos PostgreSQL con la extensión espacial PostGIS. Este tipo de índice resulta de gran utilidad para realizar un posterior análisis de las estadísticas almacenadas.

Asimismo, se ha creado un tipo de índice "*PonderatedIndex*" que permite combinar la información de otros índices existentes de forma ponderada asociando un peso a cada uno de ellos.

**LoadingControllers**

Cuando la tesela solicitada no está almacenada en la caché (diremos que se ha producido un fallo de caché) ésta debe ser pedida al servidor de mapas remoto a través de una petición WMS estándar. Esta es la tarea del *LoadingController*. Se han implementado dos de estos componentes como se describe a continuación. En un futuro se pretende desarrollar módulos que permitan utilizar un *backend* que no sea un servidor WMS, como por ejemplo un servicio WMTS.

*BasicLoadingController*

El componente *BasicLoadingController* realiza una petición WMS para obtener la imagen deseada a partir del servicio de mapas remoto. La URL de la petición se construye a partir de los parámetros de configuración del LayerSet y el *bounding box* correspondiente a la tesela solicitada.

*BufferedLoadingController*

En el proceso de generación de teselas hay diversos cuellos de botella que se pueden mejorar. Una de las partes del proceso de dibujado es el acceso a los almacenes externos de información geográfica. Ya que las consultas suelen estar optimizadas mediante índices espaciales, puede obtenerse una mejoría al reducir el número de consultas ampliando el *bounding box* de la zona a generar.

Otro beneficio obtenido indirectamente es la reducción del problema del etiquetado fraccionado o redundante provocado por el teselado de las peticiones.

El componente *BufferedLoadingController* utiliza esta técnica de generación de peticiones de mayor tamaño que la tesela a *cachear* (esta supertesela se denomina *metatile*) y posteriormente se postprocesa para aprovechar la información disponible y generar nuevas teselas.

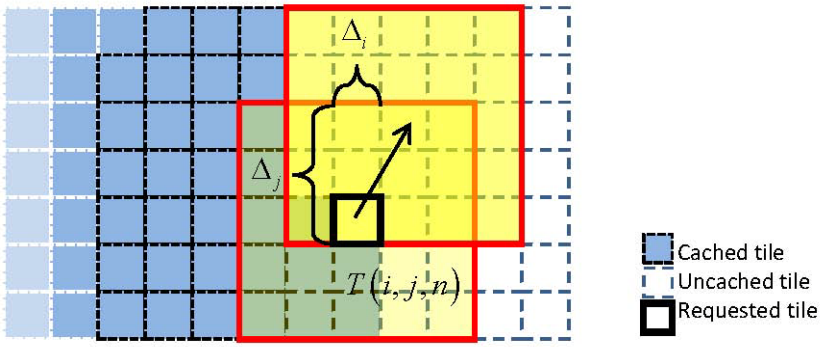


Figura 6: Estrategia de mínima correlación con la cache para la generación de *metatiles*

Las implementaciones investigadas permiten especificar el número adicional de teselas alrededor de la realmente solicitada (denominémoslo *buffer* de N teselas) que se van a pedir en una sola petición al servidor WMS. De esta manera se le pide al WMS una *metatile* de tamaño $(2N+1)^2$ teselas centrada en el elemento realmente solicitado.

En un escenario de *cache* no completa (pero no vacía) esta elección del área a generar no es muy eficiente puesto que es muy probable que algunas de las teselas próximas a la solicitada ya estén disponibles.

Partiendo de la suposición de que la zona de la tesela solicitada no está homogéneamente cargada, se ha implementado un algoritmo para la elección óptima de las *metatiles* a generar. El procedimiento, ilustrado en la Figura 1, busca obtener

en función del estado de la *caché* la *metatile* que, conteniendo la tesela solicitada, minimice la correlación espacial:

$$R_n(\Delta i, \Delta j) = \sum_{l=i-N}^{j+N} \sum_{m=j-N}^{j+N} h[l + \Delta i, m + \Delta j] \qquad (1)$$

Interpretando la correlación en (1) como una medida de la semejanza de la información contenida en ambos objetos (*metatile* y *caché*), parece evidente que la *metatile* que tenga una menor correlación espacial con el estado de la caché es aquella que proporciona la mayor información al sistema, puesto que la información que contiene es complementaria en mayor grado a la ya disponible. En la Figura 8 se ilustra la configuración con la que se consigue un mínimo en la redundancia o en la información mutua.

La implementación llevada a cabo incluye además un procedimiento en segundo plano para realizar el postproceso de las teselas adicionales. De esta forma se busca minimizar también el tiempo necesario para obtener un objeto directamente de la caché.

## MECANISMOS DE PRECARGA DE TESELAS Y DE LIMPIEZA DE LA CACHE

Los requisitos de almacenamiento y puesta en marcha de estos servicios resultan a menudo prohibitivos cuando la cartografía a servir cubre una zona geográfica extensa para un elevado número de escalas.

Por ello, algunos de los sistemas de caché estudiados, como *TileCache* o *GeoWebCache*, ofrecen la posibilidad de indicar el *bbox* del conjunto de teselas que se quiere cachear durante el proceso de *seeding* o carga inicial para las escalas indicadas. TileCache ofrece además la posibilidad de cachear las teselas inscritas en la circunferencia cuyo centro y radio se especifican.

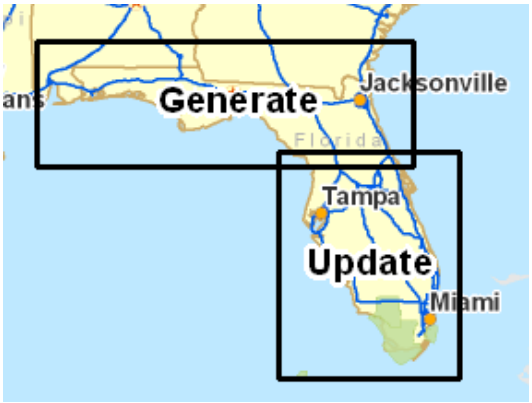


Figura 7: Selección de zonas geográficas en ArcGis para la ejecución de una tarea de *seeding* y actualización.

Es inmediato anticipar que las aproximaciones anteriores ofrecen pocos grados de libertad al administrador, resultando ineficientes para el tratamiento de zonas geográficas más complejas. Para destacar las limitaciones existentes considérese por ejemplo la solución propuesta en ArcGis[2] para el cacheo de zonas geográficas de gran tamaño, como la mostrada en la Figura 7, en la que se quieren evitar las zonas

[2] http://webhelp.esri.com/arcgisserver/9.2/dotNet/manager/publishing/tips_map_caches.htm

de agua. Obviamente se trata de una solución poco sofisticada, y no resulta óptima para el objetivo perseguido de incluir tan solo las zonas terrestres.

Como respuesta a las carencias detectadas, el sistema de caché WMSCWrapper incorpora la posibilidad de seleccionar geometrías arbitrarias. Esta aproximación permite acotar con mayor precisión las áreas de interés a cubrir por las tareas de mantenimiento de la caché, reduciendo el tiempo y los recursos necesarios para su ejecución. En la Figura 8 se muestra el interfaz de usuario para la selección de estas zonas de interés.

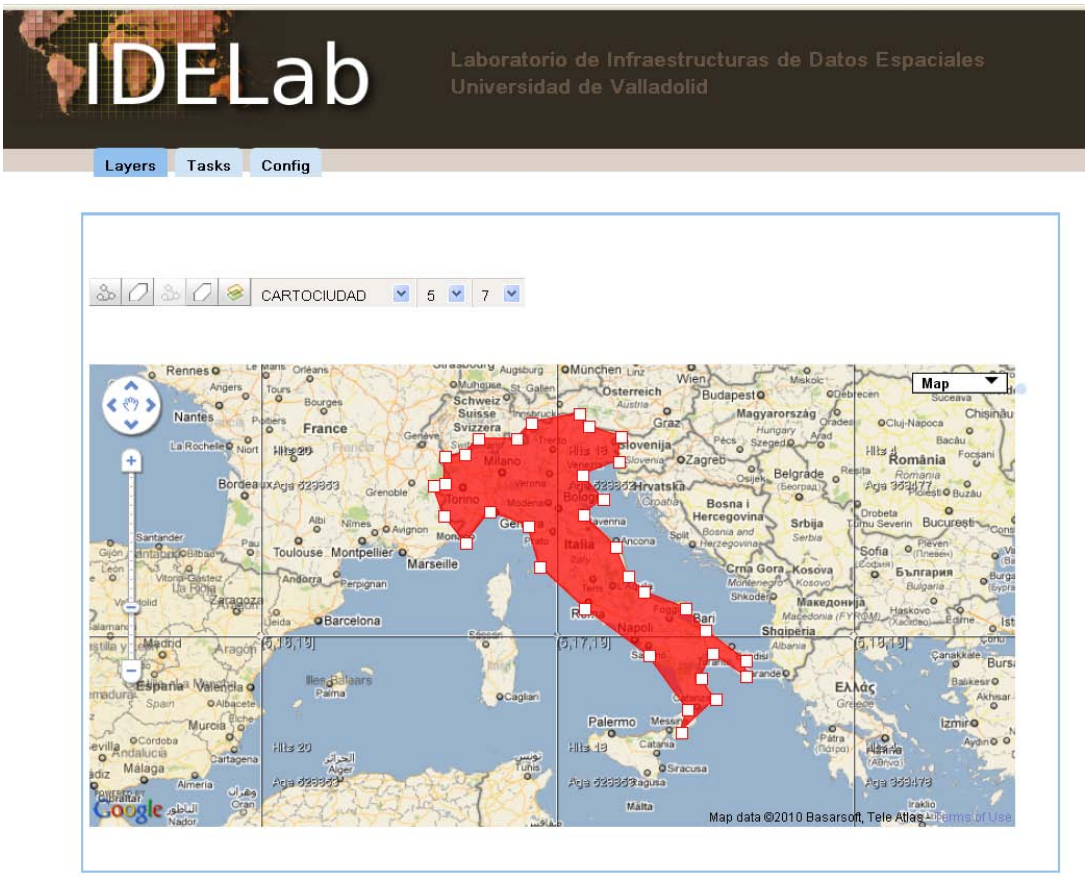


Figura 8: Interfaz gráfica de usuario para la selección de geometrías en el mapa.

Se han implementado en la caché una serie de iteradores para recorrer geometrías arbitrarias (puntos, líneas y polígonos), haciendo uso para ello de adaptaciones de los algoritmos tradicionales más comúnmente utilizados en el campo del renderizado gráfico, como son los algoritmos de Bresenham [10] y ScanLine [11] para el recorrido de líneas y polígonos, respectivamente.

Los mecanismos de mantenimiento de caché utilizan estos iteradores para ejecutar la labor concreta sobre la zona geográfica descrita por la geometría especificada, ya sea para el cacheo de teselas o el truncado de las mismas en las respectivas tareas de *seeding* o limpieza, o para la actuación de cualquier otro mecanismo de gestión.

Como se puede observar en la Figura 4, analizando los *heatmaps* de peticiones pueden identificarse *features* directrices de las peticiones de los usuarios. En el caso mostrado se aprecia que los núcleos urbanos, zonas costeras y las vías principales de comunicaciones reciben un elevado número de peticiones. WMSCWrapper permite utilizar distintas fuentes de *features* para la aplicación de mecanismos de gestión. Así, pueden utilizarse, por ejemplo, las *features* obtenidas a partir de un servicio WFS para la actuación de un mecanismo de caché sobre las teselas que intersectan dichas *features*.

## CONCLUSIONES

En este trabajo se han presentado los avances más significativos realizados en el prototipo de caché WMSCWrapper. Entre las novedades de esta versión destaca la disposición de una arquitectura más modular y flexible mediante el uso del *framework* Spring. Se han desarrollado nuevos índices para la recogida de las estadísticas de las peticiones que pueden ser explotados por los algoritmos de gestión de la caché para la realización de tareas de mantenimiento. Asimismo, los nuevos almacenes de teselas implementados permiten albergar las imágenes de la caché en base de datos, lo que permite aprovechar el indexado que éstas proveen. Dado el reciente lanzamiento del estándar WMTS de OGC y las expectativas de proliferación de este servicio, se ha implementado una interfaz compatible con el mismo. Se ha desarrollado una herramienta para la selección manual sobre un mapa de las regiones a ser tratadas por los algoritmos de precarga o de limpieza. Por otra parte, se han añadido algoritmos predictivos para la precarga de teselas a partir de *features* directrices de las peticiones de los usuarios, obtenidas a partir de diversas fuentes de datos.

## 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] R. García Martín y J.P. de Castro Fernández, “WMSCWrapper. Implementación WMS-C OpenSource para servicios WMS teselados,” *IV Jornadas de SIG libre*, Girona, Spain: 2010.

[2] OGC, “OpenGIS Web Map Service (WMS) Implementation Specification,” 2009.

[3] Lucian Plesea, “The Design, Implementation and operation of the JPL OnEarth WMS Server,” *Geospatial Services and Applications for the Internet*, Sample, J.T., Shaw, K., Tu, S., y Abdelguerfi, M., eds., Berlin: Springer, 2008, págs. 93-109.

[4] Matt Mills, “NASA World Wind Tile Structure” Available: http://www.ceteranet.com/nww-tile-struct.pdf.

[5] NASA, “WorldWind Tile System Schema” Available: http://issues.worldwind.arc.nasa.gov/confluence/download/attachments/394/world+wind+tile+systemt.gif.

[6] “Maven - Welcome to Apache Maven” Available: http://maven.apache.org/.

[7] “Sitio Web de SpringSource” Available: http://www.springsource.org/.

[8] OsGeo, “WMS Tile Caching,” *WMS Tile Caching - OSGeo Wiki* Available: http://wiki.osgeo.org/wiki/WMS_Tile_Caching.

[9] Joan Masó, Keith Pomakis, y Núria Julià, “Web Map Tiling Service Standard,” Feb. 2009.

[10] J.E. Bresenham, “Algorithm for computer control of a digital plotter,” *IBM Systems journal*, vol. 4, 2010, págs. 25–30.

[11] J.M. Lane, L.C. Carpenter, T. Whitted, y J.F. Blinn, “Scan line methods for displaying parametrically defined surfaces,” *Communications of the ACM*, vol. 23, 1980, págs. 23–34.