Atlas Comercial Comunidad Valenciana

Autor del proyecto:

Arturo Argilés Casasús

(ararca@inf.upv.es)

Director:

Moisés Pastor Gadea

Septiembre de 2010

A Moisés Pastor, por su paciencia y su aporte
para hacer realidad este proyecto.

A Lena, por estar siempre a mi lado,
éstas palabras son también suyas.

A mis padres y a mi hermana
por tantos años de apoyo.

A IVER Tecnologías,
especialmente a Adrián García.

Indice

1. Introducción

El estudio de la cartografía por el hombre se remonta a tiempos pretéritos. Entremezclados el conocimiento y la necesidad, bien por ampliar rutas comerciales, bien por marcar fronteras, tantas veces modificadas a lo largo de la historia a merced de las actividades bélicas, dio pie al conocimiento de la cartografía.

El avance de las tecnologías no ha sido ajeno a este campo, suponiendo un fuerte impacto en el manejo de información geográfica, ya que se han obtenido herramientas que han permitido crear aplicaciones relacionadas con la cartografía digital de una forma eficiente, llegando incluso a estar disponibles también en la web. Desde las organizaciones encargadas de los Sistemas de Información Geográfica (SIG) se ha buscado definir unos estándares para lograr una uniformidad que haga interoperables los sistemas desarrollados así como los que están por desarrollar. Uno de los estándares más recientes propuesto por el *Open Geospatial Consortium* es el *Web Processing Service* (WPS), que ha sido diseñado para estandarizar la forma en que los procesos SIG se ofrecen a través de Internet. Cabe tener en cuenta que, si bien la cartografía digital ya resulta un proyecto relativamente reciente debido a la dificultad que implica, trasladar esta materia al desarrollo web resulta aún más complejo, ya que el tratamiento de imágenes y la realización de los cálculos resulta mucho más lenta que si se realiza en una aplicación de escritorio.

Como ejemplo de eficiencia acerca de este campo podemos nombrar algo tan familiar como son los mapas de carreteras. Si bien no hace muchos años la lectura de estos mapas en papel resultaba algo tediosa, hoy es algo muy común y sencillo realizar consultas sobre mapas, así como las diferentes alternativas relacionadas con el cálculo de rutas entre dos puntos y otras actividades que ofrece la cartografía digital a través de un ordenador conectado a la red, basta con indicar los puntos origen y destino, así como señalar las diferentes alternativas (permite escoger la ruta más larga, la más corta, incluso la más segura) de forma que el usuario pueda estudiar sobre el propio mapa las rutas calculadas.

Cada vez son más las aplicaciones que facilitan al usuario las consultas que se pueden realizar sobre un mapa, habiendo provocado esta competencia un amplio abanico de posibilidades que hagan más atractivo al cliente el manejo de dicha aplicación frente a sus competidoras. Resulta cada día más habitual encontrar aplicaciones cartográficas tan diversas como:

- **Mapas de carreteras**, son los más empleados para realizar consultas como el cálculo de rutas entre un punto origen y otro punto destino. Para este fin, ofrece al usuario una serie de alternativas, como si se desea el cálculo de la ruta más corta, la más segura o la más rápida. Entre sus alternativas se puede indicar si se desea que el cálculo tenga en cuenta o no autopistas de peaje. Cuentan con varias alternativas para hacer más atractiva frente a sus competidoras, como son la localización de radares, información meteorológica… Son algunos ejemplos de las más utilizadas:
  - La guía Michelín (http://www.viamichelin.es)
  - La guía de Repsol-Campsa (http://www.guiarepsol.com)
  - La guía Cepsa (http://www.buenviajecepsa.com)

- La **referencia catastral** permite al usuario localizar los bienes inmuebles mediante la referencia catastral. El gobierno ofrece por medio del ministerio de economía y hacienda realizar consultas acerca de la dirección general del catastro. Se puede encontrar más información en las direcciones (http://www.catastro.meh.es
  y http://www.sedecatastro.gob.es)

- **Mapas genéricos**. Estos últimos son cada día más útiles, ya que permiten no sólo mostrar diferentes vistas de un mapa o la ruta entre dos puntos, sino también una vista satélite, que permite localizar ampliando el zoom sobre un punto del mapa hasta el extremo de mostrar imágenes de una calle como las puede ver un viandante. Son algunos ejemplos de estos últimos los que ofrecen algunos portales web como son:
  - GoogleMaps de Google (http://maps.google.es)
  - YahooMaps de Yahoo (http://maps.yahoo.com)
  - BingMaps de *Microsoft* (http://www.bing.com/maps).

Entre sus ventajas cabe destacar del movimiento que ha surgido de un tiempo a esta parte, por el que muchas páginas y aplicaciones web hacen uso de scripts que les permita mostrar estos mapas en una pequeña ventana tal que muestre la ubicación del objeto que se ofrece en la web indicada.

Ha sido tal la aceptación de este tipo de visores web aplicados a la cartografía por los usuarios de la red que ya se han adaptado, incluso, a las tecnologías móviles, así, no resulta ya extraño poder disponer de móviles con GPS y aplicaciones de este tipo en dispositivos móviles con conexión de datos, permitiendo incluso geo-localizar a otros dispositivos móviles (como sucede con la aplicación *google latitude* para teléfonos móviles *android*), así como trazar rutas en función de la distancia, tipo de vehículo (a pie, en coche, medio de transporte público) e incluso en función de la situación del tráfico.

- Existen otros visores web más específicos, entre ellos se encuentran visores ofrecidos por algunos ministerios, como el de hacienda mencionado anteriormente para el catastro, o el ministerio de fomento, que ofrece varios visores en función del campo solicitado. Así, por ejemplo, se puede estudiar un visor de servicios geodésicos en la siguiente URL (http://www.ign.es/ign/es/IGN/visor_geodesico.jsp).

  Un ejemplo de aplicación para tecnología móvil que, como indicábamos anteriormente, y de un tiempo a esta parte, ha aumentado considerablemente su oferta de mercado, es la que ha creado la entidad bancaria BBVA para dispositivos móviles, tal que una de sus ofertas es la de geo-localizar la posición del dispositivo móvil, así como la de los cajeros, propios y de otras entidades, indicando su dirección y distancia, tal que si se desea calcular la ruta a alguno de los indicados basta con seleccionarlo. Como resultado la aplicación muestra un mapa de Google en vista satélite y la ruta a seguir para llegar a la ubicación especificada señalizada sobre el mapa.

Existen muchos más tipos de visores web, también muy útiles aunque quizá algo más desconocidos pese a la información que ofrecen. Un ejemplo de un visor web de este tipo es el que ofrece la URL (http://firefly.geog.umd.edu/firemap) que informa al usuario de la localización sobre un mapamundi de los diferentes focos incendiarios que hay a lo largo del planeta, está además muy actualizado. Resulta realmente acongojador, por cierto.

- **Atlas Comercial de la Comunidad Valenciana** se trata de un visor web que encaja en el cuarto grupo de los mencionados arriba, esto es, un visor web con una funcionalidad muy específica. Se utiliza para obtener información de los diferentes municipios de la comunidad, así como la información de los establecimientos, comercios de innovación, concentraciones comerciales, antenas locales y cámaras de comercio que se extienden a lo largo de la comunidad, así como la información referente a estos (si la hay).

Existen otros visores web parecidos en cuanto a funcionalidad y diseño como el que se puede encontrar en la URL (http://sigeo.iver.es) que se encarga de la localización de explotaciones, tanto mineras como de manantiales y yacimientos mineros, entre otras, en un mapa que se extiende a lo largo de la comunidad autónoma de Extremadura. Este último visor comparte, en algunos puntos, funcionalidad con el visor que abarca este proyecto, herramientas por otro lado definidas, como veremos más adelante, por el propio *framework* sobre el que se ha desarrollado el visor. Altas Comercial de la Comunidad Valenciana forma parte de un proyecto encargado por la Cámara de Comercio de la Comunidad Valenciana a la empresa especializada en aplicaciones SIG como es Iver Tecnologías de la Información, empresa con la que colaboré trabajando precisamente en esta aplicación.

2. Especificaciones de la Interfaz

Atlas Comercial de la Comunidad Valenciana se trata, como se ha especificado, de un visor web que ofrezca al usuario una serie de herramientas para interactuar con el mapa y la información que este ofrece.

En primer lugar, deben definirse unos **márgenes** específicos para el visor como se muestran en la figura 1. Debe haber un panel herramientas (toolBarPanel) donde aparezcan los botones propios de la navegación del visor. Asimismo debe contener una barra de estado en la parte baja del navegador (footerPanel) donde se informará durante el procesamiento de los cálculos de la aplicación, de la escala utilizada, de las coordenadas donde se encuentra el ratón y la proyección empleada. Ha de crearse además un panel que permita la selección de los mapas, así como de las capas en el margen derecho del visor y la leyenda del mapa en función del que se haya seleccionado (eastPanel). Por último debe existir un panel donde se muestre el mapa (containerMapPanel).

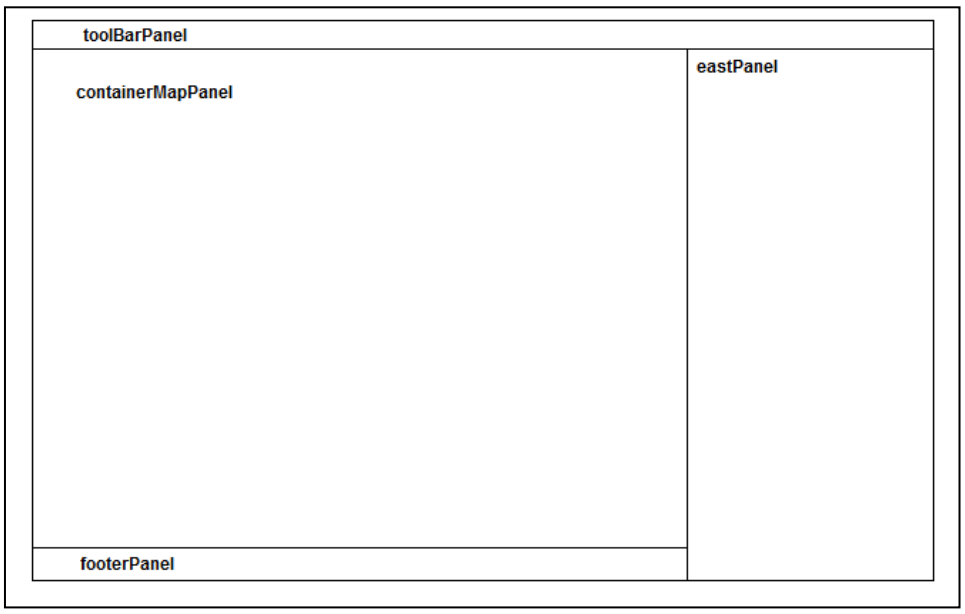

Figura 1. Márgenes del visor

Como se ha indicado anteriormente, sobre la barra de herramientas debe haber una serie de botones propios de un navegador como son:

- **Selección de idioma**, este puede ser castellano o valenciano.
- **Recarga de mapa**. De modo que, en caso de pulsarlo, vuelva a mostrar el mapa con las coordenadas definidas por defecto para su arranque. Realizando un zoom a la totalidad de la cartografía.
- **Ampliar zoom** y **reducir zoom**, de modo que en caso de ampliar ajuste el zoom, acercando el mapa, y lo aleje en caso de que se reduzca el zoom.
- **Arrastrar imagen**, que permite arrastrar el mapa mediante las acciones de pulsar y arrastrar.
- **Navegación de vistas**, tal que el usuario pueda moverse a extensiones de zoom anteriores y posteriores, o lo que es lo mismo, a vistas de cartografías anteriores y posteriores.
- **Medición de distancias** sobre la imagen del mapa.
- **Medición de áreas** sobre la imagen del mapa.
- **Limpiar el mapa**, tal que borre los elementos gráficos de la vista.
- **Recarga del mapa** debe actualizar las capas de la vista.

Además de estas funcionalidades comunes a un visor web cartográfico, existen una serie de funcionalidades específicas de la aplicación que se especifican a continuación:

- **Obtener información** de **municipios**, para ello basta con pinchar con el ratón sobre el municipio del que se desea obtener la información (se debe dotar a la selección del ratón con un pequeño margen de área seleccionable, así, en el caso de que el puntero del ratón se encuentre en los límites de varios municipios, debe mostrarse en una ventana un listado con los municipios que comprenden el área de selección del ratón, de forma que se escoja el municipio del que se desea obtener información con una nueva selección del ratón). El resultado se muestra en una nueva ventana y debe contener información relacionada con dicho municipio (nombre, extensión, una tabla demográfica…).

  También se podrá obtener información de un municipio atendiendo a su **extensión geográfica** sobre el mapa, de modo que se seleccione un municipio de una provincia (previamente indicada mediante un formulario) y

el mapa aplicará un zoom sobre el municipio del que se desea conocer su extensión.

- Sobre el mapa deben indicarse mediante unos iconos los diferentes establecimientos, concentraciones comerciales, comercios de innovación, antenas locales y cámaras de comercio que se extienden a lo largo de la comunidad, de forma que para obtener la información de cualquiera de estos baste con indicar que queremos obtener información sobre ellos seleccionándolo previamente en el *eastPanel* y arrastrando posteriormente el ratón de forma que se indique un área de búsqueda. Para la búsqueda de establecimientos y concentraciones comerciales se debe habilitar además un botón de búsqueda específico para cada uno de ellos en el *toolBarPanel*.
- Un usuario con rol de administrador podrá gestionar tanto el servicio de mapas a mostrar como las capas que pueda contener cada mapa. Podrá además indicar si una capa es consultable o no (si podemos realizar búsquedas sobre dicha capa) así como el icono y el texto que acompañe a cada capa para formar la leyenda final del mapa. El administrador podrá a su vez crear nuevos usuarios y asignarles un rol para la gestión de dicha aplicación.
- Entre las consultas que podemos realizar al mapa está la de obtención de **información WMS** de los campos de la capa previamente seleccionada. El resultado se mostrará en una nueva ventana, debe contener una serie de campos propios de la información que se puede obtener de dicho servicio.
- Debe habilitarse un **servicio de impresión**, de modo que permita exportar a imagen, obteniendo una imagen del mapa. También se podrá exportar a un fichero en formato pdf, tal que se muestre el mapa y la leyenda perteneciente a dicho mapa.
- **Servicio WMS** que permita añadir nuevas capas en el *eastPanel* a partir de una URL de un servidor WMS.
- **Centrado en coordenadas** que permita centrar el mapa en las coordenadas introducidas.
- **Street View** que permita obtener una vista panorámica en una nueva ventana del punto seleccionado con el ratón (siempre que exista dicha

vista). Es una vista ofrecida por la funcionalidad de Street View de Google Maps.

Todas estas funcionalidades deben ofrecerse en una interfaz sencilla de modo que resulte de fácil manejo y en muy poco tiempo un usuario que desconozca su funcionamiento pueda familiarizarse con ella. Con este objetivo, el mapa que se muestra al arrancar la aplicación recuerda al de otros visores ya que se ayuda de los mapas de Google y Yahoo para tal fin (aunque podrían añadirse algunos mapas nuevos, como pueden ser de referencias catastrales, mapas de software libre, de carreteras como ofrece la página del ministerio de fomento). Los márgenes del visor cuando este arranca limitan con el área que abarca la Comunidad Valenciana, pudiendo hacer zoom sobre el mismo, como ocurre con el resto de visores mencionados hasta el momento.

Como ya se ha comentado brevemente, existen una serie de objetos de búsqueda sobre los que deseamos obtener información y que explicaré a continuación a modo de glosario para que el lector se familiarice con ellos y entienda su significado de aquí en adelante, ya que hablaremos de ellos a menudo. Cabe distinguir por tanto, los siguientes objetos de interés:

- **Municipios**, indica aquellos términos jurisdiccionales regidos por un ayuntamiento. La localización del municipio se podrá realizar de dos maneras, como se ha indicado anteriormente. Bien por selección del ratón sobre el mapa, bien mediante un formulario, atendiendo a la provincia y a los municipios pertenecientes a la provincia seleccionada.
- **Establecimientos**, esto es, los comercios que se extienden por la provincia de Castellón. Son ejemplos de establecimiento las panaderías, bolserías, gasolineras, bazares, bares…
- **Comercios de Innovación**, se refiere a los comercios que se extienden en las regiones de Valencia y Alicante.
- **Concentraciones Comerciales**, se entiende así a los hipermercados, centros comerciales, grandes almacenes y mercados municipales que se extienden a lo largo de la provincia de Castellón.
- **Cámaras de Comercio** de las provincias de Castellón, Valencia y Alicante

- **Antenas Locales**, creadas por la Cámara de Comercio como servicios de ayuda para PYMES, también extendidas a lo largo de Castellón, Valencia y Alicante.
- **Servicio WMS** (*Web Map Service*), se trata de un estándar internacional que permite la generación de mapas de datos espaciales a partir de información geográfica en un archivo de imagen digital.
- **Servicios WFS** (*Web Feature Service*) es otro estándar que permite interactuar con los mapas generados por el servicio WMS mediante información vectorial.

3. Solución del problema

Antes de profundizar en la solución al problema, es conveniente comentar brevemente qué herramientas se han utilizado para el desarrollo de este proyecto. En primer lugar conviene señalar que todo el material empleado en el desarrollo de esta aplicación es gratuito ya que se trata de herramientas *Open Source*.

El desarrollo del proyecto se ha realizado sobre el entorno de desarrollo integrado **Eclipse** (http://www.eclipse.org). Dado que se trata de un visor web, el desarrollo del visor se ha realizado fundamentalmente en javascript, haciéndose uso de dos *APIs* de JavaScript como son **OpenLayers** (http://openlayers.org), adecuada para el manejo de mapas en navegadores web y **ExtJS** (http://www.sencha.com), que facilita el desarrollo de aplicaciones interactivas usando AJAX y DHTML. El código que se ha realizado, pese a estar desarrollado en un lenguaje débilmente tipado como es JavaScript, se ha realizado lo más orientado a objetos posible haciendo uso de estas librerías.

Dada la funcionalidad del visor, que comparte además algunas funciones con otros visores como el SIGEO ya mencionado, se ha desarrollado cada funcionalidad en una extensión que se añada al núcleo de la aplicación, donde se cargan las librerías del manejo de mapas y del desarrollo de interfaces. Como resultado queda una aplicación modular y ampliable sin apenas realizar modificaciones, sólo hay que indicar al núcleo que se ha añadido la nueva extensión.

Para la obtención gráfica de los mapas es necesario realizar consultas a **MapServer** o a **Geoserver**. La necesidad de utilizar ambos radica en que obtendremos las capas WMS de MapServer mientras que las capas WFS nos las proporcionará Geoserver.

La impresión de plantillas a formato pdf compuesta por el mapa mostrado por el visor con su respectiva leyenda, se ha realizado en Java, mientras que los detalles del formato de impresión se hicieron mediante **JasperReports**.

La herramienta del **servicio de administración** desde el que se puede generar nuevos mapas, así como nuevas capas, o bien modificar las capas y los mapas ya existentes (el nombre, la leyenda o el icono entre otras) fue creada en **Dojo**, que es un *framework* que permite crear aplicaciones web con Ajax. Para este proyecto sólo ha sido necesario modificar el nombre de los diferentes objetos así como realizar algunos cambios sobre las hojas de estilo (css) ya que se ha reutilizado la misma herramienta de administración que IVER Tecnologías creó para realizar la administración del SIGEO, reutilizando así el código.

En la siguiente figura se observa la solución empleada al problema de la obtención de las capas WMS y WFS mencionado anteriormente. Aplicado al proyecto, esta solución atiende al problema de localizar en el mapa los diferentes establecimientos, concentraciones comerciales, cámaras de comercio, antenas locales y comercios de innovación almacenados en la base de datos y que deben mostrarse con su icono correspondiente en el mapa.

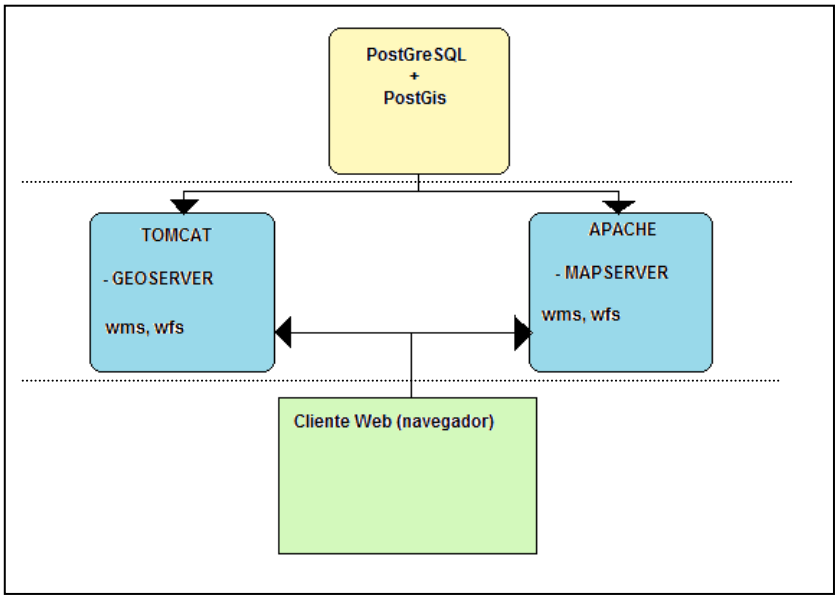

Figura 2. Obtención de datos y posicionamiento en el mapa.

Como indica la figura, es necesaria una base de datos donde almacenar la información de los diferentes objetos de búsqueda. Como se necesita además una referencia geográfica no basta con una base de datos sencilla. Es necesario crear una base de datos espacial. Para ello se hace uso de PostGIS, que es una extensión espacial en código abierto de PostgreSQL, de modo que

dota a las bases de datos generadas con este motor de un soporte para objetos geográficos.

Una explicación poco técnica aunque útil para indicar la diferencia entre una capa WMS y una capa WFS sería indicar que la capa WMS es la capa que muestra la imagen del mapa, y sobre ella se indican mediante capas WFS los diferentes objetos sobre los que trabajar (establecimientos, comercios, etc), como muestra la figura 2. Uno de los principales problemas que se presenta cuando se trabaja en cartografía digital se da cuando se trabaja con capas que presentan diferentes proyecciones, como sucede al trabajar conjuntamente con proyecciones específicas como la EPSG:900913 de GoogleMaps y proyecciones EPSG:4326 de WMS para OpenLayers. Es necesario realizar una re-proyección para poder solapar las capas.

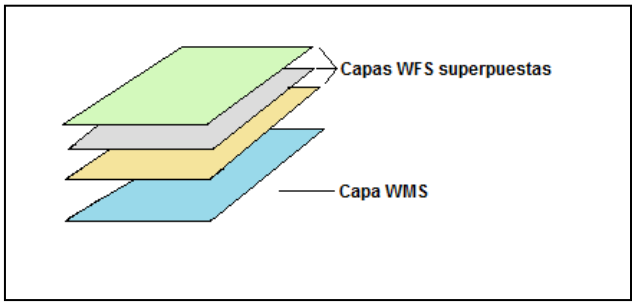

Figura 3. Sobre la capa WMS se superponen las diferentes capas WFS, para ello es necesario que todas las capas estén en la misma proyección o, en su defecto, re proyectadas.

Para utilizar los mapas de Google y Yahoo como base cartográfica es necesario definirlo como proyecciones Spherical Mercator, debido a que utilizan proyecciones de Mercator, es decir, consideran la tierra como una esfera en lugar de hacerlo como si fuera un elipsoide [1].

Debido a que utilizan proyecciones diferentes (GoogleMaps y Yahoo utilizan proyecciones EPSG:900913) es necesario, además, realizar una re-proyección[2] sobre estas capas. Una vez re-proyectadas, estas capas deben añadirse al mapa.

[1] Véase un ejemplo de cómo ha sido generado en el Anexo C
[2] Véase Anexo D

3.1 Extensión Atlas

Una vez definidas las capas WMS que debe cargar el mapa por defecto el proyecto ATLAS añade, además, algunas capas WFS nuevas, que son las correspondientes a los diferentes objetos de búsqueda (establecimientos, comercios de innovación…) de los que ya se ha hablado. Cada objeto de los mencionados necesita su propia capa WFS[3] (con proyección EPSG:23030), y se superpondrá, como indicaba anteriormente, sobre la capa WMS.

Uno de los requisitos que se especificaron en el apartado anterior era el de definir unos márgenes para el visor. Entre ellos debía haber un espacio reservado donde ubicar los botones propios de navegación del visor, además de las herramientas propias del proyecto ATLAS. Para resolver estas especificaciones se hizo uso de la librería ExtJS[4], de la que ya se ha hablado, por su amplio abanico de posibilidades que facilitan el desarrollo de aplicaciones interactivas para la web. Basta con extender de algún objeto propio de la librería, dándole valor a los atributos y definiendo las funciones que deben ejecutarse cuando se produzca el evento que las dispare.

En el Anexo D se muestra cómo se ha realizado la *toolBar*. Para ello, se ha definido en Layout.js (fichero que alberga todos los objetos que comportan el layout definido en las especificaciones del apartado anterior) la componente toolBar, que hará referencia al panel toolBar.js y el panel toolBarPanel.

Como se observa sobre el código del anexo, al inicializar los componentes se hace referencia a la componente `toolBar` propia de la clase Layout, definidia como un ToolBar. El código de ToolBar.js se puede consultar en el Anexo D.

[3] Véase Anexo D

[4] Cuando realicé el proyecto la librería se llamaba ExtJS y la versión empleada en este proyecto es la 3.0. Actualmente ya no se llama así, sino Sencha, e incluye otros paquetes además del ya mencionado, que va por su versión 3.2.1

El código de Toolbar.js, mostrado en el Anexo D, sigue el esquema de cómo se ha estructurado la definición de clases con ExtJS. Así, la clase ToolBar que se encuentra en Layout/Bar/ToolBar.js extiende de Ext.Toolbar. Tras definir los atributos propios de dicha clase se definen los controles propios del visor, ya comentados arriba, indicando el texto que aparecerá en caso de pasar el ratón sobre dichos iconos y el icono con que aparecerá. Por último se han definido los eventos de los controles previamente definidos.

Todos estos controles, que hacen referencia al *Layout* del visor web, son comunes a otros visores web, es por ello que, con el objetivo de optimizar el código, se ha optado por incluirlos en del núcleo de la aplicación al que hace referencia el index.html del visor. Dicho núcleo está compuesto, además, por las librerías *javascript* de OpenLayers y ExtJS, de las que ya se ha hablado. Así, al arrancar la aplicación se cargan los scripts que realizan la carga de mapas (GoogleMaps y YahooMaps), el núcleo de la aplicación y se le añaden las extensiones[5].

Como se ha indicado, la aplicación es **modular**, de forma que si se quiere realizar una ampliación sobre el visor web creado bastará con crear una nueva extensión y añadirla al núcleo de la aplicación para la ejecución. Se consigue además una buena utilización del código, ya que se pueden reutilizar las extensiones ya creadas en otros visores web (como ya se ha comentado la extensión de administración es una reutilización de código, realizándose tan solo ligeras modificaciones sobre la extensión).

A continuación voy se va a explicar la extensión más importante del visor, y que ocupa gran parte de la importancia del proyecto, que no es otra que la extensión propia del ATLAS y que, como se ha comentado en el punto anterior, contiene la mayoría de las especificaciones propias del visor web.

[5] Véase Anexo D

3.1.1 ExtSearchAtlas.Locale.js:

Aunque no pretende ser este documento un tutorial de programación de la librería ExtJS, se intentó seguir un estilo de programación organizada a criterio del desarrollador, estructurando los nuevos ficheros en diferentes directorios atendiendo a su funcionalidad. Así, desde el fichero principal (ExtSearchAtlas.js) se invoca a los botones de búsqueda de establecimientos, información y agrupaciones comerciales en el momento de la creación de dicha clase. Asimismo se indica qué ventanas deben mostrarse cuando se produzca el evento que las dispare. De este modo, las ventanas fueron almacenadas en un directorio *Window*, mientras que los paneles contenidos dentro de las mismas se definieron en el directorio *Panel*. La funcionalidad Información es un control, es por ello que se creó InfoBBoxControl.js y se almacena en el directorio *Control*. El contenido de esta extensión se muestra por completo en un anexo al final de la memoria.

El primer archivo a comentar es Locale.js. Dado que el usuario puede escoger el idioma de la aplicación (puede estar en español o en valenciano) ha de haber un fichero diferente para cada idioma, estos son lang-es.js para español, y lang-va.js para valenciano. Locale.js añade el fichero con el lenguaje seleccionado.

```
Locale.addScript("ideol/ide-extensions/ExtSearchAtlas/lib/Locale/lang-"+Locale.getLang()+".js");
```

3.1.2 ExtSearchAtlas.Config.js:

Ya se ha comentado anteriormente la relación entre las capas WFS superpuestas sobre la capa WMS, este fichero contiene las diferentes capas WFS con proyección EPSG:23030 que contiene la aplicación. El código de cómo se introducen estas capas se ha mostrado anteriormente cuando se ha explicado el fichero Config.js.

3.1.3 ExtSearchAtlas.js:

Se trata del fichero principal de la extensión ATLAS. Sobre él se definen los diferentes botones, controles y ventanas que comportan la funcionalidad descrita en las especificaciones de la aplicación. El evento createComponents

define los diferentes botones y las funciones que se ejecutarán cuando se produzca un evento. El evento onCreate añade los componentes definidos anteriormente al layout de la aplicación. Los componentes creados se comentan a continuación.

3.1.4 InfoBBox:

Al pulsarlo activa el control InfoBBox (puesto que es un control no se desactiva tras seleccionar el área de búsqueda con el ratón). Presenta el icono  . Tras seleccionar la capa se debe arrastrar el ratón indicando un área de búsqueda sobre el mapa. Tras realizar la búsqueda de objetos en el área marcada se analizará la capa seleccionada. Si existe algún elemento de dicha capa en el área marcada por el ratón abrirá una nueva ventana done aparecerán los resultados, como muestra la siguiente figura.

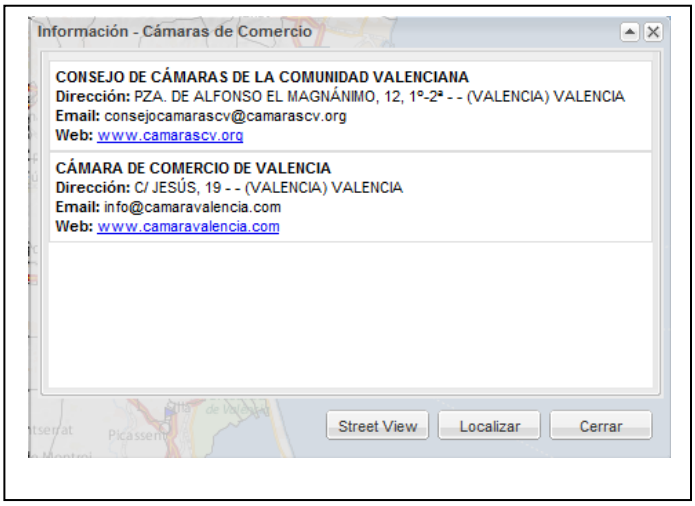

Figura 4. Resultado de la búsqueda de InfoBBox.

Como se observa en la figura, y tras seleccionar un objeto de la lista, ofrece una vista de Street View, la localización en el mapa del mismo o el cierre de la ventana.

3.1.5 aSearchButton:

Al pulsar sobre el icono abre una ventana con tres pestañas, como muestra la siguiente figura.

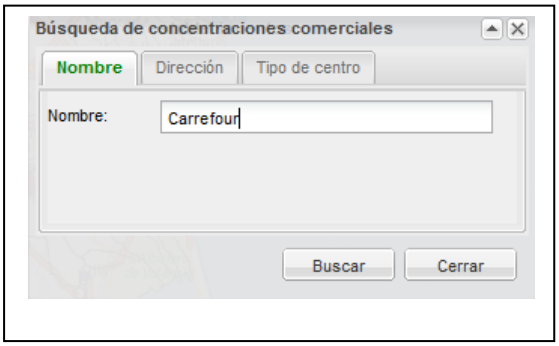

Figura 5. Pestañas búsqueda de concentraciones comerciales.

Se puede rellenar cualquiera de las tres (puede rellenarse más de una pestaña para realizar la búsqueda) y en el momento en que se edita algún campo la pestaña que se está modificando se vuelve de color verde y en negrita. En caso de que el foco pase a otra pestaña, deja de ponerse en negrita. Al realizar la búsqueda (pulsando en Buscar) muestra una ventana con los resultados, como aparece en la siguiente figura. Como el resultado de la búsqueda puede devolver muchas concentraciones, en el gridPanel se le añadió un PagingGrid que permitese cargar de una forma rápida los resultados, pudiendo cambiar de página para mostrar nuevas soluciones.

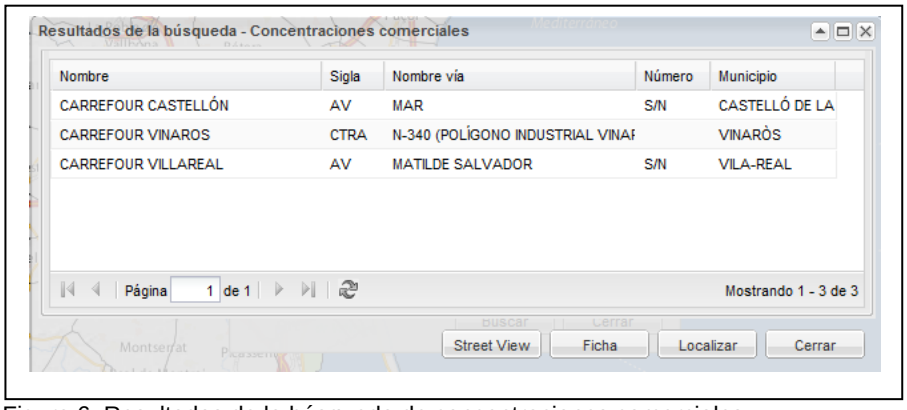

Figura 6. Resultados de la búsqueda de concentraciones comerciales

Como en el caso anterior, se puede localizar, mostrar una vista de la calle con *Street View* o mostrar una ficha de la concentración comercial seleccionada. El resultado obtenido es como el que muestra la siguiente figura.

Figura 7. Ficha de una concentración comercial

3.1.6 eSearchButton:

Similar al anterior, realiza la búsqueda de establecimientos. Al pulsar sobre el icono abre una ventana con cuatro pestañas, como muestra la siguiente figura.

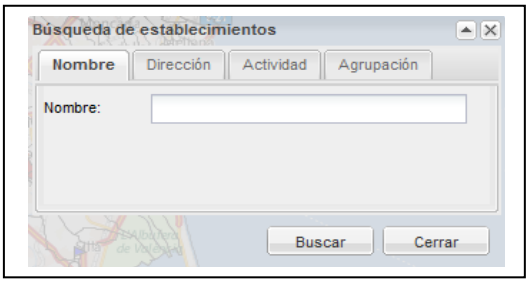

Figura 8. Pestañas búsqueda de establecimientos

Como sucedía con la búsqueda de concentraciones comerciales, en el momento en que se edita algún campo la pestaña que se está modificando se vuelve de color verde. La ficha que muestra es similar también a la que mostraba si se realizaba la búsqueda de concentraciones comerciales.

3.1.7 helpButton

Esta funcionalidad muestra un botón sobre el toolBarPanel. Al pulsarlo muestra en una nueva ventana un documento en formato html. Este documento es un manual de ayuda al usuario detallado que explica, punto por punto, el funcionamiento del visor web. Su contenido se adjunta en el apéndice al final del documento.

3.2 Extensión WMS

Esta extensión permite obtener información WMS sobre una capa seleccionada previamente. Al pulsar sobre el botón InfoWMS se activa un control (ExtWMS/Control/Info.js) que comprueba, en primer lugar, si la capa seleccionada es consultable y visible. Como se ha indicado anteriormente, se debe seleccionar la capa en primer lugar. A continuación se debe indicar sobre el mapa el punto en el que se desea obtener la información. El control debe realizar una comprobación sobre la capa seleccionada, y sólo mostrará el contenido de la capa seleccionada si la capa es visible y es consultable[6].

Si la capa seleccionada es visible y consultable se abrirá una ventana que mostrará un gridPanel con las features que mostrará como resultado. Las features a mostrar se cargan mediante la función getColumnModel.

3.3 Impresión

Como se expuso en la definición del problema, el visor web debe permitir la impresión de un mapa y su leyenda. Esta extensión comporta por una extensión en el lado cliente, donde se realiza la solicitud de la impresión del mapa que se muestra en el visor, y otra extensión en el lado servidor, donde se procesa la orden, generándose el mapa y su leyenda. Como resultado muestra el documento generado en una nueva pestaña del navegador en un formato pdf. La parte cliente[7] se generó en JavaScript, mientras que la parte servidor se realizó en Java (servlet en java).

En el anexo se puede observar, a continuación de la parte referida al lado cliente, la referida al lado servidor. Para llevar a cabo la impresión se realizó una plantilla de impresión mediante el software de código abierto **Jasper Report**. En él se definieron los márgenes de dicha plantilla, el tamaño de la leyenda y del mapa (se puede ver cómo se han descrito en el anexo). Se

[6] Véase Anexo D
[7] Véase Anexo D

generó un XML de la plantilla y se definieron los márgenes y propiedades del jasper creado en el fichero wsPrintWMC.Properties.

Como se indicaba anteriomente, desde el lado servidor se realiza la composición de los elementos que generan el mapa y la leyenda. Con el objetivo de aclarar la funcionalidad de impresión en el lado cliente, la siguiente figura muestra un diagrama que, a modo de glosario, muestra las diferentes clases que se utilizaron para realizar la impresión del documento.

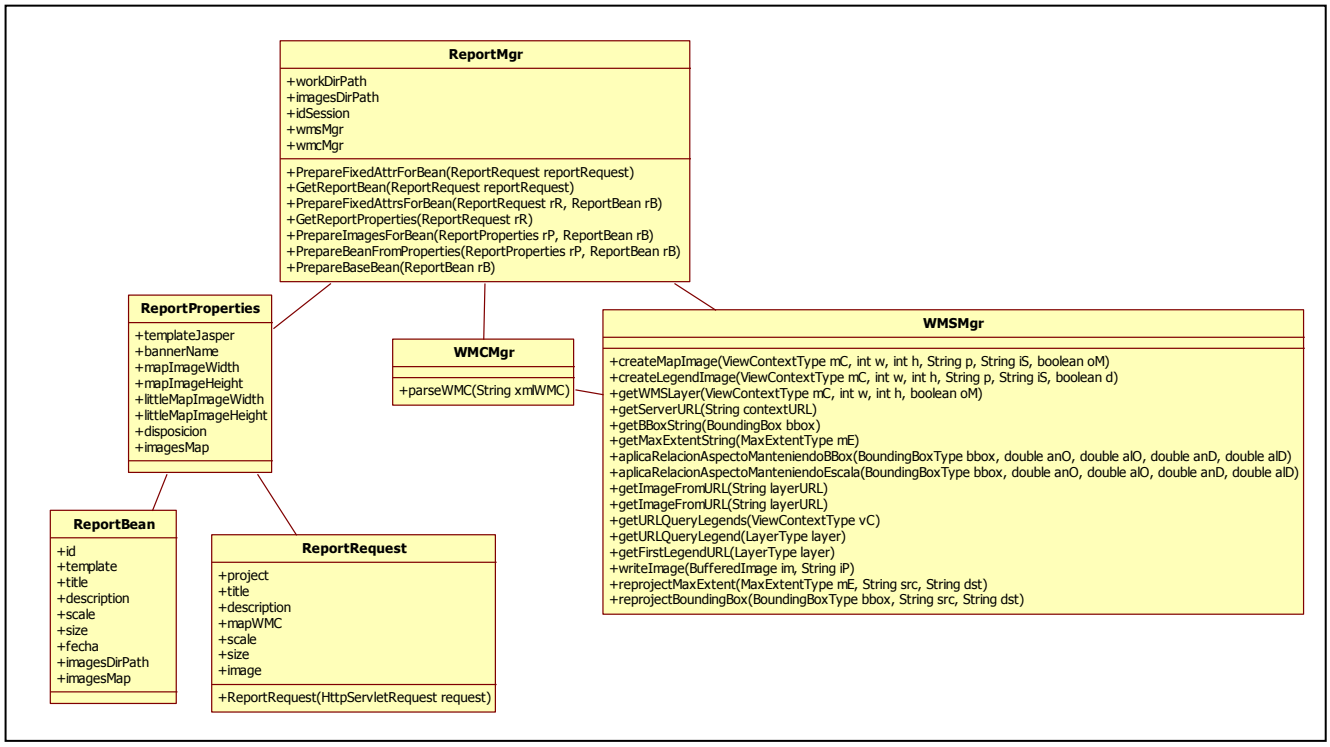

Figura 9. Glosario de ayuda para comprender el funcionamiento de la impresión de imágenes en el lado servidor.

En la figura anterior se observan una serie de clases que comentaré brevemente:

- ReportRequest realiza una petición de los atributos al *servlet* mediante el método getParameter.

- ReportBean obtiene y almacena los atributos del *jasper*.

- ReportProperties obtiene y almacena las propiedades del jasper.

A primera vista puede parecer innecesario crear dos clases para realizar las peticiones de la plantilla, ya que tanto las propiedades definidas en ReportProperties como las obtenidas en ReportBean son propiedades definidas en la plantilla del jasper. ReportProperties solicita la información que contiene el fichero wsPrintWMC*.properties* del que ya se ha hablado. De esta forma resulta un código más organizado.

La plantilla se rellenará una vez obtenidos el valor para cada uno de los atributos mediante peticiones al servlet con ReportRequest, para ello se rellenarán los atributos definidos en el ReportBean mediante el método prepareFixedAttrForBean. Sobre la imagen del mapa debe realizarse un *parseo*[8] a un servicio WMC (*WebMapContext*) para crear la imagen posteriormente sobre el ReportBean. Para evitar que la imagen del mapa, así como la de la leyenda, salga pixelada se optó por solicitar el ancho y el alto de la imagen del mapa y de la leyenda al doble para encajarlos posteriormente en el espacio definido para ambos. Todas estas funcionalidades se describen en ReportMgr, como puede observarse en el anexo.

La impresión del mapa se realiza desde la clase PrintWMCServlet, que se muestra en el anexo. Al crearse este objeto se define dónde están almacenadas las imágenes del mapa y de la leyenda, así como de la plantilla creada con **jasperReports**. El *servlet* aceptará peticiones POST, rellenando la plantilla, en función de si se trata de una imagen o de texto, invocando al objeto PrinterMgr o creando la imagen del mapa a partir del método *createMapImage*.

3.4 StreetView

En la definición del problema se indicaba que el visor debe mostrar en una nueva ventana una imagen de la dirección indicada con el ratón sobre el mapa, mostrando una vista a pie de calle, facilitando además unas herramientas que permitieran maximizar la ventana, ampliar o reducir el zoom de la imagen mostrada, y moverse por la calle con solo arrastrar el ratón sobre la imagen. Esta herramienta no es otra que la de **Street View** de **GoogleMaps**,

[8] Parseo (proviene de *parcing,* en inglés) se llama al proceso de análisis de símbolos con el fin de determinar su estructura gramatical respecto a una gramática dada.

y ofrecida por **Google** para desarrolladores web. Mediante GStreetViewPanorama podemos generar una vista Street View pasándole como parámetro una posición definida por la longitud y latitud (*lonlat*), o lo que es lo mismo, la coordenada en el eje de la X y en el eje de la Y (se puede obtener información acerca de este control en http://code.google.com/intl/es-ES/apis/maps/documentation/javascript/v2/reference.html#GStreetviewPanorama).

3.5 Trabajo futuro

Antes de hablar de posibles extensiones que completen aún más el visor web, es conveniente hacer especial hincapié en algo que ya se ha comentado a lo largo de la memoria. Este visor se caracteriza por ser **modular** y **extensible**, y por tanto la ampliación del mismo mediante nuevas herramientas y funcionalidades resulta tremendamente sencilla ya que no supone realizar cambios sobre el código ya generado.

3.5.1 Mapa localizador

Atendiendo a las funcionalidades que ofrece la librería OpenLayers, resulta interesante la opción de crear una nueva extensión que permita localizar la región del mapa sobre la que se ha hecho zoom, bien de forma manual para que el usuario trabaje con un área menor y pueda por tanto verla más grande en el espacio que ofrece el visor para el mapa, bien porque se ha pulsado al botón localizar y la propia aplicación maximiza el zoom hasta el punto que el usuario pueda perder la orientación del lugar sobre el mapa regional.

En el Mapa localizador se muestra la situación de la cartografía que se está consultando en cada momento. El área del localizador es interactiva con el área del mapa, de modo que pulsando y arrastrando sobre el localizador se genera un área rectangular, esta área rectangular se empleará para hacer zoom sobre el área del mapa.

Figura 10. Extensión de Localizador. El recuadro en Rojo indica el área del mapa que estamos consultando.

3.5.2 Sobre StreetView

Ya se ha comentado que StreetView de GoogleMaps se puede incorporar (como se incorporó al proyecto ATLAS) a proyectos de desarrollo web mediante GStreetViewPanorama. Google proporciona una herramienta que permite, además, visualizar en qué zonas está activo el StreetView mediante GStreetViewOverlay. La siguiente figura muestra un ejemplo de esta opción, sobre el mapa se visualizan las calles que tienen activo el StreetView, quedan subrayadas con líneas azules que resaltan sobre el mapa.

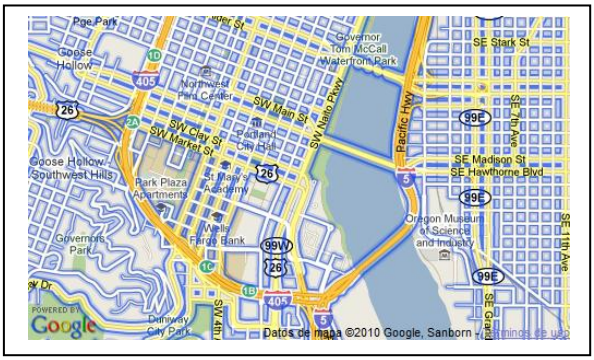

Figura 11. Las calles resaltadas tienen activado StreetView.

3.5.3 Otros mapas

Como se dijo al principio de la memoria, se podía insertar otros mapas como capas WMS, uno de los posibles mapas comerciales y cada vez más extendido es el que ofrece Microsoft con BingMaps. Cuando se realizó este proyecto Bing todavía ofrecía una versión Beta. Actualmente OpenLayers ya ofrece la posibilidad de trabajar con Bing como capa base cartográfica. La siguiente figura muestra un ejemplo de cómo quedaría un mapa con capa base Bing.

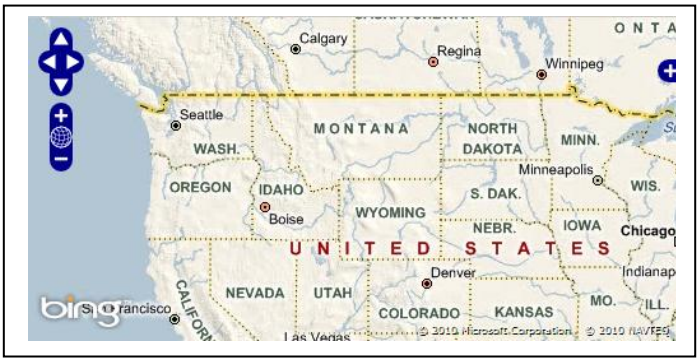

Figura 12. OpenLayers ofrece la posibilidad de trabajar con capa base Bing de Microsoft.

3.5.4 Dibujar sobre el mapa

Por último, OpenLayers ofrece las herramientas para dibujar sobre el mapa puntos, líneas, áreas o polígonos, con las posibilidades que ello conlleva. Así por ejemplo el usuario podría realizar las búsquedas que ofrece el botón de información del que ya se ha hablado sin necesidad de trazar un área rectangular, pudiendo calcular los establecimientos, cámaras, etc. trazando una figura poligonal o una línea. Un ejemplo de dibujo sobre el mapa viene ofrecido por OpenLayers en esta dirección: http://openlayers.org/dev/examples/draw-feature.html.

Estas posibles extensiones son sólo algunas de las que ofrecen OpenLayers y Google. Se podrían mejorar algunas funcionalidades con la librería ExtJS, como por ejemplo la mejora que se explicó en el Grid, facilitando la carga de datos de una forma rápida cuando ésta es muy pesada. Estas mejoras se incluirían dentro de las mejoras ofrecidas por las actualizaciones, indicar que OpenLayers es una librería compatible con los estándares XHtml, pero que existen algunos problemas propios de la librería referentes a la visualización de algunas ventanas en Internet Explorer, no ocurriendo lo mismo cuando se carga la web en otro navegador como, por ejemplo, Mozilla Firefox. Esto no son propiamente errores, ya que si se consulta en la documentación de OpenLayers ya informan que esos defectos visuales se producen al cargar la página en Internet Explorer.

4. Relación con la industria

El Atlas Comercial de la Comunidad Valenciana es, como se ha explicado anteriormente, un visor web que permite realizar búsquedas sobre los diferentes comercios y cámaras de comercio, entre otras, así como obtener información de los diferentes municipios de la comunidad, permitiendo al usuario que explote estas características sobre el mapa que desee, bien mapas de carácter genérico como los de Yahoo o Google, o bien mapas de carreteras (PNOA). El usuario al que va por tanto destinado es fundamentalmente aquellos que tengan interés en realizar este tipo de búsquedas, pero dada su **flexibilidad** a la hora de insertar nuevos mapas y nuevas capas de búsqueda, podrían perfectamente realizarse búsquedas de calles o de carreteras de la Comunidad Valenciana por cualquier usuario.

La principal ventaja que presenta este visor radica, en primer lugar, que se trata de código libre, realizado con software *OpenSource*, y por tanto, tiene un **coste cero**, esto es, completamente gratuito. También cabe destacar otra característica propia de este visor, que no es otra que su **modularidad**, característica de la que ya se ha hablado, pero que es importante remarcar ya que dota al visor de la capacidad ser ampliado sin más complicaciones que crear una nueva extensión, completamente funcional, y añadirla al visor sin que presente mayores problemas. Finalmente conviene señalar una característica que, si bien el usuario no tiene por qué apreciarla, dota al código y al desarrollo de software de una buena calidad, ya que como se ha comentado en esta memoria, su funcionalidad es apreciable desde cualquier navegador que cumpla los estándares XHTML.

La principal función del visor es la búsqueda y localización sobre el mapa de los diferentes establecimientos comerciales de la Comunidad Valenciana pertenecientes a la Cámara de Comercio. Esta funcionalidad puede ser útil para el análisis industrial, realizando estudios de crecimiento comercial en las diferentes áreas de la comunidad, así como medición de riesgos que ayuden en la toma de decisiones de futuros emplazamientos, teniendo en

cuenta no sólo la demografía de los municipios que pueblan el área de estudio, sino también la comunicación de carreteras que éstos puedan tener.

Para posibles empresarios que quieran establecer un nuevo comercio en un área de la Comunidad puede ser muy práctico realizar un estudio mediante una búsqueda sobre el mapa indicando el tipo de establecimiento y la ubicación. De esta forma puede consultar la existencia de negocios similares en dicha área.

La funcionalidad StreetView dota al visor de una característica ventajosa, ya que facilita al usuario que desee realizar la búsqueda de un comercio la posibilidad de explorar la calle donde está ubicado el objeto de búsqueda.

Otras posibilidades que ofrece un visor de estas características podría por ser muy útil además en las páginas webs de aquellos ayuntamientos de la Comunidad Valenciana que quieran potenciar el sector turístico en su comarca. Ya que no sólo ofrece información de carreteras que faciliten al usuario el acceso al municipio sin tener que recurrir a otras webs, sino que puede aportar información del propio municipio así como de los municipios colindantes. En este marco podría ampliarse el contenido de información mediante una nueva ficha como la de los establecimientos, esta vez enfocada al turismo.

5. Conclusiones

La realización de un proyecto de estas características exige, en primer lugar, iniciarse en los conocimientos de las librerías OpenLayers y ExtJS, ya que se han utilizado a lo largo de todo el proyecto (atendiendo a los manuales de ExtJS, muchas funcionalidades se podrían haber resuelto invocando una función en lugar de crear una clase y añadirle métodos, otras veces en cambio se ha creado una clase de OpenLayers para mantener un código lo más orientado a objetos posible, ya que JavaScript no es un lenguaje orientado a objetos). Además de ampliar los conocimientos de algunas herramientas y lenguajes de programación, incluídos en el programa universitario, como es Java o la realización de plantillas con JasperReports. El resultado del proyecto es un código bien organizado, modular y extensible, hasta el punto que algunas funcionalidades del visor se trata de código reutilizado de otros visores web, y algunas funcionalidades propias de este visor se han introducido en el núcleo de la aplicación dado que es una funcionalidad muy común que puede ser compartida con otros visores web.

El resultado final de este proyecto se puede probar en la siguiente URL:

http://atlas.iver.es

6. Anexo A

Manual de usuario
Atlas Comercial
de la
Comunidad Valenciana

Arturo Argilés Casasús

Indice

1. Introducción

En este manual se describen las funcionalidades y el uso de la herramienta WEB creada para la consulta del Atlas Comercial de la Comunidad Valenciana. En esta ayuda se facilita una visión global del empleo del cliente WEB a nivel de usuario y administrador, consultando el servicio básico de mapas WMS.

El cliente WEB permite consultar la cartografía de los servicios WMS implantados por otras instituciones, y de otros servicios WMS existentes y públicos en Internet.

2. Cliente WMS

El servicio de cartografía WMS (Web Map Services o Servicio de Mapas en la Red) de la IDE, se consulta mediante el cliente WEB creado a tal efecto. Se trata de un cliente WEB que se caracteriza por ser modular y extensible, por tanto será posible la inclusión de nuevas herramientas y funcionalidades en el futuro.

Este cliente de consulta no necesita instalación, es consultable desde un navegador de Internet, como Internet Explorer o Mozilla Firefox. El navegador debe tener habilitado el uso de javascript.

2.1. Características

El cliente WEB dispone de áreas perfectamente diferenciadas donde se agrupa la funcionalidad:

- Barra de herramientas: almacena las herramientas interactivas con el área del mapa.
- Área del mapa: donde se representa la cartografía mediante imágenes.
- Tabla de contenidos: muestra la colección de capas con las que se puede interactuar, y las leyendas de las mismas.
- Barra de estado: zona inferior de la aplicación donde se muestra información al usuario, como coordenadas del puntero y la escala del mapa.

- Área de administración

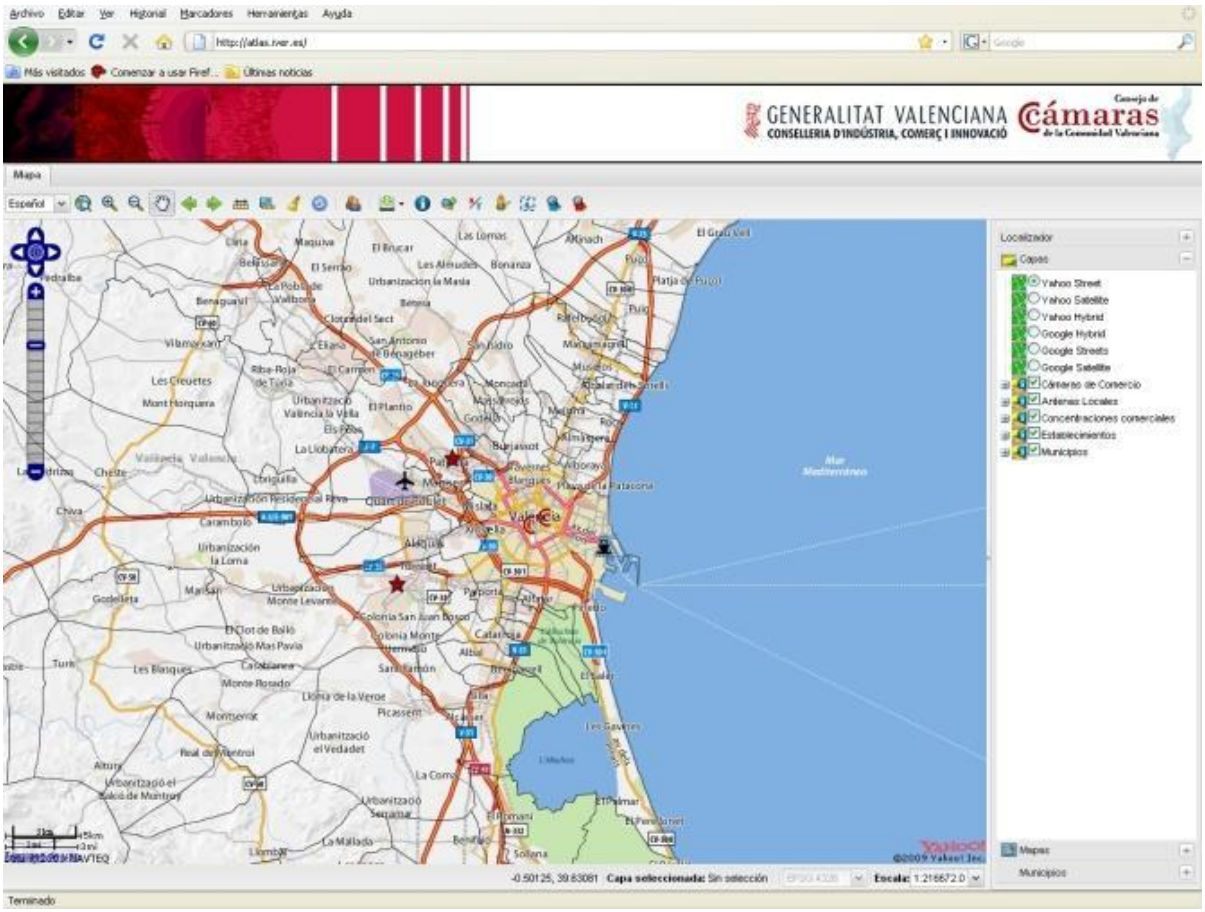

Figura 13. Vista del visor web Atlas Comercial de la Comunidad Valenciana.

2.2 Barra de herramientas

La barra de herramientas almacena las herramientas interactivas con el área del mapa. Existen herramientas que interactúan con el área de la cartografía, deben estar activas inicialmente para poder emplearlas. Sólo es posible mantener una herramienta como activa.

Figura 14. Barra de herramientas

- Herramientas de navegación: Son las herramientas típicas de navegación por la cartografía, que permiten ampliar, reducir y desplazarse por la misma

	Zoom Completo	Herramienta que realiza un zoom a la totalidad de la cartografía.
	Más zoom	Herramienta que permite ampliar la imagen y aumentar su detalle. Esta herramienta funciona mediante un clic o mediante la creación de una ventana.
	Menos zoom	Herramienta que permite disminuir la imagen.
	Vista anterior	Permite volver a extensiones de zoom anteriores.
	Vista siguiente	Permite volver a extensiones de zoom posteriores.
	Desplazar el mapa	Permite el desplazamiento por la cartografía mediante acciones de pulsar y arrastrar.

- Herramientas de medida: Son las herramientas de medidas de distancias y áreas sobre la cartografía.

	Distancia	Permite medir distancias sobre la imagen del mapa.
	Área	Permite medir áreas sobre la imagen del mapa.

- Herramientas:

	Limpiar el mapa	Borra los elementos gráficos de la vista.
	Recargar el mapa	Actualiza las capas en la vista.

	Administración	Permite acceder al área de registro.

- Herramienta de información:

	Información WMS	Facilita la información de los campos de la capa seleccionada.
	Información	Obtiene información de la capa seleccionada.

- Exportar e Importar ficheros:

	Exportar e Importar	Exporta e Importa en diferentes formatos.

- Servicio WMS

	Servicio WMS	Exporta e Importa en diferentes formatos.

-

	Centrado en coordenadas	Centra el mapa en las coordenadas introducidas

-

	*Street View*	Selecciona un punto en el mapa para obtener una vista panorámica.

- Herramientas de Búsqueda Comercial:

	Búsqueda de establecimientos	Permite realizar búsqueda de establecimientos indicando un

		criterio de búsqueda.
	Búsqueda de concentraciones comerciales.	Permite realizar búsqueda de concentraciones comerciales introduciendo un criterio de búsqueda.

- Herramienta de selección de idioma:

	Selección de idioma	Permite cambiar el idioma de la Web, puede seleccionar entre Español o Valencià.

2.3. Área del mapa

El área del mapa es donde se representa la cartografía mediante imágenes. Dispone de una herramienta con zonas sensibles para desplazamientos laterales. En la esquina superior izquierda se encuentra un punto que permite realizar desplazamientos por el mapa y modificar la escala de la vista.

Figura 15. Área del mapa con la herramienta que permite realizar desplazamientos y zoom sobre el mapa.

2.4. Tabla de contenidos

La tabla de contenidos, muestra como una pila la colección de capas con las que se puede interactuar, y las leyendas de las mismas. La tabla de contenidos puede modificarse mediante herramientas que permiten alterar el orden de visualización.

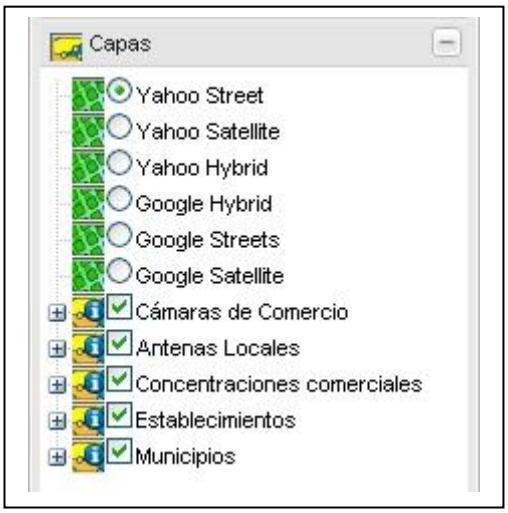

Figura 16. Tabla de Contenidos con detalle de las diferentes capas.

Puede cambiar el orden de visualización de las capas, para ello haga *click* sobre la capa deseada y arrástrela hasta la posición deseada.

La tabla de contenidos también muestra información acerca de los elementos que contiene, con una representación de la leyenda empleada. La tabla de contenidos permite mostrar u ocultar la leyenda de una capa pulsando el símbolo que precede al nombre de la misma. Si desea mostrar u ocultar una capa active o desactive el checkbox de la misma.

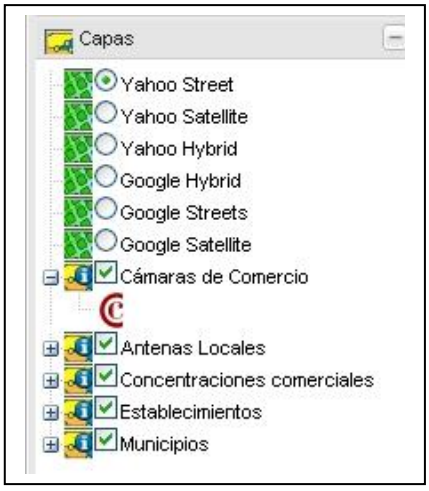

Figura 17. Detalle de la leyenda en la tabla de contenidos.

También es posible cambiar el valor de opacidad de una capa, hacer zoom a la capa, eliminarla, crear un grupo para gestionar varias capas a la vez y renombrar la capa. Para ello pulse la capa deseada y acceda al menú contextual pulsando el botón secundario del ratón.

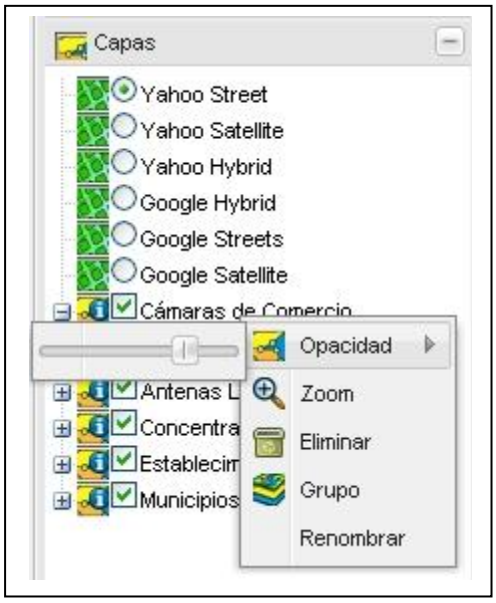

Figura 18. Edición de una capa

Desde la Tabla de Contenidos también puede modificar los Mapas que desea visualizar en la vista, para ello seleccione en el panel desplegable Mapas

Figura 19. Selección de mapas desde la tabla de contenidos.

También puede seleccionar municipios desde el panel desplegable Municipios

Figura 20. Selección de municipios.

Indique la provincia y el municipio que desee y aparece seleccionado en la vista como se muestra en la siguiente imagen

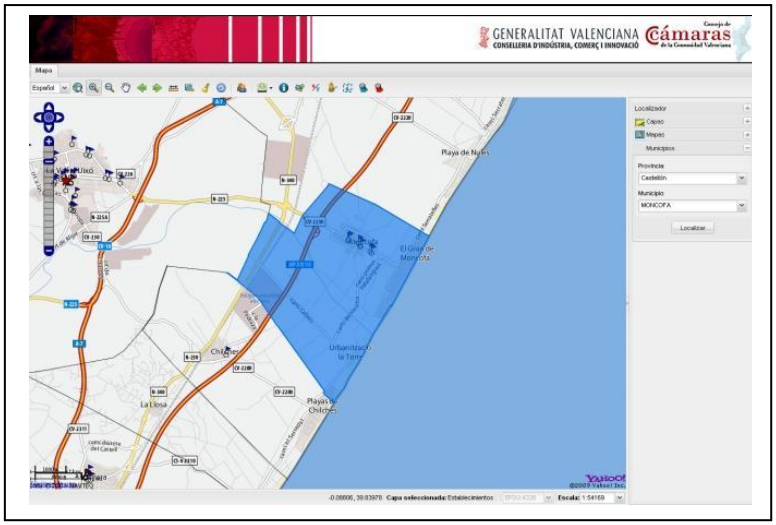

Figura 21. Área del mapa seleccionado.

2.5. Barra de estado

La barra de estado es la zona inferior de la aplicación donde se muestra información al usuario, como coordenadas del puntero y la escala del mapa.

La herramienta de escala muestra la escala a la que se visualizan los datos, y permite designar una escala de visualización especificada por el usuario. También se mostrará la distancia y/o área calculada.

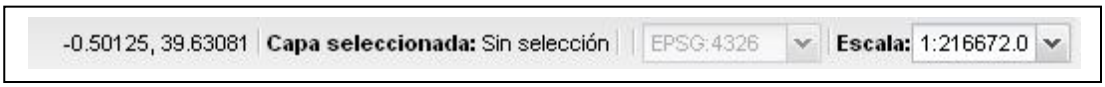

Figura 22. Barra de estado.

3. Administración

Mediante el icono de la barra de herramientas Administración  puede acceder al área de registro. Desde aquí puede controlar y gestionar la cartografía que se desea publicar en el visor de mapas. Se abre una ventana para realizar el registro

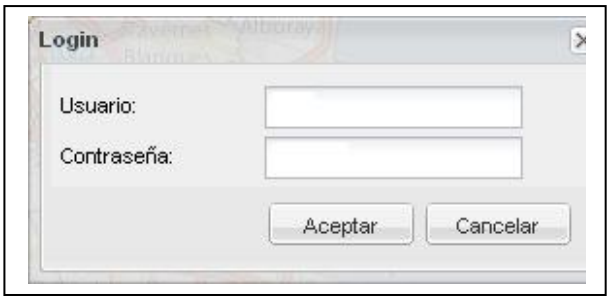

Figura 23. Área de registro al panel de Administración

Al acceder como administrador se abre una nueva pestaña como se muestra a continuación

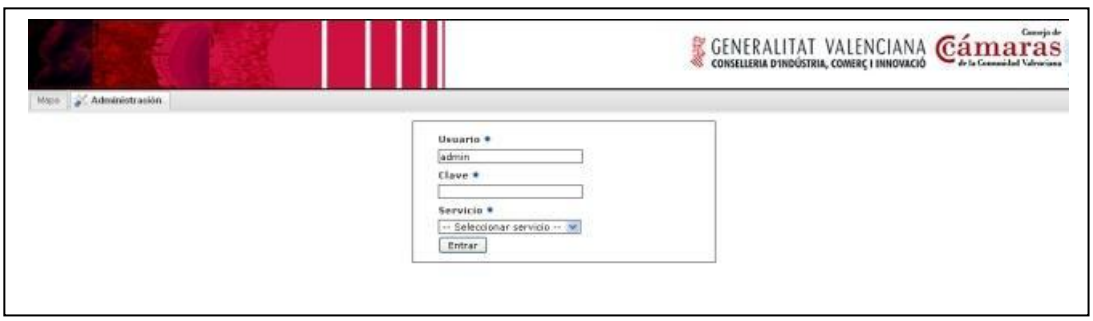

Figura 24. Detalle de la nueva pestaña que se abre al acceder como administrador.

4. Exportar e importar

Desde la herramienta Exportar e importar  puede exportar a Web Map Context (WMC) que es un estándar de OGC (http://www.opengeoespatial.org), a imagen o bien a pdf. El fichero con capas WMC puede reproducirse sobre cualquier plataforma que soporte WMC. El resultado es un archivo XML con formato específico y extensión .cml.

Figura 25. Menú ofrecido por la funcionalidad Exportar/Importar

Si selecciona la opción Exportar a imagen se abre una nueva ventana como se indica

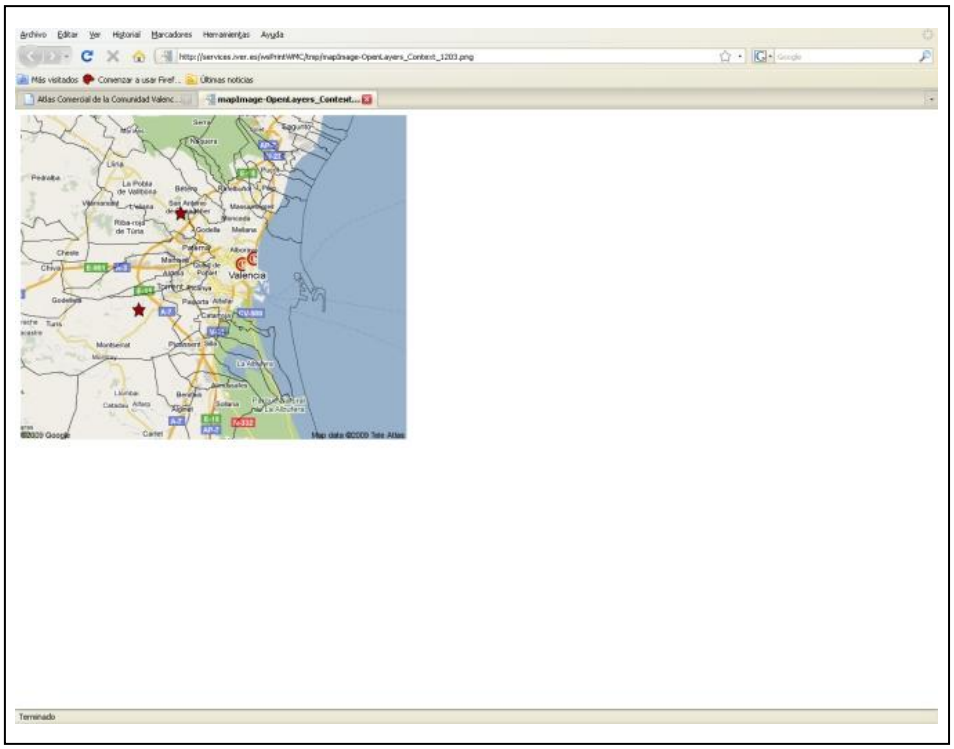

Figura 26. Resultado de Exportar una imagen.

Si Exporta a pdf se genera el pdf en una nueva ventana, esta muestra una imagen como la que se indicaba en el Anexo 6.1, desde donde puede guardar una copia del documento, imprimirlo, etc.

5. Acceso desde otros clientes WMS

Otra de las herramientas del Visor es la posibilidad de añadir capas de otros clientes WMS, concretamente del Catastro, y PNOA, y combinarlas con las ya existentes.

La herramienta para acceder a este servicio es *Servicio WMS*  pulsándola aparece la siguiente ventana:

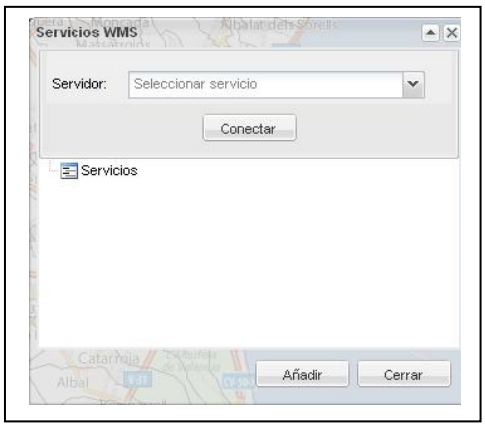

Figura 27. Servicio WMS.

Una vez seleccionado el servicio conecta con el servidor y por último basta escoger que capa deseamos añadir.

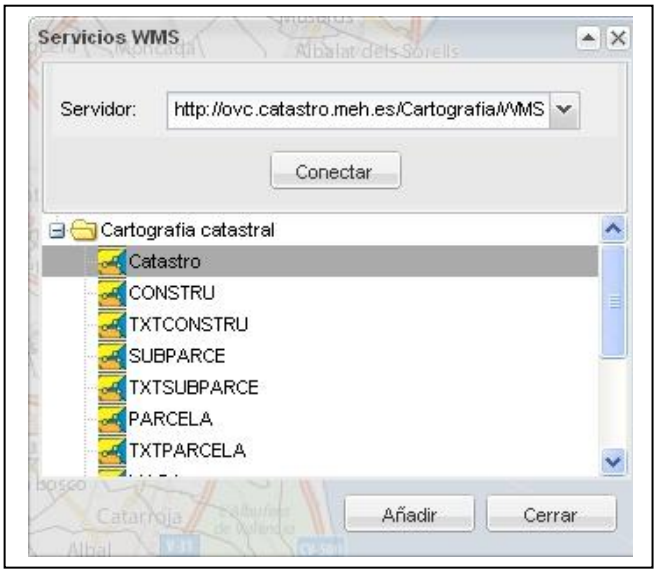

Figura 28. Detalle de conexión a un servidor WMS para escoger capas.

La capa seleccionada se añade a la vista.

6. Street View

Mediante la herramienta *Street View*  se abre una ventana donde el usuario puede visualizar una imagen de 360º, para ello seleccione esta utilidad y pulse con el ratón un punto en la vista.

Figura 29. Visualización de una calle con Street View.

La ventana con la vista panorámica puede hacerla grande e incluso hacer la pantalla completa pulsando el icono situado en la parte superior derecha  . También puede desplazarse por la vista panorámica con el ratón o bien con las teclas A y D para girar a izquierda y derecha respectivamente, W y S para desplazarse verticalmente (arriba y abajo), y con las flechas del teclado.

7. Información

La herramienta información  permite obtener información de la capa seleccionada. Para utilizar la herramienta de información, primero seleccione con el ratón en la tabla de contenidos una capa, a continuación pulse sobre la

herramienta y luego de nuevo sobre el mapa en el elemento de la capa del que se desea obtener la información.

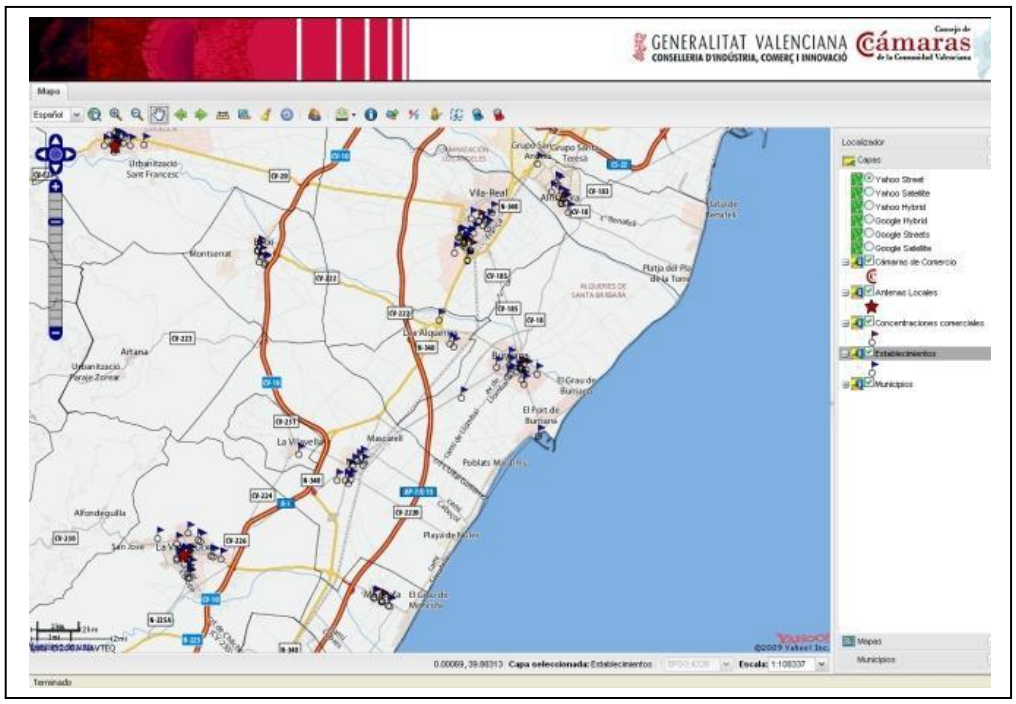

Figura 30. Debe seleccionarse la capa en primer lugar.

La Barra de estado muestra cual es la capa que tiene seleccionada, pulse la herramienta de información y haga un rectángulo en la vista

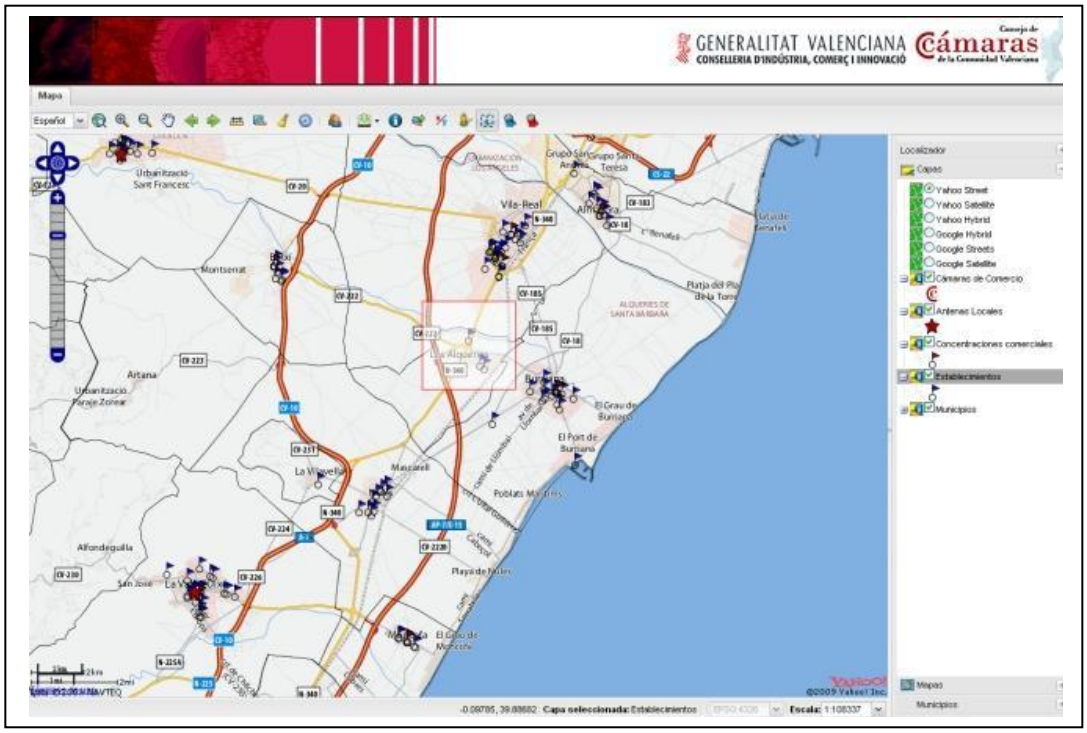

Figura 31. Selección de área arrastrando el ratón.

Aparece entonces una ventana con los establecimientos encontrados

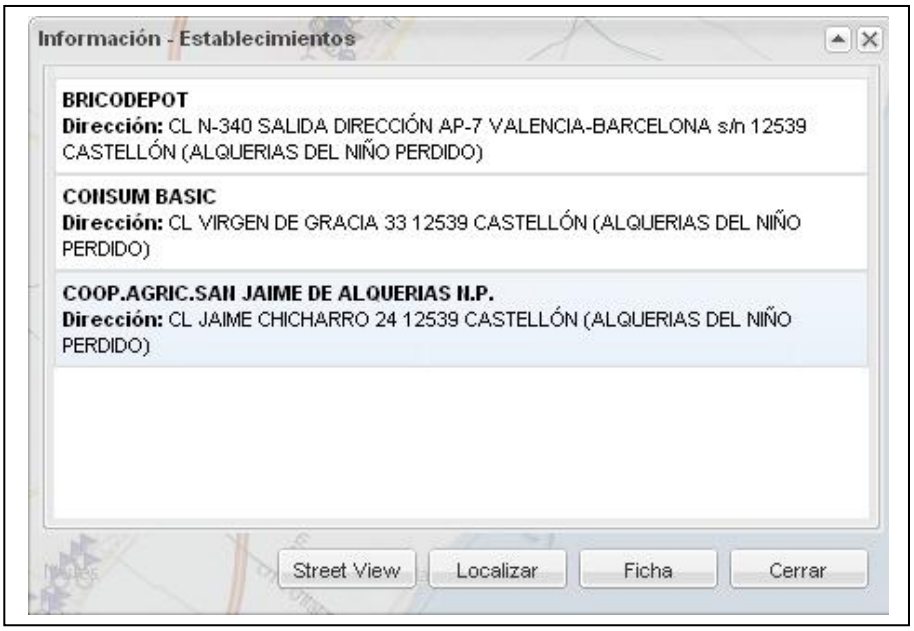

Figura 32. Ventana con el resultado de la búsqueda de establecimientos.

Desde esta ventana puede ver en una nueva ventana la vista panorámica, para ello seleccione el establecimiento deseado y pulse el botón *Street View*

Figura 33. Ventana de Street View lanzada desde la ventana resultados de la búsqueda de establecimientos.

Si selecciona el botón Localizar sitúa la vista en el establecimiento y si pulsa el botón Ficha se abre la ficha del establecimiento como se muestra a continuación

Figura 34. Detalle de ficha de establecimientos.

Si selecciona la capa Municipios en la Tabla de Contenidos, al activar la herramienta información y pulsar sobre un municipio aparece la siguiente ventana con la información correspondiente al municipio seleccionado

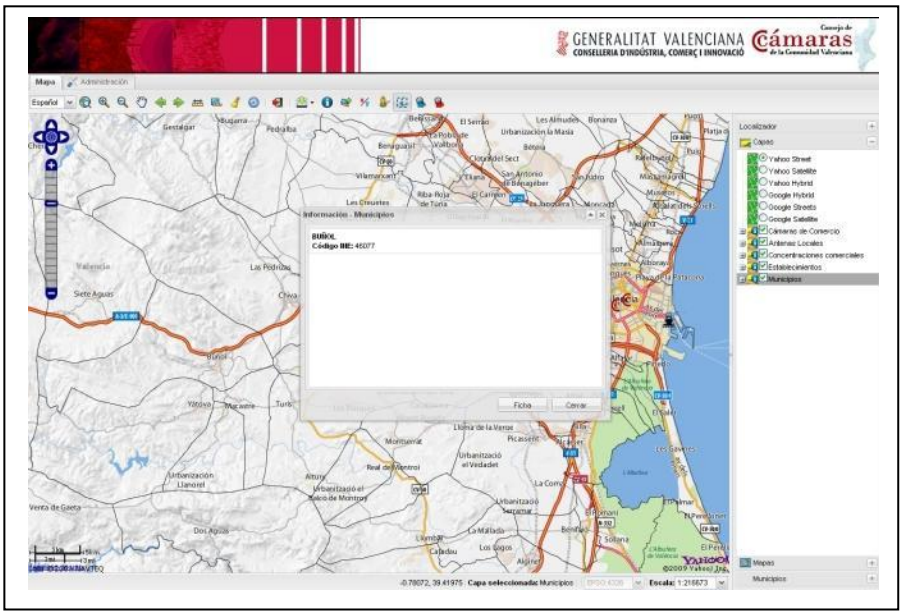

Figura 35. Información de municipios

Seleccione el municipio y pulse sobre el botón Ficha, aparecerá la infomación correspondiente al municipio como se muestra en la siguiente imagen

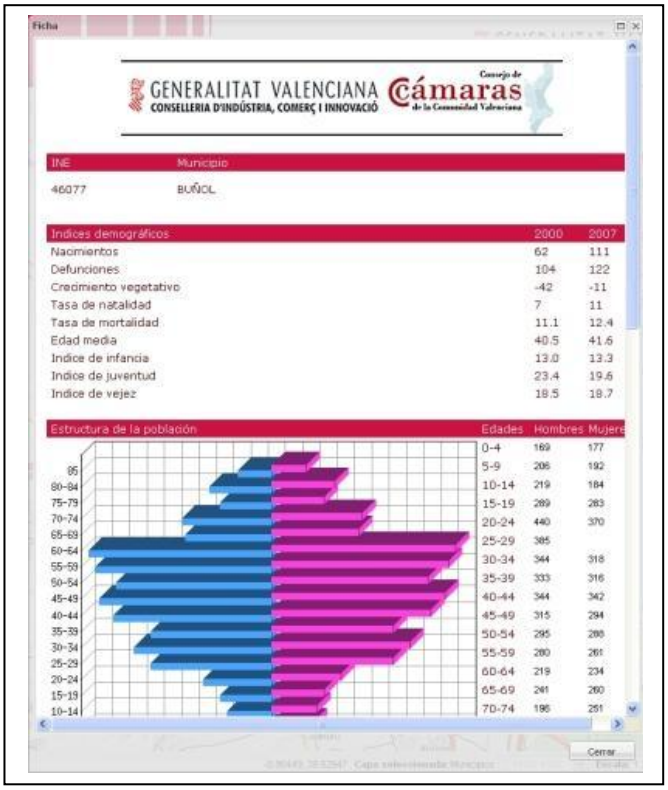

Figura 36. Ficha de municipios

8. Búsqueda comercial

A partir de los iconos de la Barra de herramientas Búsqueda comercial puede realizar búsquedas de establecimientos y concentraciones comerciales

Figura 37. Iconos de establecimientos y concentraciones comerciales, respectivamente.

Si selecciona la herramienta Búsqueda de establecimientos  se abre la siguiente ventana para que realice la búsqueda

Figura 38. Ventana de búsqueda de establecimientos.

Rellene los datos que desee para realizar la búsqueda y pulse Buscar

Figura 39. Cuando se edita un panel la pestaña se pone en verde y negrita. Si se cambia de pestaña se mantendrá en verde si el campo editado no se ha borrado.

Aparece la siguiente ventana con el resultado

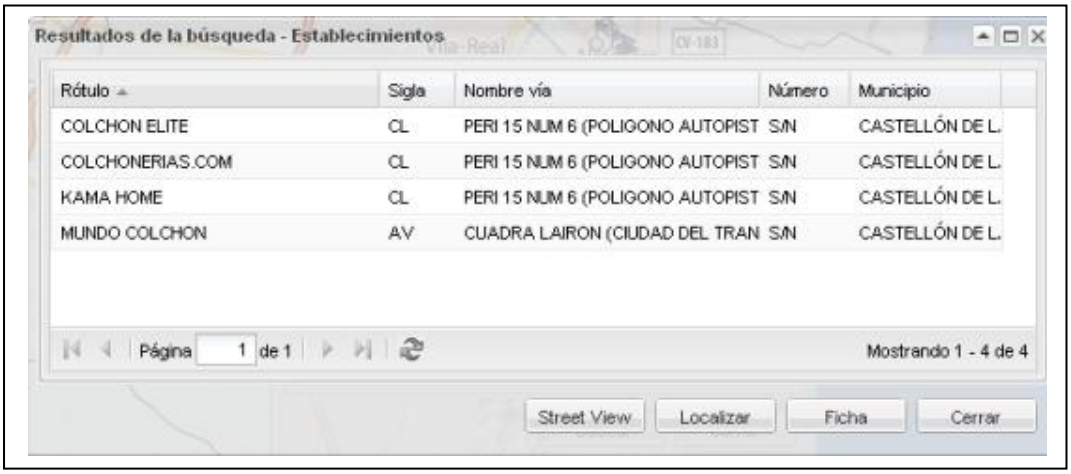

Figura 40. Resultados de la búsqueda de establecimientos.

Desde aquí puede acceder a Street View, para ello seleccione el establecimiento que desee y pulse el icono Street View

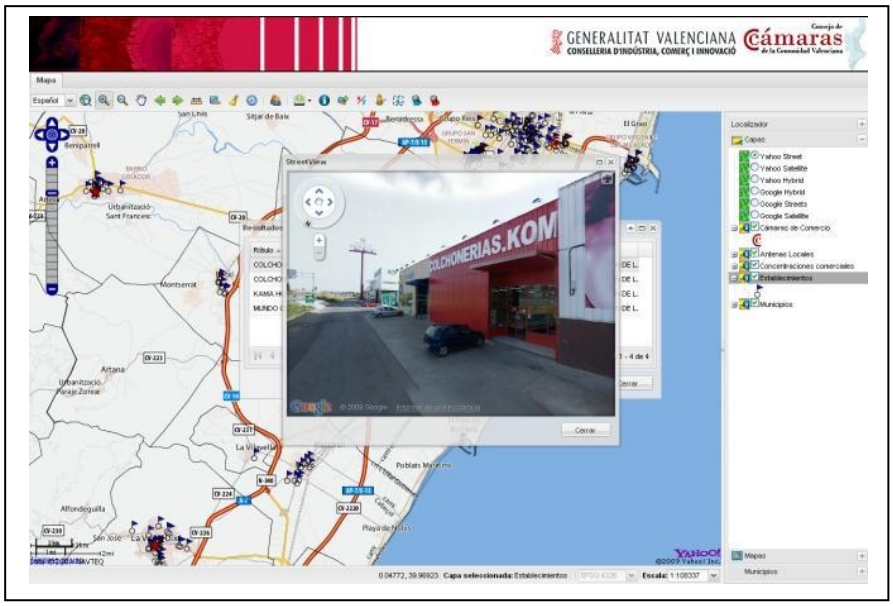

Figura 41. Detalle de Street View desde la ventana de búsqueda de establecimientos.

Si selecciona el botón Localizar situa la vista en el establecimiento y si pulsa el botón Ficha se abre la ficha del establecimiento como se muestra en la siguiente imagen

Figura 42. Ficha de un establecimiento.

Mediante la herramienta Búsqueda de concentraciones comerciales  se abre la siguiente ventana para que realice la búsqueda

Figura 43. Ventana de búsqueda de concentraciones comerciales.

Rellene los datos que desee para realizar la búsqueda de la concentración comercial y pulse Buscar

Figura 44. Edición de un panel de búsqueda de concentraciones comerciales.

Aparece la siguiente ventana con el resultado de la búsqueda

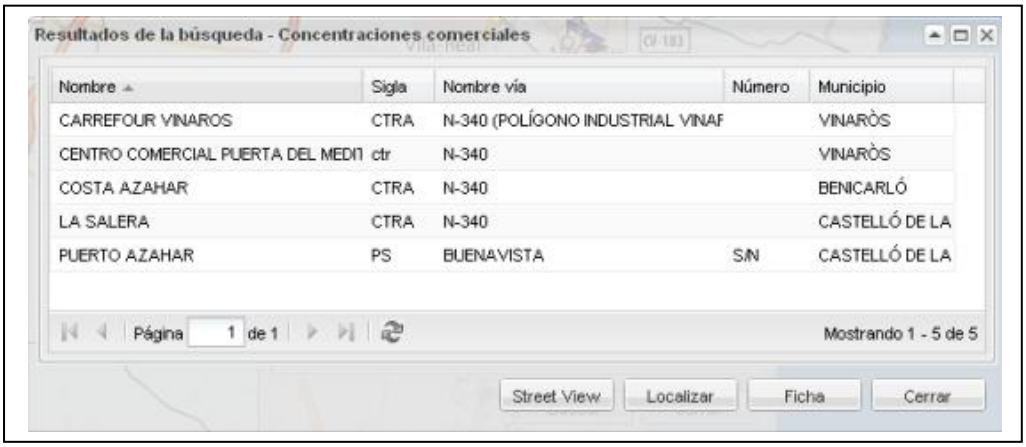

Figura 45. Resultado de la búsqueda de concentraciones comerciales.

Desde esta ventana puede acceder a *Street View*, para ello seleccione el establecimiento que desee y pulse el botón *Street View*

Figura 46. Detalle de Street View desde la ventana de búsqueda de concentraciones comerciales.

Si selecciona el botón Localizar sitúa la vista en el establecimiento y si pulsa el botón Ficha se abre la ficha de la concentración comercial como se muestra a continuación

Figura 47. Ficha de una concentración comercial.

Detalle de Impresión

Este es el resultado que se obtiene al realizar una impresión.

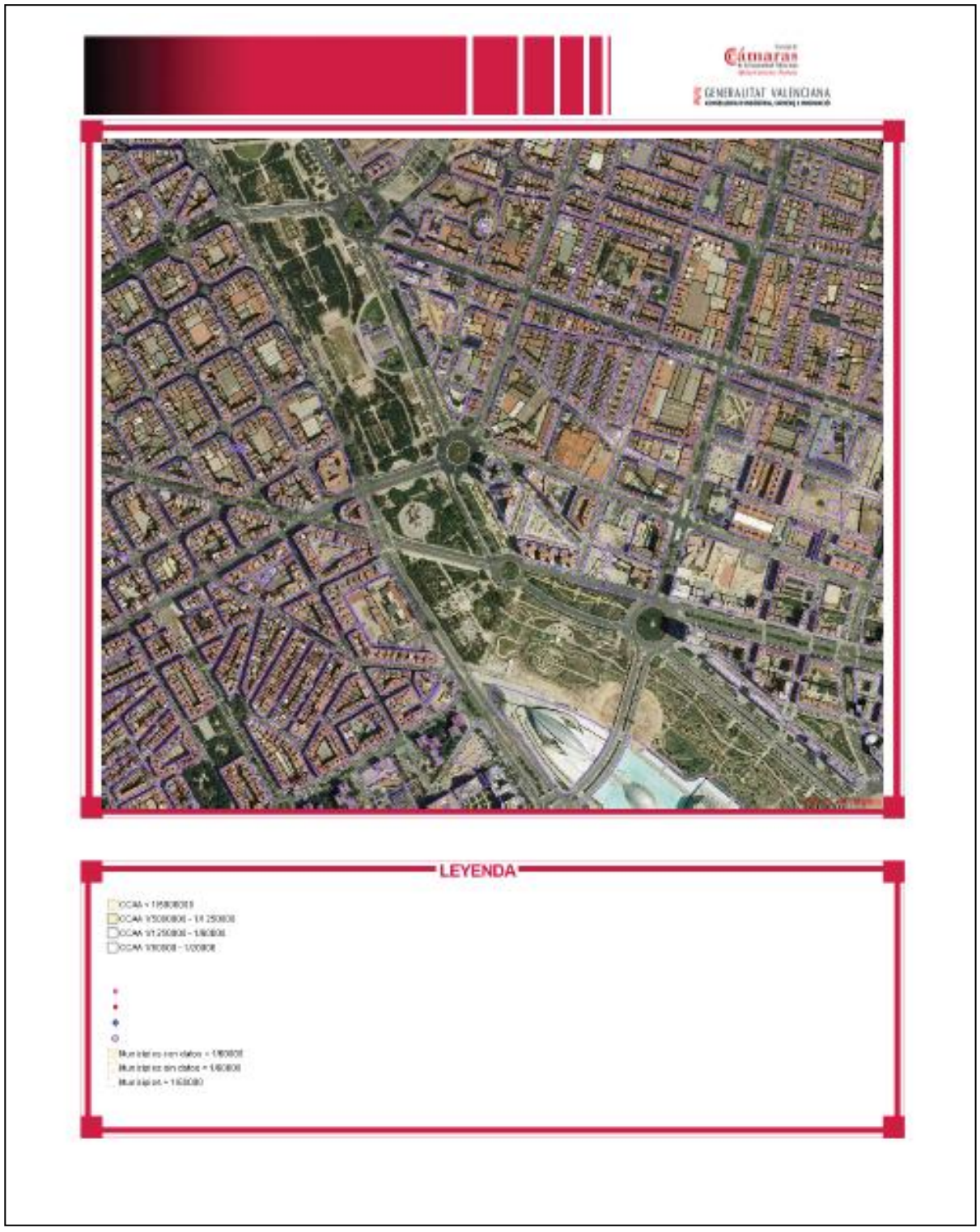

Figura 48. Detalle de impresión en formato pdf.

7. Anexo B

Manual de administración
Atlas Comercial
de la
Comunidad Valenciana

Arturo Argilés Casasús

Indice

1. Introducción

La aplicación de administración de la Infraestructura de Datos Espaciales del Atlas Comercial de la Comunidad Valenciana permite controlar y gestionar que cartografía se desea publicar en el visor de mapas

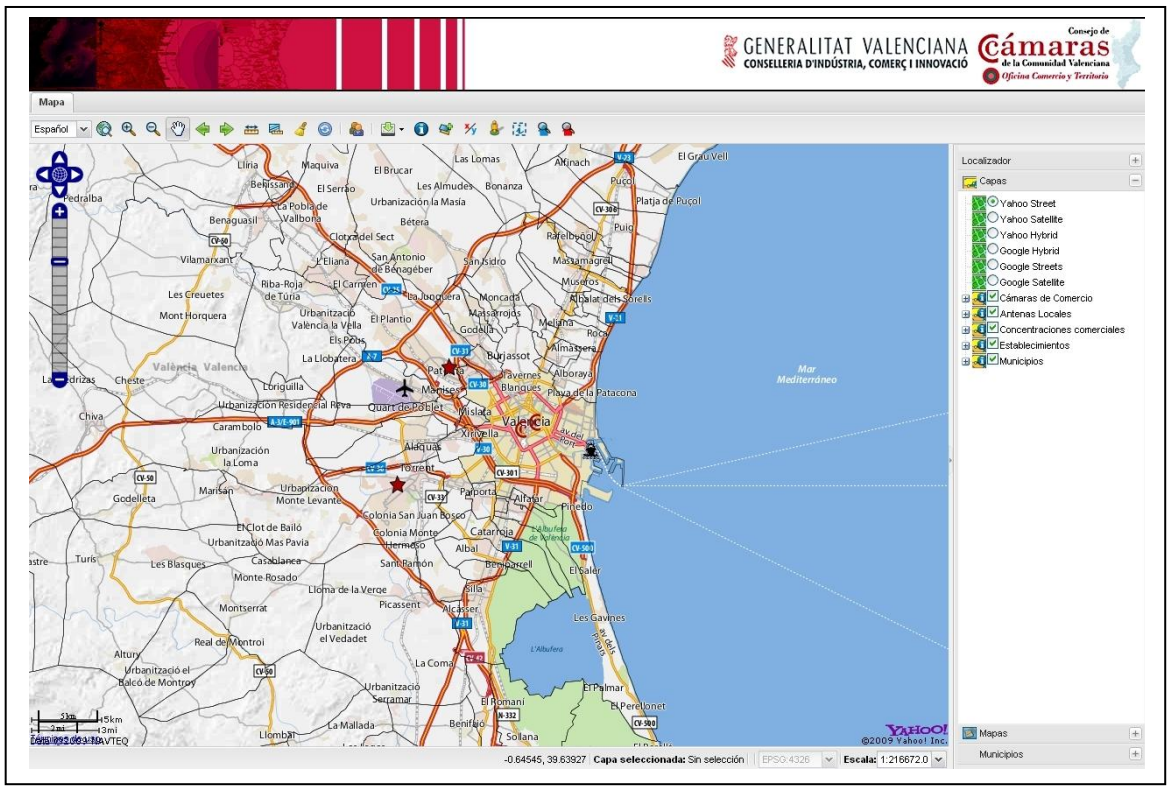

Figura 49. Detalle del visor web Atlas Comercial de la Comunidad Valenciana

Para la publicación de los datos cartográficos se utiliza el servicio de publicación de mapas estándar WMS (Web Map Server) que permite la visualización y consulta de información geográfica en remoto, produciendo mapas de datos espaciales referidos de forma dinámica a partir de información geográfica.

Para la publicación de estos servicios se dispone en el servidor de un servidor de mapas (Mapserver) y de su cartografía.

Los datos cartográficos (ficheros shape) se han almacenado de forma unificada en una carpeta del servidor de cartografía:

(Nombre de la unidad):\atlas\ortofoto

La aplicación dispone de un menú lateral situado en la parte izquierda que permite al usuario acceder a las distintas opciones, una parte central en la que se muestran los datos de la aplicación y en la que se muestra en la parte superior derecha el identificador del usuario que accede a la aplicación:

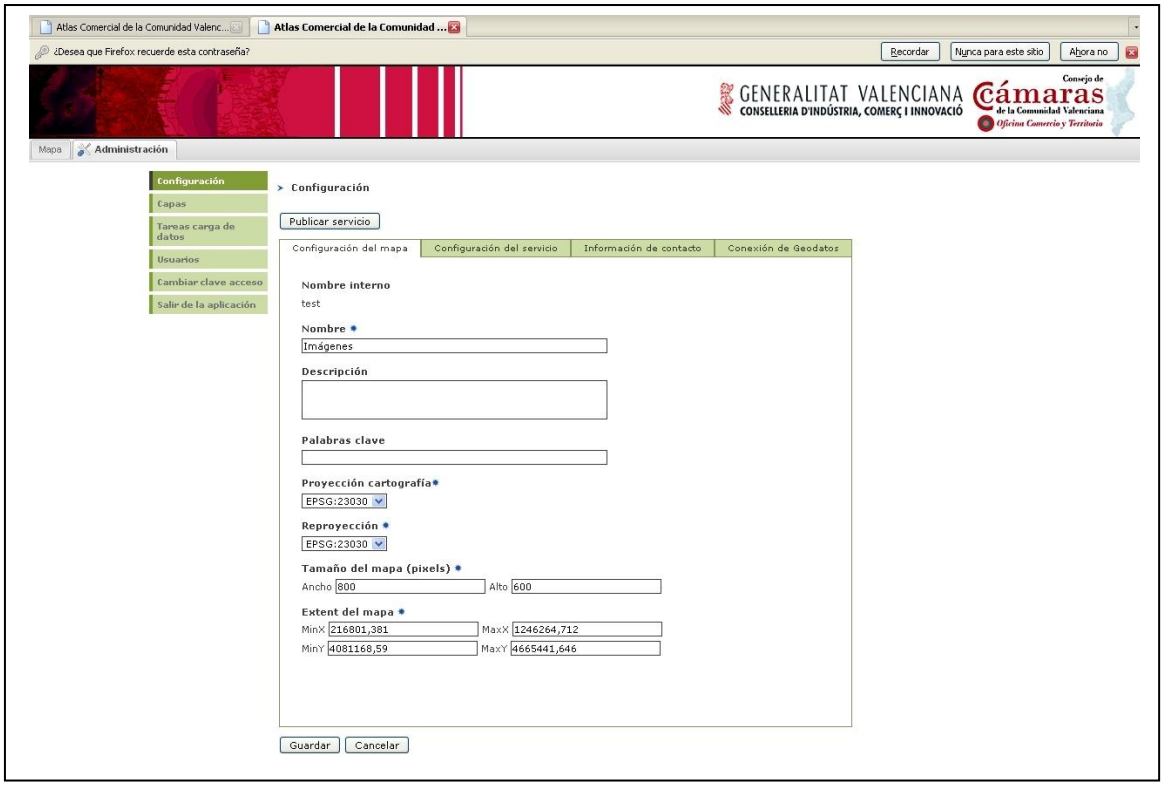

Figura 50. Vista del panel de Administración.

2. Convenciones de uso de la aplicación

La aplicación dispone de una serie de funcionalidades que son comunes en todas las pantallas para facilitar su funcionamiento a los usuarios:

Los datos que ha de rellenar el usuario de forma obligatoria se marcan con el siguiente símbolo ° junto a la etiqueta que identifica al dato:

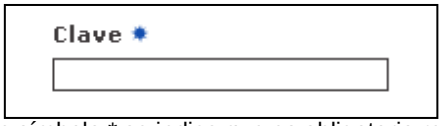

Figura 51. Con un símbolo * se indica que es obligatorio rellenar el campo

En los listados, haciendo clic en la cabecera de las columnas se puede reordenar por la columna seleccionada.

Figura 52. Detalle de cómo se puede reordenar una columna.

Al cancelar una pantalla, si se han modificado los datos se muestra una pantalla de confirmación:

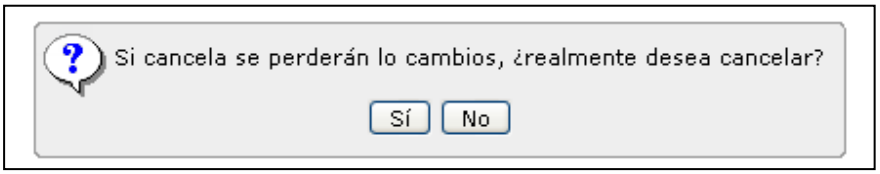

Figura 53. Esta ventana se muestra si se cancela una pantalla habiéndose modificado los datos.

Al eliminar algún dato se muestra una pantalla de confirmación:

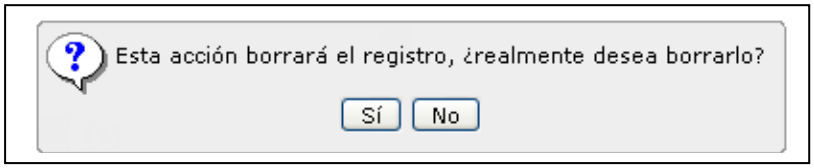

Figura 54. Ventana de confirmación al eliminar un dato.

3. Instalación de la aplicación

3.1. Requisitos previos

Para el correcto funcionamiento de la aplicación de administración se requiere tener instaladas las siguientes aplicaciones:

- Java 1.5
- Tomcat 5.5

3.2. Proceso de instalación

Para realizar la instalación de la aplicación hay que desplegar el fichero IDEAtlas_Admin.war en el servidor de aplicaciones Tomcat.

Para instalarla se pueden utilizar dos métodos: instalación manual o con la aplicación Manager de Tomcat. A continuación se detallan los pasos a realizar en ambos métodos.

3.3. Instalación manual:

Copiar en el directorio de aplicaciones de Tomcat (C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\webapps)

Reiniciar el servidor de aplicaciones para desplegar la aplicación, ejecutando la aplicación Inicio->Programas->Apache Tomcat 5.5->Monitor Tomcat:

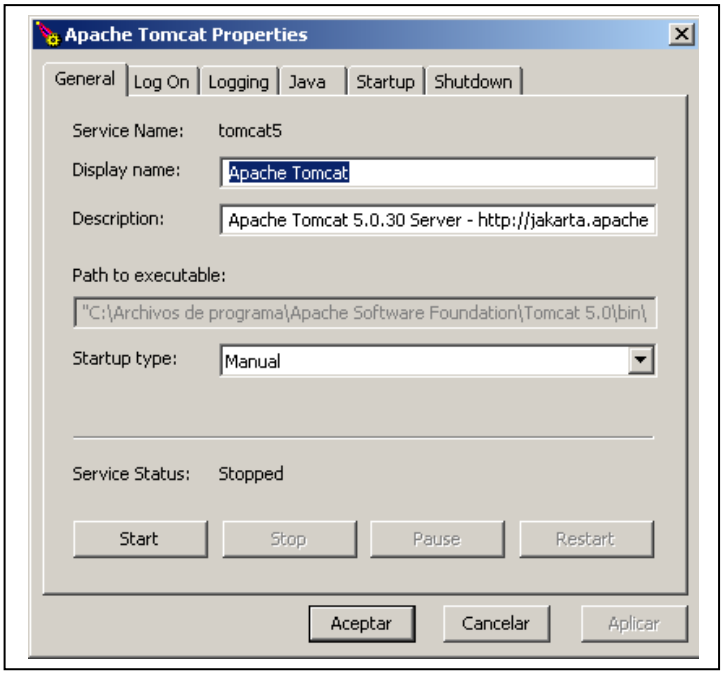

Figura 55. Ventana de propiedades de Apache Tomcat.

Instalación con la aplicación Manager de Tomcat. Tomcat dispone de una aplicación que permite gestionar las aplicaciones. Para acceder a la aplicación hay que introducir en un navegador web la siguiente URL:

http://SERVIDOR:PUERTO//manager/html

Al acceder a la aplicación se muestra una pantalla solicitando los datos del usuario.

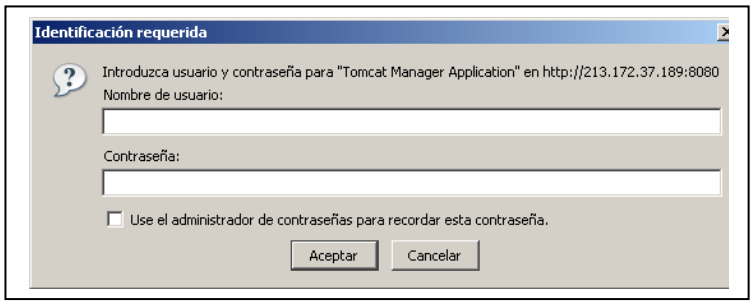

Figura 56. Ventana de identificación para acceder al Manager de Tomcat.

Los usuarios autorizados a acceder a esta aplicación se han de configurar en el fichero C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\conf\tomcat-users.xml. Este fichero tiene el siguiente formato:

```
<?xml version='1.0' encoding='utf-8'?>

<tomcat-users>

<role rolename="tomcat"/>

<role rolename="role1"/>

<role rolename="manager"/>

<role rolename="admin"/>

<user username="tomcat" password="aaaaaaa" roles="tomcat"/>

<user username="both" password="bbbbbbb" roles="tomcat,role1"/>

<user username="admin" password="ccccccc" roles="admin,manager"/>

</tomcat-users>
```

Si se quiere crear un usuario que pueda acceder a la aplicación de Manager se ha de crear una entrada de user con los datos del usuario, especificando su nombre y clave y que tenga el rol de manager:

```
<user username="usuario" password="clave" roles="manager"/>
```

Una vez validado el usuario se muestra la siguiente pantalla:

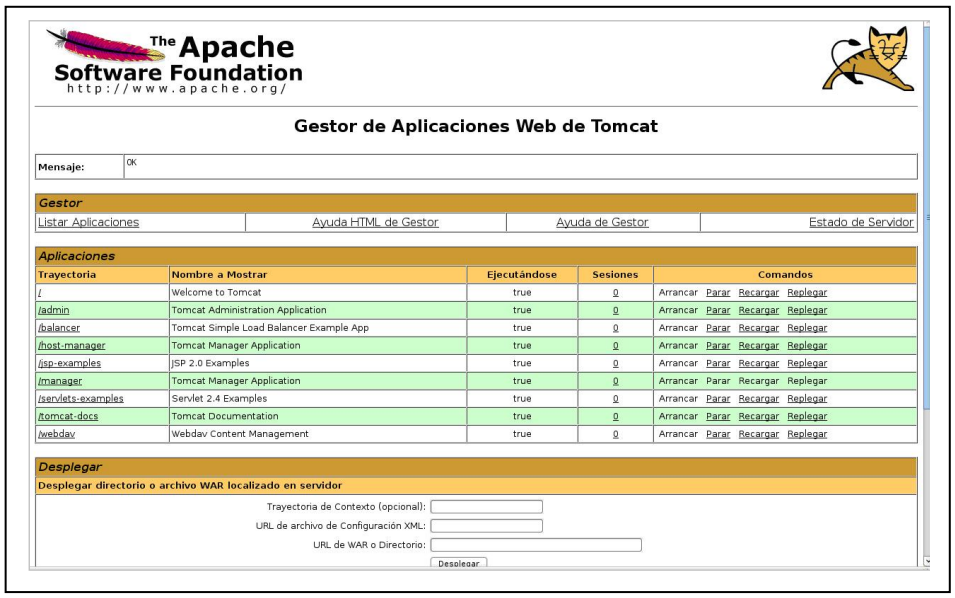

Figura 57. Gestor de aplicaciones Tomcat

En la que se pueden gestionar las aplicaciones instaladas en el servidor de aplicaciones e instalar (desplegar) nuevas aplicaciones.

Para desplegar la aplicación se ha de seleccionar el paquete con la aplicación IDEAdmin_Atlas.war, en la sección de Archivo WAR a desplegar con el botón Examinar:

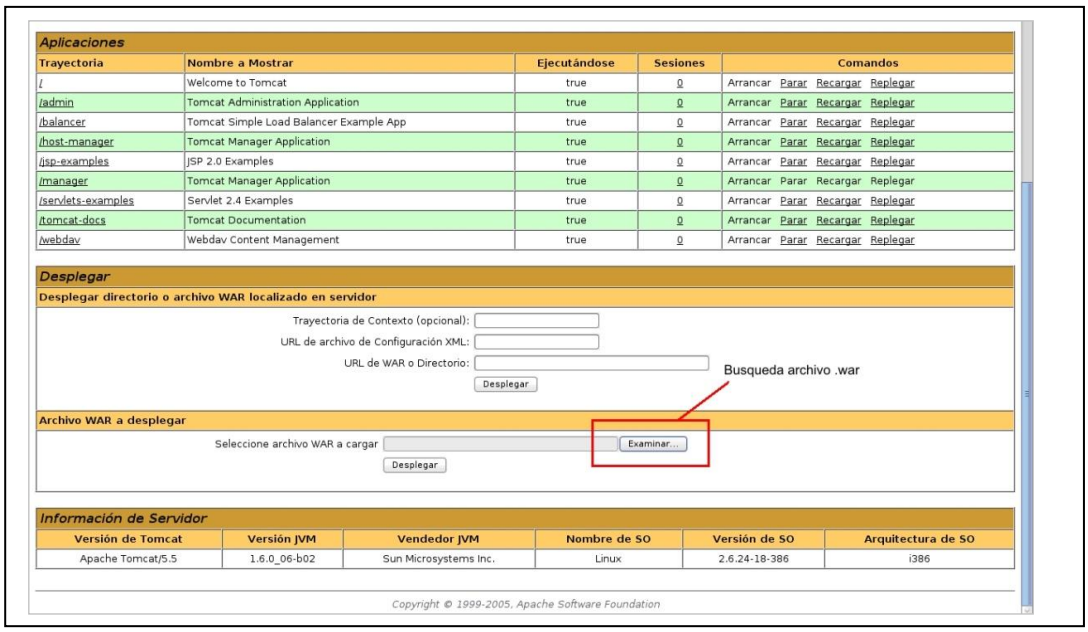

Figura 58. Debe seleccionarse el paquete con la aplicación IDEAdmin_Atlas.war

El fichero IDEAdmin_Atlas.war lo tiene que tener el usuario en su equipo, seleccionándolo del directorio en el que lo haya copiado. Una vez seleccionado, al pulsar el botón Desplegar, la aplicación se instala en el servidor de aplicaciones, apareciendo en el listado de aplicaciones.

Desde el listado de aplicaciones el usuario puede realizar las siguientes acciones con cada aplicación:

- Parar la aplicación.
- Recargar la aplicación, para actualizar la aplicación si se han modificado ficheros.
- Replegar: desinstalar la aplicación.

Al desplegarse la aplicación se crea un subdirectorio IDEAdmin con la aplicación en DIR_INSTALACION_TOMCAT/webapps, pudiendo acceder a ella con un navegador web con la siguiente URL:

http://SERVIDOR:PUERTO/IDEAdmin

en la que hay que sustituir SERVIDOR y PUERTO por el nombre del equipo y el puerto en el que está instalada la aplicación.

Antes de acceder a la aplicación hay que configurarla, como se describe en los siguientes apartados.

4. Configuración de la aplicación

La aplicación requiere que se configure la conexión con la base de datos y dirección del servidor de mapas que se utiliza para ubicar los puntos de interés en el plano.

3.5. Configurar la conexión a la base de datos

La configuración de la conexión de la base de datos se almacena en el fichero jdbc.properties que está en el directorio:

DIR_INSTALACION_TOMCAT\webapps\IDEAdmin\WEB-INF

En el fichero hay que configurar el nombre de la base de datos y los datos del usuario de la base de datos a utilizar para realizar la conexión:

jdbc.url=jdbc:postgresql://localhost:PUERTO/NOMBRE_BASE_DATOS

jdbc.username=NOMBRE_USUARIO_BASE_DATOS

jdbc.password=CLAVE_USUARIO_BASE_DATOS

dataloader.path=C:/Archivos de programa/Apache Software Foundation/Tomcat 5.5/webapps/IDEAdmin_Atlas/carga_datos

dataloader.period=600

dataloader.shp2pgsql=C:/Archivos de

programa/PostgreSQL/8.2/bin/shp2pgsql.exe

El atributo dataloader.path hace referencia al directorio del disco en el que se van a almacenar temporalmente las tareas de carga de datos. El atributo dataloader.period indica los segundos que van entre ejecución de las tareas de carga pendientes. La propiedad dataloader.shp2pgsql indica la ruta de disco donde está instalada la aplicación shp2pgsql.

Una vez modificado el fichero hay que actualizar la aplicación, bien reiniciando Tomcat o recargando la aplicación con el Manager de Tomcat, como se ha explicado en el apartado de instalación de la aplicación.

3.6. Configurar ruta visor

La configuración del servidor de mapas se establece en el fichero applicationContext.xml que está en el directorio:

DIR_INSTALACION_TOMCAT\webapps\IDEAdmin_Atlas\WEB-INF

En el fichero hay que configurar el directorio en el que está instalados los ficheros de configuración del visor de mapas el servidor de mapas, en la siguiente sección:

```
<bean id="servletManagerTarget"
class="org.iver.ide.admin.domain.manager.ServletManagerImpl">

<property name="mapBuilderWritePath">

<value>DIR_INSTALACION_TOMCAT/webapps/IDEAtlas/data/context</value>

</property>

<property name="mapBuilderReadPath">

    <value>data/context</value>

</property>

</bean>
```

El atributo mapBuilderWritePath indica el directorio de disco donde se guardan los context de MapBuilder mientras que la propiedad mapBuilderReadPath indica la parte del path de la propiedad anterior que se encuentra dentro del visor (IDEAtlas).

Una vez modificado el fichero hay que actualizar la aplicación, bien reiniciando Tomcat o recargando la aplicación con el Manager de Tomcat, como se ha explicado en el apartado de instalación de la aplicación.

7. Acceso a la aplicación

La aplicación de administración de la Infraestructura de Datos Espaciales funciona en un navegador web (Internet Explorer 6.0+, Mozilla Firefox 1.0+). Para acceder a ella se ha de introducir la siguiente URL:

http:// SERVIDOR:PUERTO /IDEAdmin/login.jsp

El usuario se ha de validar antes de poder acceder a la aplicación, introduciendo su nombre de usuario, la clave de acceso y el servicio de mapas que quiere gestionar:

Figura 59. Validación de usuario de acceso a la aplicación.

Una vez validado el usuario se accede a la pantalla de configuración general del servicio de mapas.

Cada servicio dispone de un usuario administrador por defecto llamado ******[9] y con la clave ******. Lo primero que ha de hacer cada usuario administrador al entrar en la aplicación es modificar esta clave.

También es posible entrar a la aplicación con el rol Administrador, y este es el único perfil que nos permite crear nuevos servicios. El usuario por defecto para este rol es ****** con la clave ******. Una vez dentro como administrador podremos cambiar la contraseña.

8. Gestión de servicios

En este apartado definiremos las diferentes operaciones que podemos realizar con los servicios de mapas, tales como crear, eliminar y modificar.

8.1 Creación de servicios

Accederemos a la aplicación con el rol Administrador tal y como vemos en la siguiente imagen:

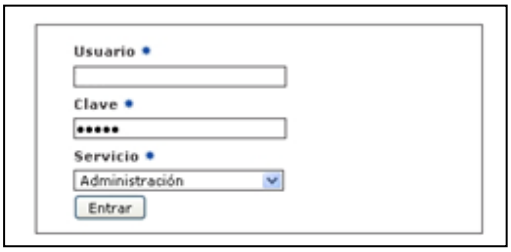

Figura 60. Validación de usuario de acceso a la aplicación como administrador.

[9] Confidencial. En este documento estos datos no se han mostrado por motivos de seguridad.

Nos encontraremos con un listado de los servicios ya creados anteriormente, a los que podemos acceder y modificar sus parámetros. También podemos crear nuevos servicios gracias al botón “Crear servicio”:

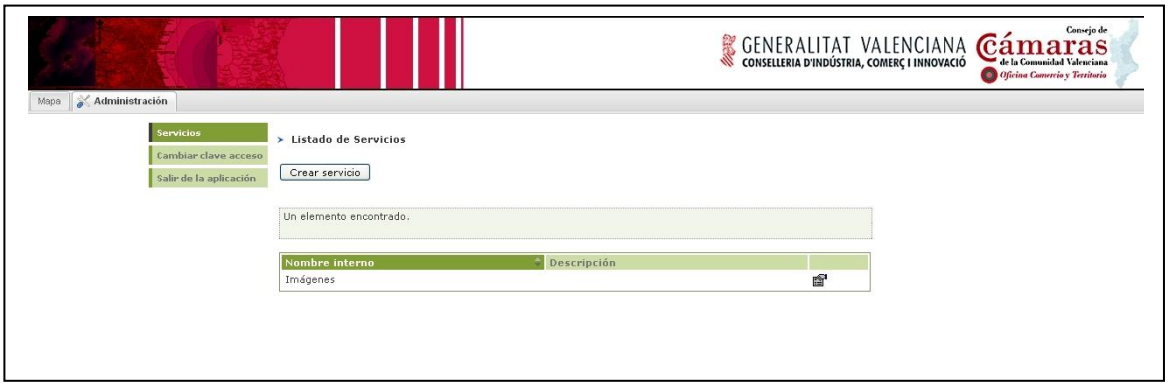

Figura 61. Listado de los servicios que ya han sido creados.

Si creamos un nuevo servicio, accederemos a la pantalla de configuración, donde se nos pedirán los datos que nos definen el nuevo servicio de mapas:

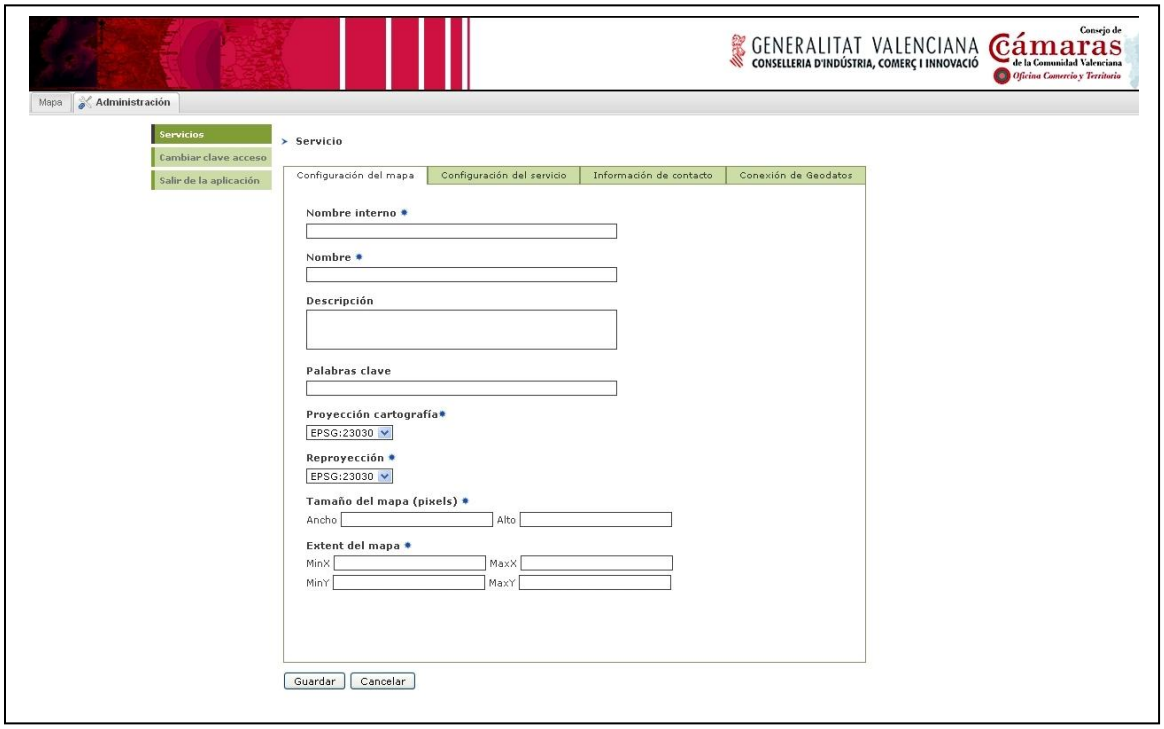

Figura 62. Detalle de creación de un nuevo servicio.

Los datos que nos piden son los siguientes:

- Nombre interno: nombre con el que se guarda el servicio.

- Nombre: nombre que se muestra al usuario.
- Descripción: breve descripción del servicio.
- Palabras clave: palabras para la búsqueda del servicio.
- Proyección cartografía: proyección en la que se encuentra los datos cartográficos.
- Reproyección: proyección en la que se representará la cartografía.
- Extent del mapa: coordenadas cartográficas de las esquinas inferior izquierda y superior derecha de la zona que queremos representar.

Una vez guardado el servicio, nos aparecerá en el listado visto anteriormente.

8.2 Eliminación de servicios

Esta opción sólo se encuentra disponible para el rol Administrador, tal y como ocurre con la creación de servicios. Eliminaremos un servicio ya creado con el botón “Eliminar”, disponible en la opción de Configuración.

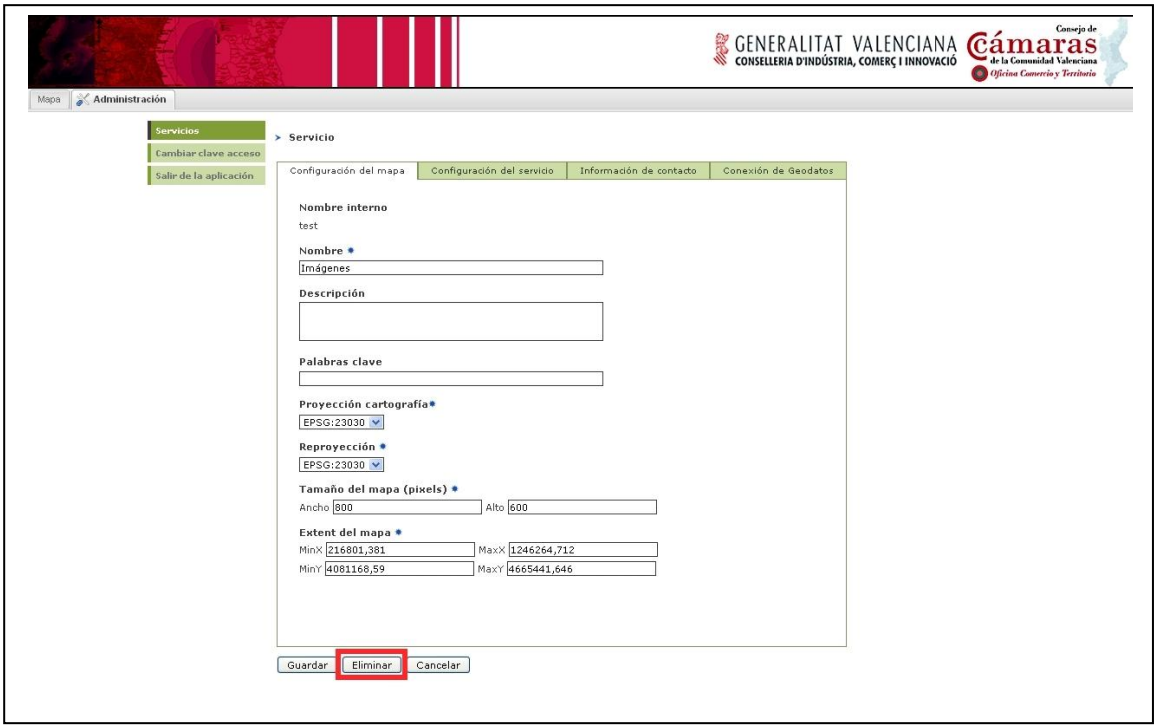

Figura 63. Detalle de eliminación de un servicio.

Una vez eliminado, el servicio dejará de aparecer en el listado de servicios disponibles.

8.3 Configuración de servicios

Al acceder a la opción de Configuración en el menú de la aplicación se pueden configurar los datos del servicio de mapa. Esta opción sólo está disponible para los usuarios de perfil administrador. Tenemos cuatro pestañas donde podemos definir diferentes elementos.

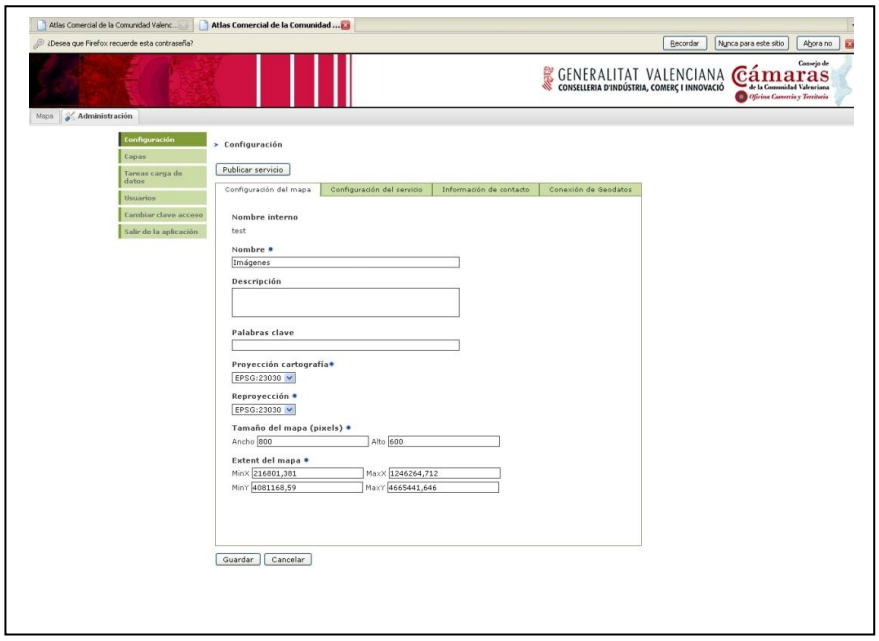

Figura 64. Configuración de servicios.

Los parámetros a configurar en la pantalla de configuración del mapa son los mismos que los listados para la creación de nuevos servicios, excepto por no estar disponible el campo de Nombre interno:

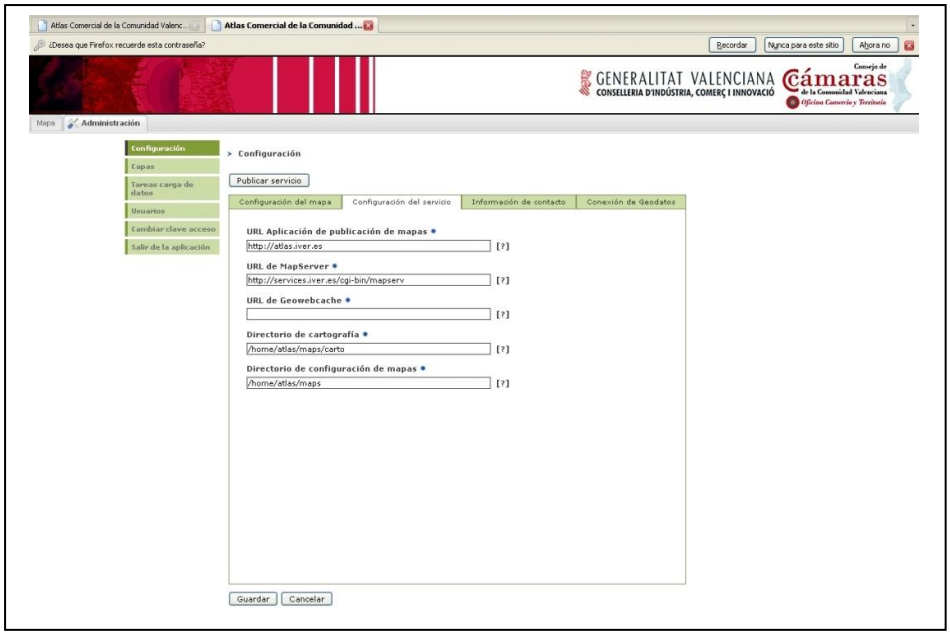

Figura 65. Se pueden configurar todos los datos de un servicio excepto el nombre interno.

Los parámetros a configurar en la pantalla de configuración del servicio son:

- URL Aplicación de publicación de mapas: URL del servidor Tomcat del servidor de cartografía.
- URL Servidor de mapas: URL del servidor MapServer del servidor de cartografía.
- Directorio de cartografía: lugar donde están los datos cartográficos a cargar (imágenes, shapes, etc) en el servidor de cartografía.
- Directorio de configuración de mapas: lugar donde se guardarán los archivos de configuración de los mapas.

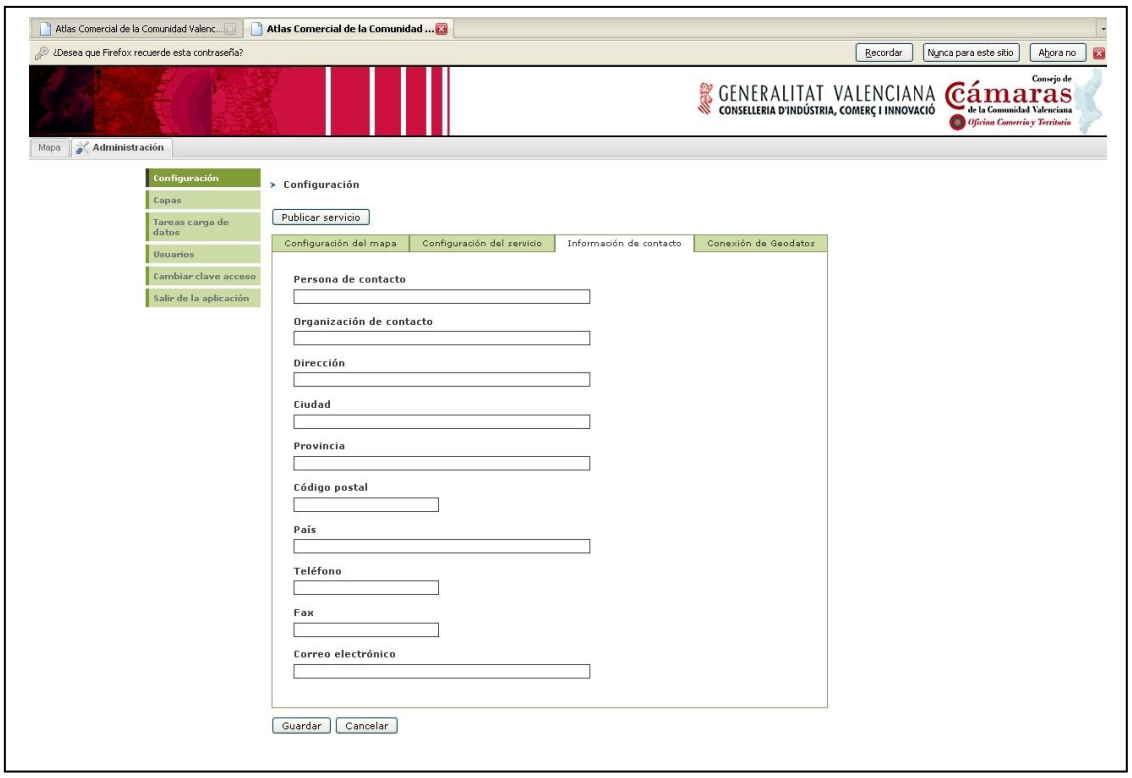

Figura 66. Detalle de configuración de servicio.

Los parámetros a configurar en la pantalla de Información de contacto son diferentes datos e información de la persona al cargo de la aplicación, tales como nombre, cargo, dirección, teléfono, correo electrónico, etc.

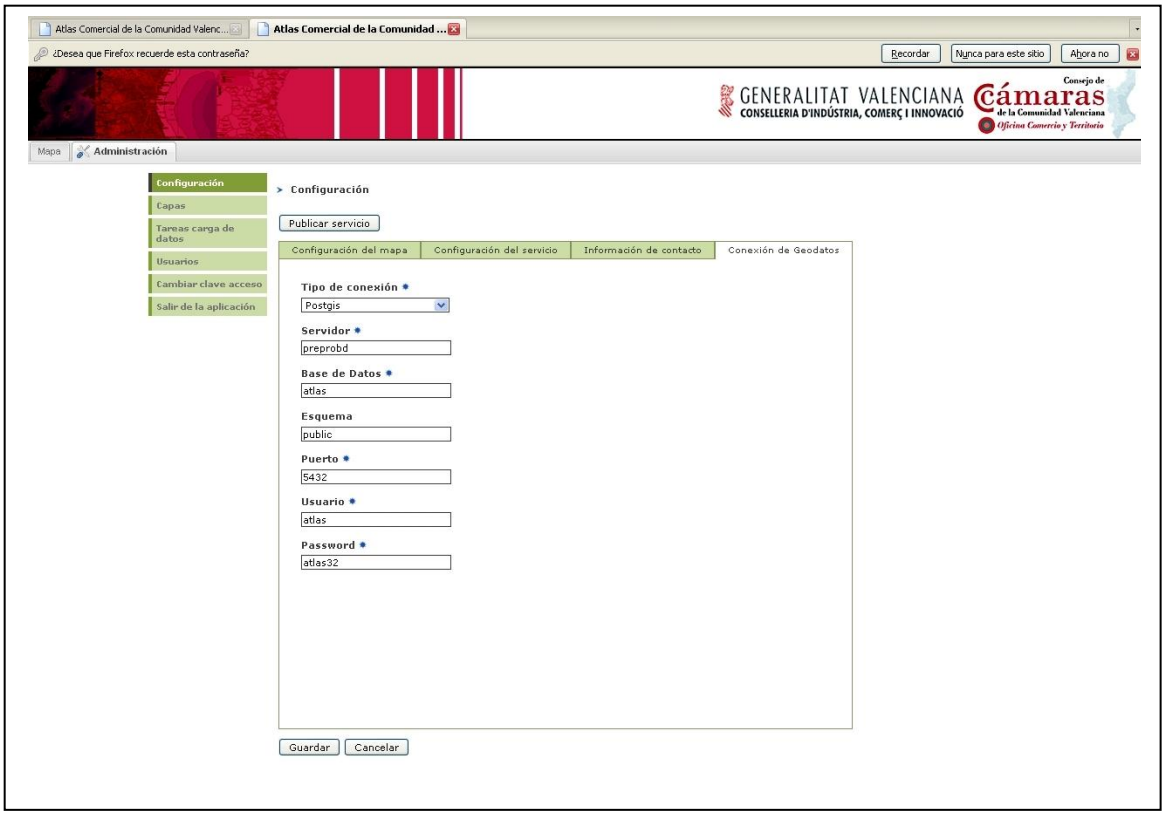

Figura 67. Detalle de pantalla de conexión de datos.

Los parámetros a configurar en la pantalla de conexión de geodatos son:

- Tipo de conexión: tipo de conexión con la base de geodatos.
- Servidor: dirección IP del servidor donde se encuentra la base de datos.
- Base de datos: nombre de la base de datos.
- Esquema: esquema donde se encuentran las tablas con los datos.
- Puerto: puerto a través del que se hace la conexión.
- Usuario: nombre del usuario de la base de datos.
- Password: contraseña del usuario de la base de datos.

8.4. Publicación del servicio de mapas

Al modificar la configuración del servicio o de las capas que componen el servicio la opción de Publicar servicio permite actualizar la configuración en el servidor de mapas para reflejar los cambios realizados en el visor de mapas.

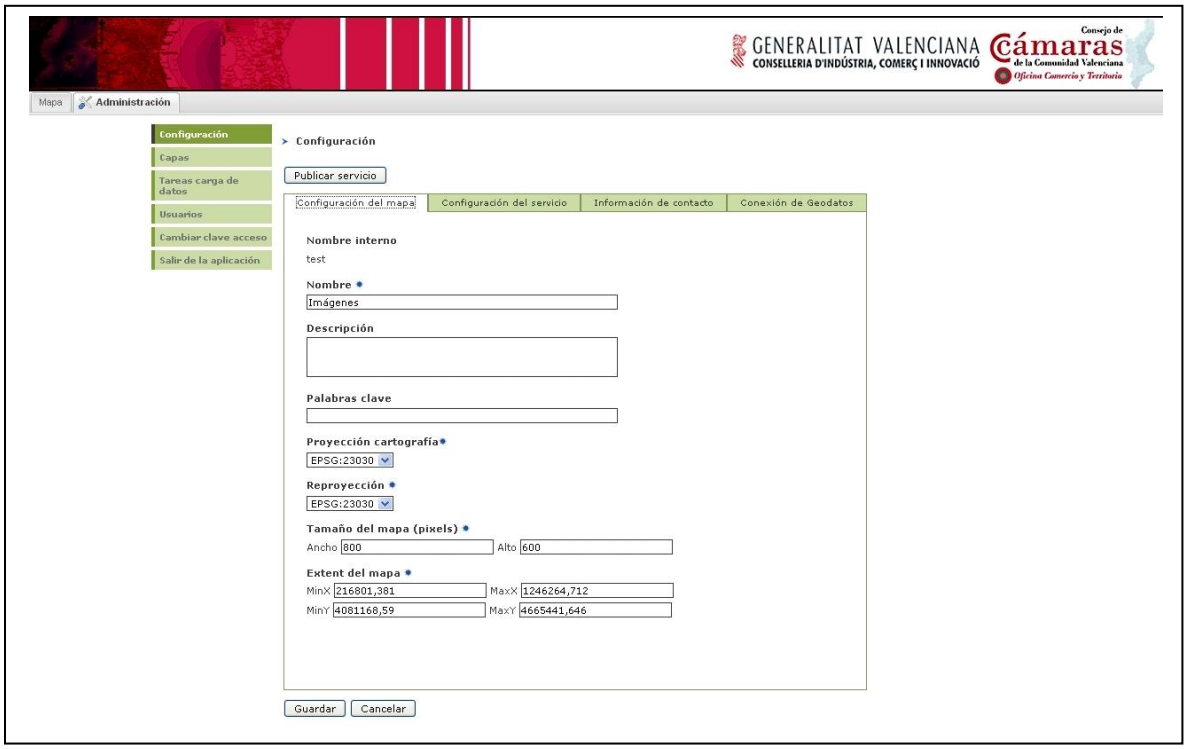

Figura 68. Configuración del mapa.

9. Carga de datos

Con esta herramienta podremos cargar nuevos datos en la base de datos. Esta herramienta permite cargar datos en formato shape, y en formato csv. Tenemos esta herramienta en el menú de la aplicación, en la parte superior izquierda. Una vez dentro, tendremos que crear una nueva tarea.

Figura 69. Carga de datos.

Lo que nos llevará a la siguiente pantalla, donde deberemos definir los parámetros de la tarea de carga de datos que queremos realizar:

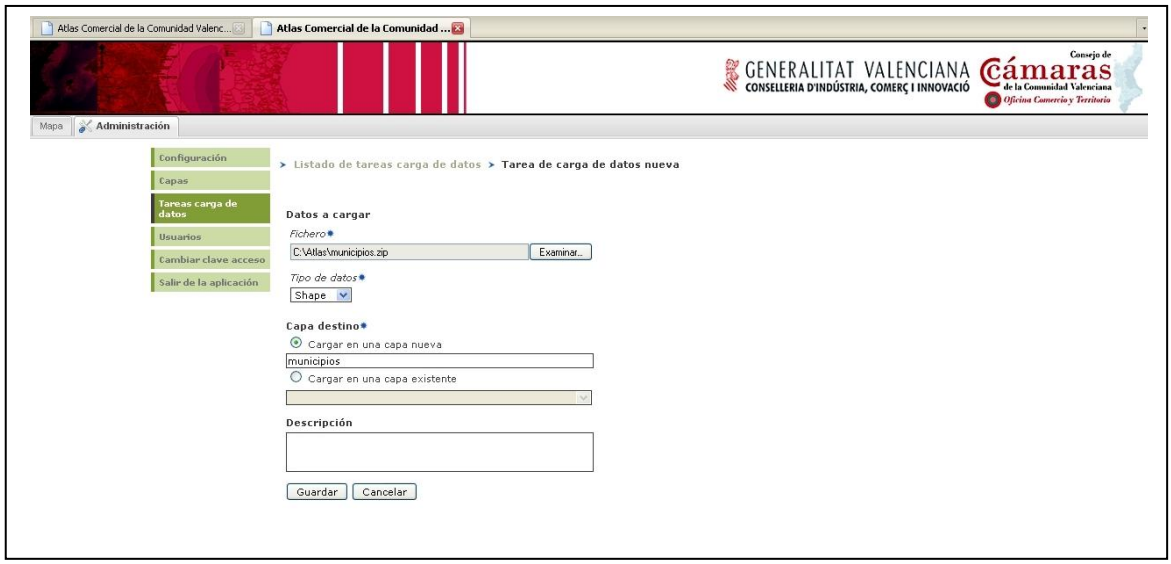

Figura 70.Debemos definir los parámetros de la tarea de carga que queremos realizar.

Los parámetros que deberemos rellenar serán los siguientes:

- Fichero: fichero que queremos cargar. En el caso del formato shape, tendremos que comprimir los ficheros necesarios (dbf, shp y shx) en un fichero zip, que es el que indicaremos en la herramienta. En el caso del formato csv, esto no es necesario.
- Tipo de datos: tipo de fichero a cargar (shape/csv).
- Capa de destino: elegimos si queremos crear una tabla nueva (indicando el nombre), o bien lo cargamos en una existente. Hay que tener en cuenta que si elegimos una tabla existente, la carga de los datos eliminará los datos anteriormente recogidos en la tabla.
- Descripción: breve descripción de los datos.

Una vez creada la tarea, se nos aparecerá listada, así como su descripción, su estado, y el resultado obtenido al finalizar el proceso.

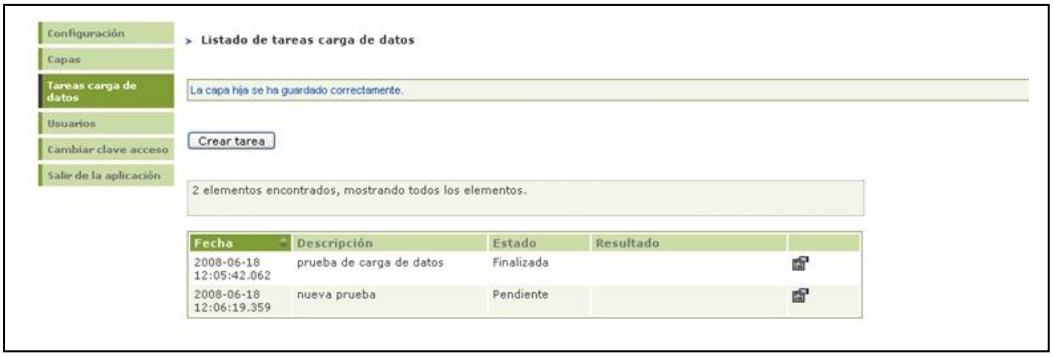

Figura 71. La tarea creada aparece listada con nombre, descripción y estado

Se deberá tener en cuenta que en el caso de la carga de datos shape, no tendrán que aparecer en el nombre de los campos caracteres no anglosajones, tales como eñes y tildes.

10. Gestión de capas

Al acceder a la opción de Capas en el menú de la aplicación se muestra un listado con las capas del servicio en el que nos encontramos:

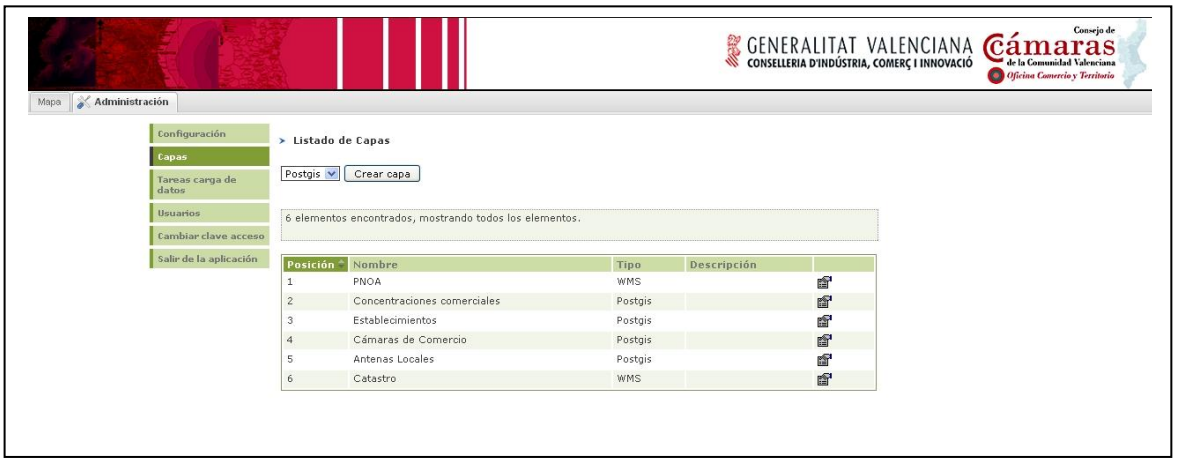

Figura 72. Listado con las capas del servicio.

Desde este listado el usuario puede crear nuevas capas y modificar las propiedades de las capas que tiene cargadas el servicio.

10.1. Crear una capa

Para crear una capa el usuario ha de seleccionar desde el listado de capas el tipo de capa (Vectorial, Raster, Tile, WMS), y hacer clic en el botón de Crear capa.

Figura 73. Selección del tipo de capa.

10.1.1 Capa vectorial

Al crear una capa en formato vectorial (shape), se muestra la siguiente pantalla:

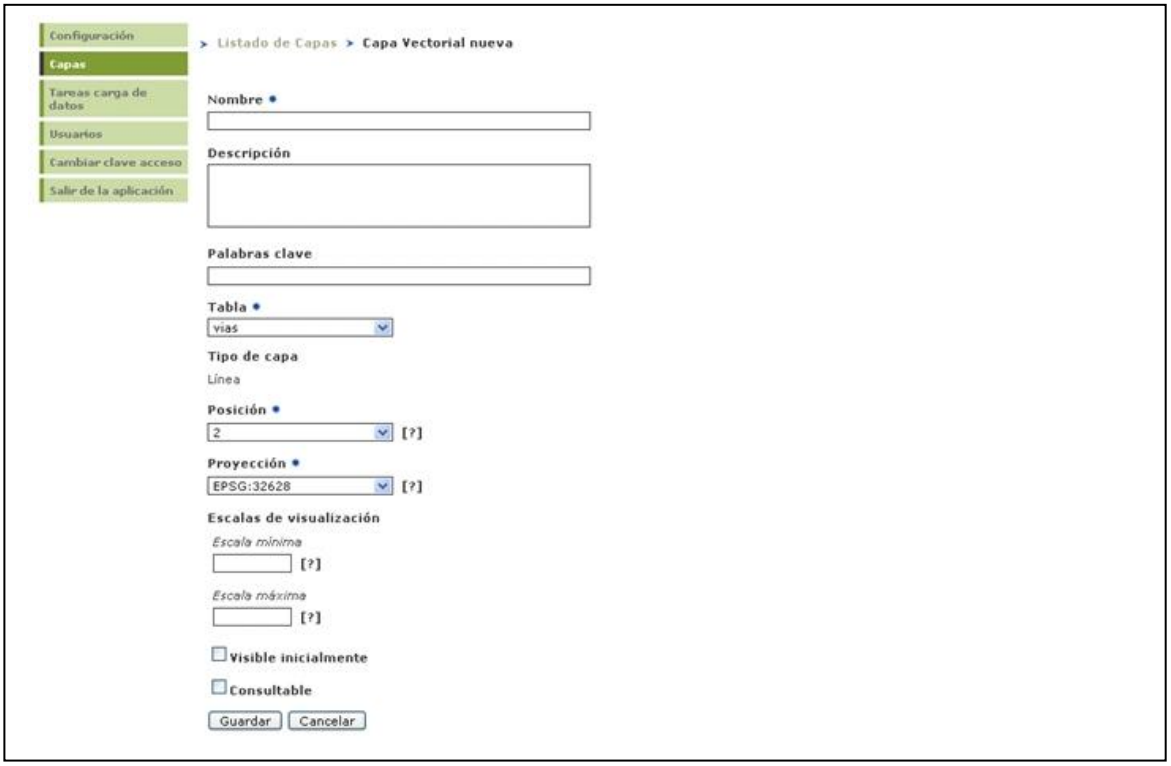

Figura 74. Detalle de creación de una capa vectorial.

En esta pantalla el usuario ha de rellenar los datos asociados a la capa:

- Nombre: nombre de la capa.
- Descripción: breve descripción de los datos.
- Palabras clave: palabras clave para la búsqueda de la capa.
- Tabla: tabla donde están los geodatos a cargar. Este listado mostrará las tablas de la base de datos a la que esta conectada el servicio en el que nos encontramos.
- Posición: posición en la leyenda. Por defecto se colocará la última.
- Proyección: proyección en la que se encuentra la cartografía.
- Escala mínima de visualización: por debajo de esta escala no se visualizará el mapa.
- Escala máxima de visualización: por encima de esta escala no se visualizará el mapa.
- Visible inicialmente: si se desea que se vea la capa al cargar el servicio.

- Consultable: si queremos o no que sean consultables sus datos alfanuméricos. Hay que tener en cuenta que se podrán consultar todos los campos que contenga la capa.

El raster debe encontrarse en el directorio de cartografía que tiene definido el servicio en el que nos encontramos. Una vez creada la capa el usuario puede acceder con el botón a la configuración de la simbología de visualización de la capa.

10.1.2 Capa raster

Al crear una capa en formato raster se muestra la siguiente pantalla:

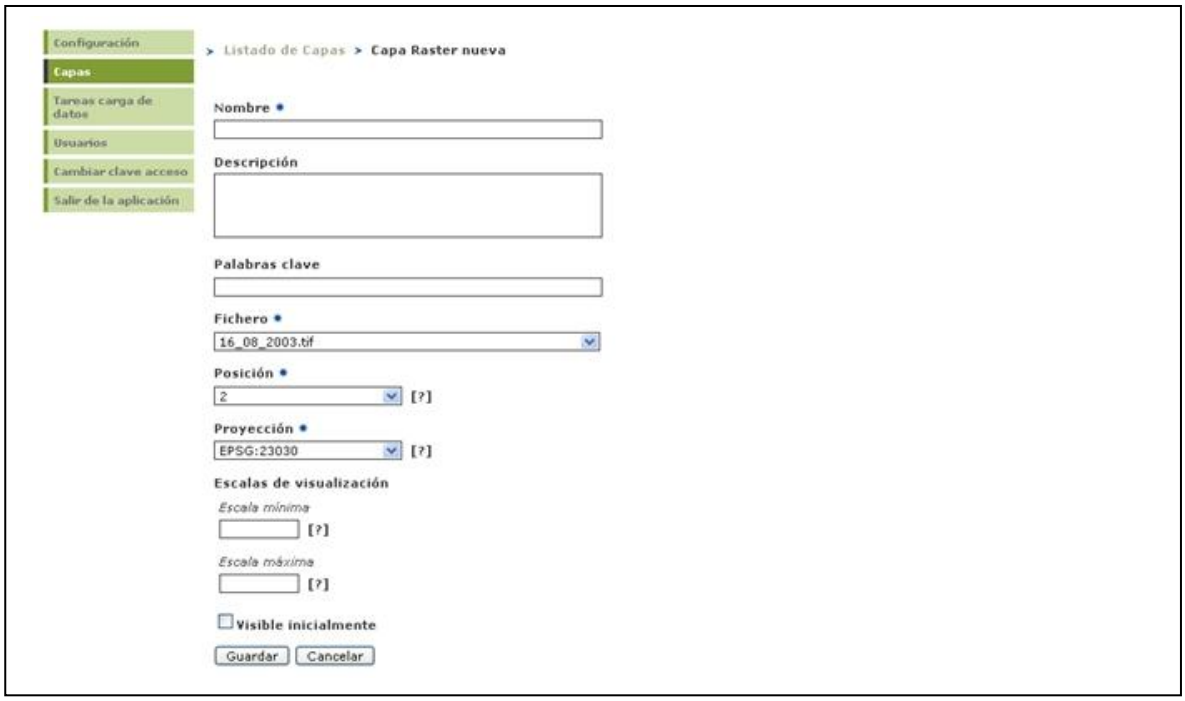

Figura 75. Detalle de creación de una capa raster.

En esta pantalla el usuario ha de rellenar los datos asociados a la capa:

- Nombre: nombre de la capa.
- Descripción: breve descripción de los datos.
- Palabras clave: palabras clave para la búsqueda de la capa.
- Fichero: imagen que queremos visualizar. Se muestra un listado con las capas en formato raster de las que se en el servidor de

cartografía para que el usuario seleccione la capa a visualizar en el mapa.

- Posición: posición en la leyenda. Por defecto se colocará la última.
- Proyección: proyección en la que se encuentra la cartografía.
- Escala mínima de visualización: por debajo de esta escala no se visualizará el mapa.
- Escala máxima de visualización: por encima de esta escala no se visualizará el mapa.
- Visible inicialmente: si se desea que se vea la capa al cargar el servicio.

10.1.3 Capa WMS

Al crear una capa conectando con otro servicio WMS se muestra la siguiente pantalla:

Figura 76. Detalle de creación de una capa conectando con otro servicio WMS.

Donde tendremos que escribir el servidor de cartografía al que nos queremos conectar. En este caso, se trato del servidor del Catastro.

Una vez conectado con el servidor de cartografía externo, nos aparecerá un listado con las capas disponibles, que podremos ir seleccionando una por una, o con las herramientas básicas de “Seleccionar todas” o “Deseleccionar todas”.

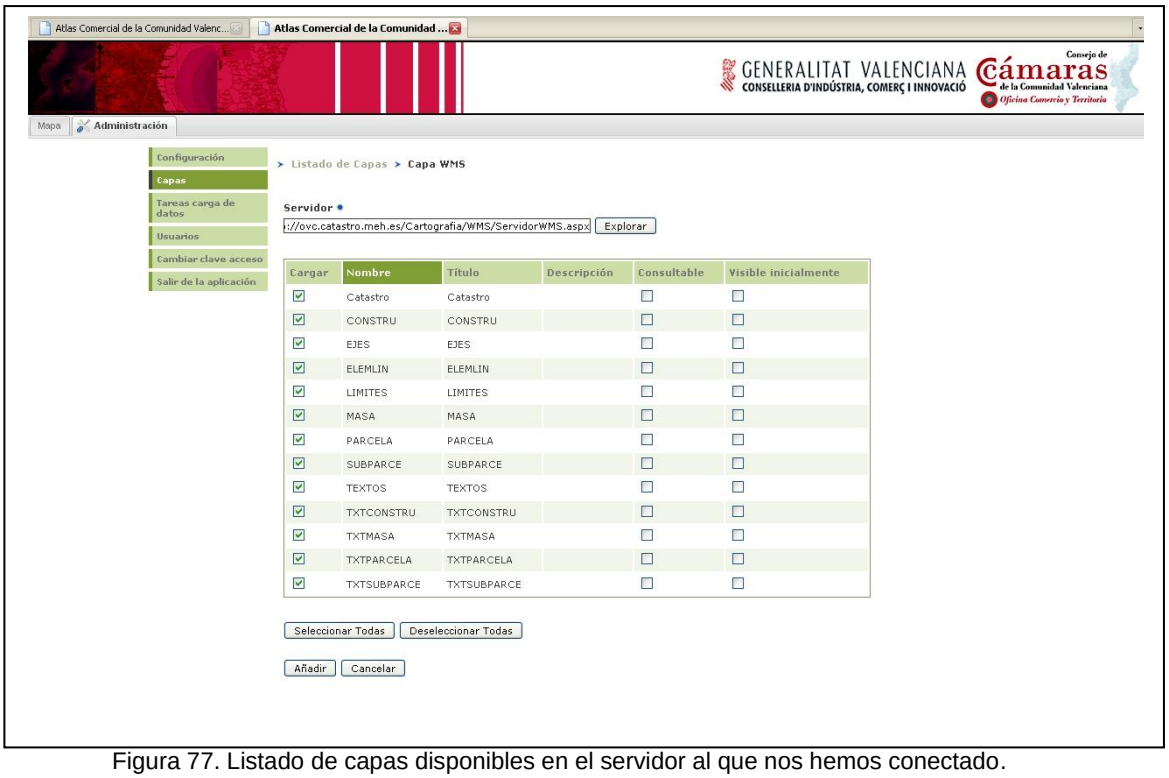

Figura 77. Listado de capas disponibles en el servidor al que nos hemos conectado.

En el listado de capas nos aparecerán todas las que hayamos seleccionado del servicio WMS.

10.1.4 Capa Tile

Un Tile es un mosaico de imágenes que toma como referencia un shape en el que se indica, mediante polígonos, la posición de cada una de las imágenes, siendo uno de los campos de la tabla asociada la dirección en la que se encuentra la imagen que va asociada a ese polígono.

El shape debe encontrarse en el directorio de cartografía que tiene definido el servicio en el que nos encontramos.

Al crear una capa de tipo Tile, se muestra la siguiente pantalla:

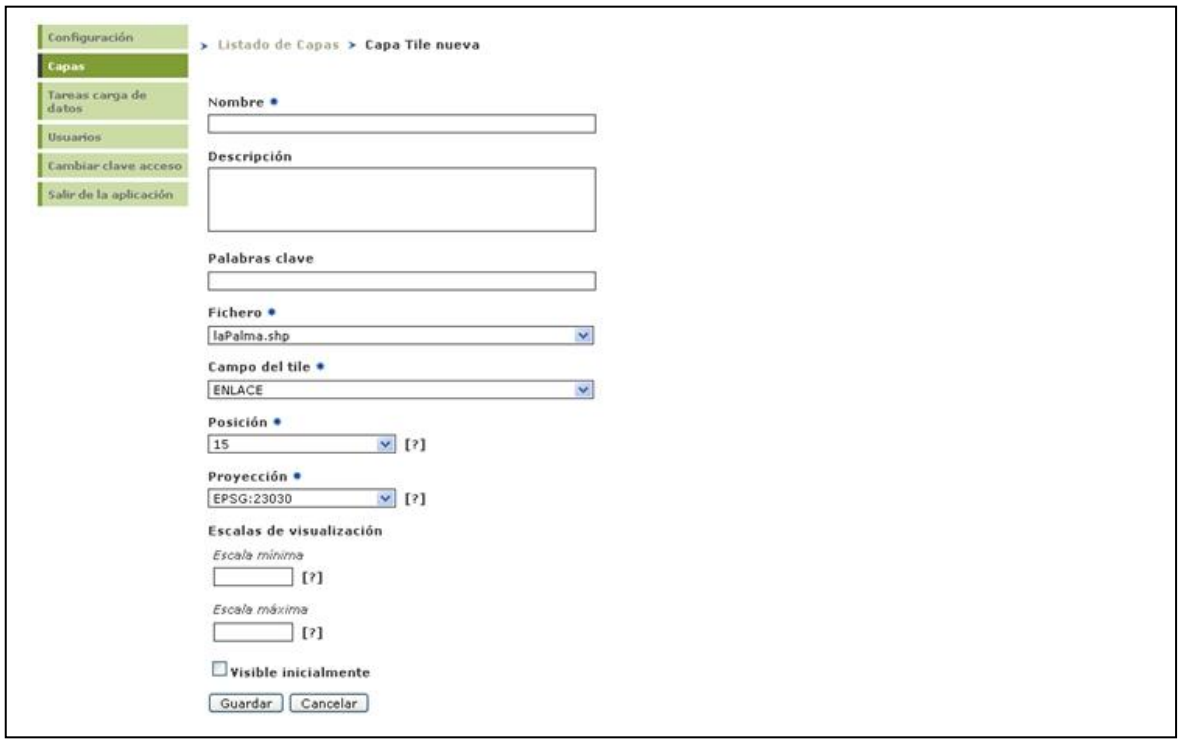

Figura 78. Detalle de creación de una capa Tile.

En esta pantalla el usuario ha de rellenar los datos asociados a la capa:

- Nombre: nombre de la capa.
- Descripción: breve descripción de los datos.
- Palabras clave: palabras clave para la búsqueda de la capa.
- Fichero: fichero shape de definición del tile. Se muestra un listado con las capas en formato shape de las que se en el servidor de cartografía para que el usuario seleccione la capa a visualizar en el mapa.
- Campo del tile: campo del shape que indica la dirección de la imagen a cargar.
- Posición: posición en la leyenda. Por defecto se colocará la última.
- Proyección: proyección en la que se encuentra la cartografía.
- Escala mínima de visualización: por debajo de esta escala no se visualizará el mapa.
- Escala máxima de visualización: por encima de esta escala no se visualizará el mapa.

- Visible inicialmente: si se desea que se vea la capa al cargar el servicio.

10.2. Modificar una capa

Esta opción es análoga a la pantalla para crear una capa, permitiendo:

- Editar los datos de la configuración de la capa.
- Acceder a la leyenda de la capa.
- Eliminar la capa para que no se visualice en el servicio de mapas. Esto sólo quita la capa del servicio de mapas no borrando los ficheros de cartografía asociados.

La única diferencia la encontramos en las capas de tipo WMS, que nos encontramos con una pantalla como la que sigue:

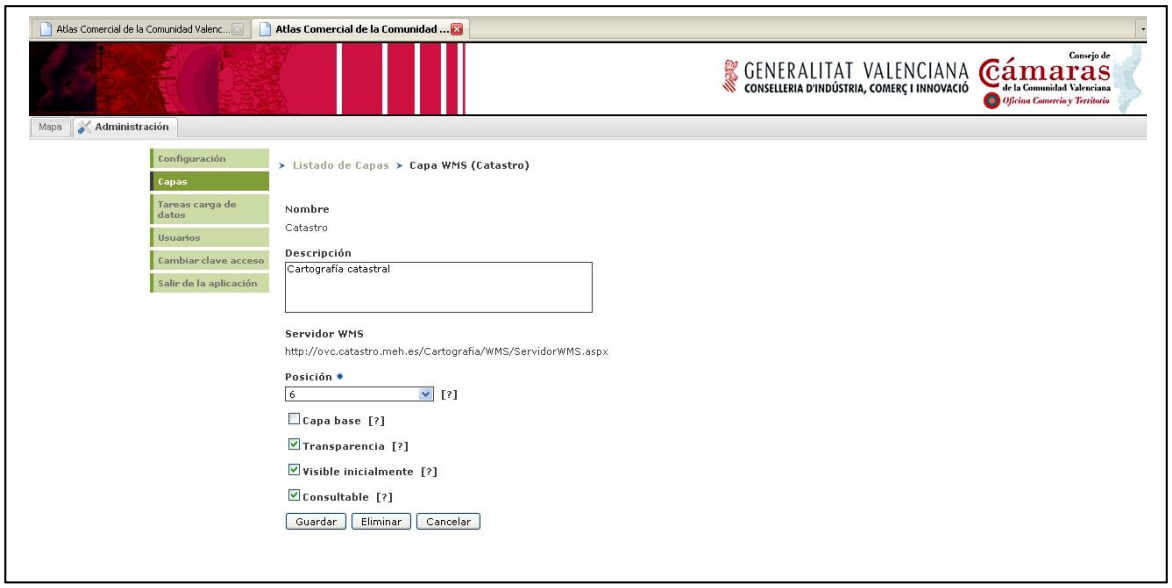

Figura 79. Capa de tipo WMS.

En la que sólo podemos modificar la descripción, la posición en la leyenda, y si queremos que se visualice al cargar el mapa o no.

10.3. Eliminar una capa

Una vez creadas las capas, si accedemos a ellas para modificar sus características tendremos habilitado un botón que nos permitirá su eliminación, tal y como vemos en la siguiente pantalla.

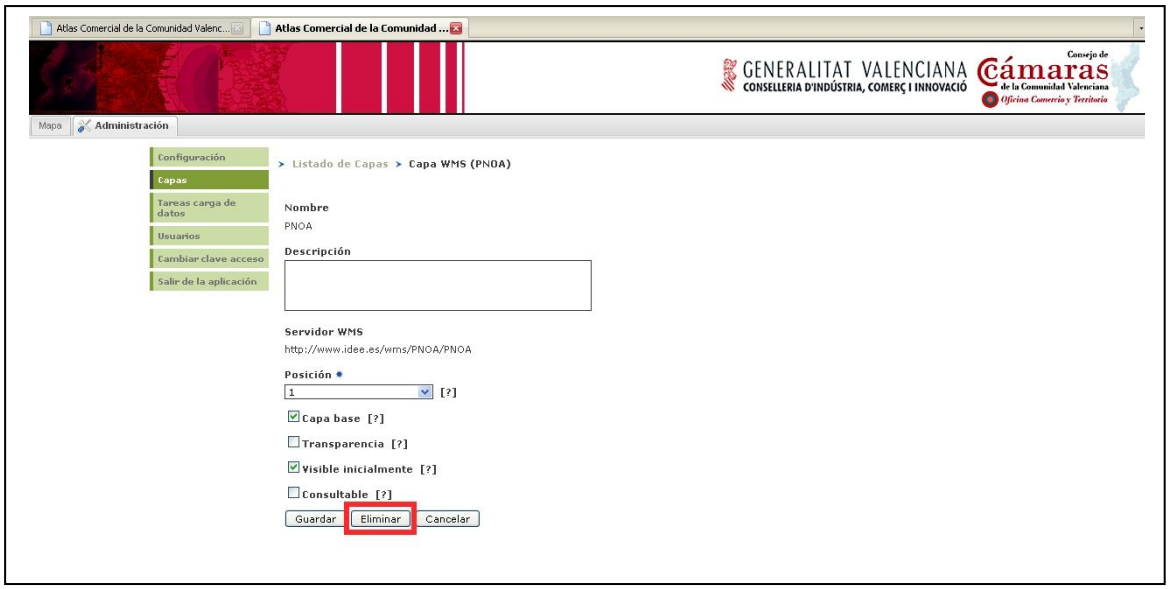

Figura 80. Eliminación de una capa.

10.4 Leyenda de una capa

Esta opción permite modificar las propiedades de visualización de la capa. Una vez creada la capa, se nos habilita el botón de leyenda, con lo que podremos acceder a las herramientas de edición de la misma.

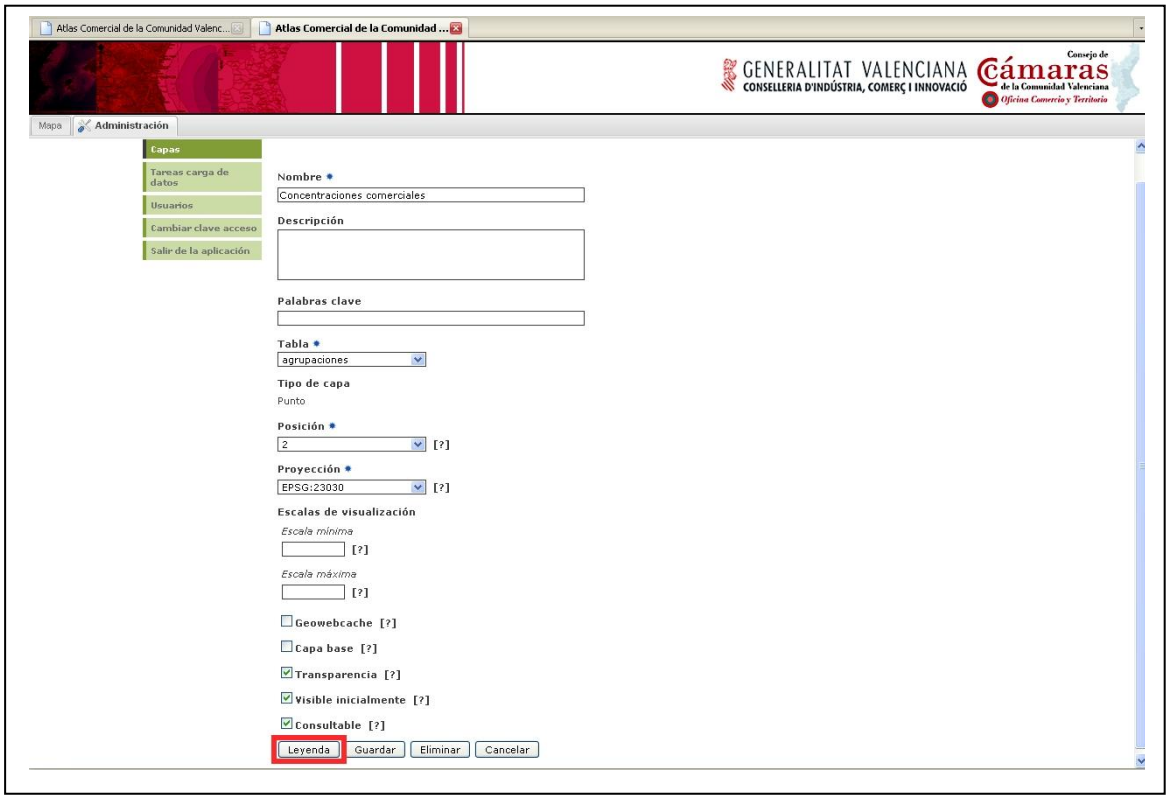

Figura 81. Modificación de las propiedades de visualización de una capa, permitiendo el acceso a las propiedades de una leyenda.

10.4.1. Tipos de leyenda

Una vez en el editor de leyendas, podemos seleccionar tres tipo de leyenda, simple, de clasificación, y por intervalos. Para aplicar cualquier cambio en el tipo de leyenda, deberemos activar el botón “Regenerar”.

- Leyenda simple: En este tipo de leyenda se visualizan todos los elementos de una capa con la misma simbología.

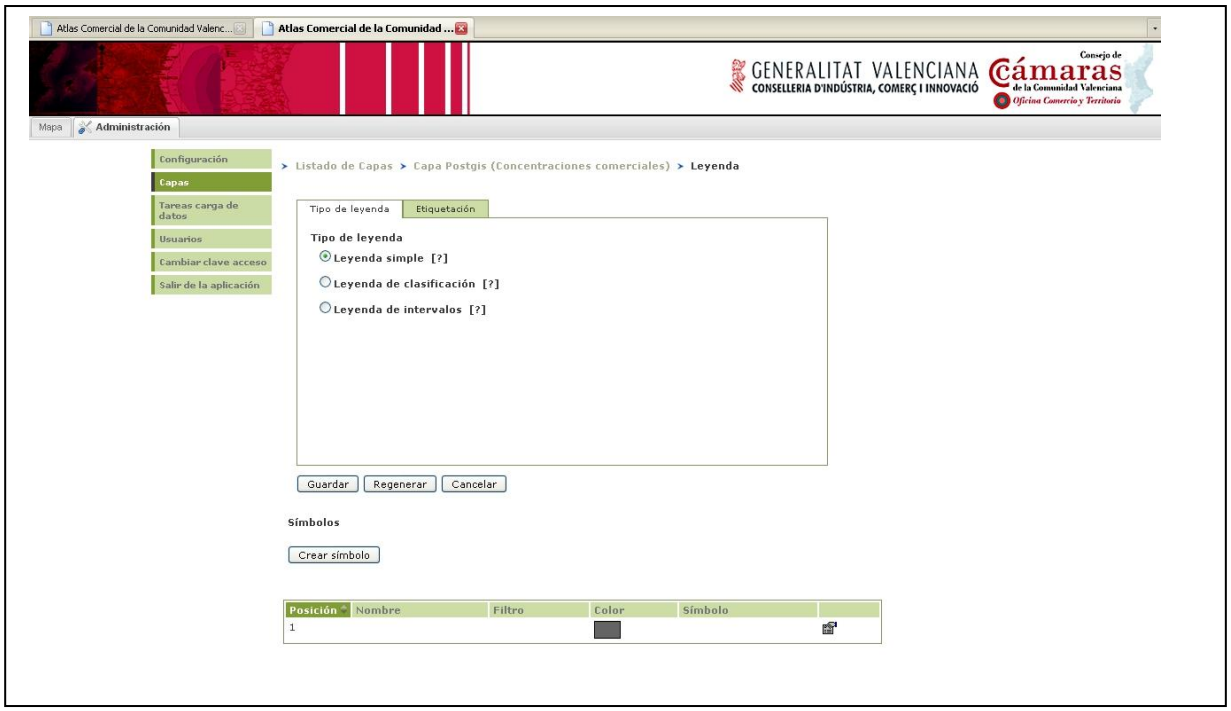

Figura 82. Tipo de leyenda: Simple.

A continuación podemos ver como se representaría un tema de polígonos con este tipo de leyenda.

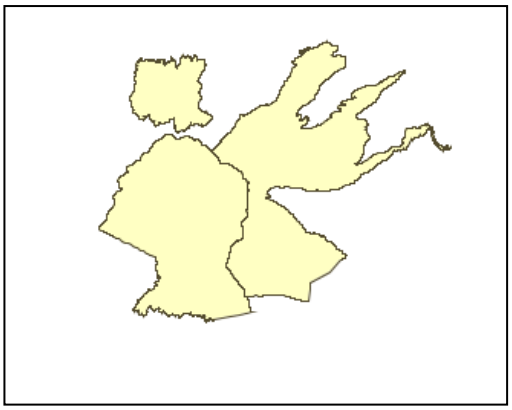

Figura 83. Ejemplo de leyenda simple.

- Leyenda de clasificación: En este tipo de leyendas se visualizan los elementos de una capa con distintos símbolos en función de los valores de alguno de los campos de la capa.

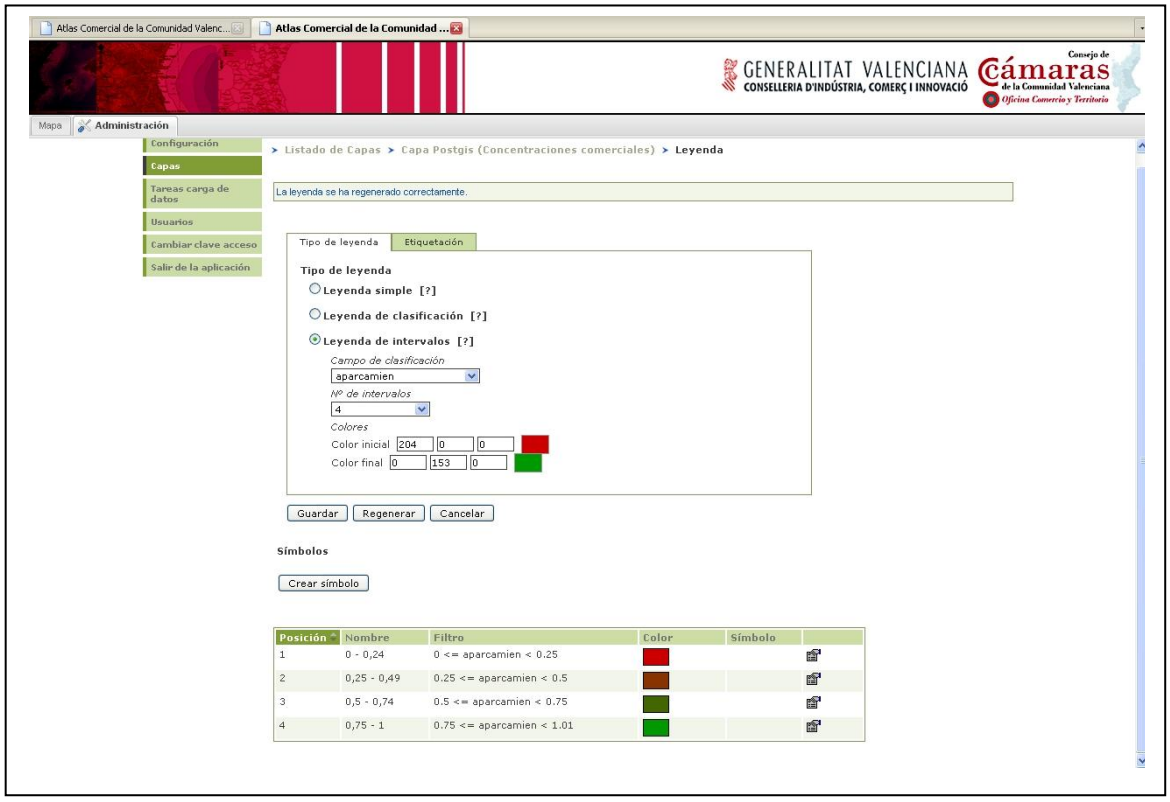

Figura 84. Leyenda de clasificación.

El usuario debe seleccionar:

- Campo de clasificación: campo por el que se clasificará la capa. Se generarán tantos símbolos como valores diferentas tenga dicho campo.
- Esquema de color: colores que se van a usar en la clasificación. Estos colores se podrá, modificar posteriormente en la edición de símbolos.

A continuación podemos ver como se representaría un tema de polígonos con este tipo de leyenda.

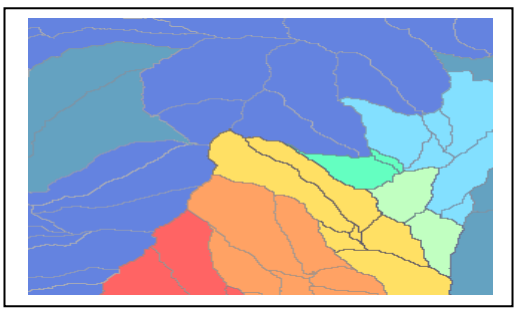

Figura 85. Ejemplo de leyenda de clasificación.

- Leyenda de intervalos: En este tipo de leyendas se visualizan los elementos de una capa con distintos símbolos en función de los valores de alguno de los campos agrupado en intervalos. Dicho campo tiene que ser numérico.

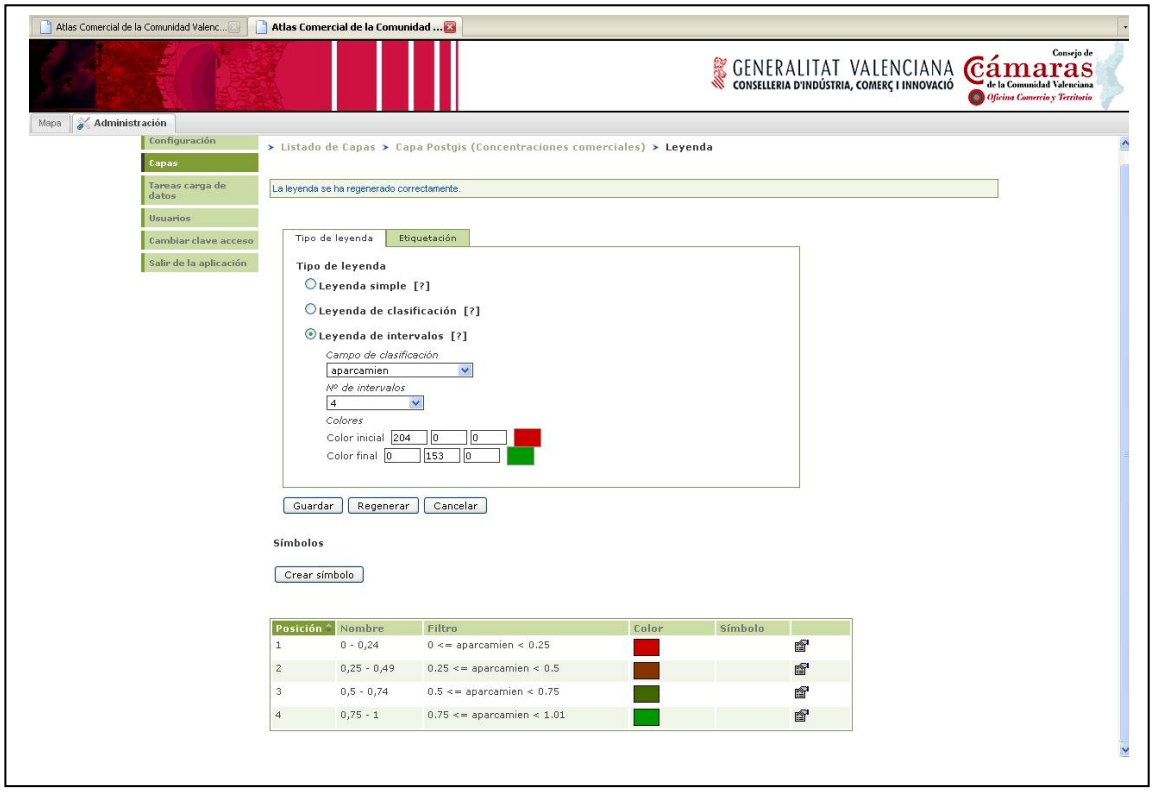

Figura 86. Ejemplo de leyenda de intervalos.

El usuario debe seleccionar:

- Campo de clasificación: campo por el que se clasificará la capa. Solo se listarán los campos numéricos.
- Número de intervalos: cantidad de intervalos que se generarán en la clasificación.
- Colores inicial y final: componentes RGB de los colores inicial y final entre los que se generarán, de forma lineal, los tonos para los intervalos. Estos colores se podrá, modificar posteriormente en la edición de símbolos.

La visualización de los elementos en el mapa sería igual que en el caso de la leyenda de clasificación.

11. Edición de símbolos

Una vez elegida el tipo de leyenda, en la parte inferior de la pantalla se nos ha generado los símbolos que vamos a emplear en la representación de la cartografía.

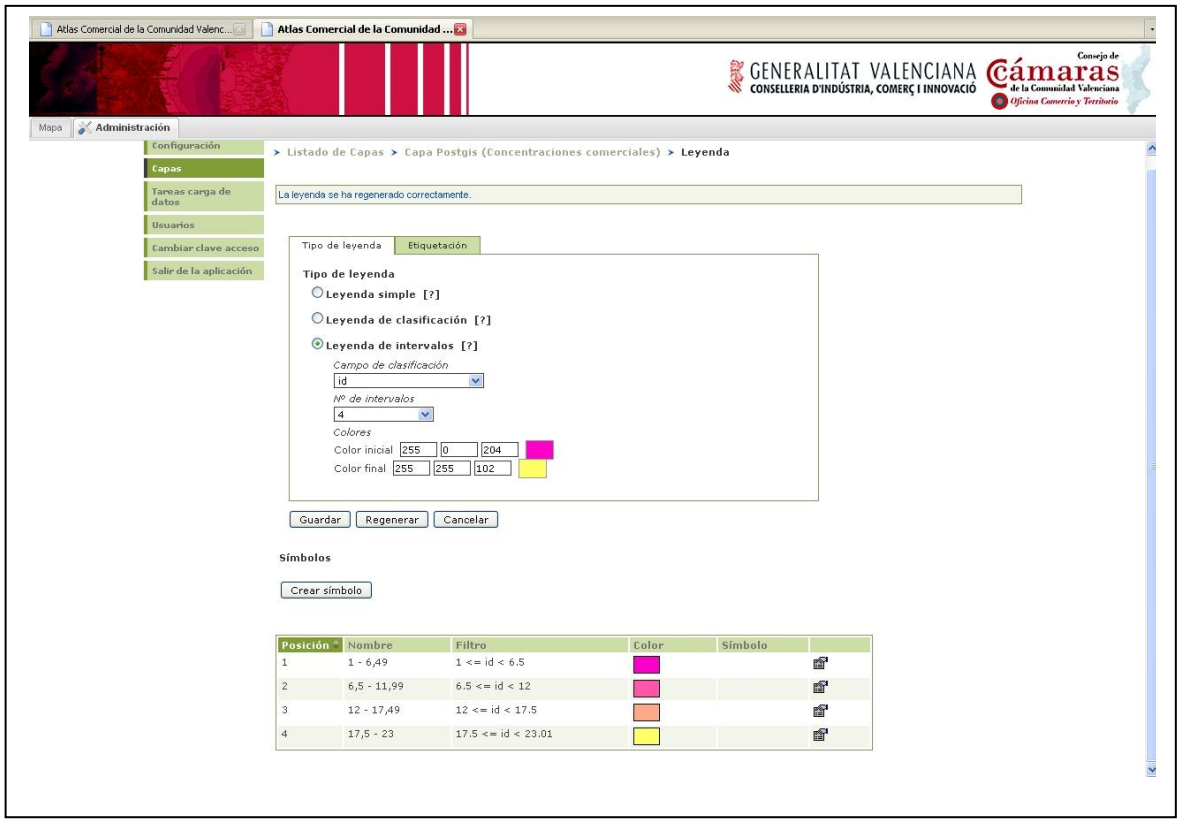

Figura 87. Edición de símbolos para visualizar los elementos de una capa.

En esta zona se muestran los símbolos utilizados para visualizar los elementos de la capa con sus características básicas:

- Nombre: nombre del símbolo que aparecerá en la leyenda.
- Posición: posición que ocupara el símbolo en la leyenda.
- Filtro: valores a los que aplicará el símbolo.
- Color: color del símbolo.
- Símbolo: nombre del símbolo especial que se aplicará a dichos elementos, que puede venir de una imagen, de una fuente, etc.

Haciendo clic en el botón de cada símbolo se accede a la pantalla de sus propiedades.

En esta pantalla el usuario ha de rellenar los datos asociados al símbolo utilizado para representar los elementos de una capa vectorial. Estos datos pueden cambiar ligeramente según el tipo de leyenda que hayamos seleccionado, así como el tipo de capa que estemos tratando, si es de puntos, de líneas o de polígonos.

A continuación podemos ver una pantalla para la edición de elementos poligonales en una leyenda de clasificación:

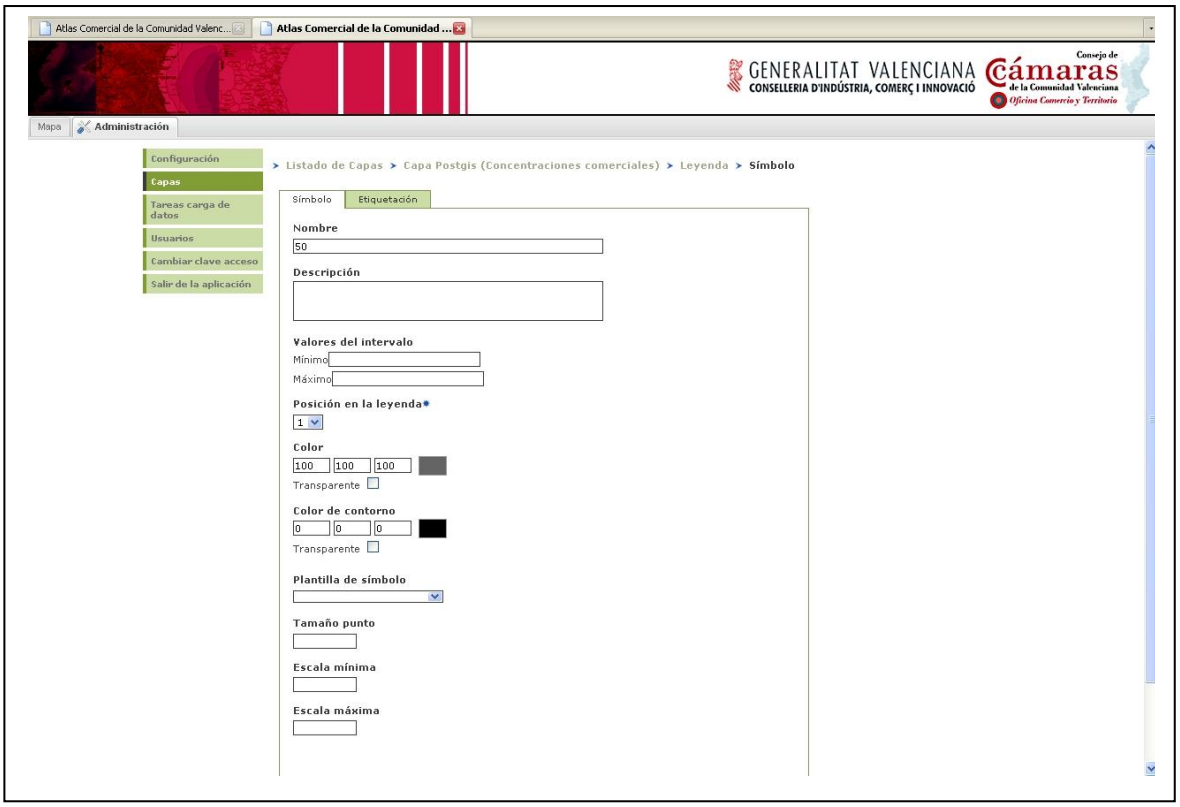

Figura 88. Edición de elementos poligonales en una leyenda de clasificación.

Los atributos que podemos definir son los siguientes:

- Nombre: nombre que aparecerá en la leyenda.
- Descripción: breve descripción del símbolo.
- Filtro: valores que se aplicarán para la aplicación del símbolo. Activo para leyendas simples y de clasificación.
- Posición en la leyenda: posición que ocupará el símbolo en la leyenda.
- Color: color de primer plano que se le aplicará al polígono.
- Color de contorno: color de contorno que se le aplicará al polígono.
- Color de fondo: color de fondo que se le aplicará al polígono.

- Transparente: hace transparente el elemento.
- Plantilla de símbolo: símbolo que se aplicará para pintar el elemento. La aplicación dispone de una librería de símbolos complejos para poder representar adecuadamente los elementos de las capas. Este campo permite especificar la plantilla a utilizar.

Es importante resaltar que algunas de estas plantillas ya tienen predefinidas algunas propiedades, como los colores utilizados.

- Grosor contorno: grosor de la línea de contorno.
- Escala mínima: escala mínima por debajo de la cual no se visualizará el elemento.
- Escala máxima: escala máxima por encima de la cual no se visualizará el elemento.

Seguidamente podemos ver un ejemplo para elementos lineales con una leyenda de intervalos:

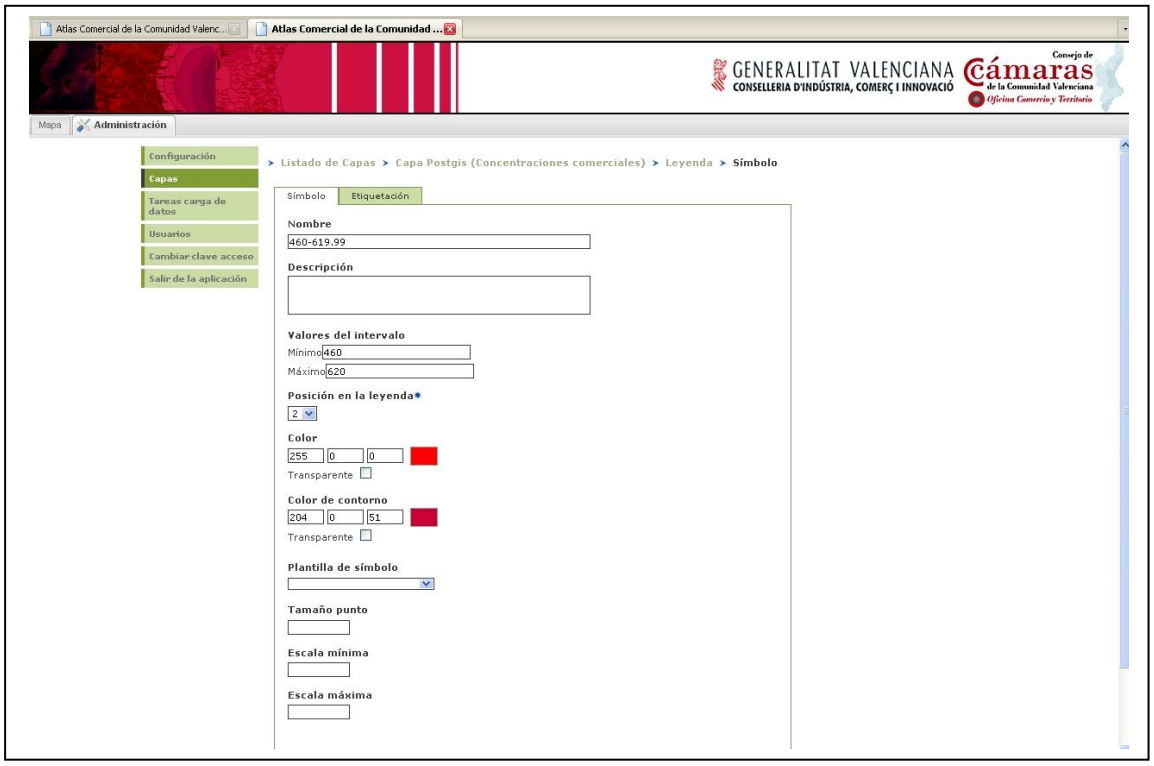

Figura 89. Elementos lineales con una leyenda de intervalos.

Los atributos que podemos definir son los siguientes:

- Nombre: nombre que aparecerá en la leyenda
- Descripción: breve descripción del símbolo
- Valores del intervalo: valores mínimo y máximo a los que se le aplicara este símbolo. Solo disponible en leyenda por intervalos.
- Posición en la leyenda: posición que ocupará el símbolo en la leyenda.
- Color: color que se le aplicará al elemento.
- Transparente: hace transparente el elemento.
- Plantilla de símbolo: símbolo que se aplicará para pintar el elemento. La aplicación dispone de una librería de símbolos complejos para poder representar adecuadamente los elementos de las capas. Este campo permite especificar la plantilla a utilizar.

Es importante resaltar que algunas de estas plantillas ya tienen predefinidas algunas propiedades, como los colores utilizados.

- Grosor contorno: grosor de la línea.
- Escala mínima: escala mínima por debajo de la cual no se visualizará el elemento.
- Escala máxima: escala máxima por encima de la cual no se visualizará el elemento.

Seguidamente podemos ver un ejemplo para elementos puntuales con una leyenda simple:

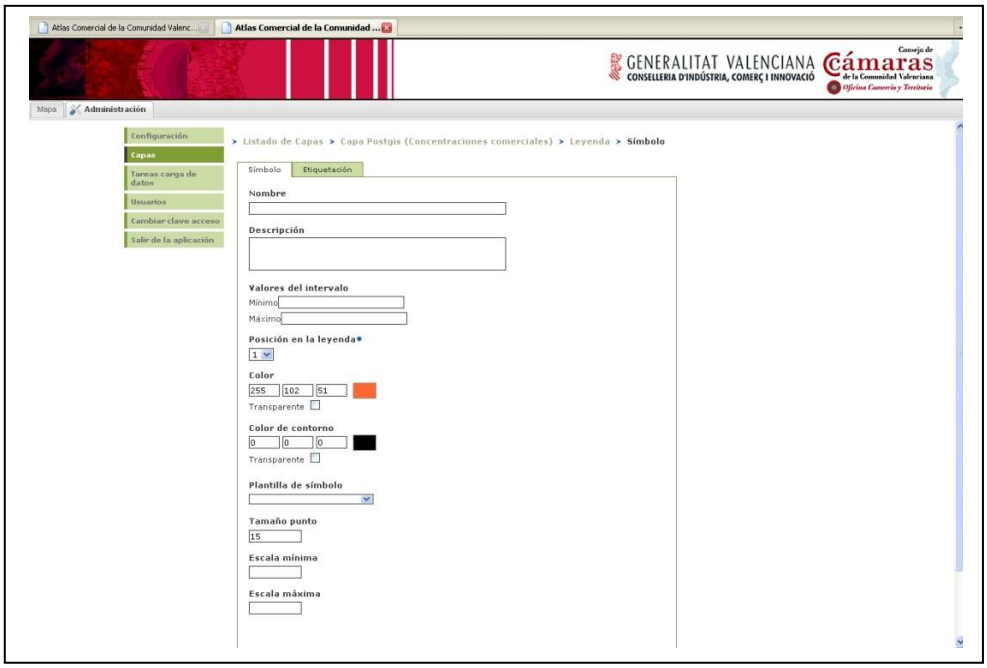

Figura 90. Elementos puntuales con una leyenda simple.

Los atributos que podemos definir son los siguientes:

- Nombre: nombre que aparecerá en la leyenda.
- Descripción: breve descripción del símbolo.
- Filtro: valores que se aplicarán para la aplicación del símbolo. Activo para leyendas simples y de clasificación.
- Posición en la leyenda: posición que ocupará el símbolo en la leyenda.
- Color: color que se le aplicará al elemento.
- Color de contorno: color de contorno que se le aplicará al símbolo.
- Transparente: hace transparente el elemento.
- Plantilla de símbolo: símbolo que se aplicará para pintar el elemento. La aplicación dispone de una librería de símbolos complejos para poder representar adecuadamente los elementos de las capas. Este campo permite especificar la plantilla a utilizar.

Es importante resaltar que algunas de estas plantillas ya tienen predefinidas algunas propiedades, como los colores utilizados.

- Tamaño punto: tamaño que se aplicará al símbolo.

- Escala mínima: escala mínima por debajo de la cual no se visualizará el elemento.
- Escala máxima: escala máxima por encima de la cual no se visualizará el elemento.

Para seleccionar los colores el usuario puede escribir el código RGB del color o hacer clic en el recuadro que representa el color, mostrándose una ventana flotante que permite seleccionar gráficamente el color. La ventana flotante consta de dos pestañas:

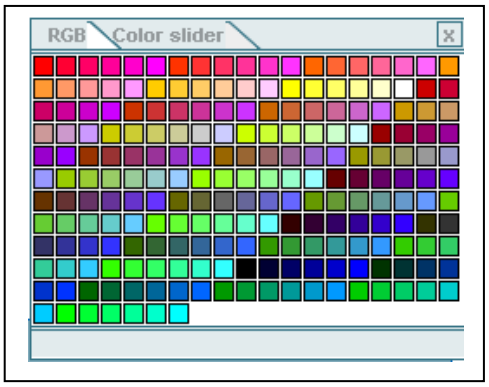

Figura 91. Tabla de colores para que el administrador escoja el color de la leyenda.

En la primera pestaña (RGB) se muestra una lista de colores predefinidos.

Para seleccionar un color se ha hacer clic sobre alguno de los colores, cerrándose la ventana flotante y rellenando las casillas del color correspondiente con el RGB del color seleccionado.

En la segunda pestaña (Color slider) el usuario puede especificar el color moviendo las barras de desplazamiento que hay en cada uno de los valores del RGB o escribiendo el valor en la casilla de texto correspondiente.

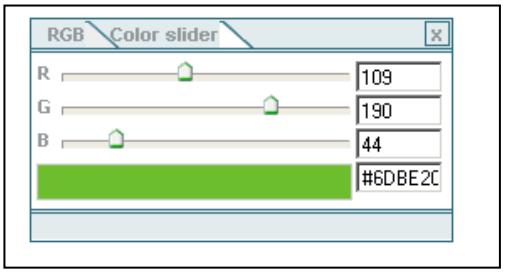

Figura 92. Selección de colores moviendo las barras de desplazamiento.

Una vez especificado el color se ha de pulsar sobre la zona coloreada para cerrar la ventana flotante, rellenándose las casillas RGB correspondientes.

11.1. Crear símbolos

No sólo podemos modificar símbolos ya creados, sino que también podemos añadir nuevos a una leyenda gracias al botón “Crear símbolo”.

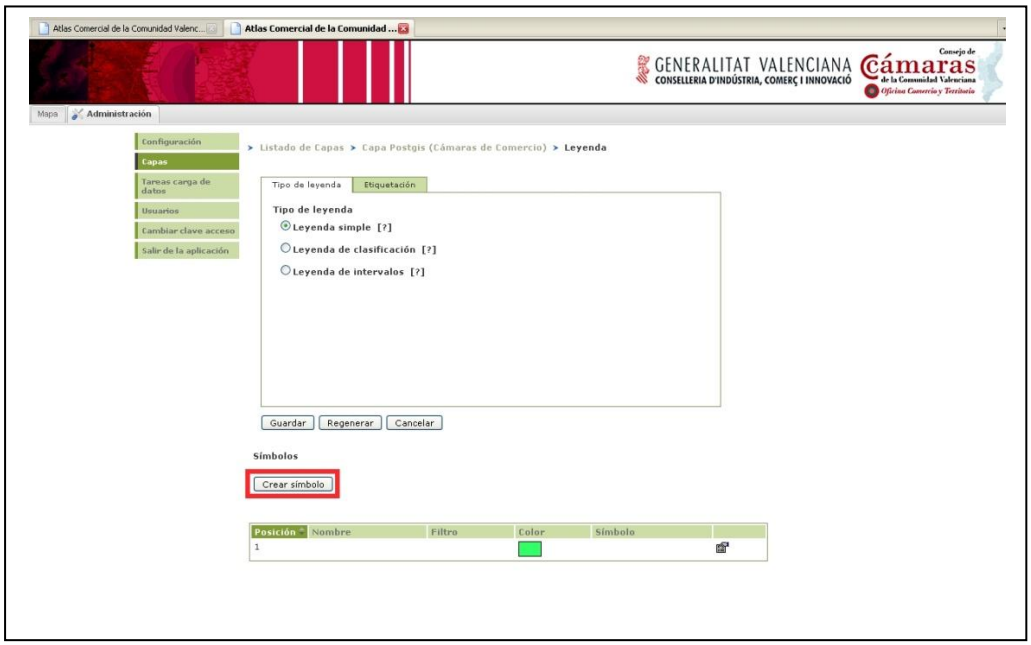

Figura 93. Se pueden crear nuevos símbolos además de modificar los ya existentes.

Tendremos que rellenar una pantalla con todos los datos necesarios para la correcta definición del símbolo, que dependerá de si se trata de un símbolo puntual, lineal o poligonal. Del mismo modo, los campos a rellenar también diferirán si se trata de una leyenda simple, de clasificación o por intervalos. Las pantallas serán las mismas que hemos vistos en el apartado anterior de edición de simbología.

Una vez creado, el símbolo nuevo aparecerá listado junto al resto.

11.2. Eliminar símbolos

La eliminación de símbolos resulta muy sencilla, ya que en la pantalla de edición de símbolos disponemos de un botón que nos elimina el símbolo que estamos editando.

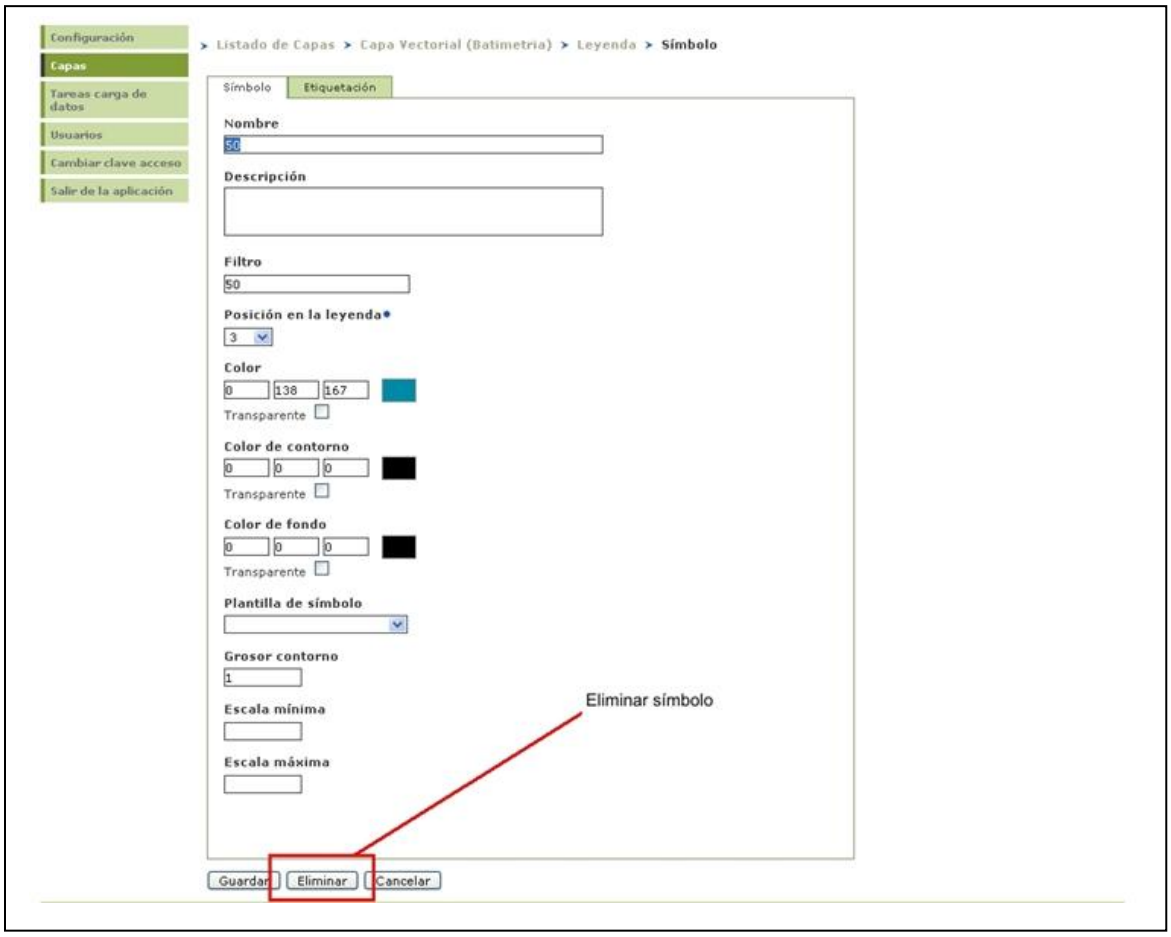

Figura 94. Eliminar símbolo.

Una vez eliminado, el símbolo ya no aparecerá listado en la leyenda.

12. Etiquetación

Esta opción permite asignar etiquetas textuales a los elementos de una capa siguiendo los elementos de un campo del tema.

En primer lugar, dentro de la pantalla de selección de tipo de leyenda, en la pestaña de etiquetación, tendremos que seleccionar unas características generales de la etiquetación:

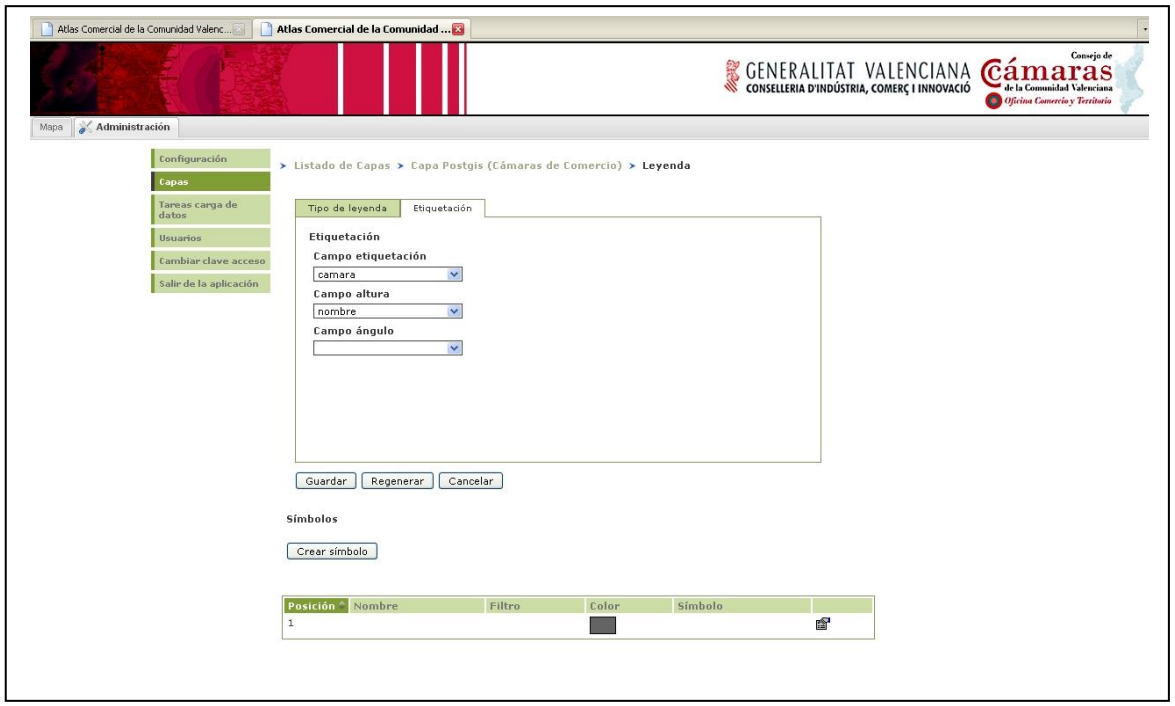

Figura 95. Etiquetación de una leyenda.

Los datos que se pueden rellenar son los siguientes:

- Campo de etiquetación: campo con el que se generará los textos de la etiquetas.
- Campo altura: si existiera, campo donde se indica el tamaño de las etiquetas.
- Campo ángulo: si existiera, campo que indica el ángulo de las etiquetas.

Estos datos son generales para todas las etiquetas que queremos que aparezcan en el mapa, pero tenemos que definirlas de una forma más concreta. Para ello, al igual que ocurría en la selección del tipo de leyenda, en la edición de símbolos disponemos de una pestaña de etiquetación donde podremos definir las etiquetas que se aplicarán a los elementos representados con dicho símbolo.

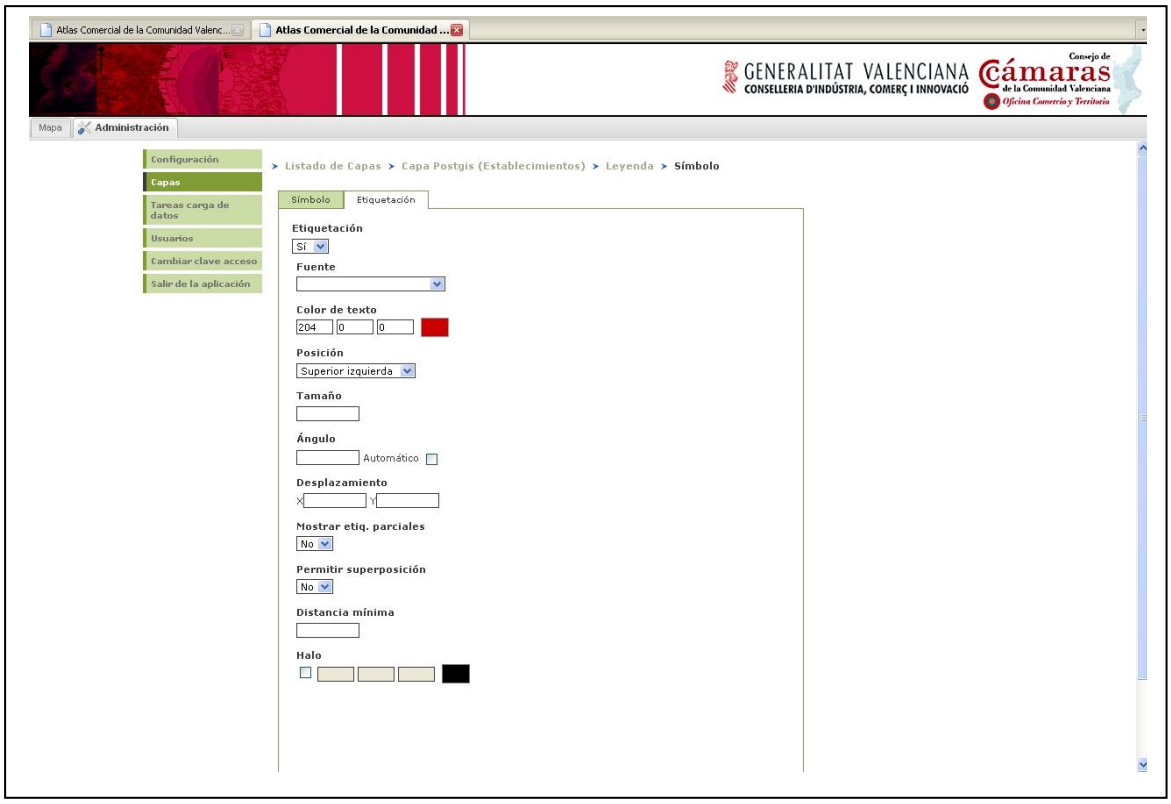

Figura 96. Etiquetación de un símbolo.

Los campos que podemos modificar en esta pantalla son los siguientes:

- Etiquetación: si se aplicarán etiquetas a un símbolo (si/no).
- Fuente: fuente con la que se escribirán los textos.
- Color de texto: color que se aplicará a los textos.
- Posición: posición del texto con respecto del elemento.
- Tamaño: tamaño del texto.
- Ángulo: ángulo a aplicar al texto (puede ser automático).
- Desplazamiento: desplazamiento en píxeles que se aplica a l texto con respecto al elemento.
- Mostrar etiq. Parciales: se muestran las etiquetas parciales en los límites del visor (si/no).
- Permitir superposición: permite la superposición de textos cuando estos no caben sin superponerse (si/no).
- Halo: línea que bordea un texto de un píxel de ancho.

13. Gestión de usuarios

Esta opción permite, en cada uno de los servicios, gestionar los usuarios que pueden acceder a la aplicación. La herramienta de gestión de usuarios la encontramos en la parte superior izquierda de la aplicación, en el menú.

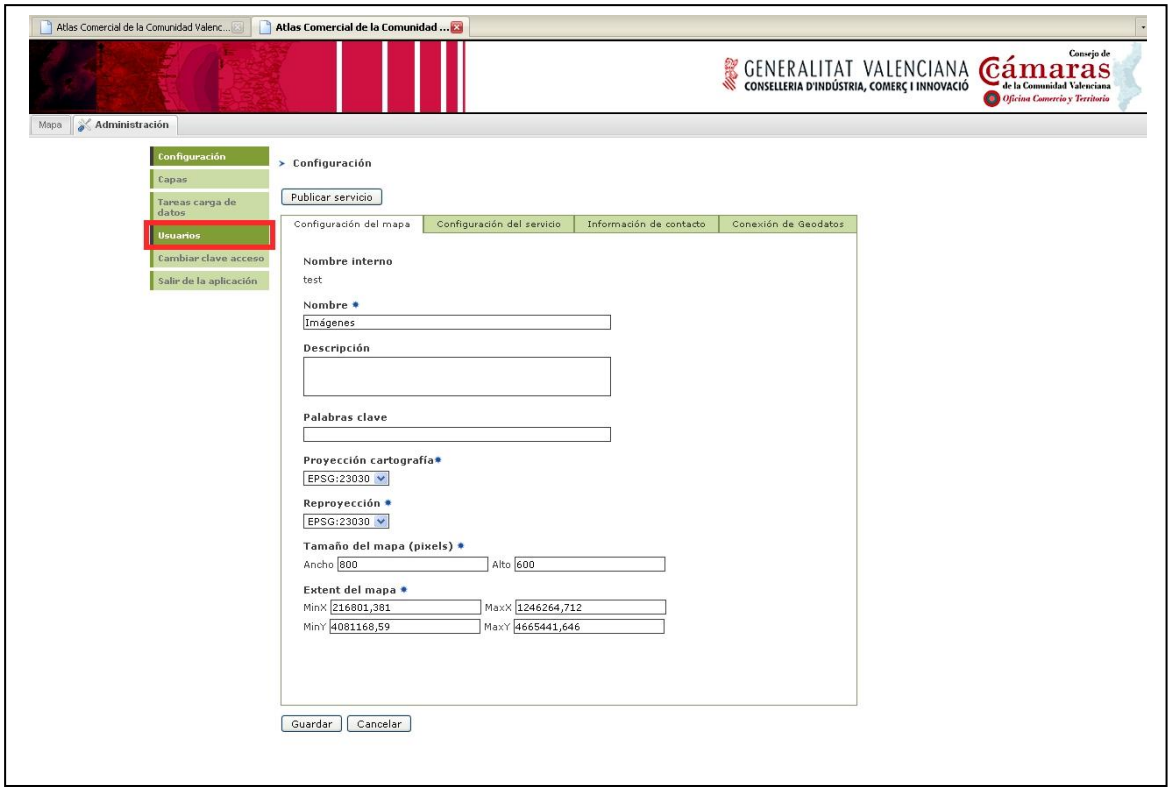

Figura 97. Gestión de usuarios.

Cada usuario sólo puede acceder al servicio al que está autorizado. Se contemplan dos perfiles de usuario distintos:

- Perfil editor: Los usuarios con este perfil pueden gestionar la configuración de capas disponibles, así como la configuración del mapa en el visor de mapas. A continuación podemos ver la pantalla del perfil editor, donde en la zona del menú de la aplicación tenemos deshabilitadas las opciones de gestión de usuarios, así como la pestaña de configuración del servicio.

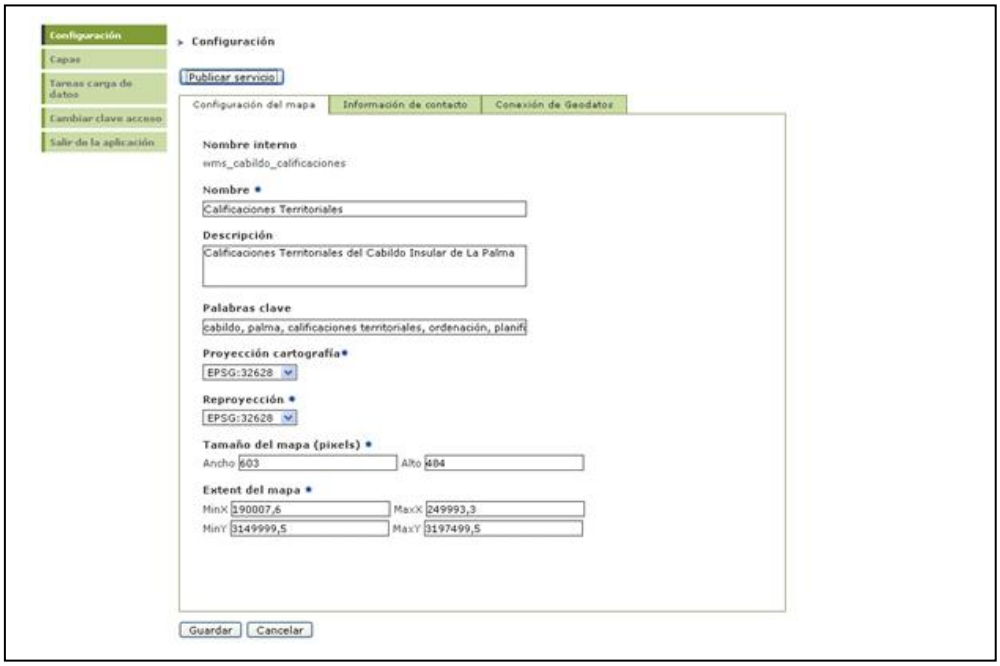

Figura 98. Pantalla del perfil editor.

- Perfil administrador: Los usuarios con este perfil, además de las funcionalidades del perfil editor, pueden gestionar la configuración del servicio de mapas y los usuarios de la aplicación para ese servicio.

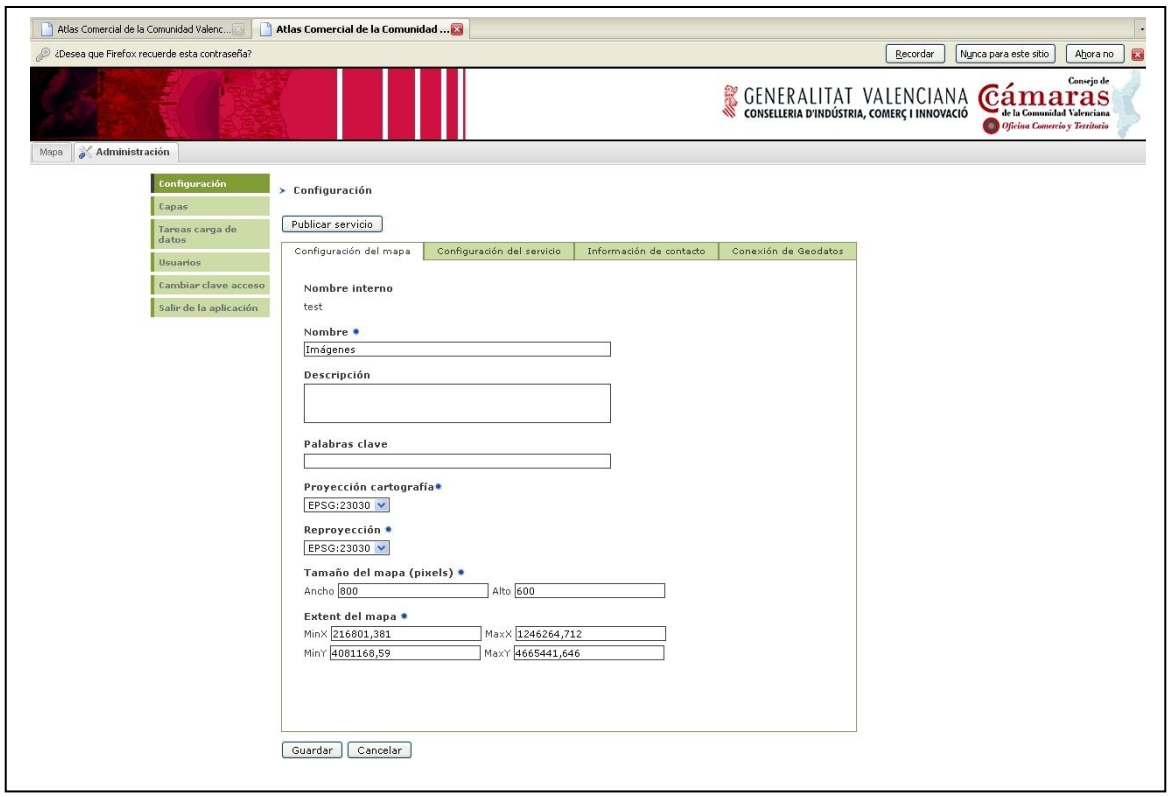

Figura 99. Pantalla para perfil administrador.

Al acceder a la opción Usuarios se muestra un listado con los usuarios que pueden gestionar los datos del servicio. Esta opción sólo está disponible para los usuarios de perfil administrador.

Figura 100. Listado de los usuarios que pueden gestionar los datos del servicio.

En la que podemos ver el nombre del usuario, una breve descripción y su rol. Desde este listado el usuario puede crear nuevos usuarios y modificar las propiedades de los usuarios  del servicio.

13.1. Crear un usuario

Para crear un nuevo usuario deberemos activar el botón correspondiente:

Figura 101. Creación de un nuevo usuario.

Con lo que se nos mostrará la siguiente pantalla

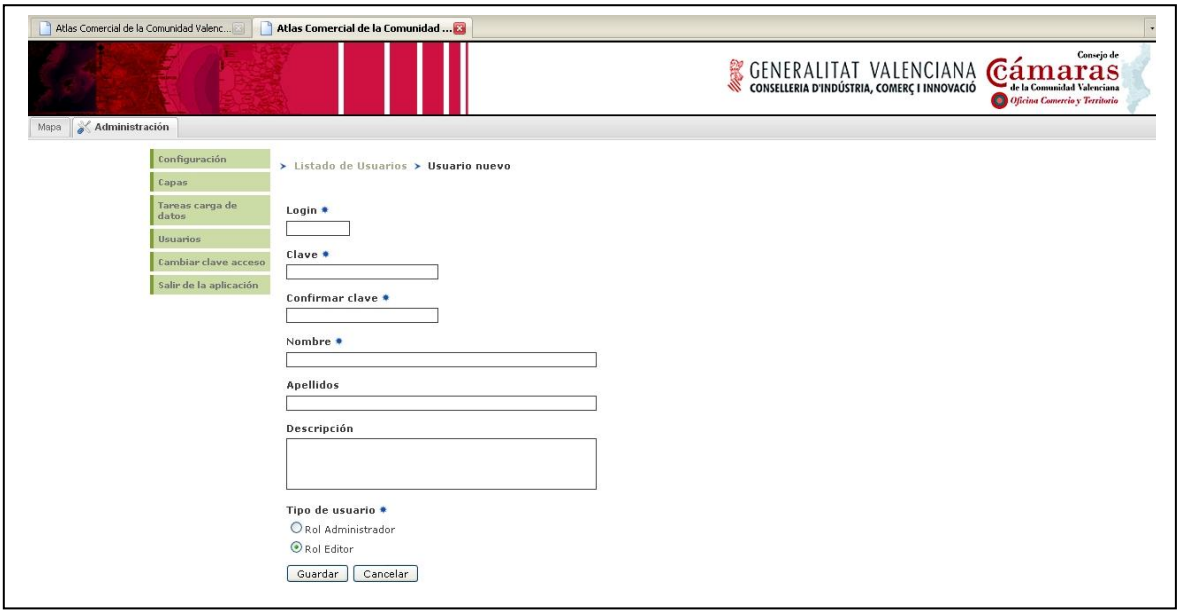

Figura 102. Datos a rellenar para el nuevo usuario.

En la que tendremos que rellenar los siguientes campos:

- Login: nombre de acceso.
- Clave: clave de acceso.
- Confirmar clave: confirmación de la clave.
- Nombre: nombre del usuario.
- Apellidos: apellidos del usuario.
- Descripción: breve descripción del usuario.
- Tipo de usuario: si es editor o si es administrador.

13.2. Modificación de un usuario

Esta pantalla es análoga a la de creación de usuarios, permitiendo modificar los datos de un usuario (excepto el login) y eliminarlo del sistema (excepto el usuario admin).

13.3. Eliminación de un usuario

Para eliminar un usuario sólo tendremos que activar el botón eliminar en la pantalla de modificación:

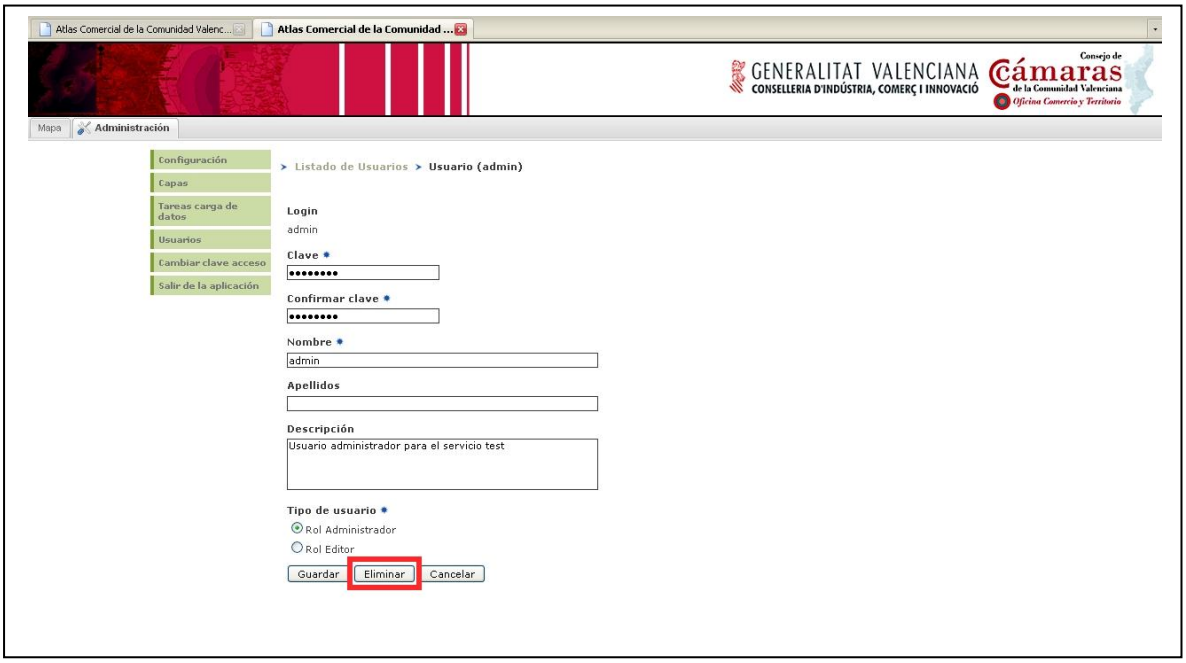

Figura 103. Eliminación de un usuario existente.

13.4. Cambiar la clave

Esta opción permite al usuario modificar su clave de acceso a la aplicación. Para ello ha de introducir la nueva clave por duplicado para evitar posibles errores a la hora de introducir la clave.

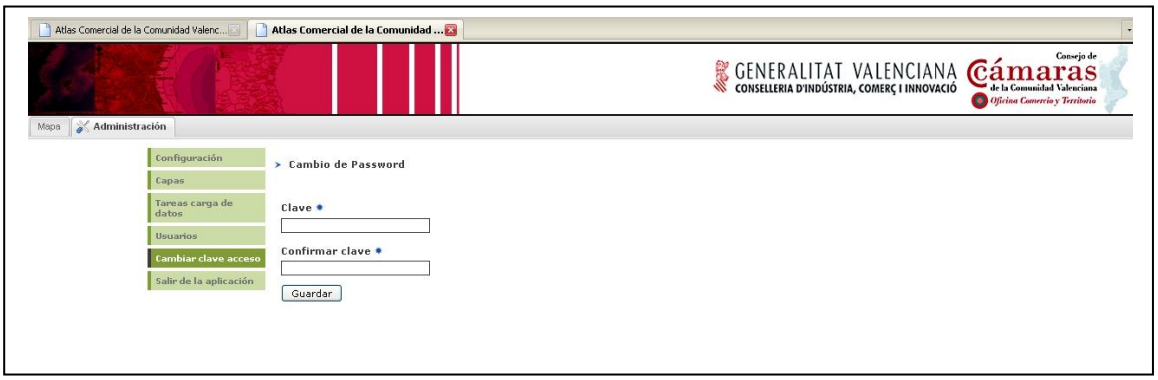

Figura 104. Modificación de la clave de un usuario.

8. Anexo C.

Solución a la proyección Spherical Mercator

Para utilizar los mapas de Google y Yahoo como base cartográfica es necesario definirlo como proyecciones Spherical Mercator.

```
/**
 * Property: sphericalMercatorOptions
 * {Array(Object)} Parámetros por defecto para la creación de un mapa
 * SphericalMercator.
 */
      sphericalMercatorOptions: {
            projection: new OpenLayers.Projection("EPSG:900913"),
            displayProjection: new
OpenLayers.Projection("EPSG:900913"),
            units: "m",
            numZoomLevels: 18,
            maxResolution: 156543.0339,
            controls: [
                      new OpenLayers.Control.KeyboardDefaults(),
                     new OpenLayers.Control.MouseDefaults({
                    'performedDrag':false}),
                      new OpenLayers.Control.ScaleLine()

                  ]
      },



/**
 * Method: loadSphericalMercatorLayers
 * Añade capas SphericalMercator a un mapa.
 *
 * Parameters:
 * map - {OpenLayers.Map} Mapa al que añadir las capas.
 */

loadSphericalMercatorLayers: function(map){
            layers = {
                  // create Google Mercator layers
                  ghyb: new OpenLayers.Layer.Google("Google
                  Hybrid",{type:    G_HYBRID_MAP, 'sphericalMercator':
                  true, numZoomLevels: 21}),
                  gmap: new OpenLayers.Layer.Google("Google
                  Streets",{'sphericalMercator': true, numZoomLevels:
                  20}),
                  gsat: new OpenLayers.Layer.Google("Google
                  Satellite",{type: G_SATELLITE_MAP,
                  'sphericalMercator': true, numZoomLevels: 22}),

                  // create Yahoo layer
                  yahoo: new OpenLayers.Layer.Yahoo("Yahoo
                  Street",{'sphericalMercator': true}),
                  yahoosat: new OpenLayers.Layer.Yahoo("Yahoo
                  Satellite",{'type': YAHOO_MAP_SAT,
                  'sphericalMercator': true}),
                  yahoohyb: new OpenLayers.Layer.Yahoo("Yahoo
                  Hybrid",{'type': YAHOO_MAP_HYB, 'sphericalMercator':
                  true})
```

```
            };
```

Re-proyección sobre las capas Spherical Mercator

Debido a que Google y Yahoo utilizan proyecciones diferentes es necesario realizar una re-proyección sobre estas capas antes de añadirlas al mapa.

```
…
// Reproyecta el maxExtent de las capas
for(var i=0; i<layers.length; i++){
      var layer = layers[i];
      layer.maxExtent.transform(new OpenLayers.Projection(projection),
      new OpenLayers.Projection(projCode));
}

this.registerMapEvents(map);
map.addLayers(layers);

if(oldBoundsMap != null){
      // Reproyecta el bounds antiguo
      oldBoundsMap.transform(new
      OpenLayers.Projection(oldProjectionMap), new
      OpenLayers.Projection(projCode));
      map.setCenter(oldBoundsMap.getCenterLonLat(),
      map.getZoomForExtent(oldBoundsMap, true));
}
else{
      // Reproyecta el bounds del contexto
      bounds.transform(new OpenLayers.Projection(projection), new
      OpenLayers.Projection(projCode));
      map.setCenter(bounds.getCenterLonLat(),
map.getZoomForExtent(bounds, true));
}

…
```

Extensión Atlas – Config.js

```
Config.ExtSearchAtlas.WFSLayers = {
            COMERCIOS_INNOVACION: new IDEOL.Layer.WFS('Comercios
            innovación','http://services.iver.es/geoserver/wfs',
                        {
                         editable: false,
                         projection: new
                         OpenLayers.Projection('EPSG:23030'),
                         geomType: 'POINT',
                         featurePrefix: 'atlas',
                          featureType: 'comercios_innovacion',
                          featureNS: 'http://localhost/atlas',
                         visibility: false,
                          displayInLayerSwitcher: false
                        }
            ),
            CAMARAS_COMERCIO: new IDEOL.Layer.WFS('Cámaras de
            Comercio','http://services.iver.es/geoserver/wfs',
                        {
                         editable: false,
```

```
                projection: new
                OpenLayers.Projection('EPSG:23030'),
                geomType: 'POINT',
                featurePrefix: 'atlas',
                 featureType: 'camaras',
                 featureNS: 'http://localhost/atlas',
                visibility: false,
                 displayInLayerSwitcher: false
               }
),
ANTENAS_LOCALES: new IDEOL.Layer.WFS('Antenas
Locales','http://services.iver.es/geoserver/wfs',
               {
                editable: false,
                projection: new
                OpenLayers.Projection('EPSG:23030'),
                geomType: 'POINT',
                featurePrefix: 'atlas',
                 featureType: 'antenas',
                 featureNS: 'http://localhost/atlas',
                visibility: false,
                 displayInLayerSwitcher: false

               }
),
ESTABLECIMIENTOS: new IDEOL.Layer.WFS('Establecimientos',
'http://services.iver.es/geoserver/wfs',
               {
                editable: false,
                projection: new
                OpenLayers.Projection('EPSG:23030'),
                geomType: 'POINT',
                featurePrefix: 'atlas',
                 featureType: 'establecimientos',
                 featureNS: 'http://localhost/atlas',
                visibility: false,
                 displayInLayerSwitcher: false
               }
),
CONCENTRACIONES_COMERCIALES: new
IDEOL.Layer.WFS('Concentraciones
comerciales','http://services.iver.es/geoserver/wfs',
               {
                editable: false,
                projection: new
                OpenLayers.Projection('EPSG:23030'),
                geomType: 'POINT',
                featurePrefix: 'atlas',
                 featureType: 'agrupaciones',
                 featureNS: 'http://localhost/atlas',
                visibility: false,
                 displayInLayerSwitcher: false
               }
),
MUNICIPIOS: new IDEOL.Layer.WFS('Municipios',
'http://services.iver.es/geoserver/wfs',
               {
                editable: false,
                projection: new
                OpenLayers.Projection('EPSG:23030'),
                geomType: 'MULTIPOLYGON',
```

```
                featurePrefix: 'atlas',
                featureType: 'municipios',
                featureNS: 'http://localhost/atlas',
                visibility: false,
                displayInLayerSwitcher: false
               }
        )
};
```

6.4.4. Definición de la interfaz

```
Ext.namespace("Layout");

/**
 * Class: Layout
 * Representa la interfaz gráfica de la aplicación.
 *
 *  Inherits from:
 *  - <Ext.Component>
 */
Layout = Ext.extend(Ext.Component, {

      /**
       * Property: ideol
       * <IDEOL.App>
       */
      ideol: null,

      /**
       * Property: toolBar
       * <Layout.Bar.ToolBar>
       */
      toolBar: null,

      /**
       * Property: toolBarPanel
       * <Ext.Panel>
       */
      toolBarPanel: null,

      //Otras Properties definidas como el tocPanel (table of
      Contents)o el mapPanel

    /**
     * Method: initComponents
     * Inicia los componentes.
     *
     */
      initComponents: function(){

            …

            this.toolBar = new Layout.Bar.ToolBar({
            layerWizard:this.layerWizard});
      }

    /**
     * Method: initPanels
     * Inicia los paneles.
     *
     */
      initPanels: function(){
```

```
            this.toolBarPanel = new Ext.Panel({
                  region: 'north',
                  layout: 'fit',
                  border: false,
                  tbar: Config.TOOLBAR ? this.toolBar : null
            });
            …
      }
```

Toolbar.js

```
Ext.namespace("Layout.Bar");

/**
 * Class: Layout.Bar.ToolBar
 *  Inherits from:
 *  - <Ext.Toolbar>
 */
Layout.Bar.ToolBar = Ext.extend(Ext.Toolbar, {

      map: null,
      id: 'toolbar',
      height: 33,
      ideol: null,
      drawManager: null,
      statusBar: null,
      localeCombo: new IDEOL.Widget.LocaleCombo(),
      layerWizard: null,
      listeners:{
           afterrender: function(){
                        this.addItem(this.localeCombo);
                        this.addItems(this.mapButtons);
                        this.addSeparator();

                        if(Config.EDITION_SUPPORT){
                              this.addItem(this.loginButton);
                              this.addSeparator();
                        }
               }
      },

      /**
       * CONTROLES DE MAPA
       *
       */
      mapButtons: {
            zoomFull: new Ext.Toolbar.Button({
                iconCls: 'zoomfull',
                tooltip: Locale.getText("txt_zoom_completo")
            }),
            zoomIn: new Ext.Toolbar.Button({
                iconCls: 'zoomin',
                tooltip: Locale.getText("txt_zoom_mas"),
                toggleGroup: 'map'
            }),
            zoomOut: new Ext.Toolbar.Button({
                iconCls: 'zoomout',
                tooltip: Locale.getText("txt_zoom_menos"),
                toggleGroup: 'map'
            }),
            pan: new Ext.Toolbar.Button({
```

```
                iconCls: 'pan',
                tooltip: Locale.getText("txt_desplazar_mapa"),
                toggleGroup: 'map',
                enableToggle: true,
                pressed: true
            }),
            back: new Ext.Toolbar.Button({
                  iconCls: 'back',
                  tooltip: Locale.getText("txt_vista_anterior")
            }),
            next: new Ext.Toolbar.Button({
                  iconCls: 'next',
                  tooltip: Locale.getText("txt_vista_siguiente")
            }),
            measureDistance: new Ext.Toolbar.Button({
                  iconCls: 'distance',
                  tooltip: Locale.getText("txt_medir_distancias"),
                  toggleGroup: 'map'
            }),
            measureArea: new Ext.Toolbar.Button({
                  iconCls: 'area',
                  tooltip: Locale.getText("txt_medir_areas"),
                  toggleGroup: 'map'
            }),

      cleanMap: new Ext.Toolbar.Button({
                  iconCls: 'cleanMap',
                  tooltip: Locale.getText("txt_limpiar_mapa")
            }),
            refreshMap: new Ext.Toolbar.Button({
                  iconCls: 'refreshMap',
                  tooltip: Locale.getText("txt_recargar_mapa")
            }),
            layerWizard: new Ext.Toolbar.Button({
                  iconCls: 'layerWizard',
                  tooltip: Locale.getText("txt_anyadir_capa"),
                  disabled: true
            })
      },
      /**
       * END CONTROLES DE MAPA
       */

      /**
       * EVENTOS CONTROLES DE MAPA
       *
       */

      setMapButtonsEvents: function(){

            this.mapButtons.zoomFull.on('click', function(object,
event){

      this.ideol.controlsManager.controls.zoomFull.trigger();
            }, this);

            this.mapButtons.zoomIn.on('click', function(object, event){
                  this.ideol.controlsManager.activateControl(this.ideol
                  .controlsManager.controls.zoomIn);
            }, this);
            this.mapButtons.zoomOut.on('click', function(object,
event){
```

```
                this.ideol.controlsManager.activateControl(this.ideol
            .controlsManager.controls.zoomOut);
            }, this);

            this.mapButtons.pan.on('click', function(object, event){

            this.ideol.controlsManager.activateControl(this.ideol.contr
      olsManager.
            controls.pan);
            }, this);
            this.mapButtons.back.on('click', function(object, event){

      this.ideol.controlsManager.controls.nav.previous.trigger();
            }, this);
            this.mapButtons.next.on('click', function(object, event){

      this.ideol.controlsManager.controls.nav.next.trigger();
            }, this);
            this.mapButtons.measureDistance.on('click',
function(object, event){
                this.ideol.controlsManager.activateControl(this.ideol
                .controlsManager.controls.measureDistance);
            }, this);
            this.mapButtons.measureArea.on('click', function(object,
      event){
                this.ideol.controlsManager.activateControl(this.ideol
            .controlsManager.controls.measureArea);
            }, this);
            this.mapButtons.cleanMap.on('click', function(object,
event){
                this.drawManager.cleanAll();
            }, this);
            this.mapButtons.refreshMap.on('click', function(object,
event){
                this.ideol.load(this.ideol.contextURL, '');
            }, this);
            this.mapButtons.layerWizard.on('click', function(object,
event){
                this.layerWizard.show();
            }, this);
      },

      /**
       * END EVENTOS CONTROLES DE MAPA
       *
       */

      /**
       * EVENTO LOGIN
       *
       */
      setLoginButtonEvent: function(){
            this.loginButton.on('click', function(object, event){
                //TODO
                Login.show();
            }, this);
      },

      setIDEOL: function(ideol){
            this.ideol = ideol;
      },
```

```
	setStatusBar: function(statusBar){
		this.statusBar = statusBar;
	},

	visibleItems: function(items, bool){
		for(key in items){
			items[key].setVisible(bool);
		}
	},

	enableItems: function(items, bool){
		for(key in items){
			items[key].setDisabled(bool);
		}
	},

	addSeparator: function(){
		this.addItem(new Ext.Toolbar.Separator());
	},

	addItems: function(items){
		for(key in items)
			this.addItem(items[key]);
	},

	initComponent: function(){
		Layout.Bar.ToolBar.superclass.initComponent.call(this);

		/*
		 * Asegura que una extension ha añadido un panel,
		 * lo hace por cada panel..
		 */
		this.layerWizard.tabPanel.on("add", function(){
			this.mapButtons.layerWizard.setDisabled(false);
		}, this);

		this.drawManager = IDEOL.Manager.Draw.getInstance();

		this.setMapButtonsEvents();
		this.setLoginButtonEvent();
	}
});
```

Index.html

Sobre la hoja de arranque de la aplicación se añaden las extensiones, cada una de ellas completamente funcional e independiente, generándose así un código modular y fácilmente ampliable.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>

   <head>

      <title>IDEOL</title>
      <script
type="text/javascript"src="http://maps.google.com/maps?file=...>
      </script>

      <script
      type="text/javascript"src="http://api.maps.yahoo.com/ajaxymap...
      ">
      </script>
      …
      <script type="text/javascript" src="ideol/IDEOL.js"></script>
      <script type="text/javascript" src="ideol/IDEOL-
Extensions.js"></script>

   </head>

   <body>
      <script>
            // BASES
            ideol.addExtension(new ExtWMS());
            ideol.addExtension(new ExtStreetView());
            ideol.addExtension(new ExtAdmin());
            ideol.addExtension(new ExtMunicipios());

            // Atlas
            ideol.addExtension(new ExtSearchAtlas());

            </script>
      </body>

</html>
```

Extensión WMS

Para la obtención de la información WMS hay que comprobar, en primer lugar, si la capa es consultable y visible. El control debe realizar una comprobación sobre la capa seleccionada.

```
…
onClick: function(evt){
            var selectedNode = this.ideol.toc.selectedNode;
            if(selectedNode != null){
                  // Un control añadido en el mapa ya contiene una
referencia al mapa
                  var arrayLayers =
this.ideol.map.getLayersByName(selectedNode.text);
                  var layer = null;
                  if(arrayLayers.length == 1)
                        layer = arrayLayers[0];

                  if(layer != null){
                    if(layer.queryable){
                        if(layer.getVisibility()){
                              var xy = evt.xy;
                              this.infoWindow.show(layer, xy);

                    }
                    else{
                        IDEOL.UtilUI.showMessageWindow(Locale.getText("
                        txt_informacion"),
                        Locale.getText("msg_capa_seleccionada_no_visibl
                        e_mapa"), Ext.MessageBox.WARNING);
                        }
                    }
                    else{
                        IDEOL.UtilUI.showMessageWindow(Locale.getText("
                        txt_informacion"),
                        Locale.getText("msg_capa_seleccionada_no_consul
                        table"), Ext.MessageBox.WARNING);
                        }
                  }
            }
            else{
                  IDEOL.UtilUI.showMessageWindow(Locale.getText("txt_in
                  formacion"),
                  Locale.getText("msg_seleccionar_capa_consultable_obte
                  ner_info"), Ext.MessageBox.WARNING);
            }

…
```

getColumnModel

Mediante este método se cargan las features a mostrar en el gridPanel, mostrando visibles las diez primeras columnas, ocultando las features geometry y fid.

```
this.getColumnModel = function(describeFeatureData){
            var columns = [];

            for(var i=0; i < describeFeatureData.length; i++){
                  var dataHeaderTitle =
describeFeatureData[i].split(".")[1];
                  if(describeFeatureData[i] == 'fid')
```

```
                        columns.push({header: 'Fid', dataIndex:
                        describeFeatureData[i], sortable: true, hidden:
                        true});
                  else if(describeFeatureData[i] == 'geometry')
                        columns.push({header: 'Geometry', dataIndex:
                        describeFeatureData[i], sortable: true, hidden:
                        true});
                  else if(columns.length < 10){
                        columns.push({header: dataHeaderTitle,
                        dataIndex: describeFeatureData[i], sortable:
                        true, editor: new Ext.form.TextField()});
                  }
                  else{
                        columns.push({header: dataHeaderTitle,
                        dataIndex: describeFeatureData[i], sortable:
                        true, hidden: true, editor: new
                        Ext.form.TextField()});
                  }
            }

            var columnModel = new Ext.grid.ColumnModel({columns:
columns});

            return columnModel;
      };

…
```

Impresión

Sobre estas líneas se muestra el código que hace referencia al lado cliente de la impresión.

```
Ext.namespace("ExtExport.Window");

ExtExport.Window.InfoReport = OpenLayers.Class({

      infoReportPanel: null,

      win: null,

      mask: null,

      wmcManager: new IDEOL.Manager.WMC(),

      printManager: new IDEOL.Manager.Print.Print(),

      initialize: function(){
            this.createComponents();
            this.prepareComponents();
      },
      createComponents: function(){

            this.infoReportPanel = new ExtExport.Panel.InfoReport();

            this.win = new Ext.Window({
                  layout: 'fit',
```

```
                bodyStyle:  'padding: 2px 2px 2px',
                modal: false,
                title: Locale.getText("txt_detalle_impresion"),
                constrainHeader: true,
                width: 300,
                collapsible: true,
                autoHeight: true,
                maximizable: false,
                resizable: false,
                draggable: true,
                closeAction: "hide",
                plain: true,
                border: false
            });
        },
        prepareComponents: function(){
            this.win.add(this.infoReportPanel);

            this.win.addButton({text:
            Locale.getText("txt_imprimir")},function (button, event){

                this.printReport();
            }, this);

            this.win.addButton({text: Locale.getText("txt_cerrar")},
            function(button, event){
                this.win.hide();
            }, this);

            this.win.on('afterrender', function(){
                this.mask = new Ext.LoadMask(this.win.body,{msg:
                Locale.getText("msg_preparando_documento")});
            }, this);
        },
            show: function(){

            if(!this.win.isVisible()){
                this.win.show();
                this.win.center();
            }

            this.win.expand();
        },

        setMap: function(map){
            this.map = map;
        },

        getPrintRequest: function(){
            var printRequest = new IDEOL.Manager.Print.PrintRequest();

            var scale = ""+this.map.getScale()+"";
            scale = "1:"+scale.split(".")[0];

            printRequest.project = Config.ExtExport.REPORT_NAME;
            printRequest.title =
            this.infoReportPanel.tituloTextField.getValue() .trim();
            printRequest.description =
            this.infoReportPanel.descripcionTextArea. getValue();
            printRequest.mapWMC =
this.wmcManager.getWMCStringMap(this.map);
```

```
            printRequest.scale = scale;
            printRequest.size =
this.infoReportPanel.comboSize.getValue();

            return printRequest;
      },

      onPrintReportSuccess: function(fileURL){
            this.mask.hide();
            window.open(fileURL);
      },

      onPrintReportFailure:  function(){
            this.mask.hide();
            IDEOL.UtilUI.showMessageWindow(
                  Locale.getText("txt_informacion"),
                  Locale.getText("msg_error_preparar_documento"),
                  Ext.MessageBox.ERROR
            );
      },

      printReport: function(){

            if(!this.infoReportPanel.getForm().isValid())
                  return;

            this.mask.show();

            var printRequest = this.getPrintRequest();

            this.printManager.printReport(
                  printRequest,
                  this.onPrintReportSuccess,
                  this.onPrintReportFailure,
                  this);
      }
});
```

Definición de los márgenes de la plantilla para un formato A4.

```
IMAGEIO_READ_TIMEOUT = 8000
MAP_IMAGE_WIDTH=535
MAP_IMAGE_HEIGHT=450
ATLAS.TEMPLATE_NAME=atlas
ATLAS.TEMPLATE_JASPER=atlasTemplate.jasper
ATLAS.BANNER=atlasBanner.png
ATLAS.MAP_IMAGE_WIDTH=502
ATLAS.MAP_IMAGE_HEIGHT=430
ATLAS.OVERVIEW_MAP_IMAGE_WIDTH=1
ATLAS.OVERVIEW_MAP_IMAGE_HEIGHT=1
ATLAS.LEGEND_IMAGE_WIDTH=502
ATLAS.LEGEND_IMAGE_HEIGHT=151
ATLAS.DISPOSICION=true
```

Report Request realiza una petición de los atributos al servlet mediante el método getParameter.

```
import javax.servlet.http.HttpServletRequest;


public class ReportRequest {

	String project = null;
	String title = null;
	String description = null;
	String mapWMC = null;
	String overViewMapWMC = null;
	String scale = null;
	String size = null;
	boolean image = false;

	public ReportRequest(HttpServletRequest request) throws
InvalidReportRequestException{
		this.project = request.getParameter("project");
		if(this.project == null)
			throw new InvalidReportRequestException();

		this.mapWMC = request.getParameter("mapWMC");
		if(this.mapWMC == null)
			throw new InvalidReportRequestException();

		this.title = request.getParameter("title");
		this.description = request.getParameter("description");
		this.overViewMapWMC =
request.getParameter("overViewMapWMC");
		this.scale = request.getParameter("scale");
		this.size = request.getParameter("size");

		String imageParam = request.getParameter("image");
		if(imageParam != null && imageParam.equals("true"))
			this.image = true;
	}

	…	//métodos get de cada atributo

	public boolean isImage() {
		return image;
	}
}
```

ReportBean obtiene y almacena los atributos del jasper.

```
import java.util.HashMap;


public class ReportBean {

	private String id;
	private String template;
	private String title;
	private String description;
	private String scale;
	private String size;
	private String fecha;
```

```
	private String imagesDirPath;

	HashMap<String, String> imagesMap = new HashMap<String,
String>();


	public ReportBean(){
	}
	…	//métodos get y set para cada atributo


	}
```

Report Properties obtiene y almacena las propiedades del jasper.

```
import java.util.HashMap;

public class ReportProperties {
	private String templateJasper;
	private String bannerName;
	private int mapImageWidth;
	private int mapImageHeight;
	private int legendImageWidth;
	private int legendImageHeight;
	private int littleMapImageWidth;
	private int littleMapImageHeight;
	boolean disposicion;
	HashMap<String, String> imagesMap = new HashMap<String,
String>();

	public ReportProperties(){
	}
	…	//métodos get y set para cada atributo
}
```

ReportMgr

```
import java.io.File;
…

public class ReportMgr {

	private static Logger logger =
Logger.getLogger(ReportMgr.class);
	static int DEFAULT_MAP_IMAGE_WIDTH;
	static int DEFAULT_MAP_IMAGE_HEIGHT;
	String workDirPath;
	String imagesDirPath;
	String idsession;
	WMSMgr wmsMgr;
	WMCMgr wmcMgr;

	static{

		try {
			DEFAULT_MAP_IMAGE_WIDTH =
			Integer.valueOf(Configuracion.getConfig().getProperty
			("MAP_IMAGE_WIDTH")).intValue();
			DEFAULT_MAP_IMAGE_HEIGHT =
			Integer.valueOf(Configuracion.getConfig().getProperty
			("MAP_IMAGE_HEIGHT")).intValue();
```

```
            } catch (NumberFormatException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            } catch (ConfiguracionException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }
      public ReportMgr(String workDirPath, String imagesDirPath,
String idsession){
            this.workDirPath = workDirPath;
            this.imagesDirPath = imagesDirPath;
            this.idsession = idsession;

            this.wmsMgr = new WMSMgr();
            this.wmcMgr = new WMCMgr();
      }
      public ReportBean getReportBean(ReportRequest reportRequest)
      throws ConfiguracionException,
      ValidateException, IOException{
            ReportBean reportBean = new ReportBean();

            prepareFixedAttrsForBean(reportRequest, reportBean);
            ReportProperties reportProperties =
getReportProperties(reportRequest);
            prepareImagesForBean(reportProperties, reportRequest,
reportBean);
            prepareBeanFromProperties(reportProperties, reportBean);
            prepareBaseBean(reportBean);
            return reportBean;
      }
      private void prepareFixedAttrsForBean(ReportRequest
      reportRequest, ReportBean reportBean){
            String title = reportRequest.getTitle();
            String description = reportRequest.getDescription();
            String scale = reportRequest.getScale();
            String size = reportRequest.getSize();

            reportBean.setTitle(title);
            reportBean.setDescription(description);
            reportBean.setScale(scale);
            reportBean.setSize(size);
      }
      public ReportProperties getReportProperties(ReportRequest
      reportRequest) throws ConfiguracionException{
            String project = reportRequest.getProject();

            ReportProperties reportProperties = new ReportProperties();

            String templateJasper = null;
            String bannerName = null;

            int mapImageWidth = 0;
            int mapImageHeight = 0;
            int legendImageWidth = 0;
            int legendImageHeight = 0;
            int littleMapImageWidth = 0;
            int littleMapImageHeight = 0;

            boolean disposicion = false;
```

```
            if(project.toLowerCase().equals(Configuracion.getConfig().g
            etProperty("ATLAS.TEMPLATE_NAME"))){
                  templateJasper =
                  Configuracion.getConfig().getProperty("ATLAS.TEMPLATE
                  _JASPER");
                  bannerName =
Configuracion.getConfig().getProperty("ATLAS.BANNER");

                  mapImageWidth =
                  Integer.valueOf((Configuracion.getConfig().getPropert
                  y("ATLAS.MAP_IMAGE_WIDTH")));
                  mapImageHeight =
                  Integer.valueOf((Configuracion.getConfig().getPropert
                  y("ATLAS.MAP_IMAGE_HEIGHT")));

                  legendImageWidth =
                  Integer.valueOf((Configuracion.getConfig().getPropert
                  y("ATLAS.LEGEND_IMAGE_WIDTH")));
                  legendImageHeight =
                  Integer.valueOf((Configuracion.getConfig().getPropert
                  y("ATLAS.LEGEND_IMAGE_HEIGHT")));

                  littleMapImageWidth =
                  Integer.valueOf((Configuracion.getConfig().getPropert
                  y("ATLAS.OVERVIEW_MAP_IMAGE_WIDTH")));
                  littleMapImageHeight =
                  Integer.valueOf((Configuracion.getConfig().getPropert
                  y("ATLAS.OVERVIEW_MAP_IMAGE_HEIGHT")));


                  String disposicionConfig =
                  Configuracion.getConfig().getProperty("ATLAS.DISPOSIC
                  ION");
                  disposicion = disposicionConfig.equals("true") ? true
: false;
            }

            reportProperties.setTemplateJasper(templateJasper);
            reportProperties.setBannerName(bannerName);

            reportProperties.setMapImageWidth(mapImageWidth);
            reportProperties.setMapImageHeight(mapImageHeight);

            reportProperties.setLegendImageWidth(legendImageWidth);
            reportProperties.setLegendImageHeight(legendImageHeight);


      reportProperties.setLittleMapImageWidth(littleMapImageWidth);

      reportProperties.setLittleMapImageHeight(littleMapImageHeight);

            reportProperties.setDisposicion(disposicion);

            return reportProperties;
      }

      private void prepareImagesForBean(ReportProperties
      reportProperties, ReportRequest reportRequest, ReportBean
      reportBean) throws ValidateException, IOException{
```

```
            ViewContextType mapContext =
wmcMgr.parseWMC(reportRequest.getMapWMC());

            int mapWidth = reportProperties.getMapImageWidth()*2;
            int mapHeight = reportProperties.getMapImageHeight()*2;

            int overViewMapWidth =
reportProperties.getLittleMapImageWidth();
            int overViewMapHeight =
reportProperties.getLittleMapImageHeight();

            int legendWidth = reportProperties.getLegendImageWidth()*2;
            int legendHeight =
reportProperties.getLegendImageHeight()*2;

            boolean disposicion = reportProperties.isDisposicion();

            File mapImage = wmsMgr.createMapImage(mapContext, mapWidth,
            mapHeight, workDirPath, idsession, false);
            File legendImage = wmsMgr.createLegendImage(mapContext,
            legendWidth, legendHeight, workDirPath, disposicion,
            idsession);
            File bannerImage = new File(imagesDirPath+reportProperties.
            getBannerName());

            reportBean.setId(mapContext.getId());

            reportBean.getImagesMap().put("mapImagePath",
            mapImage.getAbsolutePath());
            reportBean.getImagesMap().put("legendImagePath",
            legendImage.getAbsolutePath());
            reportBean.getImagesMap().put("bannerImagePath",
            bannerImage.getAbsolutePath());

      reportBean.getImagesMap().putAll(reportProperties.getImagesMap()
);
      }

      private void prepareBeanFromProperties(ReportProperties
      reportProperties, ReportBean reportBean){

      reportBean.setTemplate(reportProperties.getTemplateJasper());

            SimpleDateFormat formato = new SimpleDateFormat(
"dd/MM/yyyy");
            Date fechaActual = new Date();
            String fecha = formato.format(fechaActual);

            reportBean.setFecha(fecha);
      }

      private void prepareBaseBean(ReportBean reportBean) {
            reportBean.setImagesDirPath(this.imagesDirPath);
      }

}
```

El siguiente código hace referencia al archivo PrinterMgr, muestra cómo se completará la plantilla cuando no se trate de una imagen, generando el pdf.

```
public class PrinterMgr {

	private static Logger logger =
Logger.getLogger(PrinterMgr.class);

	ReportMgr reportMgr;

	public String generateReport(ReportRequest reportRequest, String
	workDirPath, String imagesDirPath, String reportsDirPath, String
	idsession) throws ConfiguracionException, ValidateException,
	IOException, JRException{

		reportMgr = new ReportMgr(workDirPath, imagesDirPath,
idsession);

		ReportBean reportBean =
reportMgr.getReportBean(reportRequest);

		String reportName = "report-"+reportBean.getId()+".pdf";
		String reportPath = workDirPath+reportName;

		List<ReportBean> reports = new ArrayList<ReportBean>();
		reports.add(reportBean);
		JRBeanCollectionDataSource datasource = new
		JRBeanCollectionDataSource(reports);

		String templatePath =
reportsDirPath+reportBean.getTemplate();

		HashMap<String, String> parameters =
reportBean.getImagesMap();

		JasperPrint jasperPrint =
		JasperFillManager.fillReport(templatePath, parameters,
		datasource);
		JasperExportManager.exportReportToPdfFile(jasperPrint,
reportPath);
		return reportName;
	}
}
```

Sobre estas líneas se muestra el código que genera la impresión en pdf: PrintWMCServlet.

```
public class PrintWMCServlet extends HttpServlet{

	Logger logger = Logger.getLogger(PrintWMCServlet.class);
	String WORKDIR;
	String IMAGESDIR;
	String WORKURL;
	String REPORTSDIR;
	String SESSION_ID;
```

```
	public void init(ServletConfig config) throws ServletException {
	 super.init(config);

		WORKDIR = getServletContext().getRealPath("/")+"tmp/";
	 IMAGESDIR = getServletContext().getRealPath("/")+"images/";
	 REPORTSDIR = getServletContext().getRealPath("/")+"reports/";
	}
	protected void doPost(HttpServletRequest request,
HttpServletResponse response){

	 logger.info("##### POST Request PrintWMCServlet #####");
	 String key = request.getParameter("key");
		String hostname = request.getRemoteHost();

	 if(!KeyManager.testKey(hostname, key)){
		response.setStatus(401);
		return;
	 }
	 if(WORKURL == null)
		WORKURL = "http://"+request.getServerName()+
		":"+request.getServerPort()+request.getContextPath()+"/tmp/
		";

		HttpSession session = request.getSession();
		SESSION_ID = session.getId();
		try{
		 ReportRequest reportRequest = new ReportRequest(request);
		 if(reportRequest.isImage()){
			WMSMgr wmsMgr = new WMSMgr();
			WMCMgr wmcMgr = new WMCMgr();

			ViewContextType mapContext = wmcMgr.parseWMC
		(reportRequest.getMapWMC());
			int mapWidth=
		mapContext.getGeneral().getWindow().getWidth().intValue();
			int mapHeight=
		mapContext.getGeneral().getWindow().getHeight().intValue();
			File file = wmsMgr.createMapImage(mapContext,
			mapWidth, mapHeight, WORKDIR, SESSION_ID, false);
			String imageURL = WORKURL+file.getName();

	response.getOutputStream().write(imageURL.getBytes("UTF-8"));
		 }
		 else{
			PrinterMgr printerMgr = new PrinterMgr();

			String fileName =
			printerMgr.generateReport(reportRequest, WORKDIR,
			IMAGESDIR, REPORTSDIR, SESSION_ID);

			String reportURL = WORKURL+fileName;

	response.getOutputStream().write(reportURL.getBytes("UTF-8"));
		 }
		}
		catch(InvalidReportRequestException e){
			logger.info("ERROR: Petición inválida.");
		}
		catch (ConfiguracionException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
```

```
            }
            catch (ValidateException e) {
                  logger.info("ERROR: No se ha podido validar el
contexto WMC");
            }
            catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
            catch (JRException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }

      private String convertStreamToString(InputStream is) {
             /*
              * To convert the InputStream to String we use the
BufferedReader.readLine()
              * method. We iterate until the BufferedReader return null
which means
              * there's no more data to read. Each line will appended
to a StringBuilder
              * and returned as String.
              */
             BufferedReader reader = new BufferedReader(new
InputStreamReader(is));
             StringBuilder sb = new StringBuilder();

             String line = null;
             try {
                 while ((line = reader.readLine()) != null) {
                     sb.append(line + "\n");
                 }
             } catch (IOException e) {
                 e.printStackTrace();
             } finally {
            try {
                     is.close();
                 } catch (IOException e) {
                     e.printStackTrace();
                 }
             }

             return sb.toString();
      }

      private String getFileContent(String path) throws IOException{
            FileInputStream fileInputStream = new FileInputStream
(path);
            byte[] b = new byte[fileInputStream.available()];
            fileInputStream.read(b);
            fileInputStream.close ();
            return new String (b);
      }
}
```

Extensión Atlas

ExtSearchAtlas.js

```
Ext.namespace("ExtSearchAtlas");

// Añade la traducción correspondiente de la extensión.
Locale.appendLocale(Locale.ExtSearchAtlas[Locale.getLang()]);

/**
 * Class: ExtSearchAtlas
 *
 * Descripción de la extensión.
 *
 * Inherits from:
 *  - <IDEOL.Extension>
 */
ExtSearchAtlas = OpenLayers.Class(IDEOL.Extension, {

      /**
     * Property: infoBBoxButton
     * {ExtSearchAtlas.Button.InfoBBox}
     */
      infoBBoxButton: null,

      /**
     * Property: infoBBoxControl
     * {ExtSearchAtlas.Control.InfoBBox}
     */
      infoBBoxControl: null,

      /**
     * Property: eSearchButton
     * {ExtSearchAtlas.Control.ESearch}
     */
      eSearchButton: null,

      /**
     * Property: eSearchWindow
     * {ExtSearchAtlas.Window.ESearch}
     */
      eSearchWindow: null,

      /**
     * Property: aSearchButton
     * {ExtSearchAtlas.Control.ASearch}
     */
      aSearchButton: null,

      /**
     * Property: eSearchWindow
     * {ExtSearchAtlas.Window.ASearch}
     */
      aSearchWindow: null,
```

```
/**
 * Property: gridResultWindow
 * {ExtSearchAtlas.Window.GridResult}
 */
gridResultWindow: null,

/**
 * Property: helpButton
 * {ExtSearchAtlas.Control.Button}
 */
helpButton: null,

/**
 *    Property: helpWindow
 * {ExtSearchAtlas.Window.HelpWindow}
 */
helpWindow: null,

/**
 * Constructor: ExtSearchAtlas
 *
 * Inicializa los componentes de la extensión.
 *
 */
initialize: function() {
      this.createComponents();
},

/**
 * Method: createComponents
 */
createComponents: function(){

      this.infoBBoxButton = new ExtSearchAtlas.Button.InfoBBox();
      this.infoBBoxButton.on("click", function(){
            this.activateControl(this.infoBBoxControl);
      }, this);

      this.eSearchButton = new ExtSearchAtlas.Button.ESearch();

      this.eSearchButton.on("click", function(){
            this.eSearchWindow.show();
      }, this);

      this.aSearchButton = new ExtSearchAtlas.Button.ASearch();

      this.aSearchButton.on("click", function(){
            this.aSearchWindow.show();
      }, this);

      this.helpButton = new ExtSearchAtlas.Button.Help();
      this.helpButton.on("click", function(){
            if (Locale.getLang() == 'es')
this.helpWindow.showHTML(Locale.getText("msg_ayuda"),Config.ExtS
earchAtlas.HTML_PATH_HELP_ES);
            else
this.helpWindow.showHTML(Locale.getText("msg_ayuda"),Config.ExtS
earchAtlas.HTML_PATH_HELP_VAL);
      }, this);
},
```

```
	/**
	 * Event: onCreate
	 * Este evento se lanza para indicar que la aplicación está
lista.
	 * Añadir aquí los componentes de la extensión al layout de la
aplicación.
	 *
	 */
	onCreate: function(){
		var map = this.ideol.map;

		this.gridResultWindow = new
		ExtSearchAtlas.Window.GridResult();
		this.eSearchWindow = new
		ExtSearchAtlas.Window.ESearch({gridResultWindow:
		this.gridResultWindow});
		this.aSearchWindow = new
		ExtSearchAtlas.Window.ASearch({gridResultWindow:
		this.gridResultWindow});
		this.helpWindow = new IDEOL.Tool.HTMLTemplateView();

		// Crea el control y lo añade al mapa
		this.infoBBoxControl = new
		ExtSearchAtlas.Control.InfoBBox(this.ideol);
		map.addControl(this.infoBBoxControl);

		// Añade los botones al toolbar
		var layout = this.ideol.layout;
		layout.toolBar.addItem(this.infoBBoxButton);
		layout.toolBar.addItem(this.eSearchButton);
		layout.toolBar.addItem(this.aSearchButton);
		layout.toolBar.addItem(this.helpButton);
		layout.toolBar.doLayout();
	},

	/**
	 * Event: onUpdate
	 * Este evento se lanza para indicar que la aplicación se ha
actualizado.
	 * Informar a los componentes de la extensión que lo necesiten
el nuevo objeto de la aplicación.
	 *
	 */
	onUpdate: function(){
		// Es necesario recrear el control y añadirlo al mapa
		this.infoBBoxControl.destroy();
		this.infoBBoxControl = new
		ExtSearchAtlas.Control.InfoBBox(this.ideol);
		var map = this.ideol.map;
		map.addControl(this.infoBBoxControl);
	},

	/**
	* Method: activateControl
	* Parameters:
	* control - {OpenLayers.Control}
	*/
	activateControl: function(control){
		this.ideol.controlsManager.deactivateControls();
		control.activate();
```

```
      },
      CLASS_NAME: "ExtSearchAtlas"
});
```

Button/ASearch.js

```
Ext.namespace("ExtSearchAtlas.Button");

/**
 * Class: ExtSearchAtlas.Button.ASearch
 *
 * Inherits from:
 *  - <Ext.Toolbar.Button>
 */
ExtSearchAtlas.Button.ASearch = Ext.extend(Ext.Toolbar.Button, {
      /**
       * Property: iconCls
       * {String} Icono del botón
       */
      iconCls: 'ExtSearchAtlas_Button_ASearch_Icon',

      /**
       * Property: tooltip
       * {String} Tooltip del botón
       */
    tooltip: Locale.getText("msg_busqueda_concentraciones"),

    /**
     * Constructor: ExtSearchAtlas.Button.ASearch
     *
     */
    initComponent: function(){
      ExtSearchAtlas.Button.ASearch.superclass.initComponent.call(this
      );
      }
});
```

Button/ESearch.js

```
Ext.namespace("ExtSearchAtlas.Button");

/**
 * Class: ExtSearchAtlas.Button.ESearch
 *
 * Inherits from:
 *  - <Ext.Toolbar.Button>
 */
ExtSearchAtlas.Button.ESearch = Ext.extend(Ext.Toolbar.Button, {
      /**
       * Property: iconCls
       * {String} Icono del botón
       */
      iconCls: 'ExtSearchAtlas_Button_ESearch_Icon',
```

```
      /**
       * Property: tooltip
       * {String} Tooltip del botón
       */
    tooltip: Locale.getText("msg_busqueda_establecimientos"),

    /**
     * Constructor: ExtSearchAtlas.Button.ESearch
     *
     */
    initComponent: function(){
      ExtSearchAtlas.Button.ESearch.superclass.initComponent.call(this
      );
      }
});
```

6.4.10.4 Button/InfoBBox.js

```
Ext.namespace("ExtSearchAtlas.Button");

/**
 * Class: ExtSearchAtlas.Button.InfoBBox
 *
 * Inherits from:
 *  - <Ext.Toolbar.Button>
 */
ExtSearchAtlas.Button.InfoBBox = Ext.extend(Ext.Toolbar.Button, {
      /**
       * Property: iconCls
       * {String} Icono del botón
       */
      iconCls: 'ExtSearchAtlas_Button_InfoBBox_Icon',

      /**
       * Property: tooltip
       * {String} Tooltip del botón
       */
    tooltip: {title:Locale.getText("txt_informacion"),
    text:Locale.getText("msg_obtiene_informacion_capa_seleccionada")},

    toggleGroup: 'map',

    /**
       * Property: ideol
       * <IDEOL.App> Objeto de la aplicación.
       */
    ideol: null,

    /**
     * Constructor: ExtSearchAtlas.Button.InfoBBox
     *
     */
    initComponent: function(){
      ExtSearchAtlas.Button.InfoBBox.superclass.initComponent.call(thi
      s);
      },
```

```
        /**
         * Method: setIDEOL
         * Actualiza el objeto de la aplicación del botón.
         *
         * Parameters:
         * ideol - <IDEOL.App> - Objeto de la aplicación.
         */
        setIDEOL: function(ideol){
              this.ideol = ideol;
        }
});
```

Button/Help.js

```
Ext.namespace("ExtSearchAtlas.Button");

/**
 * Class: ExtSearchAtlas.Button.Help
 *
 * Inherits from:
 *  - <Ext.Toolbar.Button>
 */
ExtSearchAtlas.Button.Help = Ext.extend(Ext.Toolbar.Button, {
        /**
         * Property: iconCls
         * {String} Icono del botón
         */
        iconCls: 'ExtSearchAtlas_Button_Help_Icon',

        /**
         * Property: tooltip
         * {String} Tooltip del botón
         */
    tooltip: Locale.getText("msg_ayuda"),

    /**
     * Constructor: ExtSearchAtlas.Button.Help
     *
     */
    initComponent: function(){
      ExtSearchAtlas.Button.Help.superclass.initComponent.call(this);
        }
});
```

Control/InfoBBox.js

```
Ext.namespace("ExtSearchAtlas.Control");

ExtSearchAtlas.Control.InfoBBox = OpenLayers.Class(OpenLayers.Control,
{

        dataViewWindow: null,

        ideol: null,
```

```
    /**
     * Property: out
     * {Boolean} Should the control be used for zooming out?
     */
    out: false,

      /**
     * Method: draw
     */
    draw: function() {
        this.handler = new OpenLayers.Handler.Box( this, {done:
this.showInfo}, {keyMask: this.keyMask} );
    },

    initialize: function(ideol) {
      this.ideol = ideol;

      this.dataViewWindow = new
ExtSearchAtlas.Window.DataView(this.ideol.map);

            OpenLayers.Control.prototype.initialize.apply(this);
    },


    checkWMSLayer: function(nodeText){
      var layer;
      var arrayLayers = this.map.getLayersByName(nodeText);

      if(arrayLayers.length == 1)
                  layer = arrayLayers[0];

      if(layer == null)
            return false;

      if(layer.queryable){
                  if(!layer.getVisibility()){

      IDEOL.UtilUI.showMessageWindow(Locale.getText("txt_informacion")
, Locale.getText("msg_capa_seleccionada_no_visible_mapa"),
Ext.MessageBox.WARNING);
                        return false;
                  }
                  return true;
            }
            else{

      IDEOL.UtilUI.showMessageWindow(Locale.getText("txt_informacion")
,Locale.getText("msg_capa_seleccionada_no_consultable"),
Ext.MessageBox.WARNING);
            }

      return false;
    },

    showInfo: function(position){
      var selectedNode = this.ideol.toc.selectedNode;
            if(selectedNode){
                  var wmsLayerName = selectedNode.text;
                  if(this.checkWMSLayer(wmsLayerName)){
                        var layerWFS =
this.getWFSLayerByName(wmsLayerName);
```

```
                        if(layerWFS != null){
                              var bounds = this.getBounds(position);

                              if(bounds != null){
                                    bounds.transform(new
OpenLayers.Projection(this.map.getProjection()), new
OpenLayers.Projection(layerWFS.projection.projCode));
                                    this.dataViewWindow.show(layerWFS,
bounds);
                              }
                        }
                  }
            }
            else{

      IDEOL.UtilUI.showMessageWindow(Locale.getText("txt_informacion")
,Locale.getText("msg_seleccionar_capa_consultable_obtener_info"),
Ext.MessageBox.WARNING);
            }
    },

    /**
     *
     */
    getWFSLayerByName: function(layerName){
      var layer = null;
      if(layerName ==
Config.ExtSearchAtlas.WFSLayers.ESTABLECIMIENTOS.name)
            layer = Config.ExtSearchAtlas.WFSLayers.ESTABLECIMIENTOS;
            else if(layerName ==
Config.ExtSearchAtlas.WFSLayers.CONCENTRACIONES_COMERCIALES.name)
            layer =
Config.ExtSearchAtlas.WFSLayers.CONCENTRACIONES_COMERCIALES;
            else if(layerName ==
Config.ExtSearchAtlas.WFSLayers.MUNICIPIOS.name)
            layer = Config.ExtSearchAtlas.WFSLayers.MUNICIPIOS;
            else if(layerName ==
Config.ExtSearchAtlas.WFSLayers.COMERCIOS_INNOVACION.name)
            layer =
Config.ExtSearchAtlas.WFSLayers.COMERCIOS_INNOVACION;
            else if(layerName ==
Config.ExtSearchAtlas.WFSLayers.ANTENAS_LOCALES.name)
            layer = Config.ExtSearchAtlas.WFSLayers.ANTENAS_LOCALES;
            else if(layerName ==
Config.ExtSearchAtlas.WFSLayers.CAMARAS_COMERCIO.name)
            layer = Config.ExtSearchAtlas.WFSLayers.CAMARAS_COMERCIO;

      return layer;
    },

    getBounds : function(position){
            var bounds = null;
            if (position instanceof OpenLayers.Bounds) {
            if (!this.out) {
                var minXY = this.map.getLonLatFromPixel(
                              new OpenLayers.Pixel(position.left,
position.bottom));
                var maxXY = this.map.getLonLatFromPixel(
                              new OpenLayers.Pixel(position.right,
position.top));
                bounds = new OpenLayers.Bounds(minXY.lon, minXY.lat,
```

```
                                                maxXY.lon, maxXY.lat);
            } else {
                var pixWidth = Math.abs(position.right-position.left);
                var pixHeight = Math.abs(position.top-
position.bottom);
                var zoomFactor = Math.min((this.map.size.h /
pixHeight),
                    (this.map.size.w / pixWidth));
                var extent = this.map.getExtent();
                var center = this.map.getLonLatFromPixel(
                    position.getCenterPixel());
                var xmin = center.lon -
(extent.getWidth()/2)*zoomFactor;
                var xmax = center.lon +
(extent.getWidth()/2)*zoomFactor;
                var ymin = center.lat -
(extent.getHeight()/2)*zoomFactor;
                var ymax = center.lat +
(extent.getHeight()/2)*zoomFactor;
                bounds = new OpenLayers.Bounds(xmin, ymin, xmax,
ymax);
            }
        }
            else { // it's a pixel
                 var offsetPoint = 14;

            if (!this.out) {

                 var min = {
                       x: position.x-offsetPoint,
                       y: position.y+offsetPoint
                 };

                 var minLonLat = this.map.getLonLatFromPixel(min);

                 var max = {
                       x: position.x+offsetPoint,
                       y: position.y-offsetPoint
                 };

                 var maxLonLat = this.map.getLonLatFromPixel(max);

                 bounds = new OpenLayers.Bounds(minLonLat.lon,
minLonLat.lat, maxLonLat.lon, maxLonLat.lat);

            } else {

this.map.setCenter(this.map.getLonLatFromPixel(position),
this.map.getZoom() - 1);
            }
        }
            return bounds;
      },

      CLASS_NAME: "ExtSearchAtlas.Control.InfoBBox"
});
```

Panel/Agrupaciones/Centro.js

```
Ext.namespace("ExtSearchAtlas.Panel.Agrupaciones");

/**
 * Class: ExtSearchAgr.Panel.Centro
 *
 * Inherits from:
 *  - <Ext.Panel>
 */
ExtSearchAtlas.Panel.Agrupaciones.Centro = Ext.extend(Ext.FormPanel, {

      frame: true,
      title: Locale.getText("txt_tipo_centro"),
      originalTitle: Locale.getText("txt_tipo_centro"),
      autoWidth: true,
      height: 95,
      buttonAlign: 'center',
      labelWidth: 70,

      layer:
Config.ExtSearchAtlas.WFSLayers.CONCENTRACIONES_COMERCIALES,

      mask: null,

      wfsManager: new IDEOL.Manager.WFS(),

      filterManager: new IDEOL.Manager.Filter(),

      storeCentros: new Ext.data.Store({
            reader: new Ext.data.JsonReader(
                  {id: 'tipo_centr'},
                  ['tipo_centr']
             ),
             sortInfo: {
                field: 'tipo_centr',
                direction: 'ASC'
             }
      }),

    comboCentros: new Ext.form.ComboBox ({
      displayField:'tipo_centr',
            valueField:'tipo_centr',
            fieldLabel: Locale.getText("txt_tipo_centro"),
            name: 'tipo_centr',
            mode: 'local',
            typeAhead: false,
          triggerAction: 'all',
          selectOnFocus:true,
          editable: false,
          anchor:'95%'
    }),

      getCentros: function(wfsFeatures){
            var blankFeature = {tipo_centr: '---
'+Locale.getText("txt_seleccionar")+' ---'};

            var centros = [blankFeature];

            for(i=0; i<wfsFeatures.length; i++){
```

```
                var centro =  wfsFeatures[i].data;
                centros.push(centro);
            }
            return centros;
        },

        onWFSQuerySuccess: function(wfsFeatures){
            var centros = this.getCentros(wfsFeatures);
            this.loadCombo(centros);
            this.mask.hide();
        },

        onWFSQueryFailure: function(response){
            this.mask.hide();
            IDEOL.UtilUI.showMessageWindow(
                Locale.getText("txt_informacion"),
                Locale.getText("msg_error_realizar_consulta"),
                Ext.MessageBox.ERROR
            );
        },

        loadCombo: function(centros){
            this.storeCentros.loadData(centros);
        },

        requestFeatures: function(){
            this.mask.show();

            var attributes = this.storeCentros.fields.keys;

            var filter = this.filterManager.getComparisonFilter(
                        OpenLayers.Filter.Comparison.LIKE,
                        'tipo_centr',
                        "*",
                        false);

            this.wfsManager.getFeaturesByFilter(
                this.layer,
                attributes,
                filter,
                this.onWFSQuerySuccess,
                this.onWFSQueryFailure,
                this
            );
        },

        initialize: function(){

            this.comboCentros.store = this.storeCentros;
            this.add(this.comboCentros);

            this.on('render', function( panel ){
                this.mask = new Ext.LoadMask(this.body);
            }, this);

            this.on('afterrender', function( panel ){
                this.requestFeatures();
            }, this);

            this.comboCentros.on('select', function(combo, record,
index){
```

```
                  if(record.data.tipo_centr != '---
'+Locale.getText("txt_seleccionar")+' ---'){
                        this.setTitle('<span style="color:
green;">'+this.originalTitle+'</span>');
                  }
                  else{
                        this.setTitle(this.originalTitle);
                  }
            }, this);

      },

      /**
      * Constructor: ExtSearchAtlas.Panel.Agrupaciones.Centro
      *
      */
     initComponent : function(){

       ExtSearchAtlas.Panel.Agrupaciones.Centro.superclass.initComponen
t.call(this);

            this.initialize();
     }
});
```

Panel/Agrupaciones/Dirección.js

```
Ext.namespace("ExtSearchAtlas.Panel.Agrupaciones");

/**
 * Class: ExtSearchAtlas.Panel.Agrupaciones.Direccion
 *
 * Inherits from:
 *  - <Ext.Panel>
 */
ExtSearchAtlas.Panel.Agrupaciones.Direccion =
Ext.extend(Ext.FormPanel, {

      frame: true,
      title: Locale.getText("txt_direccion"),
      originalTitle: Locale.getText("txt_direccion"),
      autoWidth: true,
      height: 95,
      buttonAlign: 'center',
      labelWidth: 70,

      layer: Config.ExtSearchAtlas.WFSLayers.MUNICIPIOS,

      mask: null,

      wfsManager: new IDEOL.Manager.WFS(),

      filterManager: new IDEOL.Manager.Filter(),

      direccionTextField: new Ext.form.TextField(
            {
                  fieldLabel: Locale.getText("txt_direccion"),
                  name: 'direccion',
                  enableKeyEvents: true,
```

```
                anchor:'95%'
            }
      ),

      storeMunicipios: new Ext.data.Store({
            reader: new Ext.data.JsonReader({},
                  ['nombre']
             ),
             sortInfo: {
                field: 'nombre',
                direction: 'ASC'
             }
      }),

    storeProvincias: new Ext.data.SimpleStore({
            fields: ['codigo','nombre'],
            data : [
                ['00', '--- '+Locale.getText("txt_seleccionar")+' ---
'],
              ['03','Alicante'],
              ['12','Castellón'],
              ['46','Valencia']
          ]
    }),

    comboMunicipios: new Ext.form.ComboBox ({
      displayField:'nombre',
            //valueField:'nombre',
            fieldLabel: Locale.getText("txt_municipio"),
            name: 'municipio',
            mode: 'local',
            //typeAhead: false,
          triggerAction: 'all',
          selectOnFocus:true,
          editable: false,
          anchor:'95%'
    }),

    comboProvincias: new Ext.form.ComboBox ({
            displayField:'nombre',
            fieldLabel: Locale.getText("txt_provincia"),
            value: '--- '+Locale.getText("txt_seleccionar")+' ---',
            name: 'nombre',
            mode: 'local',
            //typeAhead: false,
            typeAhead: true,
            forceSelection: true,
          triggerAction: 'all',
          //emptyText:'Seleccione una provincia...',
          selectOnFocus:true,
          editable: false,
          anchor:'95%'
    }),

    keyEventHandler: function(object, event){
            this.updateTitleColor();
      },

      getMunicipios: function(wfsFeatures){
            var blankFeature = {nombre: '---
'+Locale.getText("txt_seleccionar")+' ---'};
```

```
            var municipios = [blankFeature];

            for(i=0; i<wfsFeatures.length; i++){
                  var municipio =  wfsFeatures[i].data;
                  municipios.push(municipio);
            }
            return municipios;
      },

      onWFSQuerySuccess: function(wfsFeatures){
            var municipios = this.getMunicipios(wfsFeatures);
            this.loadComboMunicipios(municipios);
            this.mask.hide();
      },

      onWFSQueryFailure: function(response){
            this.mask.hide();
            IDEOL.UtilUI.showMessageWindow(
                  Locale.getText("txt_informacion"),
                  Locale.getText("msg_error_realizar_consulta"),
                  Ext.MessageBox.ERROR
            );
      },

      loadComboMunicipios: function(municipios){
            this.storeMunicipios.loadData(municipios);
      },

      requestFeatures: function(codprov){
            this.mask.show();

            var attributes = this.storeMunicipios.fields.keys;

            var filter = this.filterManager.getComparisonFilter(
                        OpenLayers.Filter.Comparison.EQUAL_TO,
                        'codprov',
                        codprov,
                        false);

            this.wfsManager.getFeaturesByFilter(
                  this.layer,
                  attributes,
                  filter,
                  this.onWFSQuerySuccess,
                  this.onWFSQueryFailure,
                  this
            );
      },

      updateTitleColor: function(){
            var update =false;

            var provincia = this.comboProvincias.getValue();
            var municipio = this.comboMunicipios.getValue();
            var direccion = this.direccionTextField.getValue();

            if(provincia != null && provincia.trim() != "" && provincia
!= '--- '+Locale.getText("txt_seleccionar")+' ---'){
                  update = true;
            }
```

```
            else if(direccion != null && direccion.trim() != ""){
                  update = true;
            }

            if(update){
                  this.setTitle('<span style="color:
green;">'+this.originalTitle+'</span>');
            }
            else{
                  this.setTitle(this.originalTitle);
            }
      },

    initialize: function(){

            this.on('render', function( panel ){
                  this.mask = new Ext.LoadMask(this.body);
            }, this);

            this.comboProvincias.store = this.storeProvincias;
            this.comboMunicipios.store = this.storeMunicipios;

            this.comboProvincias.on('select', function(combo, record,
index){

                  this.storeMunicipios.removeAll();
                  this.comboMunicipios.setValue("");

                  this.updateTitleColor();

                  if(record.data.codigo != '00'){
                        this.requestFeatures(record.data.codigo);

                  }
            }, this);

            this.comboMunicipios.on('select', function(combo, record,
index){
                  this.updateTitleColor();
            }, this);

            this.direccionTextField.on('keyup', function(object,
event){
                  this.keyEventHandler(object, event);
            }, this);

            this.add(this.comboProvincias);
            this.add(this.comboMunicipios);
            this.add(this.direccionTextField);
      },

    /**
     * Constructor: ExtSearchAtlas.Panel.Agrupaciones.Direccion
     *
     */
    initComponent : function(){

      ExtSearchAtlas.Panel.Agrupaciones.Direccion.superclass.initCompo
nent.call(this);

            this.initialize();
```

```
    }
});
```

Panel/Agrupaciones/Nombre.js

```
Ext.namespace("ExtSearchAtlas.Panel.Agrupaciones");

/**
 * Class: ExtSearchAtlas.Panel.Agrupaciones.Nombre
 *
 * Inherits from:
 *  - <Ext.Panel>
 */
ExtSearchAtlas.Panel.Agrupaciones.Nombre = Ext.extend(Ext.FormPanel, {

      frame: true,
      title: Locale.getText("txt_nombre"),
      originalTitle: Locale.getText("txt_nombre"),
      autoWidth: true,
      height: 95,
      buttonAlign: 'center',
      labelWidth: 70,

      nombreTextField: new Ext.form.TextField(
            {
                  fieldLabel: Locale.getText("txt_nombre"),
              name: 'nombre',
              enableKeyEvents: true,
              anchor:'95%'
            }
      ),

      keyEventHandler: function(object, event){
            var rotulo = this.nombreTextField.getValue();
            if(rotulo != null && rotulo.trim() != ""){
                  this.setTitle('<span style="color:
green;">'+this.originalTitle+'</span>');
            }
            else{
                  this.setTitle(this.originalTitle);
            }
      },
      /**
     * Constructor: ExtSearchAtlas.Panel.Agrupaciones.Nombre
     *
     */
    initComponent : function(){

      ExtSearchAtlas.Panel.Agrupaciones.Nombre.superclass.initComponen
t.call(this);

            this.nombreTextField.on('keyup', function(object, event){
                  this.keyEventHandler(object, event);
            }, this);

            this.add(this.nombreTextField);
    }
});
```

Panel/Agrupaciones/Tab.js

```
Ext.namespace("ExtSearchAtlas.Panel.Agrupaciones");

/**
 * Class: ExtSearchAtlas.Panel.Agrupaciones.Tab
 *
 * Inherits from:
 *  - <Ext.Panel>
 */
ExtSearchAtlas.Panel.Agrupaciones.Tab = Ext.extend(Ext.TabPanel, {

      activeTab: 0,
      border: false,
      autoHeight: true,

      /**
     * Constructor: ExtSearchAtlas.Panel.Agrupaciones.Tab
     *
     */
    initComponent : function(){

      ExtSearchAtlas.Panel.Agrupaciones.Tab.superclass.initComponent.c
all(this);
    }
});
```

Panel/Establecimientos/Actividad.js

```
Ext.namespace("ExtSearchAtlas.Panel.Establecimientos");

/**
 * Class: ExtSearchAtlas.Panel.Establecimientos.Actividad
 *
 * Inherits from:
 *  - <Ext.Panel>
 */
ExtSearchAtlas.Panel.Establecimientos.Actividad =
Ext.extend(Ext.FormPanel, {

      frame: true,
      title: Locale.getText("txt_actividad"),
      originalTitle: Locale.getText("txt_actividad"),
      autoWidth: true,
      height: 95,
      buttonAlign: 'center',
      labelWidth: 70,

      layer: Config.ExtSearchAtlas.WFSLayers.ESTABLECIMIENTOS,

      filterManager: new IDEOL.Manager.Filter(),

      mask: null,

      wfsManager: new IDEOL.Manager.WFS(),

      storeActividades: new Ext.data.Store({
```

```
            reader: new Ext.data.JsonReader(
                  {id: 'dactividad'},
                  ['dactividad']
             ),
             sortInfo: {
                field: 'dactividad',
                direction: 'ASC'
             }
      }),

    comboActividades: new Ext.form.ComboBox ({
      displayField:'dactividad',
            valueField:'dactividad',
            fieldLabel: Locale.getText("txt_actividad"),
            name: 'dactividad',
            mode: 'local',
            typeAhead: false,
          triggerAction: 'all',
          selectOnFocus:true,
          editable: false,
          anchor:'95%'
    }),

      getActividades: function(wfsFeatures){
            var blankFeature = {dactividad: '---
'+Locale.getText("txt_seleccionar")+' ---'};

            var actividades = [blankFeature];

            for(i=0; i<wfsFeatures.length; i++){
                  var actividad =  wfsFeatures[i].data;
                  actividades.push(actividad);
            }
            return actividades;
      },

      onWFSQuerySuccess: function(wfsFeatures){
            var actividades = this.getActividades(wfsFeatures);
            this.loadCombo(actividades);
            this.mask.hide();
      },

      onWFSQueryFailure: function(response){
            this.mask.hide();
            IDEOL.UtilUI.showMessageWindow(
                  Locale.getText("txt_informacion"),
                  Locale.getText("msg_error_realizar_consulta"),
                  Ext.MessageBox.ERROR
            );
      },

      loadCombo: function(actividades){
            this.storeActividades.loadData(actividades);
      },

      requestFeatures: function(){
            this.mask.show();

            var attributes = this.storeActividades.fields.keys;

            var filter = this.filterManager.getComparisonFilter(
```

```
                        OpenLayers.Filter.Comparison.LIKE,
                        'dactividad',
                        '*',
                        false);

            this.wfsManager.getFeaturesByFilter(
                  this.layer,
                  attributes,
                  filter,
                  this.onWFSQuerySuccess,
                  this.onWFSQueryFailure,
                  this
            );
      },

      initialize: function(){

            this.comboActividades.store = this.storeActividades;
            this.add(this.comboActividades);

            this.on('render', function( panel ){
                  this.mask = new Ext.LoadMask(this.body);
            }, this);

            this.on('afterrender', function( panel ){
                  this.requestFeatures();
            }, this);

            this.comboActividades.on('select', function(combo, record,
index){
                  if(record.data.dactividad != '---
'+Locale.getText("txt_seleccionar")+' ---'){
                        this.setTitle('<span style="color:
green;">'+this.originalTitle+'</span>');
                  }
                  else{
                        this.setTitle(this.originalTitle);
                  }
            }, this);

      },

      /**
     * Constructor: ExtSearchAtlas.Panel.Establecimientos.Actividad
     *
     */
    initComponent : function(){

      ExtSearchAtlas.Panel.Establecimientos.Actividad.superclass.initC
omponent.call(this);

            this.initialize();
    }
});
```

Panel/Establecimientos/Agrupación.js

```
Ext.namespace("ExtSearchAtlas.Panel.Establecimientos");

/**
 * Class: ExtSearchAtlas.Panel.Establecimientos.Agrupacion
 *
 * Inherits from:
 *  - <Ext.Panel>
 */
ExtSearchAtlas.Panel.Establecimientos.Agrupacion =
Ext.extend(Ext.FormPanel, {

      frame: true,
      title: Locale.getText("txt_agrupacion"),
      originalTitle: Locale.getText("txt_agrupacion"),
      autoWidth: true,
      height: 95,
      buttonAlign: 'center',
      labelWidth: 70,

      layer: Config.ExtSearchAtlas.WFSLayers.ESTABLECIMIENTOS,

      mask: null,

      wfsManager: new IDEOL.Manager.WFS(),

      filterManager: new IDEOL.Manager.Filter(),

      storeAgrupaciones: new Ext.data.Store({
            reader: new Ext.data.JsonReader(
                  {id: 'nom_agrup'},
                  ['nom_agrup']
             ),
             sortInfo: {
                field: 'nom_agrup',
                direction: 'ASC'
             }
      }),

      comboAgrupaciones: new Ext.form.ComboBox ({
      displayField:'nom_agrup',
            valueField:'nom_agrup',
            fieldLabel: Locale.getText("txt_agrupacion"),
            name: 'agrupacion',
            mode: 'local',
            typeAhead: false,
          triggerAction: 'all',
          selectOnFocus:true,
          editable: false,
          anchor:'95%'
    }),

      getAgrupaciones: function(wfsFeatures){
            var blankFeature = {nom_agrup: '---
'+Locale.getText("txt_seleccionar")+' ---'};

            var agrupaciones = [blankFeature];

            for(i=0; i<wfsFeatures.length; i++){
```

```
                var agrupacion =  wfsFeatures[i].data;
                agrupaciones.push(agrupacion);
        }
        return agrupaciones;
    },

    onWFSQuerySuccess: function(wfsFeatures){
        var agrupaciones = this.getAgrupaciones(wfsFeatures);
        this.loadCombo(agrupaciones);
        this.mask.hide();
    },

    onWFSQueryFailure: function(response){
        this.mask.hide();
        IDEOL.UtilUI.showMessageWindow(
                Locale.getText("txt_informacion"),
                Locale.getText("msg_error_realizar_consulta"),
                Ext.MessageBox.ERROR
        );
    },

    loadCombo: function(agrupaciones){
        this.storeAgrupaciones.loadData(agrupaciones);
    },

    requestFeatures: function(){
        this.mask.show();

        var attributes = this.storeAgrupaciones.fields.keys;

        var filter = this.filterManager.getComparisonFilter(
                    OpenLayers.Filter.Comparison.LIKE,
                    'nom_agrup',
                    '*',
                    false);

        this.wfsManager.getFeaturesByFilter(
                this.layer,
                attributes,
                filter,
                this.onWFSQuerySuccess,
                this.onWFSQueryFailure,
                this
        );
    },


initialize: function(){

        this.comboAgrupaciones.store = this.storeAgrupaciones;
        this.add(this.comboAgrupaciones);

        this.on('render', function( panel ){
                this.mask = new Ext.LoadMask(this.body);
        }, this);

        this.on('afterrender', function( panel ){
                this.requestFeatures();
        }, this);
```

```
            this.comboAgrupaciones.on('select', function(combo, record,
index){
                  if(record.data.nom_agrup != '---
'+Locale.getText("txt_seleccionar")+' ---'){
                        this.setTitle('<span style="color:
green;">'+this.originalTitle+'</span>');
                  }
                  else{
                        this.setTitle(this.originalTitle);
                  }
            }, this);

      },

      /**
       * Constructor: ExtSearchAtlas.Panel.Establecimientos.Agrupacion
       *
       */
     initComponent : function(){

       ExtSearchAtlas.Panel.Establecimientos.Agrupacion.superclass.init
Component.call(this);

            this.initialize();
     }
});
```

Panel/Establecimientos/Dirección.js

```
Ext.namespace("ExtSearchAtlas.Panel.Establecimientos");

/**
 * Class: ExtSearchAtlas.Panel.Establecimientos.Direccion
 *
 * Inherits from:
 *  - <Ext.Panel>
 */
ExtSearchAtlas.Panel.Establecimientos.Direccion =
Ext.extend(Ext.FormPanel, {

      frame: true,
      title: Locale.getText("txt_direccion"),
      originalTitle: Locale.getText("txt_direccion"),
      autoWidth: true,
      height: 95,
      buttonAlign: 'center',
      labelWidth: 70,

      layerMunicipios: Config.ExtSearchAtlas.WFSLayers.MUNICIPIOS,

      filterManager: new IDEOL.Manager.Filter(),

      mask: null,

      wfsManager: new IDEOL.Manager.WFS(),

      direccionTextField: new Ext.form.TextField(
            {
                  fieldLabel: Locale.getText("txt_direccion"),
```

```
                name: 'direccion',
                enableKeyEvents: true,
                anchor:'95%'
            }
      ),

      storeMunicipios: new Ext.data.Store({
            reader: new Ext.data.JsonReader({},
                  ['nombre']
             ),
             sortInfo: {
                field: 'nombre',
                direction: 'ASC'
             }
      }),

    storeProvincias: new Ext.data.SimpleStore({
            fields: ['codigo','nombre'],
            data : [
                ['00', '--- '+Locale.getText("txt_seleccionar")+' ---
'],
              ['03','Alicante'],
              ['12','Castellón'],
              ['46','Valencia']
          ]
    }),

    comboMunicipios: new Ext.form.ComboBox ({
      displayField:'nombre',
            fieldLabel: Locale.getText("txt_municipio"),
            name: 'municipio',
            mode: 'local',
            triggerAction: 'all',
            selectOnFocus:true,
            editable: false,
            anchor:'95%'
    }),

    comboProvincias: new Ext.form.ComboBox ({
            displayField:'nombre',
            fieldLabel: Locale.getText("txt_provincia"),
            value: '--- '+Locale.getText("txt_seleccionar")+' ---',
            name: 'nombre',
            mode: 'local',
            typeAhead: true,
            forceSelection: true,
            triggerAction: 'all',
            selectOnFocus:true,
            editable: false,
            anchor:'95%'
    }),

    keyEventHandler: function(object, event){
            this.updateTitleColor();
      },

      getMunicipios: function(wfsFeatures){
            var blankFeature = {nombre: '---
'+Locale.getText("txt_seleccionar")+' ---'};

            var municipios = [blankFeature];
```

```
            for(i=0; i<wfsFeatures.length; i++){
                  var municipio =  wfsFeatures[i].data;
                  municipios.push(municipio);
            }
            return municipios;
      },

      onWFSQuerySuccess: function(wfsFeatures){
            var municipios = this.getMunicipios(wfsFeatures);
            this.loadComboMunicipios(municipios);
            this.mask.hide();
      },

      onWFSQueryFailure: function(response){
            this.mask.hide();
            IDEOL.UtilUI.showMessageWindow(
                  Locale.getText("txt_informacion"),
                  Locale.getText("msg_error_realizar_consulta"),
                  Ext.MessageBox.ERROR
            );
      },

      loadComboMunicipios: function(municipios){
            this.storeMunicipios.loadData(municipios);
      },

      requestFeatures: function(codprov){
            this.mask.show();

            var attributes = this.storeMunicipios.fields.keys;

            var filter = this.filterManager.getComparisonFilter(
                        OpenLayers.Filter.Comparison.EQUAL_TO,
                        'codprov',
                        codprov,
                        false);

            this.wfsManager.getFeaturesByFilter(
                  this.layerMunicipios,
                  attributes,
                  filter,
                  this.onWFSQuerySuccess,
                  this.onWFSQueryFailure,
                  this
            );
      },

      updateTitleColor: function(){
            var update =false;

            var provincia = this.comboProvincias.getValue();
            var municipio = this.comboMunicipios.getValue();
            var direccion = this.direccionTextField.getValue();

            if(provincia != null && provincia.trim() != "" && provincia
!= '--- '+Locale.getText("txt_seleccionar")+' ---'){
                  update = true;
            }
            else if(direccion != null && direccion.trim() != ""){
                  update = true;
```

```
            }

            if(update){
                  this.setTitle('<span style="color:
green;">'+this.originalTitle+'</span>');
            }
            else{
                  this.setTitle(this.originalTitle);
            }
      },

    initialize: function(){

            this.on('render', function( panel ){
                  this.mask = new Ext.LoadMask(this.body);
            }, this);

            this.comboProvincias.store = this.storeProvincias;
            this.comboMunicipios.store = this.storeMunicipios;

            this.comboProvincias.on('select', function(combo, record,
index){

                  // El combo de municipios nunca podrá estar informado
si el combo de provincias no lo está
                  this.storeMunicipios.removeAll();
                  this.comboMunicipios.setValue("");

                  this.updateTitleColor();

                  if(record.data.codigo != '00'){
                        this.requestFeatures(record.data.codigo);

                  }
            }, this);

            this.comboMunicipios.on('select', function(combo, record,
index){
                  this.updateTitleColor();
            }, this);

            this.direccionTextField.on('keyup', function(object,
event){
                  this.keyEventHandler(object, event);
            }, this);

            this.add(this.comboProvincias);
            this.add(this.comboMunicipios);
            this.add(this.direccionTextField);
      },

      /**
     * Constructor: ExtSearchAtlas.Panel.Establecimientos
     *
     */
    initComponent : function(){

      ExtSearchAtlas.Panel.Establecimientos.Direccion.superclass.initC
omponent.call(this);

            this.initialize();
```

```
    }
});
```

Panel/Establecimientos/Nombre.js

```
Ext.namespace("ExtSearchAtlas.Panel.Establecimientos");

/**
 * Class: ExtSearchAtlas.Panel.Establecimientos.Nombre
 *
 * Inherits from:
 *  - <Ext.Panel>
 */
ExtSearchAtlas.Panel.Establecimientos.Nombre =
Ext.extend(Ext.FormPanel, {

      frame: true,
      title: Locale.getText("txt_nombre"),
      originalTitle: Locale.getText("txt_nombre"),
      autoWidth: true,
      height: 95,
      buttonAlign: 'center',
      labelWidth: 70,

      nombreTextField: new Ext.form.TextField(
            {
                  fieldLabel: Locale.getText("txt_nombre"),
              name: 'nombre',
              enableKeyEvents: true,
              anchor:'95%'
            }
      ),

      keyEventHandler: function(object, event){
            var rotulo = this.nombreTextField.getValue();
            if(rotulo != null && rotulo.trim() != ""){
                  this.setTitle('<span style="color:
green;">'+this.originalTitle+'</span>');
            }
            else{
                  this.setTitle(this.originalTitle);
            }
      },

      /**
     * Constructor: ExtSearchAtlas.Panel.Establecimientos.Nombre
     *
     */
    initComponent : function(){

      ExtSearchAtlas.Panel.Establecimientos.Nombre.superclass.initComp
onent.call(this);

            this.nombreTextField.on('keyup', function(object, event){
                  this.keyEventHandler(object, event);
            }, this);

            this.add(this.nombreTextField);
    }
});
```

Panel/Establecimientos/Tab.js

```
Ext.namespace("ExtSearchAtlas.Panel.Establecimientos");

/**
 * Class: ExtSearchAtlas.Panel.Establecimientos.Tab
 *
 * Inherits from:
 *  - <Ext.Panel>
 */
ExtSearchAtlas.Panel.Establecimientos.Tab = Ext.extend(Ext.TabPanel, {

      activeTab: 0,
      border: false,
      autoHeight: true,

      /**
     * Constructor: ExtSearchAtlas.Panel.Establecimientos.Tab
     *
     */
    initComponent : function(){

      ExtSearchAtlas.Panel.Establecimientos.Tab.superclass.initCompone
nt.call(this);
    }
});
```

Window/ASearch.js

```
Ext.namespace("ExtSearchAtlas.Window");

/**
 * Class: ExtSearchAtlas.Window.ASearch
 *
 */
ExtSearchAtlas.Window.ASearch = OpenLayers.Class({

      layer:
Config.ExtSearchAtlas.WFSLayers.CONCENTRACIONES_COMERCIALES,

      wfsManager: null,

      filterManager: new IDEOL.Manager.Filter(),

      columns:
Config.ExtSearchAtlas.Columns.CONCENTRACIONES_COMERCIALES,

    tabPanel: null,

      panelNombre: null,
      panelDireccion: null,
      panelCentro: null,

      buscarButton:  new Ext.Button({
            text: Locale.getText("txt_buscar")
      }),
```

```
	cerrarButton:  new Ext.Button({
		text: Locale.getText("txt_cerrar")
	}),

	win: null,

	gridResultWindow: null,

	initialize: function(options){
		OpenLayers.Util.extend(this, options);
		this.createComponents();
		this.prepareComponents();
	},

	createComponents: function(){
		this.wfsManager = new IDEOL.Manager.WFS();

		this.tabPanel = new
ExtSearchAtlas.Panel.Agrupaciones.Tab();

		this.panelNombre = new
ExtSearchAtlas.Panel.Agrupaciones.Nombre();
		this.panelDireccion = new
ExtSearchAtlas.Panel.Agrupaciones.Direccion();
		this.panelCentro = new
ExtSearchAtlas.Panel.Agrupaciones.Centro();

		this.win = new Ext.Window({
			layout      : 'fit',
			bodyStyle:  'padding: 2px 2px 2px',
		    modal       : false,
		    title       :
Locale.getText("msg_busqueda_concentraciones"),
		    titleCollapse: true,
		    constrainHeader: true,
		    width       : 340,
		    autoHeight  : true,
		    expandOnShow: true,
		    maximizable : false,
		    collapsible: true,
		    resizable   : false,
		    draggable   : true,
		    closeAction   : "hide",
		    plain       : true,
		    border      : false,
			items: [this.tabPanel],
			buttons: [this.buscarButton,this.cerrarButton]
		});
	},

	prepareComponents: function(){
		this.tabPanel.add(this.panelNombre);
		this.tabPanel.add(this.panelDireccion);
		this.tabPanel.add(this.panelCentro);

		// Eventos de componentes
		this.panelNombre.nombreTextField.on('keypress',
function(object, event){
			this.keyEventHandler(object, event);
		}, this);
```

```
            this.panelDireccion.direccionTextField.on('keypress',
function(object, event){
                  this.keyEventHandler(object, event);
            }, this);

            this.buscarButton.on('click', function(){
                  this.doSearch();
            }, this);

            this.cerrarButton.on('click', function(){
                  this.win.hide();
            }, this);
      },

      keyEventHandler: function(object, event){
            if (event.getCharCode() == event.ENTER)
                  this.doSearch();
      },

      getFilter: function(){
            var filter = null;
            var filters = [];

            // Nombre
            var nombre = null;
            nombre = this.panelNombre.nombreTextField.getValue();
            if(nombre != null && nombre.trim() != ""){
                  var filter = this.filterManager.getComparisonFilter(
                              OpenLayers.Filter.Comparison.LIKE,
                              'nombre',
                              "*" + nombre + "*",
                              false);
                  filters.push(filter);
            }

            // Direccion
            var provincia = null;
            provincia = this.panelDireccion.comboProvincias.getValue();
            if(provincia != null && provincia.trim() != "" && provincia
!= '--- '+Locale.getText("txt_seleccionar")+' ---'){
                  var filter = this.filterManager.getComparisonFilter(
                              OpenLayers.Filter.Comparison.LIKE,
                              'provincia',
                              provincia,
                              false);
                  filters.push(filter);

                  var municipio = null;
                  municipio =
this.panelDireccion.comboMunicipios.getValue();
                  var filter = this.filterManager.getComparisonFilter(
                              OpenLayers.Filter.Comparison.LIKE,
                              'municipio',
                              '',
                              false);

                  if(municipio != null && municipio.trim() != "" &&
municipio != '--- '+Locale.getText("txt_seleccionar")+' ---'){
                        filter.value = municipio;
                  }
                  else{
```

```
                        filter.value = '*';
                }
                filters.push(filter);
        }

        var direccion = null;
        direccion =
this.panelDireccion.direccionTextField.getValue();
        if(direccion != null && direccion.trim() != ""){
                var filter = this.filterManager.getComparisonFilter(
                        OpenLayers.Filter.Comparison.LIKE,
                        'nomvia',
                        "*" + direccion + "*",
                        false);
                filters.push(filter);
        }

        // Centro
        var centro = null;
        centro = this.panelCentro.comboCentros.getValue();
        if(centro != null && centro.trim() != "" && centro != '---
'+Locale.getText("txt_seleccionar")+' ---'){
                var filter = this.filterManager.getComparisonFilter(
                        OpenLayers.Filter.Comparison.EQUAL_TO,
                        'tipo_centr',
                        centro,
                        true);
                filters.push(filter);
        }

        filter =
this.filterManager.getLogicalFilter(OpenLayers.Filter.Logical.AND,
filters);

        return filter;
    },

    doSearch: function(){
        var filter = this.getFilter();
        this.gridResultWindow.show(this.layer, this.columns,
filter);
    },

    show: function(){
        this.win.show();
        this.win.expand();

        this.panelNombre.nombreTextField.focus(false, 700);
    },

    CLASS_NAME: "ExtSearchAtlas.Window.ASearch"
});
```

Window/DataView.js

```
Ext.namespace("ExtSearchAtlas.Window");

/**
* Class: ExtSearchAtlas.Window.DataView
*/
ExtSearchAtlas.Window.DataView = OpenLayers.Class({

      filterManager: new IDEOL.Manager.Filter(),

      WFSLAYERS_ESTABLECIMIENTOS:
Config.ExtSearchAtlas.WFSLayers.ESTABLECIMIENTOS.name,
      WFSLAYERS_CONCENTRACIONES_COMERCIALES:
Config.ExtSearchAtlas.WFSLayers.CONCENTRACIONES_COMERCIALES.name,
      WFSLAYERS_MUNICIPIOS:
Config.ExtSearchAtlas.WFSLayers.MUNICIPIOS.name,
      WFSLAYERS_COMERCIOS_INNOVACION:
Config.ExtSearchAtlas.WFSLayers.COMERCIOS_INNOVACION.name,
      WFSLAYERS_ANTENAS_LOCALES:
Config.ExtSearchAtlas.WFSLayers.ANTENAS_LOCALES.name,
      WFSLAYERS_CAMARAS_COMERCIO:
Config.ExtSearchAtlas.WFSLayers.CAMARAS_COMERCIO.name,


      TEMPLATE_DATAVIEW_PATH_ESTABLECIMIENTOS:
Config.ExtSearchAtlas.TEMPLATE_DATAVIEW_PATH_ESTABLECIMIENTOS,
      TEMPLATE_DATAVIEW_PATH_AGRUPACIONES:
Config.ExtSearchAtlas.TEMPLATE_DATAVIEW_PATH_AGRUPACIONES,
      TEMPLATE_DATAVIEW_PATH_CAMARAS:
Config.ExtSearchAtlas.TEMPLATE_DATAVIEW_PATH_CAMARAS,
      TEMPLATE_DATAVIEW_PATH_ANTENAS:
Config.ExtSearchAtlas.TEMPLATE_DATAVIEW_PATH_ANTENAS,
      TEMPLATE_DATAVIEW_PATH_MUNICIPIOS:
Config.ExtSearchAtlas.TEMPLATE_DATAVIEW_PATH_MUNICIPIOS,
      TEMPLATE_DATAVIEW_PATH_COMERCIOS:
Config.ExtSearchAtlas.TEMPLATE_DATAVIEW_PATH_COMERCIOS,

      storeConcentraciones : new Ext.data.Store({
            reader: new Ext.data.JsonReader({},
                        ['id',
                         'empresa',
                         'sector',
                         'nombre',
                         'numpol',
                         'ciudad',
                         'provincia',
                         'web',
                         'pdf',
                         'the_geom'
                         ])
      }),

      storeEstablecimientos: new Ext.data.Store({
            reader: new Ext.data.JsonReader({},
                        ['id',
                         'rotulo',
                         'sigla',
                         'nomcall',
                         'numpol',
```

```
                        'km',
                        'cp',
                        'provincia',
                        'municipio',
                        'nom_agrup',
                        'num_local',
                        'dactividad',
                        'tipo_patec',
                        'long_fach',
                        'refcat',
                        'foto1',
                        'foto2',
                        'the_geom'
                        ])
    }),

    storeAgrupaciones: new Ext.data.Store({
          reader: new Ext.data.JsonReader({},
                        ['id',
                         'nombre',
                         'sigla',
                         'nomvia',
                         'numpol',
                         'km',
                         'cp',
                         'municipio',
                         'provincia',
                         'tipo_centr',
                         'directorio',
                         'plant_tota',
                         'aparcamien',
                         'pl_ap_tot',
                         'pl_ap_int',
                         'pl_ap_ext',
                         'foto1',
                         'foto2',
                         'the_geom'
                         ])
    }),

    storeCamaras: new Ext.data.Store({
          reader: new Ext.data.JsonReader({},
                        ['nombre',
                         'direccion',
                         'codpostal',
                         'provincia',
                         'municipio',
                         'email',
                         'web',
                         'the_geom'
                         ])
    }),

    storeAntenas: new Ext.data.Store({
          reader: new Ext.data.JsonReader({},
                        ['nom_antena',
                         'perso_cont',
                         'direccion',
                         'cp',
                         'municipio',
                         'provincia',
```

```
                    'telefono',
                    'horario',
                    'the_geom'
                    ])
    }),

    storeMunicipios: new Ext.data.Store({
          reader: new Ext.data.JsonReader({},
                    [
                      'nombre',
                      'ine',
                      'the_geom'
                     ])
    }),

    storeComercios: new Ext.data.Store({
          reader: new Ext.data.JsonReader({},
                    [
                     'id',
                     'empresa',
                     'sector',
                     'nombre',
                     'numpol',
                     'ciudad',
                     'provincia',
                     'web',
                     'pdf',
                     'the_geom'
                     ])
    }),

    tplEstablecimientos: null,

    tplAgrupaciones: null,

    tplCamaras: null,

    tplAntenas: null,

    tplMunicipios: null,

    tplComercios: null,

    layer: null,

    map: null,

    wfsManager: null,

    drawManager: null,

    wktFormat: null,

    establecimientosManager: null,

    mask: null,

    arrayData: null,

    fichaWindow: null,
```

```
	streetViewWindow: null,

	dataView: null,

	panel: null,

	win: null,

	streetViewButton: null,

	buscarButton: null,

	fichaButton: null,

	cerrarButton: null,

	initialize: function(map){
		this.createComponents(map);
		this.prepareComponents();
		this.prepareTemplates();
	},

	createComponents: function(map){
		this.map=map;
		this.wfsManager = new IDEOL.Manager.WFS();
		this.drawManager = IDEOL.Manager.Draw.getInstance();
		this.wktFormat = new OpenLayers.Format.WKT();
		this.establecimientosManager = new
ExtSearchAtlas.Manager.Establecimientos(this.map);
		this.fichaWindow = new IDEOL.Tool.HTMLTemplateView();
		this.streetViewWindow = ExtStreetView.Window.Window;

		this.dataView = new Ext.DataView({
			//store: this.storeEstablecimientos,
		  //tpl: this.tpl,
			    autoWidth: true,
		    autoHeight:true,
		    singleSelect: true,
		    multiSelect: false,
		    overClass:'x-view-over',
		    itemSelector:'div.search-item',
		    emptyText: 'No items to display'
		});

		this.panel = new Ext.Panel({
			id: 'ea-view',
			layout:'fit',
		    frame:true,
		    width: 450,
		    height:250,
		    autoScroll: true
		});

		this.win = new Ext.Window({
			layout      : 'fit',
			bodyStyle:  'padding: 2px 2px 2px',
		       modal         : false,
		       title       : Locale.getText("txt_informacion"),
		       titleCollapse: true,
		       constrainHeader: true,
		       maximizable : false,
```

```
                resizable     : false,
                draggable     : true,
                collapsible: true,
                closeAction   : "hide",
                plain       : true,
                border        : false
            });
      },

      prepareComponents: function(){

            this.panel.add(this.dataView);

            this.win.add(this.panel);

            this.fichaButton = this.win.addButton({text:
Locale.getText("txt_ficha")}, function(){
                  this.requestShowFicha();
            }, this);

            this.streetViewButton = this.win.addButton({text:
Locale.getText("txt_streetview")},function(){
                  this.showStreetView();
            },this);

            this.buscarButton = this.win.addButton({text:
Locale.getText("txt_localizar")},function(){
                  this.locateElement();
            },this);

            this.cerrarButton = this.win.addButton({text:
Locale.getText("txt_cerrar")}, function(){
                  this.win.hide();
            },this);
      },

      /**
       * Method: prepareTemplates
       */
      prepareTemplates: function(){
            var tplMunicipiosContent =
this.getTplContent(this.TEMPLATE_DATAVIEW_PATH_MUNICIPIOS);
            this.tplMunicipios = new
Ext.XTemplate(tplMunicipiosContent);

            var tplAntenasContent =
this.getTplContent(this.TEMPLATE_DATAVIEW_PATH_ANTENAS);
            this.tplAntenas = new Ext.XTemplate(tplAntenasContent);

            var tplCamarasContent =
this.getTplContent(this.TEMPLATE_DATAVIEW_PATH_CAMARAS);
            this.tplCamaras = new Ext.XTemplate(tplCamarasContent);

            var tplAgrupacionesContent =
this.getTplContent(this.TEMPLATE_DATAVIEW_PATH_AGRUPACIONES);
            this.tplAgrupaciones = new
Ext.XTemplate(tplAgrupacionesContent);

            var tplEstablecimientosContent =
this.getTplContent(this.TEMPLATE_DATAVIEW_PATH_ESTABLECIMIENTOS);
```

```
            this.tplEstablecimientos = new
Ext.XTemplate(tplEstablecimientosContent);

            var tplComerciosContent =
this.getTplContent(this.TEMPLATE_DATAVIEW_PATH_COMERCIOS);
            this.tplComercios = new Ext.XTemplate(tplComerciosContent);
      },

      getTplContent: function(contentURL){
            var content;

            var request = OpenLayers.Request.GET({
            url: contentURL,
            success: function(response){content =
response.responseText;},
            failure: function(response){content = null;},
            async: false
            });
            return content;
      },

      /**
       * Method: getGeometryFromRecord
       *
       */
    getGeometryFromRecord: function(record){
      var geometryWKT = record.get("the_geom");
            var geometry = this.wktFormat.read(geometryWKT);
            return geometry;
    },

      showStreetView: function(){
            var cont = this.dataView.store.getCount();
            var records = this.dataView.getSelectedRecords();
            var record = records[0];

            if(record == null && cont > 1)
            {
                  IDEOL.UtilUI.showMessageWindow(
                              Locale.getText("txt_informacion"),

      Locale.getText("msg_seleccionar_registro_para_street_view"),
                              Ext.MessageBox.WARNING);
            }
            else
            {
                  if(cont == 1)
                        record = this.dataView.store.getAt(0);

                  var geom = this.getGeometryFromRecord(record);
                  var bounds = geom.geometry.getBounds();
                  var lonlat = bounds.getCenterLonLat();

                  this.streetViewWindow.show(lonlat,
this.layer.projection);

            };
      },
```

```
	/**
	 * Method: onEstablecimientosWFSQuerySuccess
	 *
	 * Muestra la ficha para la agrupacion una vez obtenidos los
establecimientos de la misma.
	 *
	 * Parameters:
	* wfsFeatures - {Array(OpenLayers.Feature.WFS)} Features WFS.
	 */
	onEstablecimientosWFSQuerySuccess:  function(establecimientos){
		this.mask.hide();

		this.agrupacion.data.establecimientos = [];

		if(establecimientos.length > 0){
			this.agrupacion.data.establecimientos =
establecimientos;
		}

		this.fichaWindow.show(this.layer.name, this.agrupacion,
Config.ExtSearchAtlas.TEMPLATE_PATH_AGRUPACIONES);
	},

	/**
	 * Method: requestEstablecimientosForAgrupacion
	 *
	 * Parameters:
	 * idagrupacion - {String} Id de la agrupación sobre la que se
desea obtener todos los establecimientos.
	 *
	 */
	requestEstablecimientosForAgrupacion: function(idagrupacion){
		this.mask.show();

		var attributes = ['rotulo', 'dactividad'];

		var filter = this.filterManager.getComparisonFilter(
				OpenLayers.Filter.Comparison.EQUAL_TO,
				'cod_agrup',
				idagrupacion,
				true);

		this.establecimientosManager.getEstablecimientos(
			attributes,
			filter,
			this.onEstablecimientosWFSQuerySuccess,
			this.onWFSQueryFailure,
			this
		);
	},
```

```
        /**
         * Method: onCompleteFeatureWFSQuerySuccess
         *
         * Obtiene las features de la consulta WFS.
         * Muestra la ficha correspondiente con la feature obtenida.
         *
         * Parameters:
        * wfsFeatures - {Array(OpenLayers.Feature.WFS)} Features WFS.
         */
        onCompleteFeatureWFSQuerySuccess: function(wfsFeatures){
              var completeFeature = null;

              this.mask.hide();

              if(wfsFeatures.length > 0){
                    // La ficha no necesita features preparadas. Accede
al data de las mismas directamente.
                    completeFeature = wfsFeatures[0];

                    var layerName = this.layer.name;

                    if(layerName ==
this.WFSLAYERS_CONCENTRACIONES_COMERCIALES){
                  this.agrupacion = completeFeature;

        this.requestEstablecimientosForAgrupacion(this.agrupacion.data.i
d);
                }
                else if(layerName == this.WFSLAYERS_ESTABLECIMIENTOS){
                    this.fichaWindow.show(this.layer.name,
completeFeature,
Config.ExtSearchAtlas.TEMPLATE_PATH_ESTABLECIMIENTOS);
                }
                else
                    this.fichaWindow.show(this.layer.name,
completeFeature, Config.ExtSearchAtlas.TEMPLATE_PATH_MUNICIPIOS);
              }
        },

        /**
         * Method: requestCompleteFeature
         *
         * Realiza una consulta WFS para obtener todos los atributos de
una geometría dado su id.
         *
         * Parameters:
        * id - {String} Valor del id de la feature.
         */
        requestCompleteFeature: function(id,value){

              this.mask.show();

              // Todos los atributos
              var attributes = "";

              var filter = this.filterManager.getComparisonFilter(
                          OpenLayers.Filter.Comparison.EQUAL_TO,
                          id,
                          value,
                          true);
```

```
            this.wfsManager.getFeaturesByFilter(
                        this.layer,
                        attributes,
                        filter,
                        this.onCompleteFeatureWFSQuerySuccess,
                        this.onWFSQueryFailure,
                        this
                  );
      },

      /**
       * Method: requestShowFicha
       *
       * Solicita obtener la ficha para el registro seleccionado en el
grid.
       *
       */
      requestShowFicha: function(){
            var cont = this.dataView.store.getCount();
            var records = this.dataView.getSelectedRecords();
            var record = records[0];

            if(record == null && cont > 1)
            {
                  IDEOL.UtilUI.showMessageWindow(
                              Locale.getText("txt_informacion"),

      Locale.getText("msg_seleccionar_registro_para_ficha"),
                              Ext.MessageBox.WARNING);
            }
            else
            {
                  if(cont == 1)
                        record = this.dataView.store.getAt(0);
                  /**
                   * El grid muestra las features con los atributos
necesarios para el grid.
                   * Es necesario obtener todos los atributos de la
feature para la ficha.
                   */
                  // Municipios
                  if(this.dataView.store.id == 2){
                        this.requestCompleteFeature("ine",
record.get("ine"));
                  }
                  else
                        this.requestCompleteFeature("id",
record.get("id"));
            }
      },

      locateElement: function(){
            var cont = this.dataView.store.getCount();
            var records = this.dataView.getSelectedRecords();
            var record = records[0];

            if(record == null && cont > 1)
            {

      IDEOL.UtilUI.showMessageWindow(Locale.getText("txt_informacion")
```

```
,Locale.getText("msg_seleccionar_elemento_localizar"),
Ext.MessageBox.WARNING);
            }
            else
            {
                  if(cont == 1)
                        record = this.dataView.store.getAt(0);

                  var geom = this.getGeometryFromRecord(record);
                  var bounds = geom.geometry.getBounds();
                  var lonlat = bounds.getCenterLonLat();

                  this.drawManager.drawMarker(lonlat,
this.layer.projection.getCode(), true, true);
            };
      },

      /**
       * Method: getPreparedFeatures
       *
       * Obtiene el array de objetos con el que se cargará el store
del grid a partir de features WFS.
       *
       * Parameters:
     * wfsFeatures - {Array(OpenLayers.Feature.WFS)} Features WFS.
     *
     * Returns:
     * {OpenLayers.Layer.WFS} Array de objetos con el que se podrá
cargar el store del grid.
       */
      getPreparedFeatures: function(wfsFeatures){
            var features = [];
            for(i=0; i<wfsFeatures.length; i++){
                  var feature =  wfsFeatures[i].data;
                  // TODO obtener nombre de atributo del store
                  feature.the_geom = wfsFeatures[i].geometry;
                  features.push(feature);
            }
            return features;
      },

      onWFSBBoxQuerySuccess: function(wfsFeatures){
            this.showResults(wfsFeatures);
            this.mask.hide();
      },

      onWFSBBoxQueryFailure: function(response){
            this.mask.hide();
            IDEOL.UtilUI.showMessageWindow(
                  Locale.getText("txt_informacion"),
                  Locale.getText("msg_error_realizar_consulta_bbox"),
                  Ext.MessageBox.ERROR
            );
      },

      showResults: function(wfsFeatures){
            var store;
            var tpl;

            var features = this.getPreparedFeatures(wfsFeatures);
```

```
            if(this.layer.name == this.WFSLAYERS_ESTABLECIMIENTOS){
                  tpl = this.tplEstablecimientos;
                  store = this.storeEstablecimientos;
                  store.id = 0;
                  this.fichaButton.show();
                  this.buscarButton.show();
                  this.streetViewButton.show();
            }
            else if(this.layer.name ==
this.WFSLAYERS_CONCENTRACIONES_COMERCIALES){
                  tpl = this.tplAgrupaciones;
                  store = this.storeAgrupaciones;
                  store.id = 1;
                  this.fichaButton.show();
                  this.buscarButton.show();
                  this.streetViewButton.show();
            }
            else if(this.layer.name ==
this.WFSLAYERS_CAMARAS_COMERCIO){
                  tpl = this.tplCamaras;
                  store = this.storeCamaras;
                  this.fichaButton.hide();
                  this.buscarButton.show();
                  this.streetViewButton.show();
            }
            else if(this.layer.name == this.WFSLAYERS_ANTENAS_LOCALES){
                  tpl = this.tplAntenas;
                  store = this.storeAntenas;
                  this.fichaButton.hide();
                  this.buscarButton.show();
                  this.streetViewButton.show();
            }
            else if(this.layer.name == this.WFSLAYERS_MUNICIPIOS){
                  tpl = this.tplMunicipios;
                  store = this.storeMunicipios;
                  store.id = 2;
                  this.fichaButton.show();
                  this.buscarButton.hide();
                  this.streetViewButton.hide();
            }
            else if(this.layer.name ==
this.WFSLAYERS_COMERCIOS_INNOVACION){
                  tpl = this.tplComercios;
                  store = this.storeComercios;
                  this.fichaButton.hide();
                  this.buscarButton.show();
                  this.streetViewButton.show();
            }

            store.removeAll();
            store.loadData(features);

            this.dataView.tpl = tpl;
            this.dataView.setStore(store);
      },

      doSearch: function(){

            if(this.dataView.store != null)
                  this.dataView.store.removeAll();
```

```
            this.mask.show();

            this.wfsManager.getFeaturesByBBOX(
                  this.layer,
                  this.bounds,
                  this.onWFSBBoxQuerySuccess,
                  this.onWFSBBoxQueryFailure,
                  this
            );
      },

      show: function(layer, bounds){
            if(layer && bounds){

                  this.layer = layer;
                  this.win.setTitle(Locale.getText("txt_informacion")+"
- "+this.layer.name);
                  this.bounds = bounds;

                  this.showWindow();

                  this.mask= new Ext.LoadMask(this.panel.body);

                  this.doSearch();
            }
      },

      showWindow: function(){
            this.win.show();
            this.win.expand();
      },

      setFichaType: function(param){

      }
});
```

Window/ESearch.js

```
Ext.namespace("ExtSearchAtlas.Window");

/**
 * Class: ExtSearchAtlas.Window.ESearch
 *
 */
ExtSearchAtlas.Window.ESearch = OpenLayers.Class({

      layer: Config.ExtSearchAtlas.WFSLayers.ESTABLECIMIENTOS,

      wfsManager: new IDEOL.Manager.WFS(),

      filterManager: new IDEOL.Manager.Filter(),

      tabPanel: null,

      columns: Config.ExtSearchAtlas.Columns.ESTABLECIMIENTOS,

      panelNombre: null,
```

```
        panelDireccion:   null,
        panelActividad: null,
        panelAgrupacion: null,

        buscarButton: new Ext.Button({
              text: Locale.getText("txt_buscar")
        }),

        cerrarButton: new Ext.Button({
              text: Locale.getText("txt_cerrar")
        }),

        win: null,

        gridResultWindow: null,

        initialize: function(options){
              OpenLayers.Util.extend(this, options);
              this.createComponents();
              this.prepareComponents();
        },

        createComponents: function(){
              this.wfsManager = new IDEOL.Manager.WFS();

              this.establecimientosManager = new
ExtSearchAtlas.Manager.Establecimientos();

              this.tabPanel= new
ExtSearchAtlas.Panel.Establecimientos.Tab();

              this.panelNombre= new
ExtSearchAtlas.Panel.Establecimientos.Nombre();
              this.panelDireccion= new
ExtSearchAtlas.Panel.Establecimientos.Direccion();
              this.panelActividad= new
ExtSearchAtlas.Panel.Establecimientos.Actividad();
              this.panelAgrupacion= new
ExtSearchAtlas.Panel.Establecimientos.Agrupacion();

              this.win= new Ext.Window({
                    layout      : 'fit',
                    bodyStyle:  'padding: 2px 2px 2px',
                  modal         : false,
                  title       :
Locale.getText("msg_busqueda_establecimientos"),
                  titleCollapse: true,
                  constrainHeader: true,
                  width         : 340,
                  autoHeight  : true,
                  expandOnShow: true,
                  maximizable : false,
                  collapsible: true,
                  resizable     : false,
                  draggable     : true,
                  closeAction   : "hide",
                  plain       : true,
                  border        : false,
                    items: [this.tabPanel],
                    buttons: [this.buscarButton,this.cerrarButton]
              });
```

```
	},

	prepareComponents: function(){
		this.tabPanel.add(this.panelNombre);
		this.tabPanel.add(this.panelDireccion);
		this.tabPanel.add(this.panelActividad);
		this.tabPanel.add(this.panelAgrupacion);

		// Eventos de componentes
		this.panelNombre.nombreTextField.on('keypress',
function(object, event){
			this.keyEventHandler(object, event);
		}, this);

		this.panelDireccion.direccionTextField.on('keypress',
function(object, event){
			this.keyEventHandler(object, event);
		}, this);

		this.buscarButton.on('click', function(){
			this.doSearch();
		}, this);

		this.cerrarButton.on('click',function(){
			this.win.hide();
		},this);
	},

	keyEventHandler: function(object, event){
		if (event.getCharCode() == event.ENTER)
			this.doSearch();
	},

	getFilter: function(){
		var filter = null;
		var filters = [];

		// Nombre
		var rotulo = null;
		rotulo = this.panelNombre.nombreTextField.getValue();
		if(rotulo != null && rotulo.trim() != ""){
			var filter = this.filterManager.getComparisonFilter(
					OpenLayers.Filter.Comparison.LIKE,
					'rotulo',
					"*" + rotulo + "*",
					false);
			filters.push(filter);
		}

		// Direccion
		var provincia = null;
		provincia = this.panelDireccion.comboProvincias.getValue();
		if(provincia != null && provincia.trim() != "" && provincia
!= '--- '+Locale.getText("txt_seleccionar")+' ---'){
			var filter = this.filterManager.getComparisonFilter(
					OpenLayers.Filter.Comparison.LIKE,
					'provincia',
					provincia,
					false);
			filters.push(filter);
```

```
                var municipio = null;
                municipio =
this.panelDireccion.comboMunicipios.getValue();
                var filter = this.filterManager.getComparisonFilter(
                            OpenLayers.Filter.Comparison.LIKE,
                            'municipio',
                            '',
                            false);
                if(municipio != null && municipio.trim() != "" &&
municipio != '--- '+Locale.getText("txt_seleccionar")+' ---'){
                      filter.value = municipio;
                }
                else{
                      filter.value = '*';
                }
                filters.push(filter);
          }

          var direccion = null;
          direccion =
this.panelDireccion.direccionTextField.getValue();
          if(direccion != null && direccion.trim() != ""){
                var filter = this.filterManager.getComparisonFilter(
                            OpenLayers.Filter.Comparison.LIKE,
                            'nomcall',
                            "*" + direccion + "*",
                            false);
                filters.push(filter);
          }

          //Actividad
          var actividad = null;
          actividad =
this.panelActividad.comboActividades.getValue();
          if(actividad != null && actividad.trim() != "" && actividad
!= '--- '+Locale.getText("txt_seleccionar")+' ---'){
                var filter = this.filterManager.getComparisonFilter(
                            OpenLayers.Filter.Comparison.EQUAL_TO,
                            'dactividad',
                            actividad,
                            false);
                filters.push(filter);
          }

          //Agrupación
          var agrupacion = null;
          agrupacion =
this.panelAgrupacion.comboAgrupaciones.getValue();
          if(agrupacion != null && agrupacion.trim() != "" &&
agrupacion != '--- '+Locale.getText("txt_seleccionar")+' ---'){
                var filter = this.filterManager.getComparisonFilter(
                            OpenLayers.Filter.Comparison.EQUAL_TO,
                            'nom_agrup',
                            agrupacion,
                            false);
                filters.push(filter);
          }
```

```
            filter =
this.filterManager.getLogicalFilter(OpenLayers.Filter.Logical.AND,
filters);

            return filter;
      },

      doSearch: function(){
            var filter = this.getFilter();
            this.gridResultWindow.show(this.layer, this.columns,
filter);
      },

      show: function(){
            this.win.show();
            this.win.expand();

            this.panelNombre.nombreTextField.focus(false, 700);
      },

      CLASS_NAME: "ExtSearchAtlas.Window.ESearch"
});
```

Window/GridResult.js

```
Ext.namespace("ExtSearchAtlas.Window");

/**
 * Class: ExtSearchAtlas.Window.GridResult2
 *
 */
ExtSearchAtlas.Window.GridResult =
OpenLayers.Class(IDEOL.Tool.WFSResultsGrid, {

      TEMPLATE_PATH_AGRUPACIONES:
Config.ExtSearchAtlas.TEMPLATE_PATH_AGRUPACIONES,
      TEMPLATE_PATH_ESTABLECIMIENTOS:
Config.ExtSearchAtlas.TEMPLATE_PATH_ESTABLECIMIENTOS,

      mask: null,

      streetViewButton: null,

      streetViewWindow: null,

      filterManager: new IDEOL.Manager.Filter(),

      /**
       * Property: fichaWindow
       * {IDEOL.Tool.HTMLTemplateView}
       */
      fichaWindow: null,

      fichaButton: null,

      establecimientosManager: null,

      initialize : function(options){
```

```
      IDEOL.Tool.WFSResultsGrid.prototype.initialize.apply(this,[optio
ns]);

            this.createExtendedComponents();
        this.prepareExtendedComponents();
      },

      createExtendedComponents: function(){
        this.fichaWindow = new IDEOL.Tool.HTMLTemplateView();

        this.establecimientosManager = new
ExtSearchAtlas.Manager.Establecimientos();

        this.fichaButton = new Ext.Button({
                text: Locale.getText('txt_ficha'),
                handler: function(){
                        this.requestShowFicha();
                },
                scope: this
        });

        this.streetViewWindow = ExtStreetView.Window.Window;

        this.streetViewButton = new Ext.Button({
                text: Locale.getText('txt_streetview'),
                handler: function(){
                        this.showStreetView();
                },
                scope: this
        });
      },

      prepareExtendedComponents: function(){
        this.win.buttons.unshift(this.fichaButton);
        this.win.buttons.unshift(this.streetViewButton);

        this.win.on("show", function(){
            this.mask = new Ext.LoadMask(this.grid.body);
        }, this);
      },

      showStreetView: function(){
            var countRecords = this.grid.store.getCount();
            var records = this.getSelectedRecords();
            var record = null;

            if(records.length == 1){
            record = records[0];
      }
            else if(records.length == 0 && countRecords == 1){
            record = this.grid.store.getAt(0);
      }

            if(record != null){
                  this.requestGeometryForRecord(record);
            }
            else{
            IDEOL.UtilUI.showMessageWindow(
                        Locale.getText("txt_informacion"),
```

```
		Locale.getText("msg_seleccionar_registro_para_street_view"),
                        Ext.MessageBox.WARNING);
		}
		},

		onWFSRequestGeometrySuccess: function(wfsFeatures){
			var geoms = this.getGeometriesForFeatures(wfsFeatures);

			if(geoms.length > 0){
				var geom = geoms[0];
				var bounds = geom.geometry.getBounds();
				var lonlat = bounds.getCenterLonLat();

				this.streetViewWindow.show(lonlat,
this.layer.projection);
			}
			this.mask.hide();
		},

		requestGeometryForRecord: function(record){
			this.mask.show();

			var fid = record.data.fid;

			// TODO comprobar cual es el atributo de geometría para la
capa
		var attributes = ["the_geom"];

		var filter = new OpenLayers.Filter.FeatureId({
				fids: [fid]
			});

		this.wfsManager.getFeaturesByFilter(
				this.layer,
				attributes,
				filter,
				this.onWFSRequestGeometrySuccess,
				this.onWFSQueryFailure,
				this
			);
		},

		/**
		 * Method: onEstablecimientosWFSQuerySuccess
		 *
		 * Muestra la ficha para la agrupacion una vez obtenidos los
establecimientos de la misma.
		 *
		 * Parameters:
		* wfsFeatures - {Array(OpenLayers.Feature.WFS)} Features WFS.
		 */
		onEstablecimientosWFSQuerySuccess: function(establecimientos){
			this.mask.hide();

			this.agrupacion.data.establecimientos = [];

			if(establecimientos.length > 0){
				this.agrupacion.data.establecimientos =
establecimientos;
			}
```

```
            this.fichaWindow.show(this.layer.name, this.agrupacion,
this.TEMPLATE_PATH_AGRUPACIONES);
      },

      /**
       * Method: requestEstablecimientosForAgrupacion
       *
       * Parameters:
       * idagrupacion - {String} Id de la agrupación sobre la que se
desea obtener todos los establecimientos.
       *
       */
      requestEstablecimientosForAgrupacion: function(idagrupacion){
            this.mask.show();

            var attributes = ['rotulo', 'dactividad'];

            var filter = this.filterManager.getComparisonFilter(
                        OpenLayers.Filter.Comparison.EQUAL_TO,
                        'cod_agrup',
                        idagrupacion,
                        true);

            this.establecimientosManager.getEstablecimientos(
                  attributes,
                  filter,
                  this.onEstablecimientosWFSQuerySuccess,
                  this.onWFSQueryFailure,
                  this
            );
      },

      /**
       * Method: onCompleteFeatureWFSQuerySuccess
       *
       * Obtiene las features de la consulta WFS.
       * Muestra la ficha correspondiente con la feature obtenida.
       *
       * Parameters:
      * wfsFeatures - {Array(OpenLayers.Feature.WFS)} Features WFS.
       */
      onCompleteFeatureWFSQuerySuccess: function(wfsFeatures){
            var completeFeature = null;

            this.mask.hide();

            if(wfsFeatures.length > 0){
                  // La ficha no necesita features preparadas. Accede
al data de las mismas directamente.
                  completeFeature = wfsFeatures[0];

                  var layerName = this.layer.name;

                  if(layerName ==
Config.ExtSearchAtlas.WFSLayers.CONCENTRACIONES_COMERCIALES.name){
                this.agrupacion = completeFeature;

      this.requestEstablecimientosForAgrupacion(this.agrupacion.data.i
d);
            }
```

```
            else if(layerName ==
Config.ExtSearchAtlas.WFSLayers.ESTABLECIMIENTOS.name){
                this.fichaWindow.show(this.layer.name,
completeFeature, this.TEMPLATE_PATH_ESTABLECIMIENTOS);
            }
        }
    },

    /**
     * Method: requestCompleteFeature
     *
     * Realiza una consulta WFS para obtener todos los atributos de
una geometría dado su id.
     *
     * Parameters:
     * id - {String} Valor del id de la feature.
     */
    requestCompleteFeature: function(id){

        this.mask.show();

        // Todos los atributos
        var attributes = "";

        var filter = this.filterManager.getComparisonFilter(
                OpenLayers.Filter.Comparison.EQUAL_TO,
                'id',
                id,
                true);

        this.wfsManager.getFeaturesByFilter(
                this.layer,
                attributes,
                filter,
                this.onCompleteFeatureWFSQuerySuccess,
                this.onWFSQueryFailure,
                this
            );
    },

    /**
     * Method: requestShowFicha
     *
     * Solicita obtener la ficha para el registro seleccionado en el
grid.
     *
     */
    requestShowFicha: function(){
        var cont = this.grid.store.getCount();

        var records = this.getSelectedRecords();
        var record = records[0];

        if(record == null && cont > 1)
        {
            IDEOL.UtilUI.showMessageWindow(
                    Locale.getText("txt_informacion"),

    Locale.getText("msg_seleccionar_registro_para_ficha"),
                    Ext.MessageBox.WARNING);;
        }
```

```
            else
            {
                  if(cont == 1)
                        record = this.grid.store.getAt(0);
                  /**
                   * El grid muestra las features con los atributos
necesarios para el grid.
                   * Es necesario obtener todos los atributos de la
feature para la ficha.
                   */
                  this.requestCompleteFeature(record.get("id"));
            };
      }

});
```

10. Bibliografía

- La documentación del manejo de librería ExtJS se puede encontrar en la URL: http://dev.sencha.com/deploy/dev/docs/.

- Existen además ejemplos muy prácticos para el desarrollo de interfaces en la siguiente URL: http://dev.sencha.com/deploy/dev/examples/.

- Información referente al manejo de capas WMC para utilizarlas en la impresión: 03-036r2_Web_Map_Context_Documents_WMC_version_1.0.pdf

- Para el desarrollo del StreetView y otras aplicaciones de googles hay información relevante en la ayuda para este proyecto en esta URL: http://code.google.com/intl/es-ES/apis/maps/documentation/javascript/v2/

- Ayuda acerca de la librería OpenLayers se puede encontrar en la siguiente URL: http://dev.openlayers.org/docs/files/OpenLayers-js.html

- JasperReports for Java Developers (David R. Hefflfinger) – Packt Publishing

- http://wiki.osgeo.org/wiki/Openlayers:_las_herramientas_disponibles_por_defecto

- http://mapserver.org/ogc/