Visualización en navegadores web de información geográfica en forma de mapas vectoriales activos

José Ignacio Lamas Fonte, Miguel R. Luaces, José R. Paramá

Laboratorio de Bases de Datos
Universidade da Coruña
Campus de Elviña S/N
{jilamas, luaces, parama}@udc.es

Resumen

El objetivo de este artículo es describir el análisis, diseño e implementación de un componente Java que permitirá la visualización de mapas vectoriales activos y que podrá ser utilizado en forma de applet para visualizarlos a través de Internet. El componente obtendrá los mapas en formato SVG de un servidor web de mapas conforme a la especificación *Web Map Service* (WMS). Los mapas llevaran incorporada actividad en forma de scripts de Javascript que será debidamente interpretada.

También se documentara la inclusión de este applet en un proyecto real, el *Viaxe Virtual por Galicia* de la *Biblioteca Virtual Galega*.

Palabras clave: SIG, Aplicaciones Web, SVG, WMS, Información Vectorial Activa

1 Introducción

La especificación del *OpenGeospatial Consortium* (OGC) más utilizada actualmente define la interfaz de un servicio web de mapas (*Web Map Service*, WMS [6]). Un WMS recibe una petición HTTP de un cliente en la que se solicita un mapa, recupera los objetos geográficos que componen el mapa de una base de datos o un servidor de información geográfica y, de acuerdo a unas características de estilo, genera un mapa en alguno de los formatos existentes para representar información gráfica. El OGC recomienda el formato *Scalable Vector Graphics* (SVG [5]) como formato vectorial para la representación de los mapas generados por los WMS. Un mapa representado en este lenguaje, basado en XML [3], podrá responder a eventos de usuario y cambiar su aspecto visual de forma dinámica ya que SVG permite incluir funciones descritas en un lenguaje de *script* que deben ejecutarse en respuesta a eventos del usuario.

En este artículo se describe el análisis, diseño, implementación y prueba de un componente para visualizar información vectorial activa en forma de mapas en formato SVG. Este componente puede utilizarse en forma de *applet*, en una aplicación web para sistemas de información geográfica (SIG)

y obtiene la información geográfica de un servicio de mapas que implemente el estándar WMS. El componente es responsable de la gestión de las capas que se pueden visualizar, así como de los datos del mapa (coordenadas, resolución, etc.), y de la construcción de las peticiones a un WMS de manera consistente, es decir, con los parámetros bien inicializados y las capas ordenadas convenientemente.

Se describe en primer lugar el análisis, diseño e implementación de un subsistema capaz de interpretar y dibujar un documento SVG de forma rápida y eficiente. Esto se lleva a cabo mediante un analizador sintáctico que crea una representación interna altamente optimizada del documento SVG. Una de las principales características de este módulo es su rapidez de pintado y su alta extensibilidad para poder añadir más elementos según las necesidades que se puedan tener en un futuro.

A continuación se presenta el análisis, diseño e implementación de un subsistema con la capacidad de construir peticiones a un WMS y devolver el SVG resultante. Para ello proporciona una interfaz para añadir capas, quitar capas, cambiar la resolución, las coordenadas, el fondo, y otras propiedades del mapa que queremos visualizar. Otro componente importante es el que realiza el análisis e interpretación de la actividad del mapa incluida dentro del SVG. Este componente es también el encargado de recoger la interacción del usuario con el mapa y ejecutar el *script* correspondiente sobre el elemento correspondiente. La principal característica de este módulo es el compromiso que es necesario alcanzar en la ejecución de funcionalidad descrita con lenguajes de *script*. Debemos cubrir una parte lo suficientemente amplia del lenguaje de *script* para dar servicio a la funcionalidad típica que tienen este tipo de mapas, pero no es necesario que lo interprete todo puesto que este componente se haría muy grande en tamaño y sería bastante laborioso de implementar.

Finalmente, para demostrar el funcionamiento del componente, se ha realizado la migración del *Viaxe Virtual por Galicia* de la *Biblioteca Virtual Galega* (http://bvg.udc.es) que funcionaba con una aplicación propietaria. En la actualidad está funcionando con el componente descrito en este artículo.

El resto de este artículo se estructura como sigue: en la Sección 2 se describe el trabajo relacionado que se utiliza como base en este artículo- A continuación, en la Sección 3 se presenta la arquitectura del sistema y en las siguientes secciones se describe cada uno de sus componentes: en la Sección 4 se presenta el módulo de visualización de SVG, en la Sección 5 se describe el módulo intérprete de Javascript e interacción con el usuario, en la Sección 6 se detalla el módulo cliente de WMS, y en la Sección 7 se describe el módulo de integración. A continuación, en la Sección 8 se describe el *Viaxe Virtual por Galicia* de la *Biblioteca Virtual Galega* y finalmente en la Sección 9 se presentan las conclusiones del trabajo y las líneas de trabajo futuro.

2 Trabajo relacionado

El *Web Map Service* (WMS [6]), es un estándar propuesto por el *OpenGeospatial Consortium* (OGC [4]), y describe un servicio Web que produce mapas de información georeferenciada. Un mapa por tanto es la representación visual de datos geográficos; el mapa no son los datos en sí mismos. Esta especificación define tres operaciones y la sintaxis para invocar cada una de ellas. Estas operaciones son: *GetCapabilities* que describe la información del servicio y los parámetros aceptables para las peticiones; *GetMap* que devuelve un mapa, esta es la operación más importante para el presente trabajo y será explicada con más detalle; y *GetFeatureInfo* que es una operación opcional y devuelve información acerca de las entidades geográficas mostradas en un mapa.

Por ejemplo, una petición *GetMap* típica sería así:

```
http://servidor.mapas/wms?service=WMS&VERSION=1.1.1&REQUEST=GetMap&
       LAYERS=provincias,concellos,comarcas,espaciosnaturais,praias,poboacions&
       STYLES=,,,,estilo:praias,&
       SRS=EPSG:23029&BBOX=552371,4806932,570318,4818877&
       HEIGHT=394&WIDTH=592&FORMAT=image/png
```

El resultado de lanzar esta petición *GetMap* sobre un WMS puede verse en la Ilustración 1. El significado de los parámetros de está petición es el siguiente:

- LAYERS: Se indica una lista de nombres de capa separados por comas y en el orden en que se desea que se pinten en el mapa.
- STYLES: Se indica el estilo con que cada capa va a ser pintada mediante una lista de nombres de estilo separados por comas. Si una de las posiciones de la lista la dejamos vacía la capa se pintara con su estilo por defecto. Estos estilos determinaran el color de las líneas y los rellenos de los polígonos así como los iconos con los que se representaran los puntos o el tamaño y tipo de letra de los textos que aparecerán en cada capa.
- SRS: Se indica el sistema de referencia espacial (*Spatial Reference System*) que será utilizado para representar el mapa.
- BBOX: Indica la porción de la Tierra que aparecerá pintada en el mapa, y se representa por dos coordenadas que estarán en las unidades del SRS seleccionado, siendo la primera coordenada el punto inferior izquierdo del rectángulo y la segunda el punto superior derecho.
- HEIGHT y WIDTH: Representan la resolución del mapa mediante el número de píxeles que tendrá la imagen de largo y de alto respectivamente.
- FORMAT: Se le indica el tipo MIME del formato de imagen en el que deseamos que nos devuelva el mapa.

Ilustración 1. Mapa generado por un WMS

Uno de los posibles formatos de la imagen resultante es el lenguaje SVG [5], que es un estándar propuesto por el *World Wide Web Consortium* (W3C [7]), el mismo que ha realizado la especificación HTML. SVG es un lenguaje empleado para describir gráficos vectoriales en 2D utilizando XML [3]. Permite definir tres tipos de objetos gráficos: formas geométricas vectoriales (puntos, líneas, etc.), imágenes y texto. Con SVG se pueden realizar gráficos animados y dinámicos, y presenta como una gran ventaja el uso de lenguajes *script*, que nos proporcionan un acceso

completo a todos los elementos, atributos y propiedades definidos en el estándar, con lo que facilita a un usuario la elaboración de aplicaciones tan complejas como se deseen basándose en un documento SVG. Además, existe una total compatibilidad entre esta especificación y otras definidas por el W3C, como son, por citar algunas, HTML o XHTML y XML, y eso posibilita la integración en una misma página Web de elementos SVG con HTML, o crear un documento XML que tiene incrustado una sección SVG.

En la Ilustración 2 se muestra un ejemplo de un SVG con actividad asociada.

Ilustración 2. Ejemplo de SVG.

En la parte izquierda de la ilustración se encuentra el cuerpo del documento SVG Dentro del cuerpo del elemento *svg* nos encontramos con un elemento de tipo *script* en el que se define una función denominada *change_colour* que cambia el color del relleno del elemento sobre el que se ejecute. Justo después se encuentra un elemento de agrupamiento *g* en el que se definen unas propiedades de pintado, el color del contorno negro y el del relleno gris, y se asocia a los elementos de este grupo la función *change_colour* mediante el evento *onclick*, de forma que si el usuario hace *click* sobre un elemento de este grupo se ejecutara la función con ese elemento como parámetro. Dentro del grupo se encuentra un elemento *polygon* que define la forma geométrica de un polígono mediante sus puntos, este polígono hereda las propiedades de pintado y de interacción del grupo que lo encapsula.

```
<svg viewbox=”0 0 20 20 width=”10px” height=”10px”>
  <script type=”text/ECMAScript”>
    <![CDATA[
      function change_colour(evt){
        var target = evt.target;
        target.setAttribute(“fill”, “white”);
      }
    ]]>
  </script>
  <g stroke=”black” fill=”silver” onclick=” change_colour(evt)”>
    <polygon points=”1 11 6 1 11 6 5 19 8 10 18”>
  </g>
</svg>
```

En la parte derecha de la ilustración podemos observar el resultado de pintar este SVG junto con una cuadricula de ayuda y las coordenadas de los puntos que conforman el polígono. Como podemos observar el polígono tiene un color gris. Si hiciésemos *click* encima de él se ejecutaría la función asociada a ese evento, y en consecuencia su color de relleno cambiaría a blanco.

Actualmente muy pocos navegadores soportan de forma nativa SVG, de una lista con los 21 navegadores más comunes solo dos de ellos lo soportan de forma completa y cinco de forma parcial. Además si atendemos a las estadísticas de uso de los navegadores vemos que el navegador más ampliamente usado, Internet Explorer, que según datos del año 2006 es usado por un 83% de los internautas aproximadamente no soporta de forma nativa el formato SVG.

Una de las opciones para poder visualizar SVG en un navegador es utilizar un *plug-in*. Un *plug-in* es un componente que extiende la funcionalidad del navegador para, en este caso, visualizar e interpretar la información del formato vectorial correspondiente. La ventaja que tienen es que proporcionan un alto rendimiento visualizando el formato para el que fueron construidos; la

principal desventaja es que necesitan instalación con los problemas que de ello se derivan, problemas de seguridad, problemas de usuarios que no tengan permisos suficientes para instalar nada en el equipo, etc.

La alternativa al uso de un *plug-in* es la utilización de un *applet* de java. Un *applet* es una miniaplicación escrita en Java que se ejecuta en la maquina virtual de Java asociada al navegador que esta visualizando la página en la que esta incluido el *applet*. La ventaja es que al ser una aplicación compilada se ejecuta de manera rápida. Además, la maquina virtual de Java ofrece un entorno seguro para la ejecución de los *applets*.

Dado que muchas de las aplicaciones SIG necesitan un alto grado de interacción entre el usuario y los elementos gráficos del mapa, es necesario utilizar SVG activo en estas aplicaciones. Sin embargo, no hemos encontrado ningún *applet* Java que fuera pequeño y rápido, por lo que hemos considerado necesaria la implementación de un *applet* Java para la visualización de mapas en formato SVG.

Dado que el *applet* va a tener que realizar la visualización de los elementos gráficos del SVG, se vio desde el principio que sería necesaria la utilización de alguna estructura de indexación espacial para implementar de forma eficiente este proceso de visualización. En nuestro caso decidimos utilizar un *R-Tree*, que es una estructura en forma de árbol, de la familia del *B-Tree*, que permite gestionar de forma eficiente información geométrica, como puntos, polígonos, segmentos, áreas y volúmenes en espacios multidimensionales. El R-Tree fue propuesto por Guttman en 1984 [1], motivado por el diseño de sistemas VLSI y a partir de ese momento se ha extendido a muchas otras áreas y se han propuesto múltiples variaciones. Los R-Tree se basan en que localizar un punto dentro de un rectángulo es mucho más rápido que hacerlo en cualquier otro tipo de polígono. Por ello se utilizarán rectángulos como estructura básica para el manejo de los objetos geométricos.

3 Arquitectura del sistema

El sistema esta formado por cuatro módulos principales que interactúan entre si. En la Ilustración 3 puede verse un esquema con la estructura global.

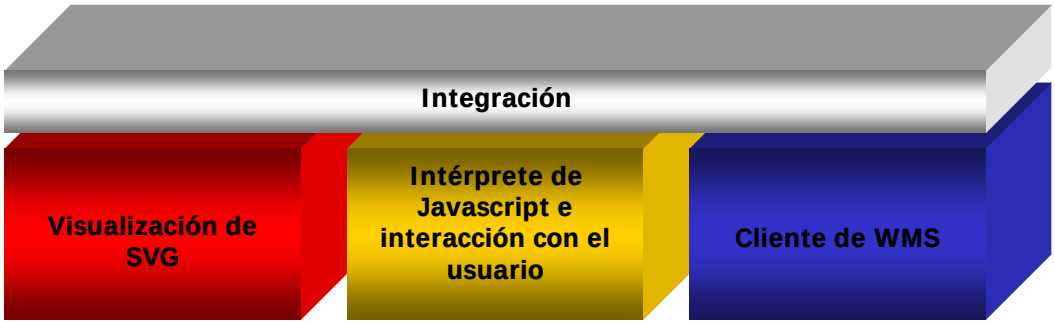

Ilustración 3. Componentes que integran el sistema.

El módulo de visualización de SVG es el encargado de analizar sintácticamente los ficheros SVG y pintarlos en la pantalla. El módulo de intérprete de Javascript e interacción con el usuario se encarga de capturar los eventos generados por el usuario y hacer que el mapa responda según la interactividad definida en el mapa activo. El módulo cliente de WMS se encarga de la

comunicación entre el *applet* y el servidor de mapas activos, así como de guardar la configuración del mapa actual (*bounding box*, *resolución*, etc.) Finalmente está el módulo de integración que se encarga de coordinar el trabajo del resto de los subsistemas y además es el punto de entrada y la fachada visible de todo el sistema, ocultando por tanto los subsistemas de más bajo nivel. A continuación pasaremos a describir más detalladamente cada uno de los módulos que conforman el sistema.

4 Módulo de visualización de SVG

El objetivo de este módulo es visualizar en pantalla información en formato SVG. Dado que SVG es un formato muy extenso el módulo deberá soportar sólo los elementos básicos de SVG que sean de uso común en este tipo de aplicaciones, ya que hacer que pintase cualquier elemento SVG llevaría demasiado tiempo de desarrollo e incrementaría innecesariamente el tamaño de este módulo; por ello la parte de SVG correspondiente a animación no será soportada pues no es usada habitualmente en aplicaciones SIG.

El módulo, además, debe implementar la funcionalidad común en las herramientas de visualización de información geográfica. En concreto, debe poder hacer cambios de escala en el mapa, localizar los elementos geográficos que componen el mapa por el lugar que ocupan en pantalla, marcar un punto de la pantalla con algún tipo de señal para resaltar una entidad geográfica sobre las otras de su mismo estilo y pintar elementos auxiliares sobre el mapa para ayudar al usuario (por ejemplo, el cuadro de selección de la nueva extensión del mapa). Además el sistema debe ser altamente extensible para poder añadir en un futuro elementos adicionales de SVG o elementos que surjan en posteriores revisiones del formato.

Un requisito fundamental de este módulo es la velocidad, tanto recuperando los objetos geográficos que componen el mapa como pintando el propio mapa por pantalla. La velocidad recuperando la información geográfica la conseguimos gracias al uso en este módulo de una estructura de indexación espacial (en nuestro caso un R-Tree). Esto hace que podamos saber en tiempo real sobre que objeto geográfico del mapa esta pasando el ratón en cada momento.

La velocidad en el pintado del mapa es fundamental principalmente porque ante un cambio en algún objeto geográfico, debido a la interactividad especificada en el propio mapa, deberemos repintar el mapa entero para que quede plasmado en pantalla dicho cambio. Así, por ejemplo, si tenemos un mapa con una capa de municipios que responden al evento *onclick* cambiando el color en el que se pintan y un usuario hace *click* en un municipio del mapa debemos: en primer lugar recuperar el municipio sobre el que hizo *click*, y pintar el mapa de nuevo con el municipio ya cambiado de color. Si este sistema es lento el usuario percibirá un retraso entre las acciones que ejecute sobre el mapa y la respuesta del propio mapa.

La forma de conseguir velocidad en este subsistema es añadiendo un paso intermedio en el proceso que pasa del fichero en formato SVG al mapa pintado en pantalla. De forma que lo que se hace es analizar el fichero SVG y construir una representación en árbol del mismo traduciendo los elementos SVG por objetos gráficos de Java. Al mismo tiempo se rellena el índice espacial con estos objetos geográficos. Este árbol de pintado intermedio se pintará de forma rápida y eficiente dado que ya esta compuesto por objetos Java listos para ser pintados y conserva todas las características del documento SVG como puede ser la herencia de los atributos de pintado de un nodo por sus hijos.

5 Módulo intérprete de Javascript e interacción con el usuario

Este módulo se encarga de la interacción del usuario con el mapa una vez que está el mapa cargado. La interacción del usuario con el sistema para añadir o eliminar capas al siguiente mapa que se pida al WMS es gestionada por el módulo cliente de WMS.

Este subsistema debe ser sencillo y no ocupar demasiado, puesto que nos interesa mantener el tamaño global del sistema lo más pequeño posible. La funcionalidad del módulo consiste en capturar los eventos producidos por el usuario, generalmente a través del ratón, y, a partir de ahí, la interacción puede ser de dos tipos: bien mediante el comportamiento activo incluido en el mapa en formato Javascript, o bien mediante herramientas definidas para modificar de alguna forma el mapa, como pueden ser herramientas para desplazar el mapa o realizar acercamientos o alejamientos. Cualquiera de estas interacciones con el mapa pueden provocar cambios en el mismo que deberán ser notificados al módulo de pintado y ocasionalmente al módulo cliente de WMS para mantener la coherencia en el próximo mapa que se pida. Además estos cambios deberán ser expresados en pantalla mediante el repintado del mapa, que se hará delegando en el módulo de pintado.

La ejecución de funciones en Javascript se dividió en dos pasos para dar una mayor velocidad. En primer lugar, cuando el módulo de pintado está analizando el documento SVG y se encuentra con una función de Javascript, esta es también analizada y convertida en una estructura de objetos Java que permitirá la ejecución rápida del código *script*. Posteriormente, cada vez que sea necesario ejecutar una función de Javascript sobre un objeto del mapa utilizará la representación interna que tenemos de esa función, por lo que nos ahorraremos el tiempo de interpretar ese Javascript cada vez que se llame a la función.

Debido a que Javascript es un lenguaje muy extenso sería muy costoso realizar un intérprete de todo el lenguaje. Además la interacción con los mapas se define utilizando un subconjunto más bien pequeño del lenguaje. Por estas razones, y teniendo siempre en mente que es vital que el sistema en conjunto ocupe poco, se decidió hacer un analizador que soportase solo los elementos necesarios de Javascript para dotar a un mapa de interactividad, pero que a su vez fuese altamente extensible para poder, en un futuro, añadir más elementos de Javascript de forma rápida y sencilla. Los elementos soportados son los siguientes: creación de variables, asignación de valores a las variables creadas, acceso a atributos de elementos geográficos que estén dentro de una variable, modificación de atributos de elementos geográficos que estén dentro de una variable, operaciones aritméticas básicas entre dos números, concatenación de dos cadenas de caracteres, comparaciones entre dos números, comparaciones de igualdad o desigualdad de dos cadenas de caracteres, e instrucciones condicionales. Faltaría por tanto implementar el soporte para: el resto de sentencias de control de flujo, operaciones entre más de dos números o cadenas de caracteres, expresiones lógicas complejas, y todas las funciones que tiene Javascript predefinidas.

6 Módulo cliente de WMS

Este módulo se encarga de la interacción con el mapa antes de analizarlo y pintarlo, la diferencia con el subsistema de interacción con el usuario es que el de interacción con el usuario interactúa con el mapa una vez este había sido analizado y pintado. Este sistema está ligado al WMS que utilicemos pues a él será a quien le pida los mapas. Por tanto tiene que permitirnos ajustar el tamaño en píxeles del mapa que vamos a pedir, así como añadir o quitar capas. También nos debe permitir establecer el *bounding box*, así como modificarlo mediante acercamientos, alejamientos y desplazamientos del mapa.

Un problema que surge es que los WMS a día de hoy no devuelven mapas SVG con interactividad incorporada, por lo que este subsistema tiene que incorporar actividad a los mapas que le devuelva el WMS. Por ello se implementó un *servlet* que se instala en el lado del servidor y actúa como un servidor de mapas activos, recibiendo peticiones *GetMap*, que delega en un WMS, y les añade la actividad que corresponda a los mapas que recupere.

Por otra parte, si pedimos a un servidor de mapas un mapa con varias capas, este nos devolverá un único SVG en el que estarán representadas todas las capas pedidas. Si un usuario desea añadir o quitar una nueva capa deberíamos generar una nueva petición con todas las capas anteriores más la que añadió el usuario. Sin embargo, la información acerca de las capas que ya habíamos pedido la tenemos dentro del SVG anterior, pero al estar mezcladas todas las capas dentro de un único SVG no podemos invalidar una única capa o añadir una capa de forma individual. Para solucionar esto, este módulo no pide mapas enteros al servidor de mapas activos, sino que divide la petición del mapa en varias peticiones, una por cada capa. De esta forma recibiremos varios SVG que analizaremos independientemente y luego pintaremos en pantalla en el orden que les corresponda. De esta forma, si el usuario quita una capa, lo único que deberemos hacer es pintar de nuevo todo el mapa, pero dejaremos sin pintar el SVG correspondiente a la capa quitada, y si el usuario decide añadir una nueva capa al mapa, se generará una única petición en la que pediremos el SVG correspondiente a esa capa y pintaremos de nuevo el mapa incluyendo la nueva capa en la posición que le corresponda.

Al ser SVG un formato basado en XML, los mapas están en formato texto y ocupan una cantidad considerable de espacio. Dado que los mapas deben enviarse por Internet desde el servidor de mapas hasta el *applet* es fundamental que ocupen lo menos posible para que no se tarde demasiado tiempo en su transferencia. Por ello nuestro servidor de mapas activos comprime los mapas antes de enviarlos al *applet*, el cual los descomprime antes de su analizado. Con esta medida disminuimos bastante el tiempo de transferencia de los mapas a través de la *web*, algo especialmente importante para conexiones con bajo ancho de banda.

Finalmente, puesto que en el SVG las coordenadas de los objetos geográficos están en referencia a la pantalla no podemos saber las coordenadas de estos objetos en el SRS del WMS solo mediante el SVG, por lo tanto este subsistema se encargara de hacer la traducción de coordenadas del mapa a la pantalla.

7 Módulo de integración

Los principales requerimientos de este subsistema son los de proporcionar un punto de acceso único a toda la funcionalidad del sistema, mantener la coherencia entre el *bounding box* del mapa que será pedido y el del que esta siendo pintado, gestionar la petición en paralelo de las capas y permitir la configuración global del sistema. El primer requerimiento implica que el sistema cliente sólo tenga que hacer uso de este subsistema para usar la aplicación sin necesidad de saber como se hacen las cosas por debajo. Por lo tanto será este subsistema el que se comunique con los otros tres subsistemas. Para mantener la coherencia entre los *bounding box* del subsistema de pintado y el de gestión de la información geográfica, este subsistema de integración deberá soportar la funcionalidad de hacer zooms y desplazamientos sobre el mapa delegando en los subsistemas de pintado y de gestión de la información geográfica. Por lo tanto las herramientas del subsistema de interacción con el usuario que modifiquen el *bounding box* deberán llamar a este subsistema, como por ejemplo las herramientas de zoom y de desplazamiento.

Para ahorrar tiempo en la carga de los mapas la petición de las capas se realiza en paralelo utilizando varios hilos de ejecución, delegando en el subsistema de cliente WMS y en el subsistema de visualización de SVG. Este módulo se encargará por tanto de gestionar y lanzar nuevos hilos de ejecución cuando sea necesario. Lanzar un hilo de ejecución por cada capa de un mapa puede llegar a sobrecargar el servidor de mapas y degradar el tiempo que se tarda en pedir el mapa, por ello se permite configurar el número máximo y mínimo de capas que el sistema va a pedir simultáneamente.

8 La Biblioteca Virtual Galega

La Biblioteca Virtual Galega [2], puesta en marcha en Febrero de 2002, pretende ayudar a paliar la escasez de contenidos relativos al idioma y literatura gallegos en la Web. Es un proyecto amplio desarrollado por el *Laboratorio de Bases de Datos* de la *Universidade da Coruña* y financiado por la *Deputación Provincial de A Coruña*. Los principales objetivos de ésta biblioteca virtual son:

- Facilitar el acceso a la Literatura Gallega, incluidas las publicaciones de autores clásicos.
- Actuar como una plataforma de publicación que permita tanto a autores conocidos como noveles la publicación electrónica de sus obras.

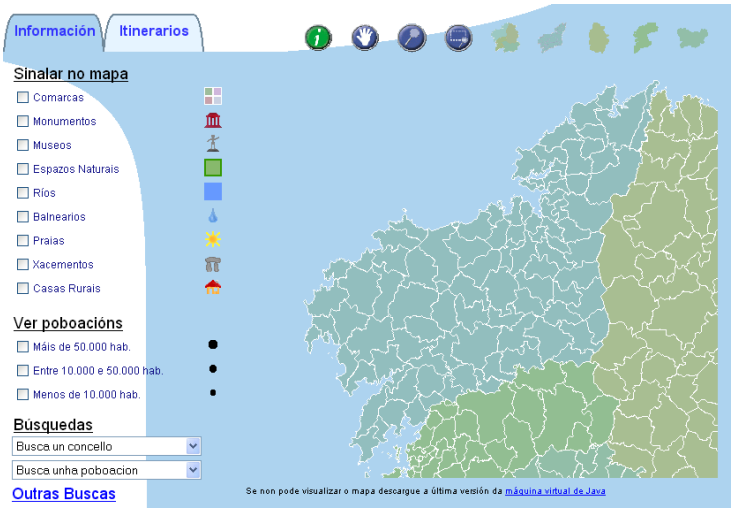

Ilustración 4. Página principal del Viaxe Virtual por Galicia.

El *Viaxe Virtual por Galicia* es una aplicación *web* integrada en la *Biblioteca Virtual Galega* que permite la visualización de información geográfica de Galicia, así como la consulta de datos de los elementos que se visualizan en el mapa. Para ello proporciona una página de inicio donde se visualiza un mapa inicial predeterminado así como una serie de herramientas para su modificación y una página de búsqueda donde se pueden buscar diversos elementos para visualizar información relativa a los mismos. Esta aplicación permite además agregar y quitar capas del mapa, realizar desplazamientos, acercamientos o alejamientos, encuadrar una determinada zona del mapa, visualizar información alfanumérica de las entidades que componen el mapa al pulsar con el ratón sobre ellas, marcar en el mapa los ayuntamientos seleccionados cambiándoles el color, señalar en el mapa otros elementos, y realizar búsquedas de elementos por el nombre, completo o parcial.

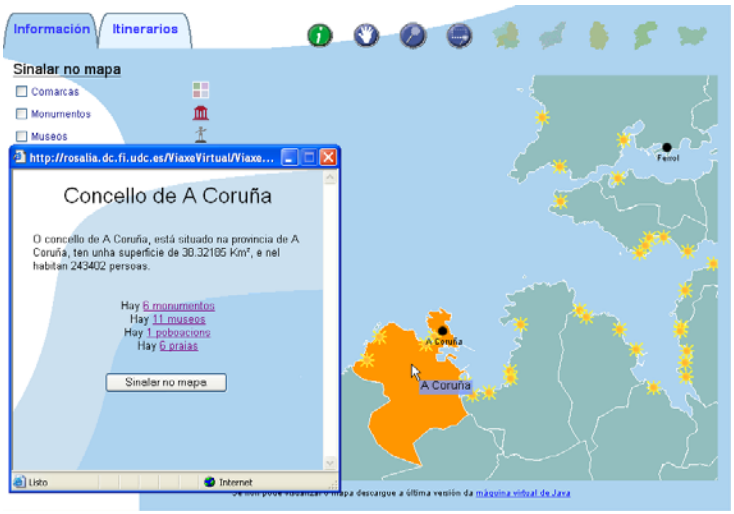

Ilustración 5. Ejemplos de actividad en los mapas.

En la Ilustración 4 podemos ver la página principal de la aplicación, tal y como se muestra cuando entramos en la aplicación. En la parte superior derecha hay una serie de botones que nos permitirán interactuar con el *applet*. Los cuatro primeros nos permiten cambiar el modo de actuación del *applet*: el primero de ellos hace que el *applet* ejecute la interactividad definida por el propio mapa vectorial activo; el segundo nos permite desplazar el mapa sin variar la escala del mismo; el tercero nos permite acercarnos o alejarnos del mapa al pulsar sobre él con el botón izquierdo o derecho del ratón respectivamente; y el cuarto nos permite encuadrar una determinada zona del mapa. Los cinco siguientes iconos nos permiten encuadrar el mapa de forma automática en Galicia o en cualquiera de las provincias. Además, estos botones quitarán todas las capas cargadas por el usuario dejando solo las capas iniciales. En la parte superior izquierda de la pantalla hay dos pestañas que nos permiten variar el tipo de capas que podemos cargar en el mapa. La pestaña *información* nos permitirá cargar capas de información general y realizar búsquedas de municipios, poblaciones y búsquedas avanzadas. La pestaña *itinerarios* nos permite cargar capas de rutas literarias, itinerarios y de carreteras.

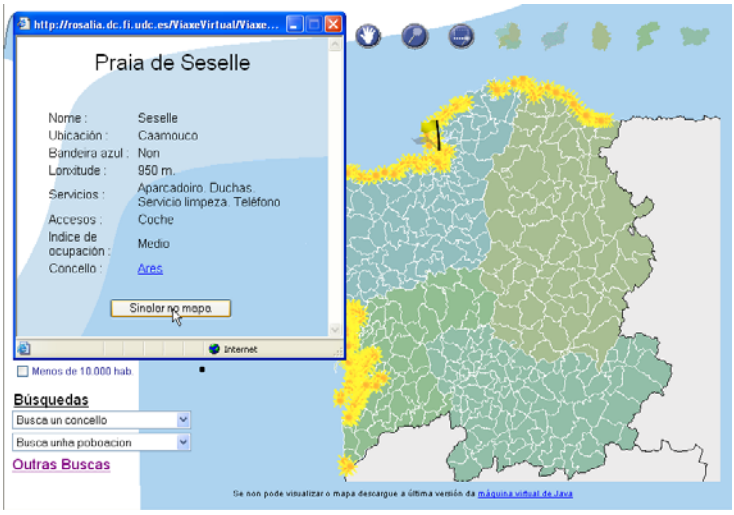

Ilustración 6. Marcado de un punto del mapa.

En la Ilustración 5 podemos ver las principales formas de interactividad que vienen embebidas dentro de los mapas activos. En este ejemplo podemos ver las formas en que los mapas representados por el *applet* reaccionan ante los eventos de usuario. Así los municipios cambian de color al entrar el puntero del ratón dentro de ellos, recuperando su color original al salir. Además se muestra un mensaje de texto con el nombre del elemento que esta siendo señalado por el usuario en cada momento, en este caso nos indica que el municipio sobre el que se encuentra el ratón es el de A Coruña. Por ultimo, al hacer *click* en un elemento del mapa se abre una nueva ventana del navegador mostrando información más completa acerca del mismo.

En la Ilustración 6 puede verse un ejemplo de la funcionalidad que ofrece el *applet* para marcar un punto dentro del mapa. En el ejemplo aparece la playa de Seselle marcada mediante una bandera amarilla.

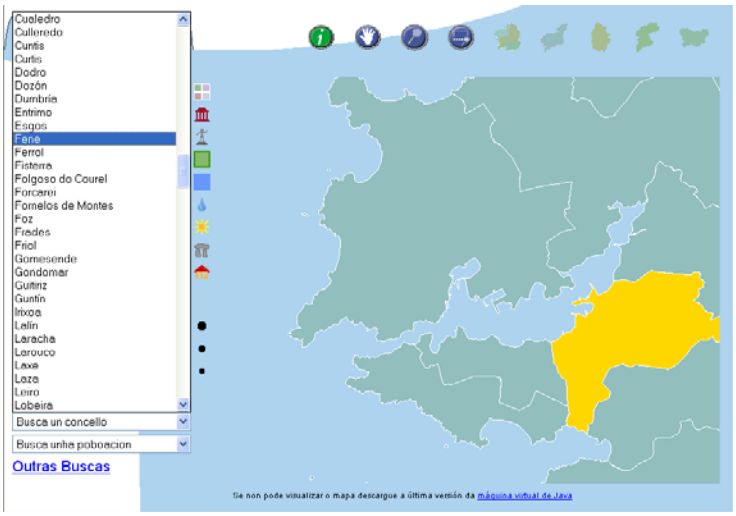

Ilustración 7. Búsqueda de municipio.

En la Ilustración 7 podemos ver un ejemplo de la búsqueda de municipios, en el ejemplo buscamos el municipio de Fene y este cambia inmediatamente de color pasando de un color verde a amarillo. De igual forma se permite la búsqueda de poblaciones, solo que al ser puntos aparecerán representadas en el mapa por una bandera.

9 Conclusiones y trabajo futuro

En este trabajo se ha desarrollado un componente Java para la visualización de mapas activos en formato SVG. La principal utilidad de este componente es su utilización como *applet* dentro de una aplicación SIG en *web*, lo que permitirá la visualización de estos mapas activos a través de Internet, con la única condición de que el usuario tenga instalada la maquina virtual de Java. Las características principales de este componente son:

- Permite la visualización en navegadores de mapas vectoriales activos. Estos mapas, a diferencia de lo que están en formato *raster*, permiten que el usuario interactúe con ellos utilizando el ratón.
- Ofrece herramientas para la navegación por el mapa mediante zooms y desplazamientos.
- Realiza la gestión de la información geográfica permitiendo agregar o quitar capas. De esta forma oculta al usuario la comunicación con el WMS del que extrae los mapas.

- Gracias a la compresión de las comunicaciones entre el WMS y el *applet* conseguimos reducir el tamaño de los mapas aproximadamente a un 10%-30% de su tamaño original, con la consecuente disminución en los tiempos de transferencia de los mapas a través de Internet.
- Gracias al uso de un *R-Tree* como indexador espacial se consiguen un acceso a los elementos del mapa prácticamente instantáneo.
- Gracias a la optimización realizada en el pintado de mapas el sistema ofrece una respuesta más que aceptable al manejar mapas activos de gran tamaño (del orden de un mega y medio).

Hay que destacar que cuando los navegadores Web incorporen de forma nativa el formato SVG este proyecto dejara de tener utilidad, ya que de el pintado del mapa y de la interpretación de la actividad se encargara el propio navegador. Sin embargo las ideas y algoritmos utilizados en los módulos de gestión de la información geográfica y las herramientas utilizadas para modificar el mapa seguirán siendo útiles, pero en vez de implementarlas en un *applet* tendrían que ir, por ejemplo, en una librería que se usase en la aplicación Web.

Tras la realización de este proyecto se plantean las siguientes líneas de trabajo:

- Añadir los elementos de SVG y de Javascript que no están soportados por el parser para que este sistema se pueda utilizar para pintar cualquier tipo de SVG y no esté restringido a aplicaciones geográficas.
- Realizar una mejor implementación de la parte que se encarga de añadir actividad a los mapas para que soporte la inclusión de interactividad en las capas de forma similar a como soportan los WMS los estilos.

Referencias

[1] Antonin Guttman, “R-trees: a dynamic index structure for spatial searching”, Proceedings of the 1984 ACM SIGMOD international conference on Management of data, June 18-21, 1984, Boston, Massachusetts

[2] Biblioteca Virtual Galega (BVG), página principal, http://bvg.udc.es/

[3] Extensible Markup Language (XML) 1.0 (Fourth Edition), publicación web, http://www.w3.org/TR/2006/REC-xml-20060816/

[4] OpenGeospatial Consortium (OGC), página principal, http://www.opengeospatial.org/

[5] Scalable Vector Graphics (SVG) 1.1 Specification, publicación web, accesible en http://www.w3.org/TR/SVG/

[6] Web Map Service (WMS), publicación web, http://www.opengeospatial.org/standards/wms

[7] World Wide Web Consortium (W3C), página principal, http://www.w3.org/