Integración de APIs políglotas de mapas en Google Web Toolkit: IDELabMapstractionGWT

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

La continua evolución de los ordenadores y otros dispositivos, unida a las mejoras constantes que están presentando los distintos navegadores web han aumentado en gran medida las posibilidades que pueden ofrecer estos navegadores. Dentro del campo de los Sistemas de Información Geográfica, esta evolución ha pasado de imágenes estáticas de mapas a complejos globos virtuales que dan una visión más detallada del terreno en la que se puede apreciar su orografía.

Sin embargo, a la hora de desarrollar este tipo de aplicaciones, pueden surgir problemas a causa del lenguaje de programación utilizado. Javascript presenta diferentes problemas, como los que surgen entre las diferentes formas que tienen los distintos navegadores de interpretarlo, o por ciertas inconsistencias debidas al bajo nivel de tipado. Por ello se ha buscado una alternativa que pueda solventar estos problemas. Esta alternativa pasa por integrar la librería IDELabMapstraction dentro del framework Google Web Toolkit. Este framework permite programar aplicaciones web en Java, que posteriormente se compilan, obteniendo el código Javascript equivalente, abstrayendo al desarrollador de las diferencias entre unos navegadores y otros.

Al utilizar Java como lenguaje de programación, resulta más sencillo crear aplicaciones más complejas y robustas. Los mapas se integran fácilmente dentro de la cuidada interfaz de usuario de GWT, de la que se pueden aprovechar todos sus componentes para poder generar mashups más complejos y vistosos.

En este trabajo se ha integrado la librería IDELabMapstraction como un nuevo módulo para GWT, denominado IDELabMapstractionGWT, en el que también se han diseñado distintos controles genéricos que se pueden asociar al mapa para que los usuarios puedan interactuar con él de forma intuitiva. Como resultado de esta integración se ha creado el visor MirameDuero, un completo visor web realizado en colaboración con la Confederación Hidrográfica del Duero.

Palabras clave: IDELabMapstraction, *Google Web Toolkit, Javascript, navegador web, visor.*

ABSTRACT

The continued evolution of computers, coupled with constant improvements in web browsers has increased the potential of these browsers. In the Geographical Information Systems field, this evolution has moved from static images of maps to complex virtual globes that give a more detailed view of the terrain and the orography.
However, when developing complex applications can arise problems because of the programming language used. Javascript have different problems, such as the differences between different web browsers engines. Therefore, it was necessary to find an alternative. The problem was solved integrating the IDELabMapstraction library into the Google Web Toolkit framework. This framework allows to developers to use Java as programming language. Then the code is compiled, getting the Javascript equivalent code, abstracting the developer from the differences between web browsers.
By using Java as a programming language, it is easier to create more complex and robust web applications. The maps are easily integrated into the GWT user interface, which can take advantage of all its components in order to generate more complex and colorful mashups.
This work has included the IDELabMapstraction library in the Google Web Toolkit framework as a module called IDELabMapstractionGWT, which also provides several generic controls that can be associated to the map so that users can interact with it intuitively. This module allows developing more easily sophisticated viewers as the viewer MirameDuero, a complete web map viewer developed in collaboration with the Hydrographic Confederation of the Duero.

Key words: IDELabMapstraction, *Google Web Toolkit, Javascript, map viewer, web browser.*

INTRODUCCIÓN

La complejidad que está adquiriendo Mapstraction [1,2] con las aportaciones que se están realizando con IDELabMapstraction[1] y otros proyectos es cada vez mayor. En anteriores versiones de la librería se han ido integrando, por ejemplo, nuevos proveedores [3] o la interactividad a los mapas [4]. En este trabajo se trata el siguiente paso que se está dando para ofrecer una mayor funcionalidad y flexibilidad desde la librería, su integración dentro del compilador de aplicaciones Javascript *Google Web Toolkit*.

Google Web Toolkit es un *framework* creado por Google que facilita la creación de componentes web Javascript [5]. El desarrollo de componentes Javascript suele resultar un proceso tedioso, ya que cada navegador tiene sus propias peculiaridades y el programador de las aplicaciones debe comprobar que funcionan correctamente en cada uno de los navegadores. Además, existen otros problemas para los desarrolladores, debidos a que este lenguaje está muy poco *tipado* y la orientación a objetos no está totalmente integrada [6,7].

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

Por ello Google Web Toolkit proporciona una herramienta que permite al desarrollador programar sus aplicaciones web en lenguaje Java, que posteriormente al compilarlas devuelven el código Javascript y HTML equivalente, como se puede ver el esquema de la Figura 1. Además, es el propio GWT el encargado de hacer que el Javascript generado funcione correctamente en los distintos navegadores.

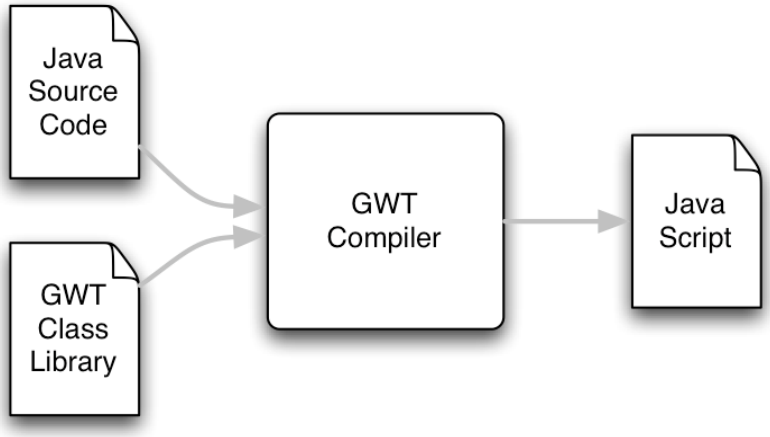

Figura 1 Esquema de la filosofía de Google Web Toolkit

Para que sea posible la integración de los mapas ofrecidos por la librería IDELabMapstraction dentro del *framework* de *Google Web Toolkit,* será necesario crear un módulo para éste en el que se indiquen los pasos a seguir para mapear los objetos creados con GWT a sus equivalentes en Javascript. Una vez creado este módulo, será posible integrar los mapas de forma sencilla en cualquier página web como un único componente totalmente operativo.

El objetivo principal de este desarrollo es la integración de los mapas que proporciona IDELabMapstraction como un componente más de GWT, de forma que sea fácilmente incrustable dentro de cualquier otro contenedor proporcionado por el *framework*. Esta implementación se beneficia de la simplicidad a la hora de programar en este lenguaje y el ahorro de tiempo para el programador que supone no tener que preocuparse por las diferencias existentes a la hora de interpretar el Javascript por parte de los distintos navegadores. Pero este objetivo principal plantea la consecución de otros retos secundarios, como puede ser la creación de controles auxiliares que permitan una interacción sencilla con el mapa, así como la gestión de los eventos que se producen en el mapa y que se deben propagar hasta el componente GWT.

Otro objetivo secundario de esta nueva implementación se refiere a la posibilidad de crear sobre el componente básico, una arquitectura desarrollada para el componente que permita la creación de nuevas funcionalidades más avanzadas en el futuro directamente sobre el componente GWT, sin necesidad de desarrollarlas en el Javascript nativo de la librería. Estos objetivos ya mencionados, unidos a la posibilidad de distribuir este componente de forma totalmente libre para que cualquier programador sea capaz de crear sus *mashups* de mapas o colaborar en el desarrollo de nuevas funcionalidades hacen que este nuevo paso adelante sea a priori, muy interesante.

El resto del documento se estructura de la siguiente manera. En primer lugar se presenta una comparativa entre las ventajas y desventajas de desarrollar este tipo de aplicaciones con GWT o directamente en Javascript. A continuación se hace una presentación del *framework Google Web Toolkit*, remarcando sus bondades para este tipo de aplicaciones. La siguiente sección se centrará en el diseño de la solución desarrollada en este trabajo, así como su desarrollo. Posteriormente se mostrará un breve caso de uso de este desarrollo, un prototipo basado en el visor CuídameDuero

de la Confederación Hidrográfica del Duero. Finalmente se mostrarán unas breves conclusiones y posibles líneas futuras de este trabajo.

JAVASCRIPT VS GWT

Uno de los inconvenientes que presentan estas APIs (*Application Programming Interface*) para clientes ligeros es que están escritas en Javascript, un lenguaje de programación que presenta algunos inconvenientes a la hora de realizar aplicaciones complejas, ya que no está totalmente orientado a objetos y no es un lenguaje *tipado* [6,7]. Además, el hecho de que sea un lenguaje interpretado por parte del navegador hace que las diferencias entre los motores Javascript de los navegadores lleguen a ser aspectos críticos a la hora de programar estas aplicaciones y hagan perder gran cantidad de tiempo a los desarrolladores por tener que comprobar que las aplicaciones funcionan en los distintos navegadores y corrigiendo las posibles incompatibilidades que puedan surgir entre unos y otros. Una de las alternativas que pueden acabar con las desventajas de estos clientes ligeros es la posibilidad de desarrollarlos dentro del *framework Google Web Toolkit* (GWT)[2]. Con esta herramienta es posible desarrollar aplicaciones web utilizando para ello el lenguaje Java, por lo que los dos primeros problemas explicados anteriormente desaparecen. Estas aplicaciones son finalmente compiladas y convertidas a código Javascript comprimido y compatible con todos los navegadores. De esta forma, la labor para el desarrollador de estas aplicaciones web basadas en mapas será mucho más sencilla. Sin embargo, el primer paso para poder desarrollar estas aplicaciones desde GWT es integrar la interfaz que ofrece Mapstraction como un componente más, que podrá ser utilizado de forma genérica dentro del *framework.*

En [8] se presenta una problemática similar, en la que se presenta la disyuntiva de crear un GIS con tecnologías GWT o Javascript. Tras un análisis de los pros y los contras de ambas alternativas, los resultados fueron favorables a la creación de esta herramienta utilizando GWT. Razones como las incompatibilidades entre navegadores, inconsistencias entre los tipos de datos en Javascript o la dificultad para encontrar programadores con experiencia en este lenguaje hicieron que la balanza se decantara en favor de GWT.

Dado que la complejidad que la librería IDELabMapstraction es cada vez mayor, y que el lenguaje Java ofrece ventajas sobre Javascript a la hora de la gestión de la orientación a objetos, resulta interesante llevar a cabo la integración de la librería dentro de este *framework* para ofrecer la posibilidad de crear *mashups* de mapas más complejos, pero creados de forma más sencilla gracias a las posibilidades que ofrece GWT.

GOOGLE WEB TOOLKIT

Google Web Toolkit es una potente herramienta para el desarrollo de aplicaciones Javascript de forma sencilla a través de un entorno de desarrollo en lenguaje Java. La arquitectura de GWT está basada en cuatro componentes principales (Ver Figura 2).

[2] http://code.google.com/intl/es-ES/webtoolkit/

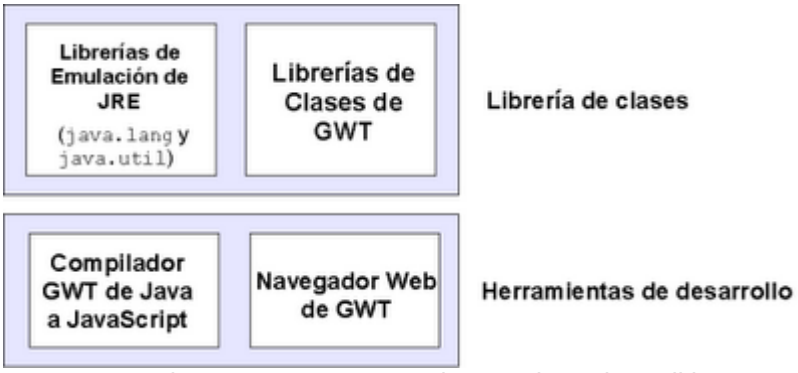

Figura 2 Componentes de *Google Web Toolkit*

Los componentes son:

- **Compilador GWT Java-a-JavaScript:** El Compilador GWT Java-a-JavaScript traduce del lenguaje de programación Java a JavaScript. El compilador se utiliza cuando necesites correr tu aplicación en modo web.

- **Navegador web *hosted* de GWT:** El Navegador web *hosted* de GWT permite correr y ejecutar GWT aplicaciones en modo *hosted*, donde lo que estás corriendo son *bytecodes* de Java sobre una máquina virtual sin compilarlos a JavaScript. Para lograr esto, el navegador GWT incrusta un controlador de browser especial (un control del Internet Explorer sobre Windows o un control de Gecko/Mozilla sobre Linux) con *hooks* dentro de la máquina virtual de Java.

- **Emulación de librerías JRE:** GWT contiene implementaciones en JavaScript de las librerías de clases más usadas en Java, incluyendo la mayoría de las clases del paquete java.lang y un subconjunto de clases del paquete java.util. El resto del estándar de librerías de Java no es soportado nativamente con GWT. Por ejemplo, las clases de los paquetes como java.io no se utilizan en aplicaciones web ya que estas acceden a recursos en la red y al sistema de archivos local.

- **Librería de clases de interfaz de usuario de GWT:** Las librerías de clases de interfaz de usuario de GWT son un conjunto de interfaces y clases personalizadas que te permiten crear *widgets* para el navegador, como botones, cajas de texto, imágenes, y texto. Éste es el núcleo de las librerías de interfaz de usuario para crear aplicaciones GWT.

GWT es capaz de emular en Javascript los paquetes más básicos de Java java.lang o, en parte, java.util para llevar a cabo la parte no visual de las aplicaciones. Para la parte visual, GWT ofrece un completo soporte capaz de realizar prácticamente cualquier tipo de aplicación, ofreciéndose un completo conjunto de *widgets*, como botones, tablas o árboles, que son organizados mediante un amplio abanico de paneles [5]. A partir de ese código, se genera el código Javascript equivalente para poder mostrar la aplicación en el navegador. Con GWT se puede depurar el código que se ejecuta en el navegador en el propio entorno de desarrollo, como si de código Java nativo se tratara. En la Figura 3 se puede ver un resumen del proceso de creación de aplicaciones con GWT.

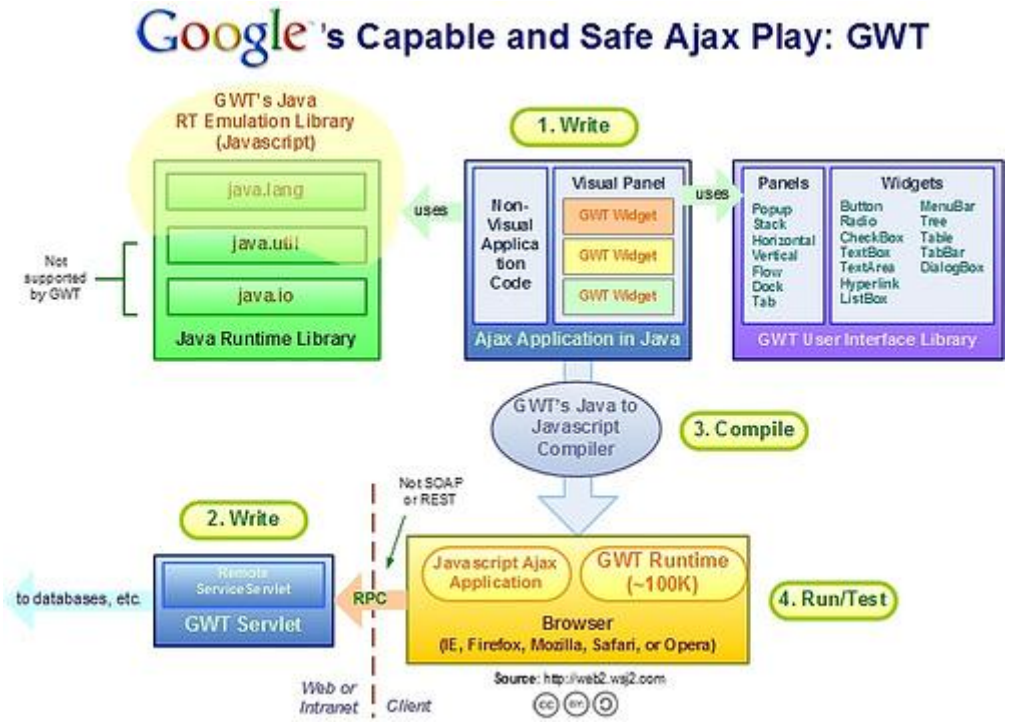

Figura 3 Esquema del proceso de creación de aplicaciones con *Google Web Toolkit*

GWT ofrece distintas maneras de crear *widgets* personalizados. La forma más fácil es crear *widgets* compuestos por grupos existentes *widgets* básicos y añadiendo una lógica de interacción con ellos. También es posible desarrollar *widgets* utilizando las *interfaces* Java de bajo nivel de utilizadas por los propios componentes GWT estándar o las *interfaces* de muy bajo nivel de JavaScript. Al crear un *widget* personalizado es necesario encontrar su lugar en la jerarquía de clases de GWT. En la Figura 4 se muestran las clases abstractas básicas de los componentes gráficos de GWT.

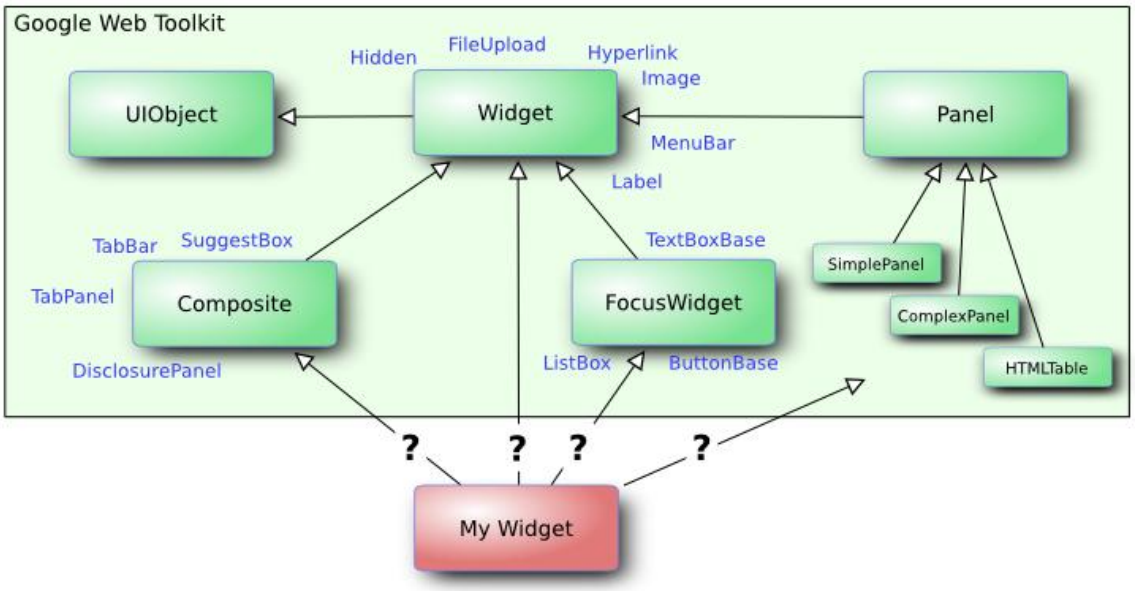

Figura 4 Arquitectura de los componentes gráficos de *Google Web Toolkit*. Tomada de [9]

GWT también da la posibilidad de hacer que las aplicaciones diseñadas tengan acceso a una parte externa alojada en un servidor. GWT ofrece dos maneras diferentes para comunicarse con un servidor vía HTTP. Utilizar el *framework* GWT RPC para hacer transparente las llamadas a los *servlets* Java y dejar que GWT se

ocupe de detalles de bajo nivel como la serialización de objetos. Alternativamente, se pueden utilizar las clases *GWT HTTP client* para construir y enviar peticiones HTTP personalizadas.

DISEÑO DE LA SOLUCIÓN

La integración de componentes de tipo mapa dentro de GWT es un proceso que ya ha sido implementado por otras soluciones, como Google Maps[3] y OpenLayers[4], pero con la desventaja de que son aproximaciones orientadas a un único cliente de mapas. También existe una aproximación denominada GWT-Ext Maps[5], que se basa en Mapstraction para la integración de mapas dentro de la librería GWT-Ext. Al mismo tiempo, hay aproximaciones similares para otras librerías Javascript de alto nivel, como el proyecto GeoExt[6] para ExtJS[7], que es una librería nativa en Javascript que también ofrece abstracción entre los distintos navegadores al desarrollador.

Una vez estudiados los diferentes componentes de mapas existentes, se decidió llevar a cabo el desarrollo de forma similar al que se ha seguido para la integración del cliente de OpenLayers[8] dentro de un componente GWT. El proceso consiste en mapear cada tipo de objeto Javascript y sus distintos métodos a clases Java con sus métodos correspondientes. De esta forma resulta sencilla e intuitiva la integración de la librería dentro de un módulo GWT.

El proceso seguido es sencillo, como se puede ver en la Figura 5, y tan sólo consta de dos elementos intermedios. El primero consiste en crear la clase que se encargue de mapear los métodos que expone el API Javascript en Java. De esta forma, será posible acceder a los métodos que se exponen en la librería original desde cualquier aplicación Java. En este proceso hay que tener muy en cuenta los tipos de datos tanto de los parámetros de entrada como de los de retorno, ya que esta limitación no se tiene en cuenta en Java. Esta clase se encarga internamente de transformar estos tipos de datos a objetos *JSObject*. La clase *JSObject* es la encargada de mapear la clase Object Javascript en Java. De esta forma, esos objetos se podrán enviar posteriormente al código Javascript sin problemas. Finalmente, se encarga de realiza la llamada al segundo de los elementos de este proceso, la clase que ejecuta el código nativo Javascript.

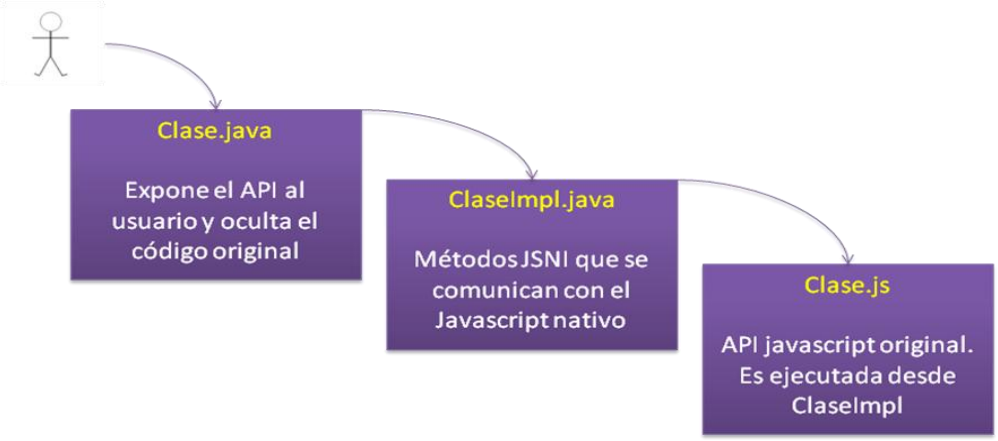

Figura 5 Esquema del mapeo desde Objetos Javascript a Objetos GWT

[3] http://code.google.com/p/gwt-google-apis/wiki/MapsGettingStarted
[4] http://sourceforge.net/projects/gwt-openlayers/
[5] http://www.gwt-ext.com/demo-maps/
[6] http://www.geoext.org/
[7] http://www.sencha.com/products/js/
[8] http://openlayers.org

La segunda de las clases es la encargada de ejecutar el código Javascript que ofrece el API de Mapstraction cuando éste es requerido por la aplicación. Para ello, hace uso del *Javascript Native Interface* (JSNI). Este *interface* permite inyectar código Javascript directamente dentro de las aplicaciones GWT [5]. JSNI en una técnica poderosa, pero debe utilizarse con moderación, porque la escritura de código Javascript consistente es notoriamente difícil. El código JSNI es potencialmente menos portable entre los distintos navegadores, más propenso a desperdiciar memoria, menos amigable para las herramientas de Java, y más difícil de optimizar para el compilador.

Con la utilización de este proceso se consigue ofrecer desde GWT la misma interfaz que ofrece la librería Javascript de forma nativa de una forma sencilla y sin necesidad de llevar a cabo ningún tipo de modificación en la librería original. Además, esta estrategia es altamente escalable, ya que cualquier ampliación en el API de Mapstraction se puede reflejar rápidamente en este módulo de forma poco intrusiva. Otra gran ventaja, es que al tener completamente integrada el API original, se pueden implementar nuevas funcionalidades a partir de este módulo escritas íntegramente sobre GWT, sin necesidad de escribirlas en Javascript, con las ventajas que ello conlleva en cuanto a simplicidad y solidez del código.

Este proceso se ha seguido estrictamente con todas las clases integradas dentro de la librería IDELabMapstraction a excepción de la clase Mapstraction, que es la que mapea el visor. Este hecho se da porque esta clase es la que debe ser un elemento a integrar dentro de la interfaz de usuario (UI, del inglés *User Interface*) de GWT. La jerarquía de los componentes de la UI es la que se muestra en la Figura 4 Arquitectura de los componentes gráficos de *Google Web Toolkit*. Tomada de [9], en la que se puede observar cómo se dividen principalmente en dos grandes grupos, los *widgets* y los paneles, el mapa se encuadrará lógicamente en el primero de los grupos. De entre los distintos tipos de *widgets*, se ha decidido que la clase que defina el mapa heredará directamente de la clase *composite*, que es la más adecuada para estos casos. Los *composites* son de lejos la manera más efectiva de crear *widgets*. Es posible combinar grupos de *widgets* dentro de un *composite* que es en sí mismo un *widget* re-utilizable. *Composite*, es un *widget* especial que puede contener otros componentes (por lo general, un panel) pero se comporta como si él fuera su *widget* contenido. Es preferible usar *Composite* que intentar crear *widgets* complejos usando subclases de panel ya que normalmente controla de una mejor forma los métodos que son publicados.

De esta forma, en el propio constructor de la clase Mapstraction crea la capa necesaria para mostrar el mapa y se integra perfectamente dentro del *layout* creado para la aplicación con la composición de los diferentes paneles que ofrece GWT. Del resto del proceso de carga y visualización de los mapas se encarga el código Javascript ya integrado en Mapstraction. Una vez que el mapa se ha creado se puede disponer de todas las funcionalidades que ofrecen los *widgets* de GWT, así como acceder a todos los métodos que ofrece el API de Mapstraction.

También se han desarrollado varios componentes asociados a los mapas que facilitan la labor del desarrollador y que se integran fácilmente en la aplicación, como pueden ser los controles para aumentar y disminuir el zoom, asignar un nuevo *bounding box* al mapa, añadirá capas a los mapas, la gestión de las *features*, etc. En la Figura 6 se puede ver una captura de una de las demos creadas para mostrar las capacidades de estos componentes.

Figura 6 Captura de una demo de IDELabMapstraction GWT

Los componentes gráficos que se pueden asociar a los mapas que se han desarrollado hasta el momento son los siguientes:

- **Controles de Zoom**: Hay de 2 tipos, el primero presenta dos botones, para aumentar o disminuir el nivel de zoom uno por uno. El segundo de ellos despliega un cuadro de texto en el que indicar que nivel de zoom se desea.
- **Gestión de las *features*:** Este control cuenta con un selector para indicar el tipo de geometría que se quiere añadir al mapa y un botón para iniciar el proceso de creación de *features*. También incluye otro botón para activar y desactivar el modo de edición.
- **Eventos propagados:** Es un cuadro de texto en el que se van notificando los eventos que tienen lugar en el mapa. Está orientado principalmente para labores de desarrollo.
- **Gestión del centro del mapa:** Despliega un cuadro de texto en el que indicar las coordenadas sobre las que centrar el mapa.
- **Gestión del *bounding box*:** Despliega un cuadro de texto en el que indicar las coordenadas del *bounding box* que se desee mostrar.
- **Gestión de las capas:** Permite mantener una lista de capas asociadas a los mapas y mostrarlas u ocultarlas sólo seleccionando un *checkbox*. También incluye un botón que despliega un cuadro de texto en el que introducir nuevas capas.

Estos componentes se han desarrollado heredando a partir de alguno de los componentes de la UI de GWT, por lo general del *Button*, añadiendo en su constructor una referencia a un mapa de Mapstraction. De esta forma, al realizar una acción sobre el componente, éste directamente realiza las acciones necesarias sobre

el mapa. De esta forma, cualquier desarrollador tiene a su disposición una lista de componentes que puede añadir a los mapas para que éstos sean más accesibles para los usuarios de su página web de una forma sencilla y ahorrándoles trabajo. Al mismo tiempo, los desarrolladores pueden crear nuevos componentes similares fácilmente, que pueden ser posteriormente integrados dentro del módulo si se considera pertinente.

Todos estos componentes, así como el mapa han sido desarrollados como un módulo más para GWT, por lo que para utilizarlos tan sólo es necesario el fichero JAR en el que se distribuirá e incluirlo dentro de cualquier proyecto realizado con GWT para poder utilizarlo sin ningún tipo de restricción. Gracias a la integración de estos nuevos componentes en este módulo, resulta muy sencillo crear vistosos mapas sobre los que el desarrollador puede construir complejas aplicaciones web con muy poco esfuerzo.

EL VISOR CUÍDAME DUERO

Un buen ejemplo de la utilidad de los componentes desarrollados dentro del proyecto IDELabMapstractionGWT es el prototipo de visor CuídameDuero[9], que ha sido llevado a cabo con la Confederación Hidrográfica del Duero, cuyo objetivo es aprovechar este visor para poder hacer pública la información geográfica que la CHD posee de la cuenca del Duero con una solución independiente del proveedor de cartografía base y del software visor de mapas.

Este visor se aprovecha de la sencillez que ofrece GWT para desarrollar cuidadas interfaces de usuario. De esta forma el mapa se integra perfectamente entre otros controles diferentes, como puede ser un gestor de capas superpuestas o los controles para poder centrar el mapa en un determinado punto.

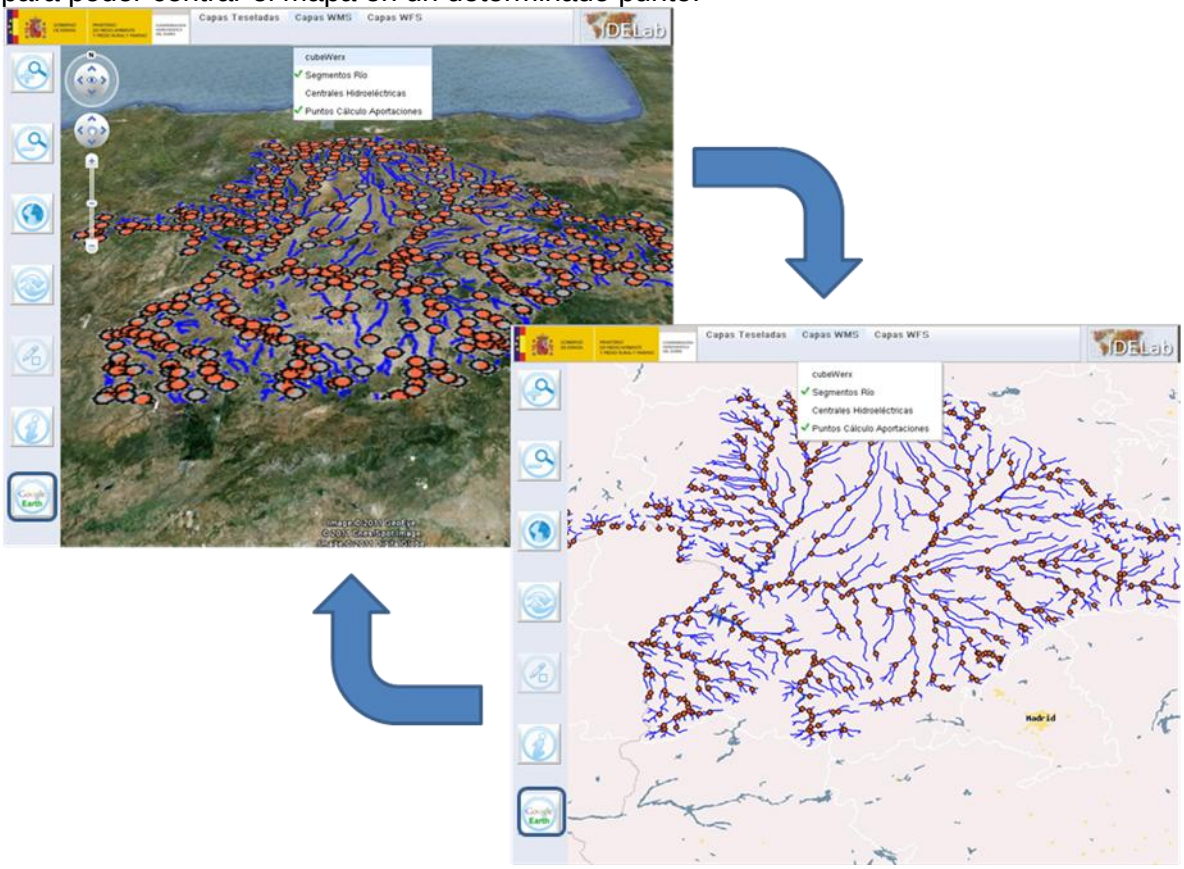

Figura 7 Captura del visor CuídameDuero

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

Sin embargo, la principal característica de este visor es que permite, gracias a Mapstraction, poder cambiar el cliente de mapas sobre el que se visualiza la información tan sólo con pulsar un botón. Con este control, se puede cambiar de una cartografía de base en 2D sobre un cliente OpenLayers a un cliente Google Earth de forma sincronizada, pudiéndose visualizar en ambos clientes la misma información geográfica, como se puede apreciar en la Figura 7.

Esta Unión de las características de Google Web Toolkit con las de Mapstraction hace que las aplicaciones resultantes sean vistosas y útiles, al mismo tiempo que sencillas de desarrollar y testear, por lo que resulta un gran avance poder utilizar este módulo para GWT.

CONCLUSIONES

El desarrollo IDELabMapstraction GWT ofrece una nueva posibilidad de crear mapas apoyados en el API Universal de mapas Mapstraction. Ahora estos mapas se pueden integrar dentro de cualquier aplicación que se desee crear utilizando Google Web Toolkit, por lo que el espectro de desarrolladores que tendrán la posibilidad de acceder a las ventajas que ofrece este API se verá aumentado considerablemente. Además, se ofrecen varios componentes gráficos que se pueden asociar fácilmente a los mapas y que ofrecen la posibilidad de interactuar con el mapa de forma sencilla.

Las ventajas que ofrece el hecho de programar en Java sobre GWT en lugar de hacerlo sobre el Javascript nativo son numerosas, pero principalmente caben destacar las ventajas de utilizar un lenguaje *tipado* y realmente orientado a objetos para llevar a cabo los desarrollos. El *widget* GWT que representa los mapas es totalmente integrable dentro de cualquiera de los elementos que ofrece la interfaz de usuario de GWT. Además, se han desarrollado una serie de componentes auxiliares para los mapas que facilitan las labores de los desarrolladores que deseen crear componentes auxiliares para los mapas, como pueden ser controles de zoom o para modificar el *bounding box*. Las ventajas que ofrece el modo de depuración *hosted* son visibles por la sencillez con la que se depuran estas aplicaciones comparado con las herramientas de depuración de código Javascript.

De esta manera queda mucho más clara la interfaz que ofrece cada uno de los componentes creados, así como la posibilidad de crear una estructura jerárquica de herencia mucho más clara y útil. La oportunidad de aumentar las posibilidades a partir de esta arquitectura sin necesidad de modificar el código Javascript original también puede hacer que la librería resulte atractiva a un mayor número de desarrolladores.

En definitiva, esta nueva integración añade nuevos atractivos a la librería original que pretenden que su uso se extienda más rápidamente y pueda llegar a considerarse una alterativa seria para cualquier desarrollador a la hora de llevar a cabo un *mashup* basado en mapas.

AGRADECIMIENTOS

El desarrollo de este trabajo ha sido posible gracias a la financiación de la Confederación Hidrográfica del Duero en el marco del Proyecto "Creación del Aula Confederación Hidrográfica del Duero para el avance de las tecnologías espaciales en la gestión del agua".

REFERENCIAS

[1] A. Turner, “mapstraction's Profile - GitHub,” *Mapstraction project's profile at GitHub* Available: http://github.com/mapstraction.

[2] A. Turner, “Mapstraction - a javascript library to hide differences between mapping APIs.” Available: http://mapstraction.com/.
[3] M. Pérez Monje, *Desarrollo de una librería Javascript para la creación de componentes Web de mapas que soporten los servicios oficiales de cartografía.*, Universidad de Valladolid, 2009.
[4] P. López Escobés, *Aplicación de técnicas de neogeografía en un sistema de gestión de contenidos web*, Valladolid (Spain): Universidad de Valladolid, 2009.
[5] R. Hanson y A. Tacy, *GWT in Action: Easy Ajax with the Google Web Toolkit*, Manning Publications, 2007.
[6] S. Jensen, A. Møller, y P. Thiemann, “Type Analysis for JavaScript,” *Static Analysis*, Springer, 2009, págs. 238-255.
[7] E. Burnette, *Google Web Toolkit: Taking the Pain Out of Ajax*, The Pragmatic Bookshelf, 2006.
[8] Pieter De Graef, “Web GIS: from Javascript to GWT,” 2009.
[9] M. Grönross, *Book of Vaadin*, Vaadin development team, 2010.