SigWeb 2.0: AJAX + OpenToro.

Moisés D. Díaz Toledano[(1)], *Marcos Velasco González*[(1)], *Vitalino Lázaro Pereira*[(1)] y *Angel Domínguez Fernández*[(1)]

[(1)] Subdirección de Sistemas Informáticos, Empresa Pública Desarrollo Agrario y Pesquero, Consejería de Agricultura y Pesca, Junta de Andalucía.

RESUMEN

En esta comunicación se describe el desarrollo de un nuevo visor SIG para la web, integrándole AJAX y el componente libre de publicación web OpenToro.

Palabras clave: *SIG, visores web, AJAX , OpenToro, publicación web.*

ABSTRACT

This communication presents the development of a new Internet GIS viewer that integrates a good set of AJAX functionalities and integrates OpenToro, a open source component for web publication.

Key words: *GIS, web viewer, AJAX, OpenToro, web publication.*

0.- INTRODUCCIÓN.

La web ha demostrado ser un medio excepcional para distribuir información, pero muy incómoda y deficiente como soporte para las aplicaciones. Ante el fracaso de XFORMS y otros estándares, AJAX (acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML (JavaScript y XML asíncronos, donde XML es un acrónimo de e**X**tensible **M**arkup **L**anguage)) se ha convertido en la tecnología web de referencia para dar un paso más en conseguir que la web sea un *soporte real para aplicaciones*.

En esta comunicación vamos a describir los elementos que hemos desarrollado para pasar de un visor SigWeb 1.0 centrado en el mapa casi como único elemento a un visor SigWeb 2.0, más parecido a una aplicación, y que intenta integrar los SIGs con los procesos de gestión y los modelos de datos alfanuméricos.

La interacción entre el visor SigWeb 2.0 pasa a ser similar a la que muestra el siguiente gráfico:

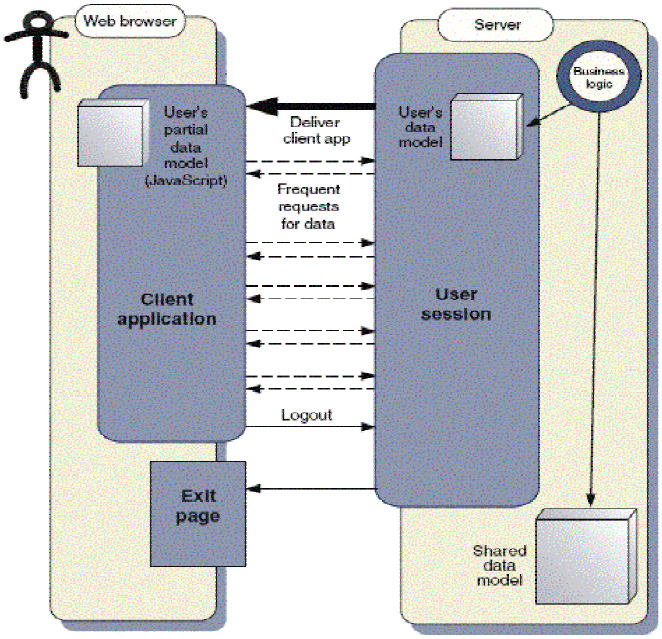

Fig. 1. Interacción entre una página con AJAX y un servidor web.

Este es el esquema típico de una aplicación que hace uso de AJAX, tenemos un cliente que a través de JavaScript se conecta al servidor y este devuelve los resultados de la petición tras procesarla. En nuestro caso, estas peticiones serán mapas, datos alfanuméricos, formularios e incluso otras aplicaciones web, sólo recargándose la parte de la página que así lo requiera.

1.- VISOR SIGWEB 1.0.

Nuestro visor sigweb 1.0, desarrollado con HTML 'tradicional' basada en tablas *y* frames, dividía la página en 3 zonas básicas:

Una barra de herramientas.
Un árbol de capas.
Un área de visualización de mapas.

Adicionalmente la aplicación abría ventanas flotantes para realizar ciertas operaciones.

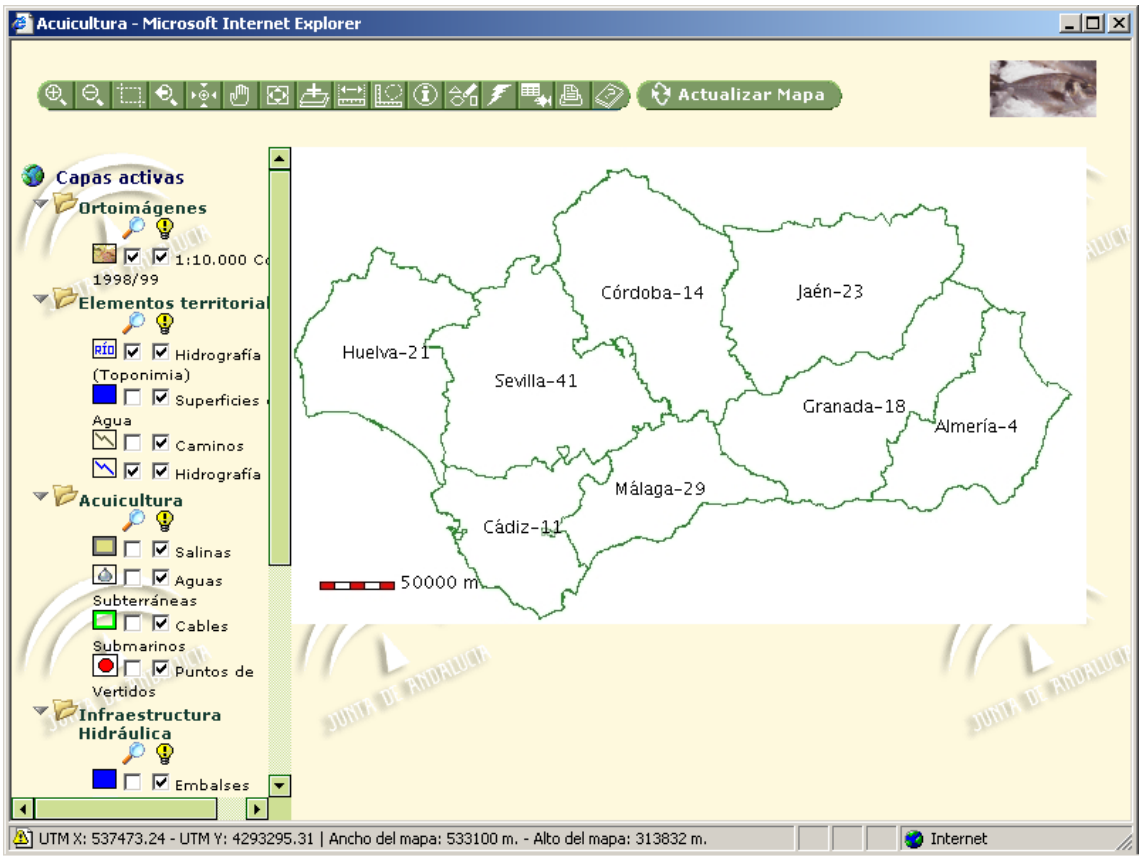

Figura 2. Imagen del visor SigWeb 1.0.

La barra de herramientas es un conjunto de botones que nos permitía realizar operaciones sobre el mapa y la lista de capas.

Podríamos clasificar estas herramientas en 2 grupos:

- Funciones primarias cartográfica (básicamente navegación cartográfica).
- Funciones con componente alfanumérico.

Entre las funciones primarias cartográfica tenemos las típicas de cualquier visor:

- Acercarse. Esta herramienta acerca el mapa al punto seleccionado sobre el mapa, centrándose en él.
- Alejarse. Esta herramienta aleja el mapa, tomando como punto central el punto seleccionado.
- Zoom a la caja. La herramienta zoom a la caja permite definir un rectángulo sobre el mapa, de modo que el mapa se acerque a ese rectángulo.
- Extensión anterior. Permite recuperar el mapa previo a la última acción de desplazamiento o zoom realizada.
- Centrar mapa. Permite centrar el mapa en el punto seleccionado.
- Desplazar. Permite mover el mapa en la dirección en la que se arrastre.
- Ver toda la extensión. Permite al usuario mostrar el mapa con toda su extensión geográfica inicial (en nuestras aplicaciones suele ser la Comunidad Autónoma Andaluza completa).
- Añadir capas. El usuario puede determinar cuáles de entre todas las capas disponibles en la aplicación, desea mantener como activas, esto es, que aparezcan en el menú de la izquierda

Adicionalmente nuestros visores ligeros implementaban para cada aplicación un conjunto de funciones con componente alfanumérico:

- Herramienta de Información. Permite obtener una descripción de la información relevante para cada una de las capas activas y seleccionadas en el momento de la petición sobre un punto concreto del mapa.
- Localizador. Permite seleccionar entre un conjunto de filtros hasta localizar un elemento que será el que se muestre en el mapa.
- Buscador. Permite buscar elementos introduciendo varios parámetros en la búsqueda. El resultado es un conjunto de elementos que cumplen con estos parámetros. Podremos seleccionar cualquiera de ellos y ser visualizado en el mapa.

El conjunto de todas estas funciones, hacían al visor completo, en cuanto a los requisitos habituales de los visores web. Sin embargo, tanto el desarrollo tecnológico, como complejidad de las aplicaciones que desarrollábamos, así como los recursos consumidos por los visores SIG nos hicieron identificar una nueva dirección.

2.- VISOR SIGWEB 2.0.

El visor sigweb 2.0 no elimina las funcionalidades implementadas por el visor 1.0. Su objetivo es otro: crear una plataforma que permita desarrollos más rápidos, robustos, adaptables, extensibles y más usables para el usuario, integrando el visor SIG en los procesos de gestión y los modelos de datos alfanuméricos. ¿Cómo conseguir estos objetivos? Básicamente:

- Integrando el OpenToro (producto software libre), que nos ha permitido realizar de forma sencilla:
  - La publicación web (automatizada) de contenidos alfanuméricos almacenados en bases de datos.
  - La integración de potentes funcionalidades AJAX en el lado del cliente.
- Desarrollando una estructura JavaScript más potente y flexible para la integración de nuevas funciones.
- Mejorando el HTML generado para adaptarnos a la técnica “tableless” haciendo un mayor uso de CSS.
- Desarrollando un diseño más usable.

2.1.- Diseño usable.

La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.

Básicamente ha consistido en:

- Uso de colores menos cansinos y de mayor contraste para mejorar la visualización.
- Feedback al usuario. El visor ligero indica en todo momento qué está haciendo. De esta forma se evita que si existen tiempos largos de cargas el usuario no sepa por qué no ve una nueva imagen.
- Definición de zonas de mayor consistencia semántica de tal forma que por ejemplo los botones que afectan a un elemento se encuentren cerca, se definan visualmente la zona de carga de datos alfanuméricos, etc.

El visor 2.0 en cierto sentido pasa a parecerse más a una aplicación de esta forma podemos conseguir reutilizar la experiencia del usuario en el uso de programas informáticos, es decir en exponer al usuario ante lo conocido y facilitar por tanto el aprendizaje de lo que se le muestra.

2.2.- HTML y Hojas de estilo.

Dentro del esfuerzo realizado para conseguir desarrollos más rápidos, y adaptables, se ha pasado de una arquitectura de HTML 'tradicional' basada en tablas y frames a una arquitectura basada en capas que separe la estructura de un documento de su presentación, es decir hemos usado CSS.

Las **hojas de estilo en cascada** (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

2.3.- Integración OpenToro.

Tal y como decíamos anteriormente en esta comunicación, no sólo de mapas vive un SIG. El componente alfanumérico es también muy importante.

El OpenToro es un Publicador Web de Bases de Datos, es decir una herramienta que nos permite el desarrollo ágil y automatizado de aplicaciones web que acceden a bases de datos. Usar el OpenToro significa simplemente olvidarse de escribir innumerables SQLs y JSPs cada vez que queremos realizar una web con acceso a base de datos. Lo podemos ver es como un generador automático de interfaces web funcionales.

El OpenToro nos permite realizar las operaciones básicas para la publicación de información en la web tomando dicha información desde una base de datos:

- Gestión de listados.
- Visualización de registros.
- Creación de formularios.
- Gestión de tablas de base de datos.

OpenToro implementa un conjunto de funciones AJAX que lo hacen especialmente interesante para la Web 2.0: recargar combos n-dependientes, campos estilo

textsuggest (como en Google suggest), carga embebida (embedded load) de formularios, listados y registros (es decir, no hace falta recargar una página para cargar en ella nuevos listados, formularios y/o registros) .

El funcionamiento del OpenToro se basa en la metainformación, es decir, para poder trabajar con las tablas que tengamos en la base de datos tenemos que describirlas en ficheros xml del OpenToro (llamadas **metavistas**) de forma que este tenga conocimiento tanto de su estructura como de algunos datos (relaciones con otras tablas, etc).

Podemos ver la arquitectura básica del OpenToro en la siguiente imagen:

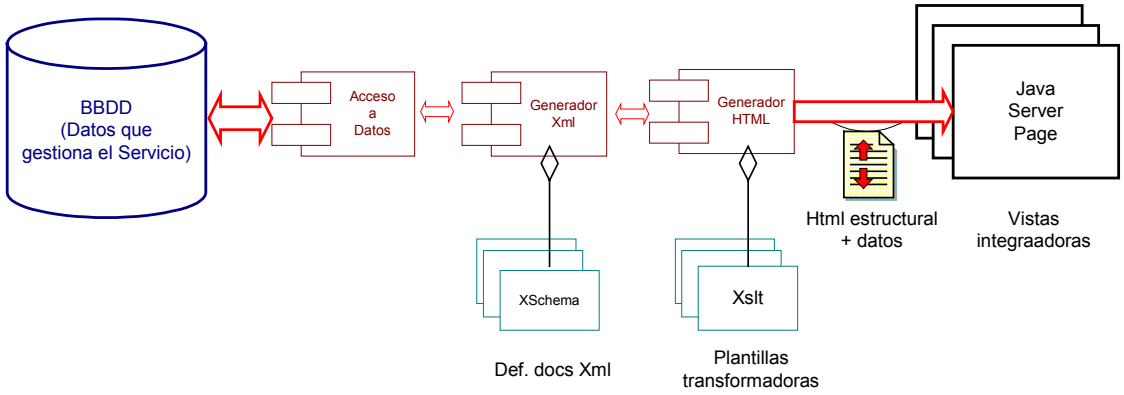

Fig. 3. Diagrama básico de subcomponentes del OpenToro.

Arquitectónicamente podríamos ver la integración del OpenToro de la siguiente forma:

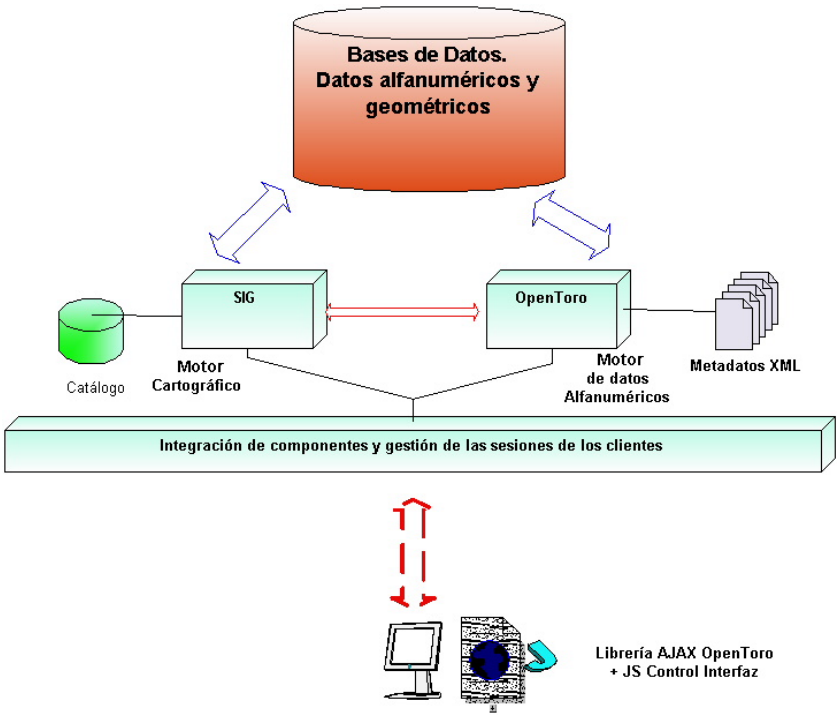

Fig. 4. Modelo integración OpenToro-SIG.

Un ejemplo de pantalla automáticamente generada por el OpenToro a partir de una metavista es el listado (con sus correspondientes filtros) que se muestra a continuación:

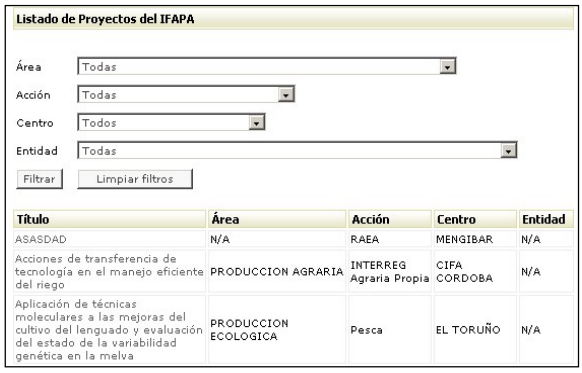

Fig. 5. Ejemplo de listado generado por OpenToro.

La integración del OpenToro tiene un fuerte impacto en el visor web. En los siguientes puntos vamos a describir los más importantes.

2.3.1.- GetFeatureInfo.

Nuestros visores web se apoyan en un servidor de mapas, éste no sólo nos proporciona los mapas sino que también nos facilita la información asociada a un punto (el 'típico' GetFeatureInfo del estándar WMS).

Hasta ahora, esta información nos era devuelta en formato HTML pero una vez integrado OpenToro en nuestra arquitectura hemos modificado esta funcionalidad del servidor de mapas para poder aprovechar la potencia de OpenToro en este sentido.

Cada feature obtenida por el servidor de mapas se corresponde con una layer que a su vez se corresponde con una metavista y usando la clave de la feature podemos pedirle al OpenToro que nos genere el HTML formateado del registro.

OpenToro nos permite indicar los campos que se muestran en la información, hacerla 'human-readable', y modificando la metavista correspondiente, además, dado el uso que realiza OpenToro de plantillas xslt para la generación del html disponemos de una gran independencia entre la información y la visualización permitiéndonos realizar cambios visuales sin más que modificar las metavistas, las plantillas xslt o las css adecuadas.

Fig. 6. Ejemplo de GetFeatureInfo usando OpenToro.

2.3.2.- Localizadores.

Otra de las funcionalidades que se han beneficiado de la integración de OpenToro y sus funciones AJAX han sido los localizadores.

Hasta ahora, cuando usábamos los localizadores, trabajábamos con ventanas flotantes que era preciso refrescar para conseguir la carga de combos relacionados.

Con la integración de OpenToro y sus funciones AJAX obtenemos dichas recargas de forma automática sin necesidad de refrescar la ventana. Además, se ha procedido también a cargar el formulario en la misma pantalla del visor de forma que tenemos una visión mucho más integrada.

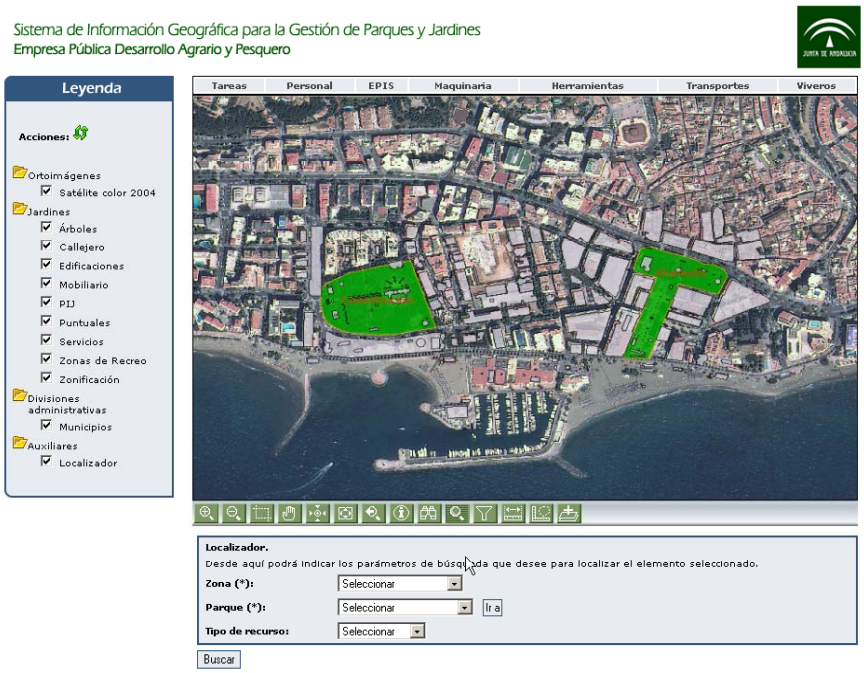

Fig. 7. Ejemplo de Localizador generado con OpenToro

OpenToro nos facilita el desarrollo de esta funcionalidad pues una vez tenemos definido los filtros necesarios para llegar al elemento que queremos localizar no tenemos más que crear las metavistas necesarias, de esta forma, OpenToro nos genera el formulario en base al cual seleccionaremos el registro que queremos localizar en el mapa.

Con el registro ya seleccionado usaremos las librerías AJAX desarrolladas a tal efecto para obtener el mapa donde podremos ver localizado el registro.Con estos cambios la generación de localizadores se ha automatizado enormemente disminuyendo la aparición de errores y los tiempos de desarrollo.

2.3.3.- Buscadores.

Los buscadores también se han visto afectados por las mejoras introducidas en los localizadores pues su proceso se realiza de forma similar aunque algo más compleja.

Al igual que en el caso anterior también nos apoyamos en OpenToro y sus funciones AJAX por lo que una vez creadas las metavistas necesarias para crear el formulario de búsqueda tenemos de forma automática la recarga de combos y la búsqueda obteniendo así un listado de registros que se carga en la propia pantalla.

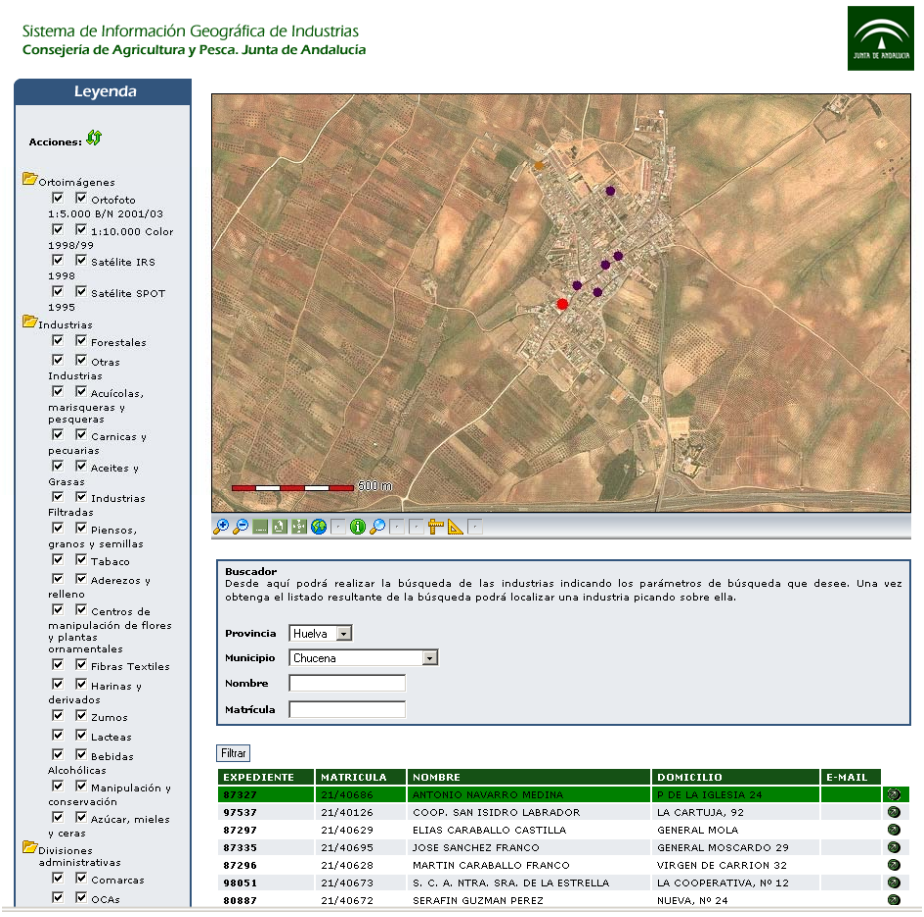

Fig. 8.- Ejemplo de resultado de buscador con OpenToro.

Como observamos en la imagen superior una vez tenemos el listado podemos localizar el registro que deseemos picando sobre él. Igualmente, podemos obtener la información asociada al registro usando de nuevo OpenToro y sus funciones AJAX que nos mostrarán el HTML con la información asociada al registro embebida en la propia página.

2.3.4.- Gestión de contenidos

La integración de OpenToro ha supuesto un gran paso adelante en la gestión de la información **alfanumérica** en nuestros visores. Apoyándonos en OpenToro y sus funcionalidades hemos automatizado la gestión de registros.

Describiendo las metavistas necesarias disponemos de la posibilidad de dar de alta, modificar o eliminar registros, generándose de forma automática los formularios necesarios.

Como podemos ver OpenToro nos ha facilitado el acercamiento a las aplicaciones de gestión dentro en todo momento de nuestro visor sig.

2.3.5.- Navegación por contenidos alfanuméricos.

Otro aspecto importante a señalar es la navegación entre los contenidos alfanuméricos.

OpenToro nos permite mapear en las metavistas las relaciones existentes entre las tablas descritas. Con este mapeo podemos generar los links necesarios que nos permitan navegar entre los contenidos. Así, por ejemplo, podemos navegar por los maestro – detalle sin necesidad de un gran esfuerzo por nuestra parte en el desarrollo y sin necesidad de recargar páginas.

2.4.- Control de la interfaz en el cliente.

La implementación de las funcionalidades en el visor requieren de una estructura en el lado del cliente para gestionar la interacción con el usuario y el servidor.

La idea que buscamos era tener una estructura sólida y centralizada que nos permitiera añadir fácilmente nuevas herramientas, consiguiendo de esta forma gestionar todas las peticiones desde un único punto. Para ello decidimos apoyarnos en objetos JavaScript.

Se ha definido un gestor central, Manager, que actúa como punto de entrada de la petición del usuario capturando los eventos correspondientes y lanzando o parando la herramienta adecuada. Simplificando, el Manager es un objeto JavaScript encargado de gestionar las herramientas de las que dispone el cliente además de pasar el flujo de control a la herramienta seleccionada o activa.

Esto simplifica los desarrollos, si se quiere añadir una nueva funcionalidad sólo se ha de crear el objeto encargado de gestionarla y dar de alta dicho objeto en el Manager. La estructura diseñada nos evita tener que capturar eventos e integrar la herramienta en el visor pues el Manager ya se encarga de estos menesteres.

3.- EL VISOR COMO PARTE DE LA PLATAFORMA SIG.

Nuestro visor se encuentra englobado dentro de una plataforma SIG. Esta plataforma, que denominamos AndaluSIG, cuenta con diversos elementos entre los que podemos distinguir los siguientes:

1. El catálogo
2. El Servidor de Mapas.
3. GisCore. Componente propio con funcionalidades SIG.
4. Visores pesados
5. ACA 1.1

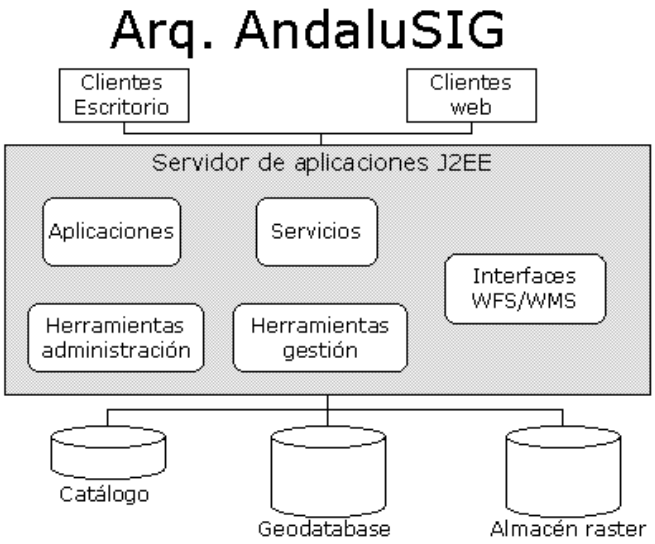

Fig. 9. Arquitectura AndaluSIG.

El catálogo es la herramienta dedicada a describir las capas existentes y su comportamiento en cuanto a reglas, estilos, etc... todo lo necesario para la correcta configuración de un SIG. Está formado por un conjunto de tablas donde mapeamos la información necesaria y es usado tanto por nuestros visores ligeros como por los pesados.

Contamos también con visores pesados basados en Jump y Udig que nos permiten desarrollar aplicaciones de escritorio.

Por último señalar que contamos con una herramienta de gestión para el catálogo, ACA 1.1 (Administración del Catálogo AndaluSIG), gracias a esta herramienta podemos configurar el catálogo de forma que se adapte a nuestras necesidades.

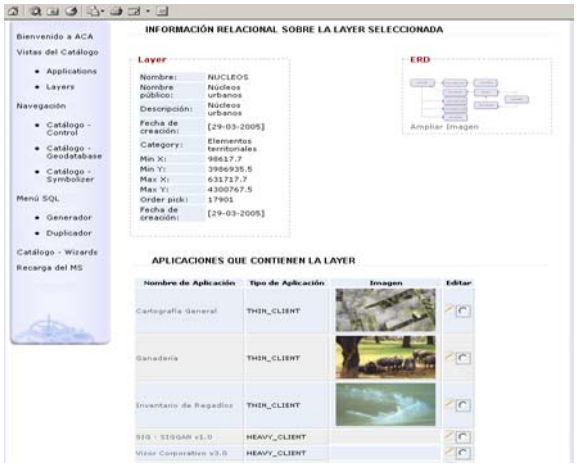

Fig. 10. Pantalla del Administrador del Catálogo.

4.- CONCLUSIONES.

Para concluir indicar que nuestro nuevo visor 2.0 es un paso importante en el acercamiento de los visores SIG a las aplicaciones de gestión, todo ello facilitado por la integración de AJAX y OpenToro. Con esta integración hemos creado una plataforma que nos permite desarrollos más rápidos, robustos, adaptables, extensibles y más usables para el usuario.

De cara al futuro nuestra intención es aumentar el número de herramientas disponibles implementando funcionalidades típicas de los visores pesados como la obtención de mapas temáticos y mapas filtrados. Así esperamos dar los pasos necesarios para que el visor web se asemeje lo máximo posible en cuanto a funcionalidad a los visores pesados.

AGRADECIMIENTOS

Todo proyecto es el resultado de una organización, no sólo de las personas que han realizado la última versión de una herramienta. Por lo tanto agradecer su trabajo a los compañeros del Área de Desarrollo Portales-GIS: Abel Moreno Suárez, Antonio Redondo Lora, Sergio Porras González, Francisco Morales Mazo, Marcos Boza Domínguez, Oscar David Fernández Martínez, José Juan Corpas Martos, Sonia Duarte Rodríguez, Felipe Arancón, Miguel Ángel Germán Serrano, Juan Manuel Sánchez Martagón, Francisco Bermúdez Castro, Ángel Nevado Pérez, Ignacio Gámez Ramírez y Javier Cámara Pérez.

Igualmente agradecer a la dirección de la Empresa Pública Desarrollo Agrario y Pesquero su apoyo en todos estos desarrollos tecnológicos, y finalmente mostrar nuestro agradecimiento a otras personas que trabajaron en el desarrollo de la plataforma AndaluSIG.