Universidad Zaragoza
1542

Proyecto Fin de Carrera

Infraestructura de edición, visualización y análisis de información geográfica orientada a gestores de administraciones públicas: aplicación a la gestión de información urbana en IDEZar.

Autor

David de Juan Martín

Director y ponente

Director: Juan López de Larrinzar Galdámez
Ponente: Pedro R. Muro Medrano

Escuela de Ingeniería y Arquitectura

2012/2013

A los compañeros de GeospatiumLab con los que he compartido unos entrañables descansos y especialmente a Mª José, David y Juan que me han encaminado durante todo el proyecto.

A los compañeros y amigos por todos los buenos ratos que hemos pasado durante la carrera.

Y principalmente a mi familia, por ser la mejor familia que uno pueda tener.

Infraestructura de edición, visualización y análisis de información geográfica orientada a gestores de administraciones públicas: aplicación a la gestión de información urbana en IDEZar.

RESUMEN

Este Proyecto Final de Carrera (PFC) se ha realizado en GeoSpatiumLab, una empresa de base tecnológica, especializada en el tratamiento digital de la información geoespacial y georreferenciada y sus ámbitos de aplicación. Cuenta con una amplia experiencia en el ámbito de las Infraestructuras de Datos Espaciales (IDE) con especial relevancia al caso del desarrollo y gestión de la Infraestructura de Datos Espaciales del Ayuntamiento de Zaragoza (IDEZar).

El presente proyecto surge como solución al problema de la gestión y análisis de la información geográfica dentro de los gestores de datos espaciales, con un especial énfasis en la Infraestructura de Datos Espaciales del Ayuntamiento de Zaragoza (IDEZar). El nuevo sistema diseñado y desarrollado está especialmente orientado a usuarios sin conocimientos en Sistemas de Información Geográfica (SIG).

En concreto, para implementar este nuevo sistema se ha trabajado sobre la herramienta gvSIG. gvSIG tiene como objetivo el desarrollo de Sistemas de Información Geográfica (SIG) de código libre. Está escrito en lenguaje Java y trabaja con el concepto de plugin lo que permite adicionar y sustraer funcionalidad de un modo sencillo y natural.

gvSIG tiene un gran potencial para la gestión de Sistemas de Información Geográfica, pero al igual que todas las herramientas SIG, su problema reside en que su utilización es compleja para el personal sin la formación adecuada en estos entornos. Por ello se han desarrollado e integrado una serie de herramientas intuitivas para la edición y análisis de información geográfica. Entre las que se encuentran herramientas que permiten el acceso de manera sencilla y abstracta a datos con independencia de su origen, ráster (servicios teselados WMS-C y WMTS) y vectoriales, así como para la exportación de la información en formatos ligeros, todo ello con interfaces gráficas orientadas a los usuarios.

El sistema final obtenido ha sido integrado satisfactoriamente en la infraestructura IDEZar, cumpliendo su propósito de agilizar y simplificar las tareas de gestión y edición cartográfica, habiendo sido utilizada ya en diferentes casos reales con un resultado óptimo.

ÍNDICE

I MEMORIA

II ANEXOS

III GLOSARIO, FIGURAS Y BIBLIOGRAFÍA

I MEMORIA

1. INTRODUCCIÓN

El presente documento tiene como objetivo recoger toda la información relacionada con la realización de este Proyecto Fin de Carrera (PFC) titulado "Infraestructura de edición, visualización y análisis de información geográfica orientada a gestores de administraciones públicas: aplicación a la gestión de información urbana en IDEZar[1]". En esta introducción se abordará el contexto, tanto profesional como tecnológico, en el que se ha desarrollado dicho proyecto.

1.1. Contexto profesional

Este PFC se ha llevado a cabo en GeoSpatiumLab S.L.[2], una empresa especializada en el desarrollo de servicios basados en la localización, donde el carácter espacial de la información es el elemento principal. Surge como Spin-Off de la Universidad de Zaragoza, entre cuyos objetivos está facilitar la transferencia de tecnología generada por el Grupo de Sistemas de Información Avanzados (IAAA[3]), perteneciente al Departamento de Informática e Ingeniería de Sistemas. El Grupo de Sistemas de Información Avanzados es un grupo de I+D+i (Investigación, desarrollo e innovación) de carácter multidisciplinar, pero con un marcado perfil informático.

GeoSpatiumLab tiene en su cartera de clientes al Instituto Geográfico Nacional[4], la Confederación Hidrográfica del Ebro[5] y del Duero, entre otros. Además lleva varios años colaborando con el Ayuntamiento de Zaragoza[6] en IDEZar, la infraestructura de datos espaciales de la ciudad.

Es precisamente dentro de este contexto de la infraestructura de datos espaciales de Zaragoza (IDEZar) donde el presente proyecto surge con el objetivo de realizar una herramienta que facilite la gestión de los datos georreferenciados creando además un

[1] http://www.zaragoza.es/ciudad/idezar/presentacion.htm
[2] http://www.geoslab.com/
[3] http://iaaa.cps.unizar.es/showContent.do?cid=presentacion.ES
[4] http://www.ign.es/ign/main/index.do
[5] http://www.chebro.es/
[6] http://www.zaragoza.es/sedeelectronica/

flujo natural de trabajo, pudiendo utilizarse además esta herramienta para la gestión de otras infraestructuras de datos espaciales.

1.2. Contexto tecnológico

Para una mejor comprensión del proyecto es necesaria una breve explicación de su ámbito tecnológico que permita una visión global de su alcance y repercusión. El proyecto ha consistido en la realización de una herramienta SIG que posibilite la gestión de IDEs, con el caso de uso en concreto de la IDE de Zaragoza IDEZar.

1.2.1. Sistemas de Información Geográfica (SIG)

Los Sistemas de Información Geográfica (SIG[7]) o en inglés *Geographic Information System (GIS)*, son un conjunto de herramientas que permiten integrar, editar, almacenar, analizar, mostrar y compartir la información geográficamente referenciada.

La principal razón para utilizar un SIG reside en su facilidad para la gestión de información espacial. Su funcionamiento se basa en la separación de la información en diferentes capas temáticas, las cuales almacena independientemente, permitiendo de este modo trabajar con ellas de forma sencilla y rápida. [1]

Existen dos formas de almacenar los datos en un SIG: ráster y vectorial. Los datos ráster se centran más en las propiedades del espacio que en la precisión de la localización. Este modelo de datos es muy adecuado para la representación de variables continuas en el espacio. El modelo ráster divide el espacio en celdas regulares donde cada una de ellas representa un valor único. Mientras en los datos vectoriales el interés de las representaciones se centra en la precisión de localización de los elementos geográficos sobre el espacio. En este modelo la información geométrica se almacena en forma de elementos geométricos (básicamente puntos, líneas y polígonos, aunque existen tipos mucho más elaborados) definidos mediante sus coordenadas numéricas en un sistema de coordenadas concreto.

[7] http://www.ign.es/ign/layoutIn/actividadesSistemaInfoGeografica.do

1.2.2. Infraestructura de Datos Espaciales (IDE)

Una Infraestructuras de Datos Espaciales (IDE[8]) es el conjunto de tecnologías, políticas, estándares y recursos humanos para adquirir, procesar, almacenar, distribuir y mejorar la utilización de la información geográfica. Las IDE facilitan el transporte de información geoespacial además de promover el desarrollo social, económico y ambiental del territorio.

Para que una IDE pueda ser operacional, son necesarios acuerdos entre diferentes organizaciones con el objetivo de que exista una coordinación y administración de la información a escalas locales, nacionales y supranacionales. Además es necesario que exista un conjunto de software y servicios que permitan explotar la información recogida en una IDE.

1.2.3. IDEZar

IDEZar[9] es la Infraestructura de Datos Espaciales de Zaragoza, que tiene entre sus objetivos principales facilitar a los ciudadanos el acceso a la información referente a la ciudad localizándola sobre el mapa, permitiendo ofrecer servicios y aplicaciones las cuales proporcionan un valor añadido a los diferentes contenidos que el Ayuntamiento ofrece a la ciudadanía.

La iniciativa IDEZar nace en el año 2004 con la finalidad de llevar a cabo la implantación de una Infraestructura de Datos Espaciales a nivel local. Hoy en día IDEZar es una referencia Europea de IDE en administración local ya que ha evolucionado notablemente y ofrece un amplio catálogo de servicios tanto para mejorar la experiencia del día a día de la ciudadanía con la ciudad como para dar soporte a la gestión municipal interna.

Cabe destacar los siguientes servicios y funcionalidades:

- Toda información o acción puede ser georreferenciada y accedida aprovechando su potencial geoespacial tanto por parte de los ciudadanos (localización de quejas y sugerencias, creación de mapas colaborativos, etc.) como en acciones de uso interno: desde las herramientas de gestión municipal transversales (portal

[8] http://www.idee.es/web/guest/introduccion-a-las-ide
[9] http://www.zaragoza.es/ciudad/idezar/presentacion.htm

interno), a través de aplicaciones para áreas específicas (como la herramienta de planificación de incidencias de la Unidad de Planificación de Policía Local o la gestión de información urbana realizada por el Servicio de Movilidad Urbana).

- Ofrece herramientas que aportan un valor añadido a la información urbana: cálculo de rutas multimodales por la ciudad (Cómo Moverse en Transporte Público), red de contaminación atmosférica, etc.

- Permite mostrar información en tiempo real para ayudar a la ciudadanía a la toma de decisiones ("Zaragoza al instante[10]"): intensidad de tráfico, farmacias de guardia, taxis libres cercanos, afecciones, ocupación de zonas de estacionamiento, etc.

IDEZar se encuentra totalmente integrada dentro del entorno tecnológico del Ayuntamiento de Zaragoza y focaliza sus esfuerzos en llevar a cabo un acercamiento de la Web Municipal a la ciudadanía y en hacerle llegar información de calidad, fácilmente comprensible basándose en la interoperabilidad de servicios y la simplicidad en la interacción con los componentes.

Las líneas de trabajo entorno a la infraestructura se centran en realizar las actuaciones necesarias para mejorar la usabilidad de los servicios y aplicaciones que forman parte de IDEZar. Es por ello que los objetivos marcados se basan en ofrecer al ciudadano mejores tiempos de respuesta, datos de calidad fácilmente comprensibles, información en tiempo real, etc.

En definitiva, IDEZar es una infraestructura transversal que permite gestionar y publicar información georreferenciada dando soporte a todos los servicios clave de una *Smart City*[11].

[10] http://www.zaragoza.es/ciudad/noticias/detalleM_Noticia?id=220048
[11] https://en.wikipedia.org/wiki/Smart_city

2. TRABAJO REALIZADO

En este capítulo se detallará el trabajo realizado a lo largo de este PFC. Desde el estudio del arte y el análisis de los requisitos del sistema, pasando por las fases de diseño e implementación y terminando con los casos de uso y las pruebas a las que se sometió el sistema.

2.1. Estudio del Estado del Arte

Antes de comenzar con el proyecto, se realizó un estudio del estado del arte actual, es decir las técnicas y programas existentes hoy en día relacionados con la temática del proyecto. Este estudio permitió aportar nuevos puntos de vista al proyecto así como nuevas ideas no planteadas en un inicio. Se puede ver en más detalle en el Anexo B – Estado del Arte.

La temática del proyecto presente es la elaboración de una herramienta SIG intuitiva que sea sencilla de utilizar para personal sin un perfil técnico ni especializado en este tipo de entornos. Aunque ya desde un principio se apostaba por trabajar sobre gvSIG, se hizo una valoración en profundidad de los diferentes softwares que existen actualmente en el mercado, realizando comparativas entre ellos para conocer si la decisión a priori había sido la más acertada. La respuesta fue afirmativa, gvSIG era la herramienta más adecuada sobre la que basar el proyecto [2], debido en gran medida a las siguientes características: es un software de código libre, con una gran comunidad de desarrolladores [3], que trabaja sobre el concepto de plugin/extensión permitiendo ampliar fácilmente su funcionalidad. Además está orientado al usuario final, tanto a nivel de interfaz de usuario como de funciones implementadas, lo cual es ideal para alcanzar el objetivo.

El problema de gvSIG, como el de cualquier software SIG es que es demasiado complejo y específico para un usuario sin conocimientos suficientes en estos entornos. Ahí es donde entra en juego el presente proyecto en parte con el objetivo de solucionar este problema.

2.2. Análisis de requisitos

Una vez realizado el estudio del estado del arte para conocer las herramientas actuales en el mercado, y partiendo de los requisitos del sistema a más alto nivel se pasó a definirlos en detalle. Estos requisitos como en todo proyecto software, se dividen en requisitos funcionales y no funcionales. Los requisitos funcionales son aquellos que establecen el comportamiento del sistema, mientras, los no funcionales son aquellos que establecen otros aspectos que no forman parte directamente del funcionamiento.

A continuación se expone un resumen de los requisitos, en el Anexo C – Análisis del Sistema se pueden ver en más detalle:

2.2.1. Requisitos funcionales

- El nuevo sistema debe permitir la edición de la información vectorial y de su simbología de forma rápida y sencilla, orientado a usuarios no expertos en SIG
- El nuevo sistema debe proporcionar herramientas de análisis garantizando el guardado y cargado de capas indistintamente de la fuente de datos, como pueden ser mapas de servicio teselados y capas vectoriales en formatos ligeros, abstrayéndoselo al usuario.
- Integración con el flujo de gestión de IDEZar.

2.2.2. Requisitos no funcionales

- Implementar el nuevo sistema sobre gvSIG.
- El nuevo sistema debe funcionar sobre Windows

2.3. Arquitectura general del sistema

El sistema diseñado tiene dos objetivos principales, el primero permitir la gestión de la información geospacial de forma intuitiva y sencilla, facilitando la tarea al personal que no tiene conocimientos SIG. El segundo permitir su integración para la gestión de IDEs, en concreto en el caso de uso de la infraestructura IDEZar, posibilitando un flujo de trabajo completo desde la creación y edición de la información georreferenciada hasta su publicación y consumo en aplicaciones para la ciudadanía.

Para conseguir estos objetivos se trabajó sobre el software gvSIG, ampliando su funcionalidad y diseñando un sistema adecuado para el entorno en el que se requería. Por lo tanto, fue necesario estudiar la arquitectura de gvSIG [4] para poder conocer cómo se iban a integrar las nuevas herramientas sobre él. En esta sección se muestra una visión general de la arquitectura de gvSIG y del nuevo sistema, así como el modo de integrarlo en el entorno para la gestión de información urbana.

La arquitectura de gvSIG se puede abstraer hasta un nivel superior donde se encuentran 3 entidades principales, la interfaz de usuario (GUI), FMap y el modelo interno de datos (core). El interfaz de usuario (GUI) representa la parte visual de la aplicación y permite al usuario interaccionar con los datos. FMap es el motor de la aplicación. Incluye todas las clases necesarias para manejar objetos SIG, desde dibujar la cartografía hasta acceder a los datos. Se compone de un gestor de herramientas, capas y orígenes de datos. Por último, el modelo interno de datos (core), sirve de puente entre la aplicación y las fuentes de datos. Contiene las clases necesarias para acceder a los datos, escribir datos en una fuente, así como las propiedades de acceso a fuentes remotas. La arquitectura de gvSIG está explicada con mayor detalle en el Anexo C – Análisis del Sistema.

El esquema de la arquitectura de gvSIG se puede ver en la Figura 1.

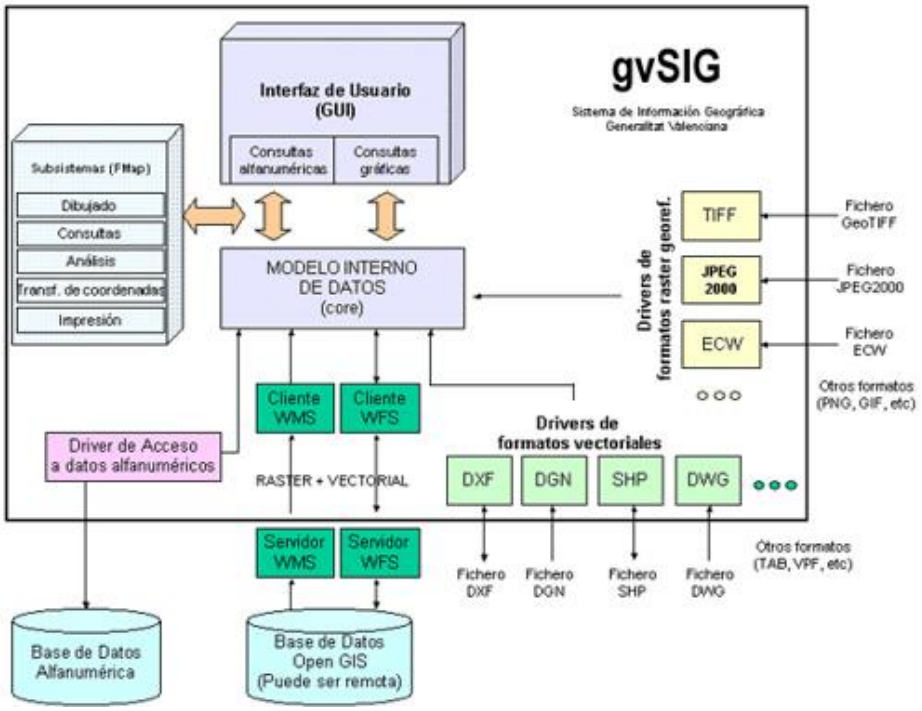

Figura 1 Arquitectura de gvSIG

Debido a que la distribución actual de gvSIG no integra soporte a servicios de mapas teselados (*Web Map Service-Cache* y *Web Map Tiled Service*) ni a ficheros en formato ligero (*GeoJSON)* estos fueron diseñados y desarrollados. Estos servicios y formatos permiten ofrecer la información geográfica (ráster y vectorial) de forma rápida y es por ello por lo que son utilizados en la infraestructura IDEZar.

La integración de estos nuevos desarrollos en el sistema, junto con las nuevas herramientas de edición y gestión, se realizaron de manera natural gracias al modelo de gvSIG basado en extensiones, el cual permite añadir nuevas funcionalidades sin la necesidad de modificar nada de lo creado previamente. Los clientes ráster de servicios teselados conectan las bases de datos de estos servicios con el core de gvSIG. Mientras los drivers de *GeoJSON* pasan a formar parte de los drivers de formatos vectoriales. Las nuevas herramientas de edición y gestión quedan integradas en la interfaz de usuario, utilizando por debajo los métodos e instrumentos proporcionados por FMap para la gestión y la edición. De este modo se obtiene la arquitectura del nuevo sistema, ver Figura 2.

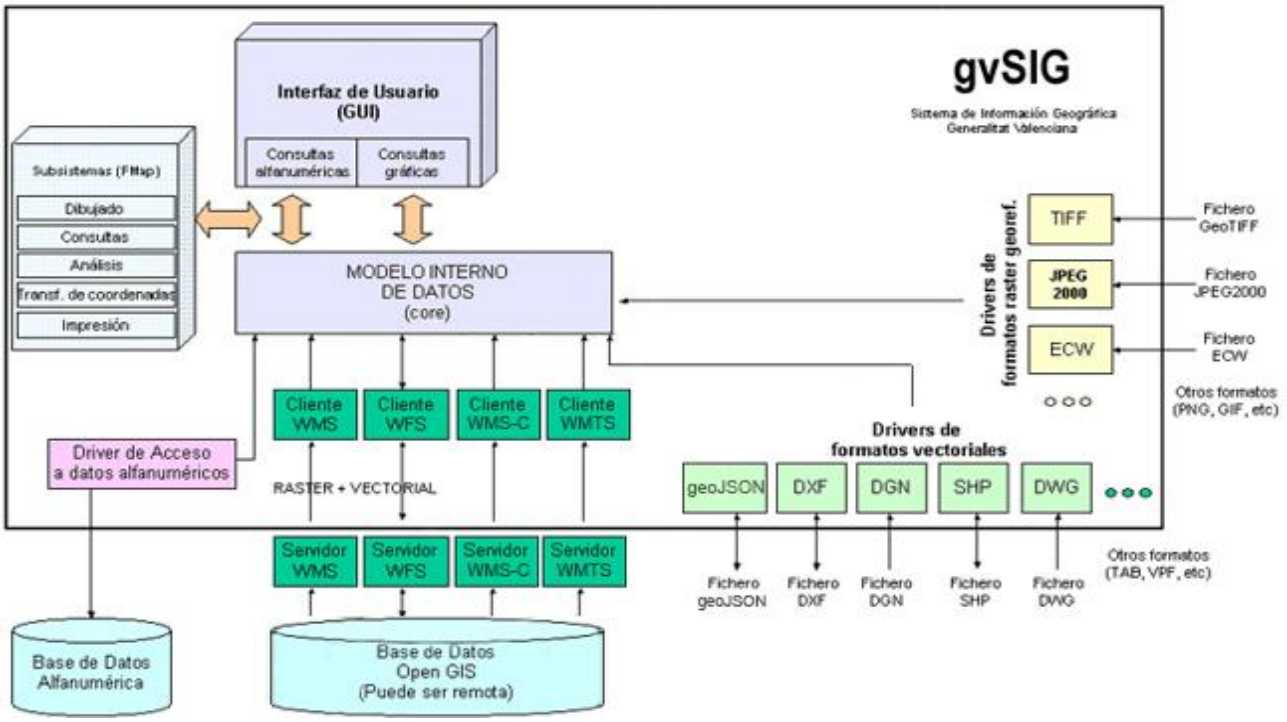

Figura 2 Arquitectura del nuevo sistema

La integración del nuevo sistema en la infraestructura IDEZar es inmediata ya que al desarrollar las herramientas para poder trabajar con los formatos principales

utilizados en IDEZar se consigue un flujo de trabajo perfecto, como se puede observar en la Figura 3, en las tres fases gestión, publicación y consumo de la información.

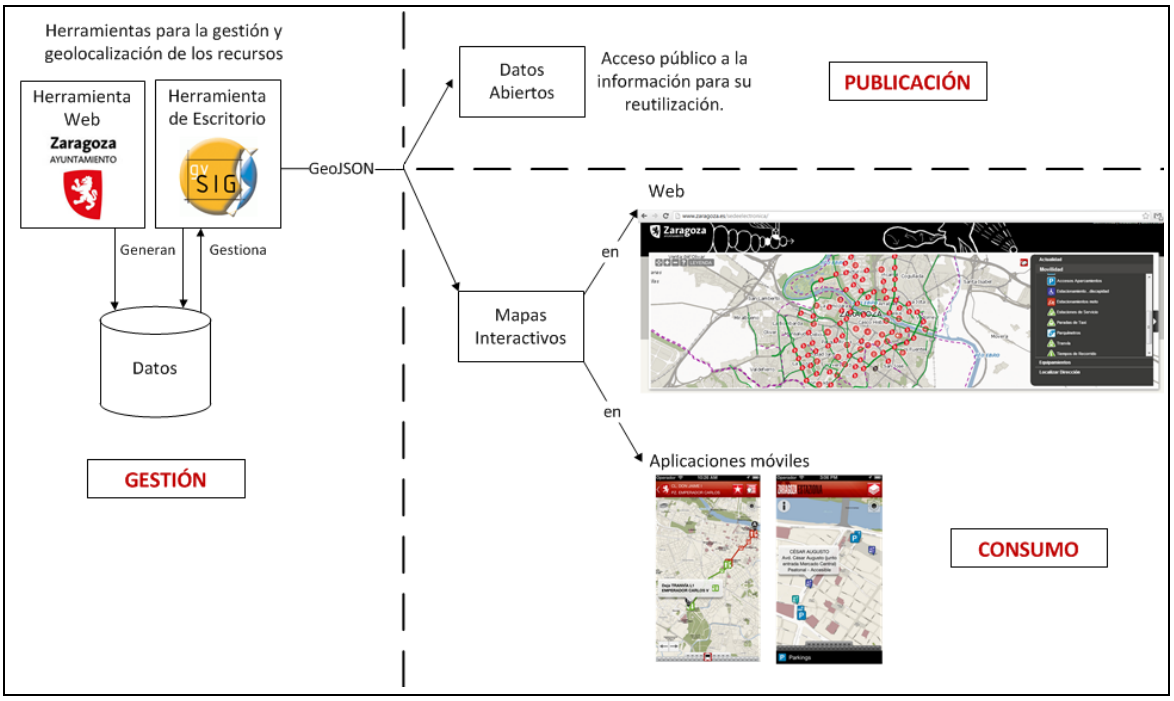

Figura 3 Flujo de gestión de los datos georreferenciados en IDEZar

El nuevo sistema se encarga de generar y gestionar la información georreferenciada y exportarla en formatos ligeros para su publicación en la plataforma datos abierto de Zaragoza (esta plataforma permite la reutilización de esta información por parte de la ciudadanía y las empresas) y para su consumo por los ciudadanos a través de los mapas interactivos de la web municipal y su aplicaciones móviles.

2.4. Diseño y desarrollo

Conociendo claramente las entradas y salidas necesarias del nuevo sistema para su integración en la gestión de IDEs y siendo que gvSIG permite trabajar sobre el concepto de extensión para añadir funcionalidades, se han diseñado e implementado una serie de extensiones enfocadas a cubrir estas necesidades.

Dependiendo del problema a solucionar se perfilaron diferentes extensiones, en el Anexo D – Diseño e implementación de las extensiones se explican con mayor detalle, las cuales se exponen a continuación.

2.4.1. Extensiones cliente WMS-C y WMTS

Se realizaron dos extensiones que permitieran el cargado de mapas teselados a partir de los servicios *Web Map Service-Cache* (WMS-C) y *Web Map Tile Service* (WMTS). Estos dos estándares de servicios de mapas teselados (o tileados) no estaban implementados en gvSIG ya que han surgido en los últimos años como solución al problema que poseía el estándar *Web Map Service* (WMS).

Hasta hace 3 años únicamente se utilizaba el *Web Map Service* (WMS) [5] como el estándar de servicios Web para producir mapas dinámicamente a partir de datos espaciales referenciados geográficamente. Cuando se diseñó este servicio, el área de los mapas mostrados en pantalla era relativamente pequeño, pero con el aumento de la velocidad de transmisión de los datos y de la resolución de las pantallas ha aumentado también el tamaño de los mapas a generar provocando que el servicio no fuera suficientemente eficiente en situaciones en las que el tiempo de respuesta y de pintado del mapa eran importantes

El principal inconveniente del WMS reside en su sencillez: raramente hay dos peticiones me mapas WMS iguales, con lo que el servidor no puede aprovechar las respuestas anteriores para despachar rápidamente las nuevas peticiones. Con el aumento del número de usuarios, el servidor recibe diversas peticiones concurrentes para mapas similares pudiendo llegarse al colapso por exceso de trabajo. Algunos fabricantes vieron estas ineficiencias y aparecieron diversas estrategias que discretizan el espacio. Así aparecieron los estándares *Web Map Service-Cache* (WMS-C) y *Web Map Tile Service* (WMTS) [6].

La mejora radica en que mientras en el servicio WMS se realizaba una única petición al servidor indicando la extensión del mapa y sus coordenadas, calculando y renderizando de este modo en el servidor la imagen pedida y devolviéndosela al cliente. En los servicios teselados WMS-C y WMTS ya existen unas imágenes prerrenderizadas para ciertos niveles de resolución, evitando de este modo el generar una imagen nueva por cada petición. Además estos servicios componen el mapa en pequeños tiles, los cuales son pedidos en paralelo para formar después el mapa, reduciendo así el tiempo de descarga.

2.4.1.1. Diseño y funcionamiento

Para la interacción con los servicios WMTS y WMS-C se siguió el patrón de diseño que ya existía en el componente del cliente WMS desarrollado en gvSIG. En la ventana que el usuario tiene para seleccionar la fuente de datos de una nueva capa (ver Figura 4) se añadieron dos nuevas pestañas "WMS-C" y "WMTS". Una vez seleccionado el servicio, se introduce la URL del servicio a conectar en el cuadro de texto y se conecta.

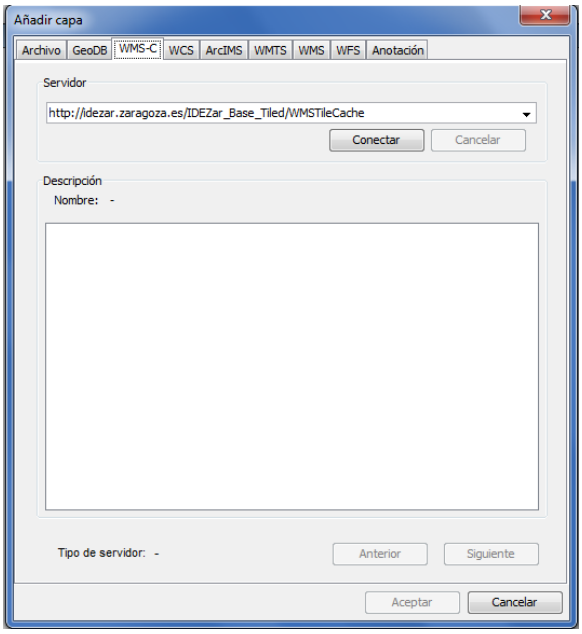

Figura 4 Interfaz gráfico para conectarse a un servicio WMS-C

En ese momento la aplicación pide el *capabilities*. El *capabilities* es un fichero en formato XML que contiene toda la información que puede devolver el servicio: capas, formatos de proyección, estilo y formato de imagen. El *capabilities* es parseado y mostrado al usuario de manera interactiva para que le sea sencillo la selección de los parámetros. Una vez terminada la selección se pulsa el botón aceptar, con los parámetros seleccionados por el usuario y se genera una petición al servicio para que devuelva los tiles que conforman el mapa en ambos casos. Una vez recogidos los tiles se escalan y pintan en la posición de pantalla adecuada, mostrando de este modo el mapa.

2.4.2. Extensión para la importación y exportación en formato ligero GeoJSON

Además de la inclusión de servicios ráster en el nuevo sistema se desarrollaron las herramientas necesarias para interactuar con información vectorial almacenada en ficheros *GeoJSON*. *GeoJSON* es un formato de intercambio geoespacial basado en *JSON* (*JavaScript Object Notation*), el cual es un formato ligero para el intercambio de datos que se ha hecho popular gracias a su simplicidad frente a XML. Los servicios integrados en IDEZar utilizan este formato para el intercambio de la información georreferenciada permitiendo su visualización en diferentes aplicaciones y mapas interactivos.

Para afrontar la importación y exportación siguiendo el estilo utilizado en gvSIG para otros ficheros, se implementaron tres drivers, dos para la exportación y uno para importar capas en este formato.

2.4.2.1. Exportación

Como se ha indicado previamente, se desarrollaron dos drivers de exportación [7], que afrontan dos problemas diferentes. El primer driver consiste en la exportación de la capa cargada en gvSIG tal cual en formato *GeoJSON*, es decir el fichero creado al exportar contiene las *features* de la capa origen con todos los atributos que poseían.

El segundo driver de exportación se desarrolló con el propósito de permitir una mayor integración de la nueva herramienta en el entorno IDEZar. Este driver permite la exportación de una capa en formato *GeoJSON* pero con un formato específico utilizado en la infraestructura IDEZar. De este modo se consigue una continuidad en el proceso edición con la herramienta, ya que una vez terminada la edición se puede exportar la capa generada a un formato específico que permite directamente su consumo en diferentes aplicaciones y servicios. La importancia de este formato reside en la inclusión del estilo de dibujado dentro de cada *feature*, de esta forma se indican las características con las que se tienen que pintar la información georreferenciada. El formato del *GeoJSON* utilizado en IDEZar está explicado en detalle en el Anexo D – Diseño e implementación de las extensiones.

2.4.2.2. Importación

El driver de importación permite cargar una capa en formato *GeoJSON*. En gvSIG para cargar una capa a partir de un fichero hay que ir a la ventana añadir capa y en la pestaña "Archivo" buscar la ruta del fichero a importar (ver Figura 5). gvSIG carga todos los drivers de importación que posee para saber si puede abrir el fichero seleccionado. El funcionamiento del nuevo driver se basa en convertir el fichero *GeoJSON* en *shapefile* de este modo gvSIG interactúa con un tipo de archivo que reconoce. Es decir se utiliza el driver de *shapefile* que incorpora gvSIG como puente entre los *GeoJSON* y el sistema y todo esto de forma transparente al usuario. Para un mayor detalle de la implementación consultar el Anexo D – Diseño e implementación de las extensiones.

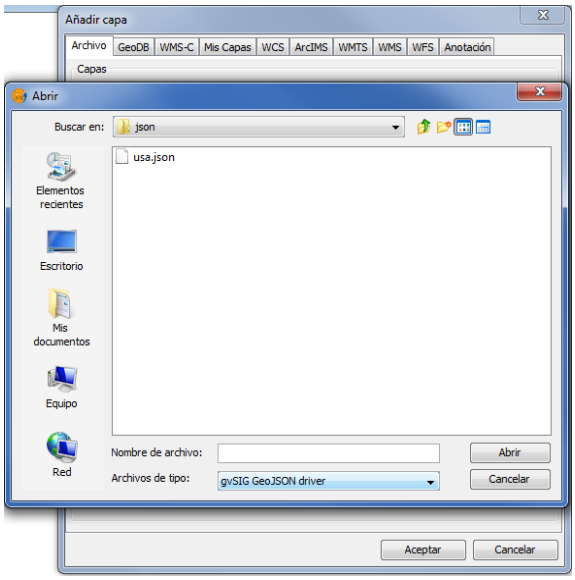

Figura 5 Importación de un GeoJSON a través de la pestaña Archivo

2.4.3. Extensión para cargar y guardar capas independientemente del origen de datos

Una vez se soportan nuevos formatos de información en el sistema es preciso agilizar y facilitar la tarea de utilizar estas fuentes de datos externas. Es por ello que se desarrolló esta extensión cuyo principal objetivo es permitir el cargado y guardado de capas de información vectorial o ráster de forma transparente para el usuario, a partir de diferentes fuentes de datos. De este modo se pueden guardar capas de datos preconfiguradas evitando tener que volver a configurarlas cada vez que se vayan a utilizar en un nuevo proyecto de gvSIG.

2.4.3.1. Interfaces

Se han creado dos tipos de interfaces para esta extensión. La primera interfaz, la cual se utilizará para guardar una capa que ya esté cargada en el sistema y configurada de forma deseada con el objetivo de recuperarla cuando se desee. La segunda interfaz permite esta recuperación de forma fácil e intuitiva.

La primera interfaz (ver Figura 6), consiste en una opción añadida en el cuadro de opciones que aparece cuando se pulsa con el segundo botón del ratón sobre una capa.

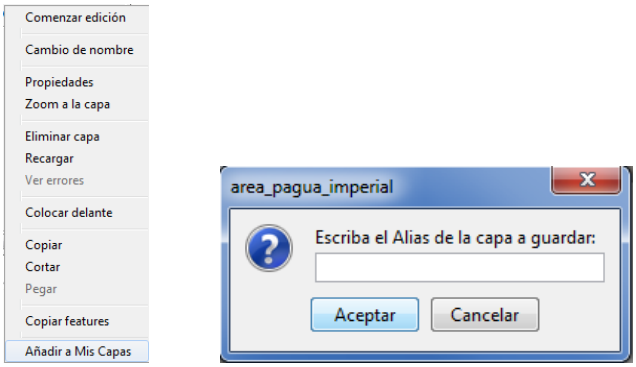

Figura 6 Menú de opciones sobre una capa y la ventana para introducir su Alias

Esta opción llamada "Añadir a Mis Capas" al ser seleccionada abre un diálogo de texto para que el usuario introduzca un Alias a la capa. Cabe destacar que no se permite introducir un nombre ya utilizado en otra capa guardada, con el objetivo de evitar la eliminación o sobrescripción de capas ya creadas.

La segunda interfaz (ver Figura 7), consiste en una pestaña nueva en la ventana de añadir capa.

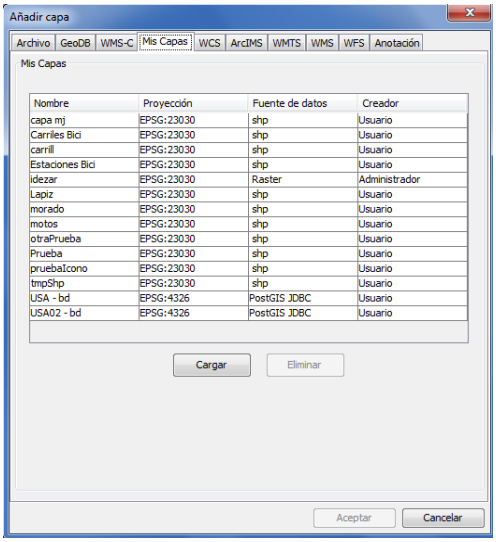

Figura 7 Interfaz gráfica que permite cargar una capa preconfigurada

En esta pestaña aparecerá la lista con todas las capas guardadas, las cuales se podrán añadir al presente proyecto de gvSIG pulsando doble click sobre ellas o seleccionándolas y pulsando el botón cargar. Además del Alias con el que se ha guardado la capa, la pestaña muestra la información de la proyección de cada capa, el tipo de la fuente de datos y su creador.

2.4.3.2. Tipos de usuarios

El creador indica si la capa la ha creado un administrador o un usuario estándar. Los administradores son los que crean las capas preconfiguradas que después los usuarios estándar las cargan para poder trabajar sobre ellas sin conocer su fuente de datos y configuraciones.

Por ello se implementaron dos herramientas diferentes, una para los administradores y otra para los usuarios estándar. Ambas permiten el guardado y el cargado de capas preconfiguradas, la diferencia radica en que las capas generadas con la herramienta del administrador no pueden ser eliminadas por el usuario estándar, con el objetivo de evitar posibles errores por parte de este, mientras las capas creadas por él sí que se pueden eliminar.

2.4.3.3. Funcionamiento

Las capas se almacenan en ficheros XML en la carpeta “Mis Capas” del directorio de preferencias de gvSIG. En este fichero se guardan todas las propiedades de la capa que son necesarias para recuperarla como la ruta a la fuente de datos, sus configuraciones, estilos de dibujado si tiene, etc.

2.4.4. Extensión para la carga del Overview Map

Esta extensión muestra un *Overview Map* (ver Figura 8) de las capas cargadas en ese momento en el programa, es decir una vista en miniatura del mapa que está cargado.

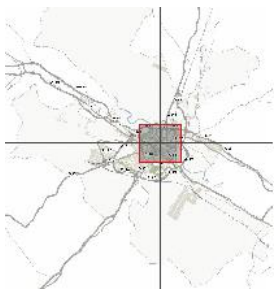

Figura 8 Vista en miniatura del mapa

El primer diseño de esta extensión mostraba todas las capas cargadas en ese momento en la vista, pero se consideró poco eficiente ya que al ser de dimensiones reducidas, cuantas más capas se mostrasen menos información se apreciaba, perdiendo de ese modo su utilidad. La decisión tomada para solucionarlo fue simplemente mostrar la capa ráster cargada en el nivel más bajo, porque es la capa que contendrá más información debido a que se suele poner abajo la capa ráster sobre la que se irán cuadrando las demás capas vectoriales.

En esta extensión es necesario conocer el momento en el que se añade, elimina o mueve una capa, para poder refrescar la imagen si la última capa ráster fuera cambiada.

2.4.5. Extensión para la edición y gestión de datos vectoriales y su simbología

Una vez se pueden acceder a servicios de mapas teselados y a información vectorial en formato *GeoJSON* es necesario diseñar nuevas herramientas que permitan la edición y gestión de datos vectoriales y su simbología de forma sencilla, por ello se desarrolló una barra de edición colocada en el lateral derecho de gvSIG.

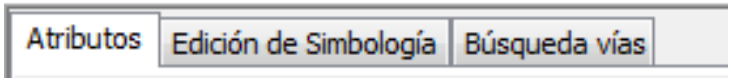

Figura 9 Pestañas de la herramienta de gestión de edición

Esta barra está formada por tres pestañas (ver Figura 9) con diferentes funcionalidades. La primera "Atributos" permite la edición rápida de las *features* de una capa vectorial. La segunda pestaña, "Edición de simbología" permite editar fácilmente la simbología asociada a una capa, es decir asociar colores, formas o imágenes a las *features* de una capa. Por último la pestaña de "Búsqueda vías" permite buscar una dirección y devuelve una serie de resultados posibles, permitiendo seleccionar uno y centrar así el mapa sobre la dirección indicada.

Este interfaz facilita en gran medida realizar todas estas funciones de edición que no eran triviales ni sencillas de realizar en la distribución oficial de gvSIG.

2.4.5.1. Diseño y funcionamiento

Las pestañas de “Atributos” y de “Edición de simbología” muestran información referente a una capa vectorial, en concreto a la que está seleccionada en ese momento. Es por ello que para su correcto funcionamiento es necesario conocer en cada instante si se ha cambiado la selección de la capa, se ha eliminado o añadido una nueva. De este modo si no hay ninguna capa seleccionada, estas dos pestañas aparecen bloqueadas. Esto se realiza implementando unos *listeners* específicos de las capas que aporta gvSIG. Los *listeners* en Java permiten escuchar eventos producidos por alguna clase y actuar al respecto. A continuación se explican más detalladamente cada una de las tres pestañas:

Pestaña Atributos

La pestaña “Atributos” (ver Figura 10) muestra las *features* de una capa vectorial, permitiendo navegar sobre ellas y editarlas.

A partir del fichero fuente de datos, se cargan en memoria todas las *features* de la capa. De este modo se carga la información de dicha estructura en la extensión y se van mostrando los valores de los diferentes campos de cada *feature* por pantalla.

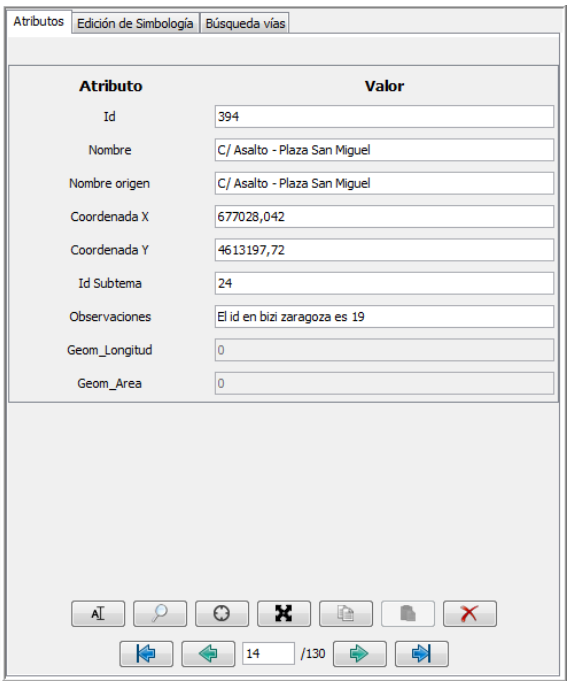

Figura 10 Pestaña Atributos

Como se aprecia en la captura de la herramienta, en la parte superior se muestran los nombres de los atributos que tiene la capa y a su lado los valores de la *feature* seleccionada en ese momento.

Para poder modificar estos valores se optó por obligar a poner la capa en edición, ya que en un principio se permitía la edición de los valores en cualquier momento, pero se observó que el funcionamiento dejaba de ser intuitivo y se permitía realizar una funcionalidad fuera de su ámbito, como es modificar datos fuera de la edición. La modificación de los datos se realiza en memoria y es al terminar la edición de la capa cuando los datos de la memoria son volcados sobre la fuente origen de los datos (*shapefile*, base de datos, etc)

En la parte inferior se encuentra el panel de botones, estos botones han sido los mínimos necesarios para poder facilitar en gran medida la edición, evitando botones y funcionalidades inútiles que sobrecargaran el interfaz. A continuación se van a explicar la función de cada botón y por qué se decidieron tales funcionalidades.

Comenzando por la primera fila de botones, el primer botón permite asignar un Alias al nombre del atributo, los ficheros *shapefiles* como las bases de datos solo permiten un número máximo de caracteres para sus atributos, esta funcionalidad permite asignar nombres más intuitivos a los atributos ayudando al usuario a la comprensión de lo que representan los atributos. El segundo botón corresponde a la búsqueda de *features* [9], permite buscar por el valor de un atributo específico, de este modo se puede acceder a una *feature* deseada rápidamente. El tercer botón permite centrar el mapa sobre la *feature* que está seleccionada en ese momento, manteniendo la escala del mapa actual. El cuarto botón permite centrar el mapa sobre la *features* que está seleccionada en ese momento haciendo zoom sobre ella. El quinto botón, que solo está habilitado cuando la capa está en modo edición, permite copiar los valores actuales de una *feature*. El sexto botón, permite pegar la *feature* copiada con el botón copiar, por lo que solo se habilitará cuando se haya copiado alguna *feature*. El último botón elimina la *feature* seleccionada.

La segunda fila de botones corresponde a los botones para la navegación entre *features*. El primero selecciona la primera *feature* y centra el mapa sobre ella. El segundo botón selecciona la *feature* anterior y centra el mapa sobre ella. El tercer botón hace lo mismo pero sobre la *feature* siguiente. Y el último botón selecciona la última *feature* y de nuevo centra el mapa sobre ella.

Por último se enumeran a continuación brevemente las decisiones más importantes de diseño que se llevaron a cabo en esta herramienta:

- La edición de los atributos de una *feature* solo se permite si la capa está en edición.
- El centrado sobre una *feature* solo se produce cuando se navega sobre ellos con las flechas, se deshabilitó esta funcionalidad cuando se cambiaba de una capa a otra porque dejaba de ser útil para el usuario y empezaba a ser engorroso.
- Al importar una capa vectorial, siempre se selecciona la primera *feature*, pero cuando se navega entre capas se recuerda la última *feature* sobre la que se ha estado en cada capa, permitiendo un uso más lineal.
- Al crear una *feature* nueva en una capa, la herramienta muestra el formulario vacío de esta nueva *feature*, permitiendo una edición rápida.
- Al eliminar una *feature*, se selecciona y muestra los valores de la *feature* anterior.

Pestaña de edición de simbología

En esta pestaña se recoge de una manera concisa las herramientas necesarias para la edición de simbología única para una capa (ver Figura 11). Se permite seleccionar un color o una imagen para las *features*. Así como el tamaño, las unidades y el ángulo en grados de la simbología.

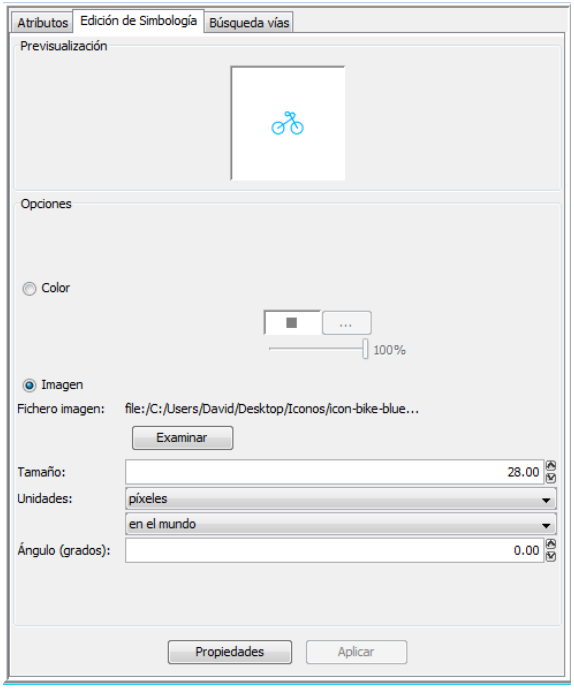

Figura 11 Pestaña de Edición de Simbología

Con esta pestaña se pretende que el usuario pueda modificar la simbología de una capa rápidamente, sin tener que estar navegando por complicadas ventanas para al final conseguir el mismo resultado.

Pestaña de búsqueda de vías

Esta pestaña (ver Figura 12) permite buscar una dirección postal y centrar el mapa sobre ella utilizando un *geocoder*. Al centrar el mapa aparece el icono en el mapa indicando la localización exacta del lugar que se ha buscado. El *geocoder* utilizado es el que está implementado en IDEZar, pero la extensión se diseñó permitiendo incluir más en el futuro.

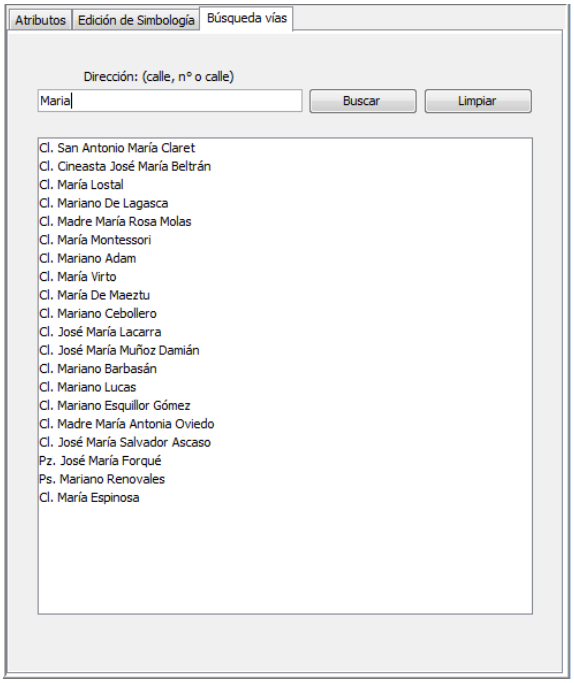

Figura 12 Pestaña de búsqueda de vías

Su funcionamiento reside en una conexión http al servicio SRW del *geocoder* [10], a donde se realizan peticiones a través del protocolo HTTP-GET. Todo el funcionamiento detallado queda explicado en el Anexo D – Diseño e implementación de las extensiones.

El *geocoder* de IDEZar principalmente utiliza el sistema de coordenadas EPSG: 23030, por lo que si se está trabajando en gvSIG sobre otro sistema de coordenadas, el resultado obtenido no tendría sentido. Es por ello que se comprueba que se está trabajando en dicho sistema de coordenadas, si no es así, se realiza una transformación de coordenadas de la proyección EPSG:23030 a la que se esté trabajando.

2.5. Configuración de la infraestructura

Una vez desarrolladas las nuevas herramientas se pasó a configurar el nuevo sistema. Como se ha mencionado repetidamente gvSIG está formado por extensiones las cuales van añadiendo funcionalidades a la aplicación. El problema reside en que no siempre es útil tener sobrecargada la aplicación de funcionalidades y barras de herramientas ya que esto perjudica al usuario final, principalmente a los que únicamente utilizan unas pocas herramientas del programa. Ahí es donde entra en juego la configuración de gvSIG [11]. Todas las extensiones contienen un fichero llamado config.xml en el cual se recoge la configuración de la extensión. Esta configuración indica si la extensión ha de estar activa (si se debe cargar), si aparecerá en un menú y en cuál o si estará en una barra de herramientas, su posición en ella y su icono.

Se han ocultado las barras de herramientas menos útiles, es decir no se cargan en el interfaz en un inicio al menos que el usuario en tiempo de ejecución lo haga a través del menú "Ver/Barra de herramientas". Además se han aligerado las barras eliminando botones de funcionalidades demasiado complejas o muy poco útiles, y se han renombrado las que no tenían un nombre descriptivo, como por ejemplo "gvSIG" que era la barra de herramientas que permitía cargar y guardar proyectos. De este modo el menú que permite ocultar o mostrar estas barras de herramientas muestra unos nombres más intuitivos facilitando la usabilidad.

Se pueden observar en las siguientes capturas las barras de herramientas y los botones que aparecían cuando había una capa en edición en el sistema (ver Figura 13), al no ser necesarios la gran mayoría se prescindieron de ellos obteniendo un interfaz mucho más limpio e intuitivo (ver Figura 14) orientando de este modo una vez más el sistema al personal sin conocimientos en los entornos SIG.

Figura 13 Antes de la configuración

Figura 14 Después de la configuración

Por último, también se planteó la posibilidad de reordenar y renombrar los menús ya que la usabilidad tal vez fuera más intuitiva si las opciones se organizaran en los menús de otra forma, por ejemplo la opción “nueva capa” aparecía en el menú “Vista” en vez de en el menú “Capa”. A parte de ese cambio que se ha realizado, al ser los cambios de reorganización de los menús muy subjetivos se han dejado los demás como estaban originalmente y se ha redactado un manual para los administradores del sistema donde se recogen los pasos a realizar para poder configurar de forma rápida los menús y barras de herramientas así como deshabilitar extensiones. Este manual se encuentra en el Anexo G – Manual del Administrador.

2.6. Caso de uso

Ilustrar un caso de uso es una técnica muy buena en sistemas interactivos para expresar la interacción que tiene el usuario (actor) al hacer uso del nuevo sistema y mostrar que se han solventado sus necesidades.

A continuación se explican dos casos de uso realizados. El primer caso elegido ha sido el Servicio de Movilidad Urbana del Ayuntamiento de Zaragoza. La misión principal de este servicio es la planificación, organización, ordenación, gestión y el control del tráfico y del transporte público desde una visión integral de la movilidad urbana, garantizando el correcto desplazamiento de los ciudadanos y la accesibilidad a Zaragoza.

Por ejemplo en el caso concreto de las líneas de transporte, el técnico municipal, el cual no tiene habilidades ni conocimientos en entornos SIG, tenía que utilizar el software gvSIG para gestionar las líneas. Lo cual se convertía en una ardua tarea ya que gvSIG no estaba diseñado para este tipo de usuarios.

Antes del nuevo sistema, se utilizaba el servicio WMS para obtener el mapa base para la georreferenciación. Como se ha explicado previamente, el servicio WMS no es el más eficiente hoy en día debido al gran tamaño de los mapas que devuelve, produciendo de este modo un gran retraso temporal. Esto se ha solventado gracias a la implementación de los servicios WMS-C y WMTS. El nuevo sistema al soportar estos dos servicios permite un cargado mucho más rápido del mapa, implicando también una

navegación por él mucho más fluida (ya que cada vez que se cambia de posición el mapa, este tiene que ser descargado y pintado), lo cual facilita en gran medida la labor de edición.

La navegación por el mapa también se ha agilizado gracias a la integración de un *Overview Map*, la vista en miniatura del mapa permite al usuario saltar rápidamente de un punto a otro del mapa, frente a arrastrar el mapa con el cursor hasta el lugar deseado, evitando de este modo peticiones y dibujados de mapa innecesarios reduciendo el tiempo de edición y mejorando la experiencia.

Además gvSIG no poseía una herramienta sencilla para la edición y gestión de datos. La nueva herramienta implementada permite navegar, realizar búsquedas y editar las *features* de una manera intuitiva agilizando de este modo el flujo de trabajo de la gestión. Esta herramienta también contiene un *geocoder*, el cual al igual que el *Overview Map* agiliza la navegación por el mapa, ya que si se quiere editar una *feature* localizada en una calle específica, se puede buscar esta calle con el *geocoder* para que centre el mapa sobre ella.

Por otro lado, ahora el usuario tiene disponibles gracias a Mis Capas una serie de capas preconfiguradas, sabiendo que si carga la capa “paradas” cargará las estaciones de autobuses perfectamente configuradas para poder editarlas rápidamente, sin necesidad de tener que importar el fichero que contiene las geometrías y configurar su estilo cada vez que se quiera editar.

Una vez terminada la edición, la nueva herramienta de exportación en formato *GeoJSON* con estilo de dibujado permite crear una pipeline perfecta ya que se puede exportar directamente en el formato en el que se va a consumir la información.

El flujo de trabajo para la gestión de las líneas de transporte publico sería:

- Se carga el mapa base con el servicio WMS-C.
- Se cargan las capas preconfiguradas de las líneas de transporte público y de las paradas.
- Se abre el *Overview Map* para tener una vista en miniatura del mapa y navegar rápidamente por él.
- Se abre la nueva herramienta para la edición y gestión (ver Figura 15).

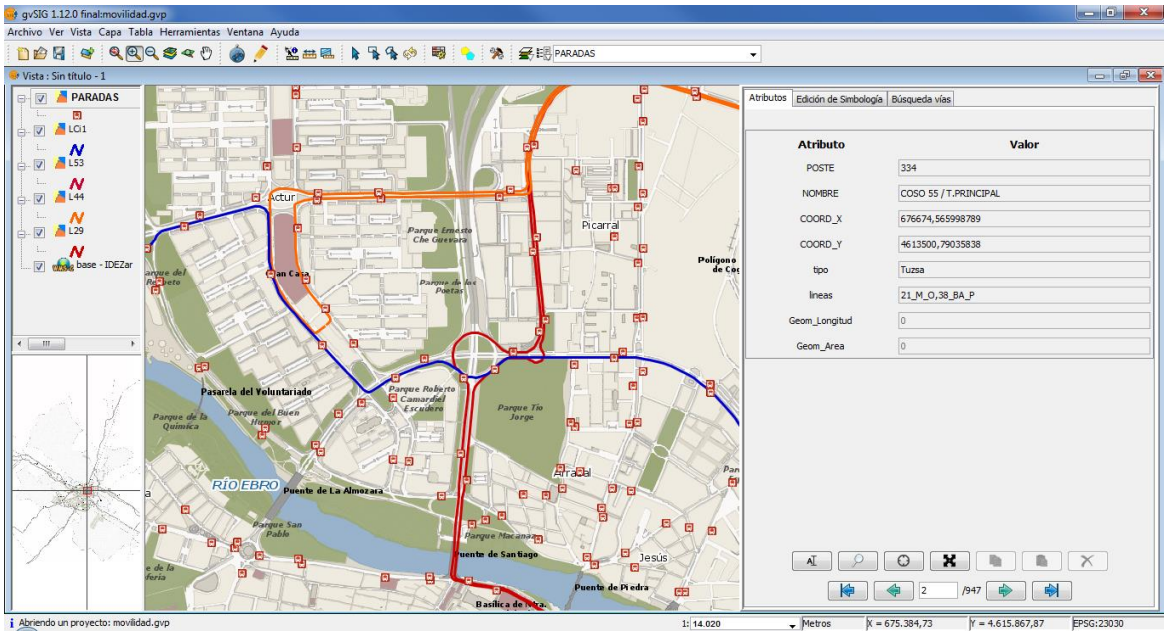

Figura 15 Edición de las paradas y líneas de autobuses

- Se edita la capa de las líneas de transporte público y las paradas con esta nueva herramienta, la cual permite la edición de forma sencilla, mostrando al mismo tiempo el mapa y la información textual asociada al elemento que se está editando en ese momento. Además incorpora herramientas que facilitan la edición como navegar, copiar, pegar o realizar búsquedas fácilmente entre los distintos elementos de la capa de trabajo.
- Una vez terminada la edición se exporta en *GeoJSON* con estilo de dibujado (ver Figura 16), al exportarlo con estilo permite que los visores en los que se utilizará puedan interpretar su color o icono dibujándolos de ese modo.

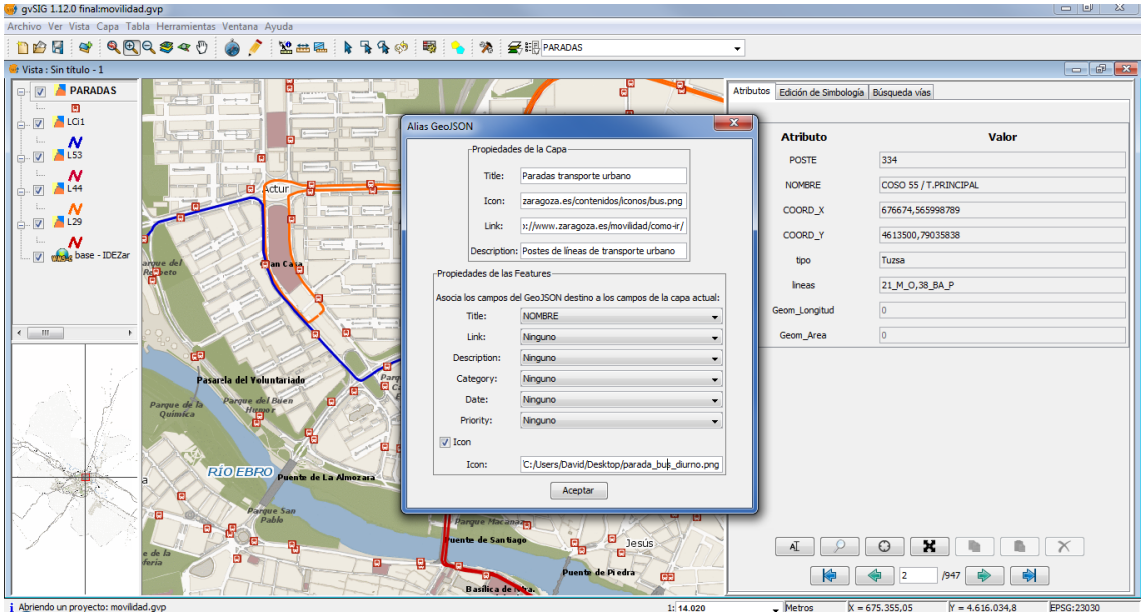

Figura 16 Exportación de las paradas de autobuses a GeoJSON con estilo

- El nuevo fichero *GeoJSON* es utilizado por visores interactivos avanzados y aplicaciones web para mostrar su información, facilitando el día a día a la ciudadanía. Se utiliza este formato para ofrecer la información de forma rápida y dinámica de cara al usuario final. Un ejemplo de estas aplicaciones web es la herramienta "Cómo moverse en transporte público" [12] (ver Figura 17)

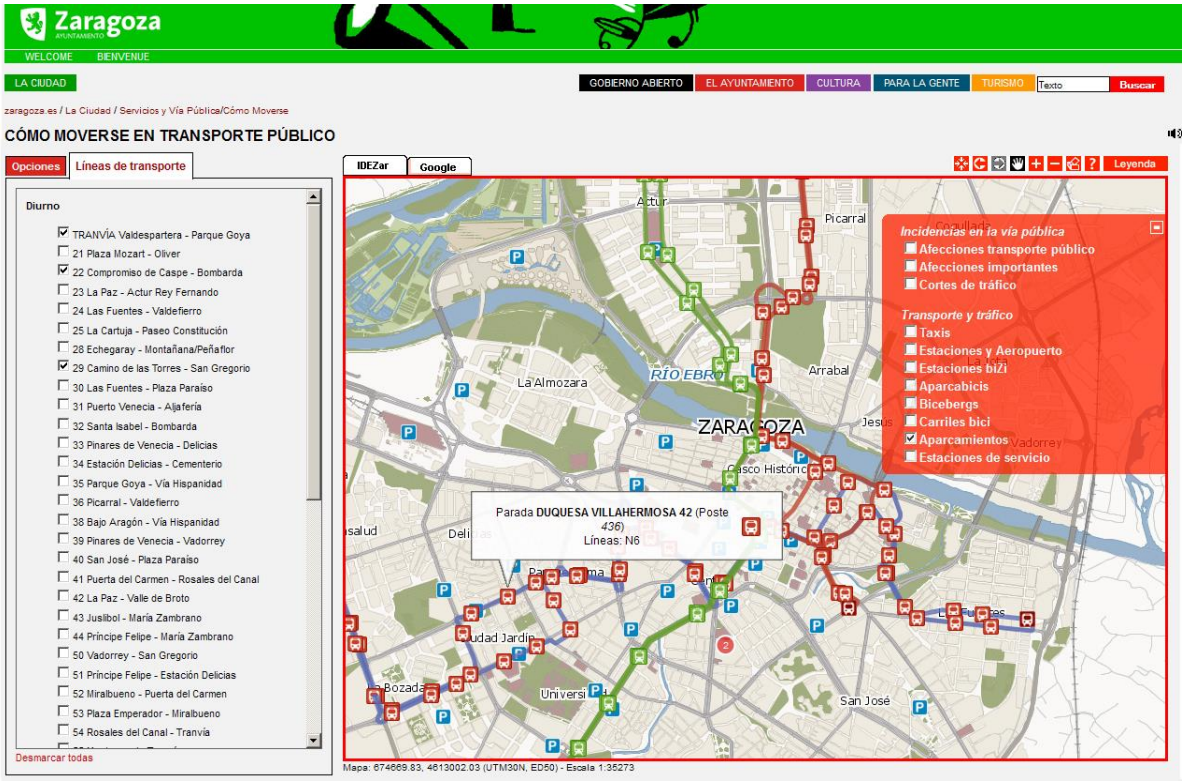

Figura 17 Aplicación web "Cómo moverse en transporte público"

- También es usado en aplicaciones móviles como por ejemplo las aplicaciones Zaragoza Estaziona[13] o Zaragoza Rutas[14] (ver Figura 18), de ahí la importancia en exportar el resultado final en formato *GeoJSON*, un formato ligero que permite una transmisión eficaz y rápida de los datos evitando tiempos largos de espera.

[12] http://www.zaragoza.es/ciudad/viapublica/movilidad/como-ir/
[13] Descarga de la aplicación para iOS:
https://itunes.apple.com/es/app/zaragoza-estaziona/id643623742?mt=8
Descarga de la aplicación para Android:
https://play.google.com/store/apps/details?id=es.zaragoza.estaziona
[14] Descarga de la aplicación para iOS:
https://itunes.apple.com/es/app/zaragoza-rutas/id643590940?mt=8
Descarga de la aplicación para Android:
https://play.google.com/store/apps/details?id=es.zaragoza.rutometromultimodal

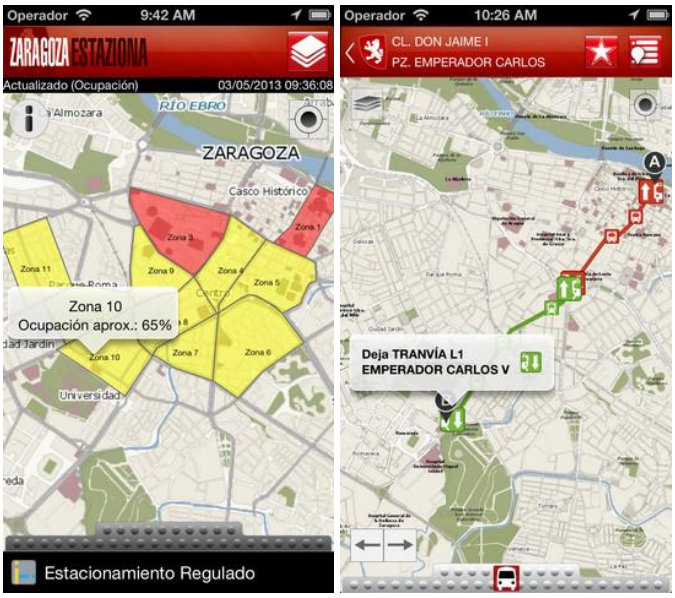

Figura 18 Aplicaciones móvil Zaragoza Estaziona y Zaragoza Rutas

El otro caso de uso realizado ha sido referente al área de medio ambiente, donde se ha generado toda la información que muestra el nuevo mapa verde[15] de Zaragoza, una nueva iniciativa del Ayuntamiento de Zaragoza que intenta reflejar la sostenibilidad y la calidad del medio ambiente urbano con el objetivo de convertir a la ciudad en la capital verde Europea en el año 2014. A continuación se explica en detalle la gestión y edición de la capa Zgz Anda, esta capa muestra las rutas periurbanas por Zaragoza las cuales son unas rutas sustentadas en la práctica del senderismo y la utilización del transporte público.

- Se carga el mapa base de Zaragoza a través del servicio WMS-C. Se carga la capa vectorial "zgzanda" a través de la herramienta de "Mis Capas" ya que es una capa que ha sido preconfigurada por un administrador (ver Figura 19), el cual utilizando también esta nueva herramienta ha generado las rutas y asociado un estilo específico a cada ruta utilizando la pestaña desarrollada de edición de simbología, guardando el resultado en una capa de Mis Capas, para que el técnico del ayuntamiento pueda trabajar rápidamente sobre ella.

[15] http://www.zaragoza.es/ciudad/medioambiente/parques/

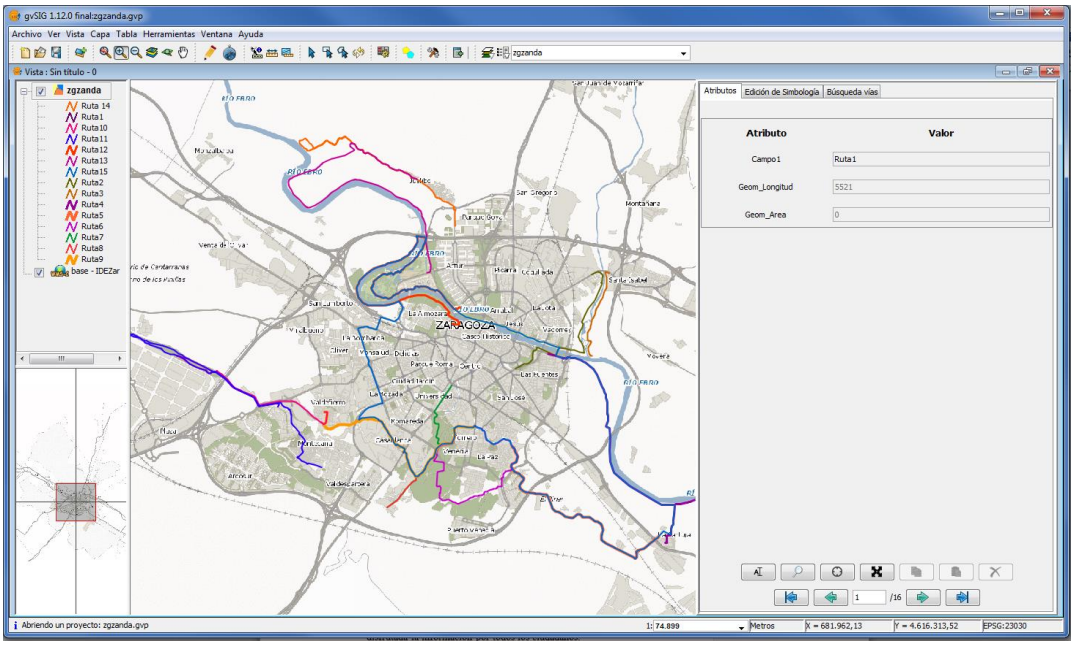

Figura 19 Edición de la capa ZgzAnda

- Se exporta la capa en *GeoJSON* con estilo (ver Figura 20), para poder ser visualizada en cualquier visor de IDEZar.

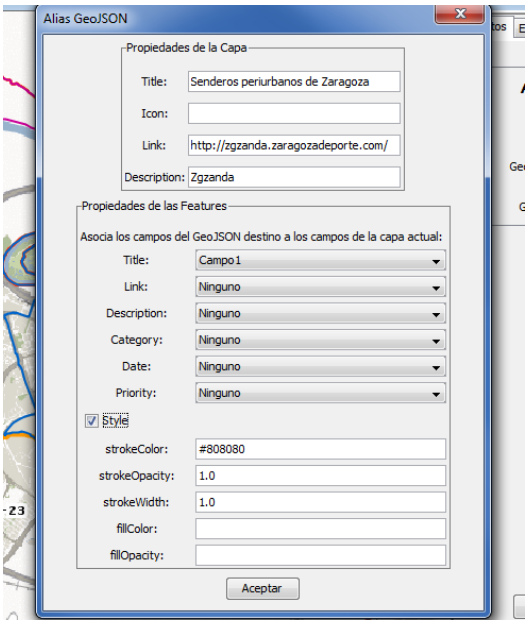

Figura 20 Exportación de la capa en formato GeoJSON con estilo

- Finalmente la información georreferenciada de los senderos es accesible en el Mapa Verde de la ciudad (ver Figura 21), desde la web municipal para poder ser disfrutada la información por todos los ciudadanos.

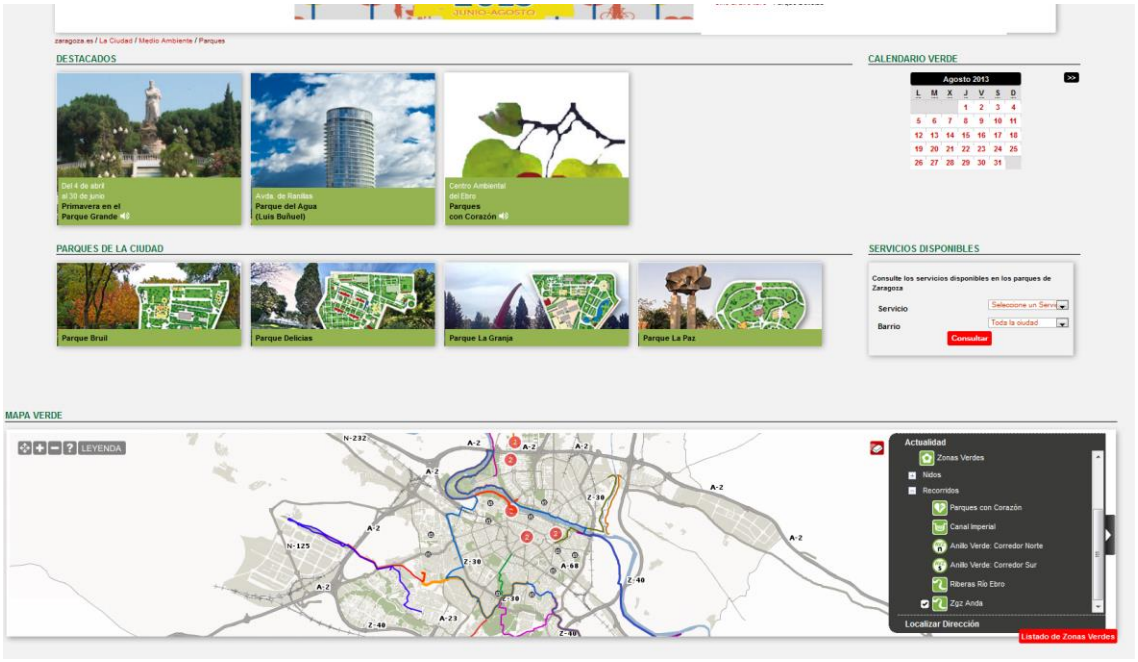

Figura 21 Mapa Verde de Zaragoza en la web del ayuntamiento

Con estos casos de uso se puede observar que la herramienta desarrollada es una solución tecnológica transversal que no se restringe a un área municipal en concreto, sino que es utilizada por gran parte de estas como son las áreas de movilidad, medioambiente, participación de la ciudadanía, etc.

2.7. Pruebas funcionales

Las pruebas funcionales son pruebas específicas cuyo objetivo es evaluar y validar que el software cumple sus funciones y lo que se ha especificado. Estas pruebas se realizan de manera manual a través del interfaz gráfico del software.

Las fases llevadas a cabo en estas pruebas son:

- Diseño del plan de pruebas: en esta fase se identifican las características que se quieren probar del nuevo software, diseñando las pruebas con el objetivo de encontrar defectos.
- Ejecución: una vez diseñadas las pruebas se ejecutan manualmente, como si de un usuario del sistema se tratara.
- Gestión de incidencias: se anotan los defectos encontrados al realizar las pruebas con el objetivo de solucionarlos.

- Volver a ejecutarlas hasta que no haya incidencias: una vez solucionado los defectos se vuelven a ejecutar las pruebas. Este ciclo se repite hasta que no aparecen más incidencias.

Se diseñó una hoja de cálculo donde se indicaban en diferentes columnas la descripción/propósito de la prueba, las acciones previas, acciones que hay que ejecutar, los resultados esperados y comentarios si se habían producido incidencias. Esta tabla se puede ver al completo en el Anexo E – Pruebas funcionales.

2.8. Tests de usabilidad

Los tests de usabilidad son una técnica que ayuda a determinar la facilidad de uso de un nuevo producto revelando problemas reales a los que se enfrenta un usuario al utilizar la aplicación bajo condiciones normales de uso.

El funcionamiento de los tests es el siguiente, se les da a los usuarios una serie de tareas reales a realizar dentro de un escenario de actuación. Una vez terminadas las tareas se les realiza unas preguntas para poder evaluar el grado de usabilidad y aceptación del nuevo sistema.

Para la realización de las pruebas se ha cogido un grupo de usuarios sin perfil técnico ni conocimientos SIG previos y se les ha puesto en contexto explicándoseles previamente la utilidad del sistema. Las tareas que tuvieron que hacer para poder evaluar la usabilidad de las nuevas herramientas de edición fueron:

Tarea 1. Cargar la capa preconfigurada de estaciones bici de Zaragoza, a través de Mis Capas.

Tarea 2. Cargar la capa ráster de Zaragoza llamada “IDEZar”, a través de Mis Capas.

Tarea 3. Asociar un icono a los puntos que representan las estaciones bici con un tamaño adecuado. (En el escritorio se ha facilitado una carpeta llamada Iconos, con un icono bici)

Tarea 4. Cambiar el nombre de la estación “Pabellón Puente” por “Cambio realizado”.

Tarea 5. Eliminar la estación “Plaza España”.

Tarea 6. En la dirección “Pablo Neruda 30” añadir una nueva estación.

Tarea 7. Exportar la nueva capa en *GeoJSON* con estilo.

Tarea 8. Añadir la capa a Mis Capas, con el nombre “Estaciones bici con estilo”

Las preguntas realizadas tras la realización de las pruebas fueron:

- ¿Te han parecido lo suficientemente sencillas de usar las herramientas de edición?
- ¿Qué problemas has encontrado al realizar las tareas?
- ¿Qué cambios propondrías en las nuevas herramienta
- ¿Añadirías alguna funcionalidad nueva?

Los usuarios valoraron la dificultad de las tareas definidas obteniendo cada una el promedio mostrado en la Figura 22, las valoraciones se han puntuado del 1 al 5 siendo 1 muy difícil de realizar y 5 muy fácil.

Tarea	Promedio
1	4,5
2	4,7
3	4,6
4	3,7
5	5
6	1,3
7	4,8
8	4,2

Figura 22 Tabla de puntuaciones promedio de cada tarea

Por otro lado, las respuestas obtenidas a las preguntas formuladas coincidían en que las nuevas herramientas eran bastante intuitivas con una sencilla interfaz que identificaban claramente las acciones que se podían desarrollar en la edición de capas. El problema que más les ha surgido ha sido el añadir una nueva *feature*, ya que la opción no está integrada en las nueva herramientas, sino que está en la barra de herramientas de dibujado, confundiendo a los usuarios debido a su nula experiencia previa con el programa. Debido a esto los usuarios opinaron que estaría bien la integración de las herramientas de dibujado en la barra de gestión. Respecto a la última pregunta sobre añadir alguna funcionalidad los usuarios, sugirieron la posibilidad de insertar registros en la capa desde una hoja de cálculo.

3. CONCLUSIONES

En este último capítulo de la memoria se analizarán los resultados obtenidos, se indicarán posibles líneas de trabajo futuras y se expondrá una valoración personal del PFC actual.

3.1. Resultados obtenidos

El objetivo planteado al inicio del proyecto se ha alcanzado con éxito. El nuevo sistema desarrollado permite la edición y gestión de la información georreferenciada de una manera sencilla y ágil, facilitando las labores a los usuarios finales. Además con la implementación de los clientes de servicios de mapas teselados y los drivers para la importación y exportación en formato ligero *GeoJSON*, ha permitido integrar perfectamente la nueva herramienta en la infraestructura IDEZar, obteniendo unos resultados satisfactorios en los casos de uso en los que ha sido probada.

También se han redactado manuales debido a la complejidad de gvSIG y a la poca documentación existente, con el objetivo de facilitar la tarea a futuros desarrolladores, usuarios y administradores del sistema. El manual para montar el código fuente de gvSIG en Eclipse y poder así comenzar a desarrollar una extensión se encuentra en el Anexo F – Manual del Desarrollador. En el manual del administrador explica a los administradores la manera de realizar una configuración personalizada del sistema, este manual está en el Anexo G – Manual del Administrador. Por último se escribió el manual de usuario para explicar la utilización de las nuevas herramientas implementadas, este manual se puede consultar en el Anexo H – Manual de Usuario.

3.2. Líneas futuras

A lo largo de la realización del presente proyecto y durante las pruebas con los usuarios han ido surgiendo nuevas propuestas en las que poder seguir desarrollando y

mejorando el sistema. Algunas de estas nuevas posibilidades que se han recogido son las siguientes:

- Integración de los mapas de GoogleMaps y las vistas de StreetView en el nuevo sistema. Los mapas de Google hoy en día son los más utilizados en el mundo por los usuarios finales, pero siguen sus propias proyecciones por lo que ahora mismo no están soportadas en gvSIG. Además permitir ver las vistas de StreetView de un punto concreto del mapa también podría ser interesante para obtener una geolocalización más exacta.

- Una sugerencia frecuente de usuarios no habituados con gvSIG es la inclusión de las herramientas de dibujado (creación de los puntos, líneas o polígonos) en la nueva barra de gestión, agrupando de este modo ya todas las funcionalidades en una misma herramienta y evitando tener las herramientas dispersas.

- La aparición de un Maptip (un bocadillo con texto dentro) con la información asociada a un elemento, cuando este es seleccionado en el mapa.

- El proyecto gvSIG también tiene su versión móvil, la cual es ideal para la captura y actualización de datos en campo. Sería muy interesante la inclusión de esta aplicación dentro de la infraestructura IDEZar, donde ayudaría principalmente a la creación de nuevos datos georreferenciados.

Además al lanzarse una puesta en marcha del sistema en el Ayuntamiento de Zaragoza, se recogerán sugerencias por parte de los usuarios permitiendo una mejora y ampliación del sistema con las nuevas ideas.

3.3. Valoración personal

Este proyecto ha supuesto la culminación de toda la formación académica recibida durante estos últimos 5 años, y ha supuesto poner en práctica todos los conocimientos adquiridos durante este tiempo en un proyecto real que va a ser utilizado en la vida real y va a ayudar en gran medida a sus usuarios.

El proyecto además me ha permitido adentrarme en el terreno de los SIG y la información georreferenciada, ampliando en gran medida mis conocimientos y aportando a mi formación un valor añadido.

El hecho de que se haya trabajado sobre un proyecto *Open Source* también ha implicado valorar aún más la documentación producida por un desarrollador, ya que en este caso la documentación de gvSIG era muy escasa y pobremente explicada, lo cual hizo el trabajo más difícil. Y no solo la documentación sino la creación de un código limpio y bien comentado ya que en este proyecto se tuvo que estudiar una gran cantidad de código ajeno, el cual tampoco estaba en las mejores condiciones de presentación. Por otro lado el observar código de otras personas con más experiencia ha permitido también aprender ciertas técnicas de programación y diferentes perspectivas para abordar y solucionar grandes problemas.

Al ser un proyecto de una cierta envergadura se ha puesto en práctica también la gestión de proyectos, definiendo metodologías de trabajo que permitieran agilizar el proceso.

En definitiva, la experiencia ha sido gratificante ya que realizando este proyecto, además de trabajar junto con personas excelentes que me han guiado y ayudado, se han obtenido una gran cantidad de conocimientos desde conocimientos SIG, técnicas eficientes de programación hasta de gestión y desarrollo de proyectos.

II ANEXOS

ANEXO A
GESTIÓN DEL PROYECTO

En un proyecto software de una envergadura relevante, es muy importante una buena planificación para conseguir unos buenos resultados en el tiempo deseado. En este anexo se explica la metodología de trabajo seguida y las herramientas utilizadas durante el desarrollo del proyecto.

A. 1. Metodología de trabajo

El proyecto se desarrolló en la empresa GeoSpatiumLab, la cual tiene una larga experiencia con proyectos de gran envergadura, por lo que se siguieron algunas de sus metodologías para la gestión del proyecto. Además GeoSpatiumLab con la coordinación del Responsable de la Calidad y el Medioambiente, ha promovido un Sistema Integrado de Gestión de la Calidad y el Medioambiente sobre la base de los requisitos de las Normas UNE-EN-ISO 9001:2008 (noviembre de 2008) y UNE-EN-ISO 14001:2004 (diciembre de 2004), para todas las actividades que desarrolla, que se complementa con los requisitos de la norma ISO/IEC 15504 "*Software Process Improvement Capability Determination*" (SPICE), nivel de capacidad 2, con el objetivo primordial de satisfacer a sus clientes mediante la prestación de servicios de la máxima calidad y respeto al entorno. Este Sistema Integrado de Gestión obtuvo su certificación formal en el año 2011.

En concreto para el proceso de desarrollo de este proyecto se utilizó el ciclo de vida de desarrollo iterativo incremental. Esta metodología se basa en la construcción de secciones reducidas de software que son probadas y van ganando tamaño con el tiempo, facilitando de este modo la detección de errores. Además permite la modificación de algunas especificaciones por parte del cliente, permitiendo un desarrollo más flexible al cambio.

gvSIG está implementado en el lenguaje de programación Java, por lo que el proyecto al completo se realizó sobre este lenguaje. Al ser Java un lenguaje orientado a

objetos, se utilizó el modelo OMT (*Object Modeling Technique*) durante el desarrollo del proyecto.

La primera etapa de este proyecto fue la formación por parte del proyectando de las tecnologías y estándares de mapas, así como la investigación y estudio de la arquitectura de gvSIG. Al no tener un conocimiento base geográfico y el estudio de código de gvSIG escrito por terceros provocó que esta etapa se alargara en el tiempo más de lo estimado inicialmente.

Una vez se tuvieron los conocimientos necesarios para empezar a trabajar se pasó a realizar un análisis del sistema a implementar, especificando los requisitos. Sabiendo ya cómo trabaja gvSIG y los objetivos que se querían alcanzar se propusieron los primeros diseños del sistema. Después se pasó a la implementación por módulos. Es decir, gvSIG al trabajar sobre un modelo basado en extensiones, donde se permiten crear extensiones para ampliar su funcionalidad sin alterar el código original de la aplicación, permitió dividir los diferentes objetivos a conseguir en extensiones, reduciendo de este modo el tamaño de los módulos a desarrollar y permitiendo un mejor desarrollo iterativo.

A lo largo de todo el proyecto se han llevado a cabo control de versiones y copias de seguridad de los elementos generados, como son código y documentación. Además la empresa cuenta con un control de esfuerzos permitiendo conocer el tiempo dedicado al proyecto.

El proyecto se ha desarrollado entre noviembre de 2012 y agosto de 2013. El número de horas dedicadas aproximadamente han sido de 920. En el siguiente diagrama de Gantt (ver Figura 23) se puede ver la planificación que se ha llevado acabo.

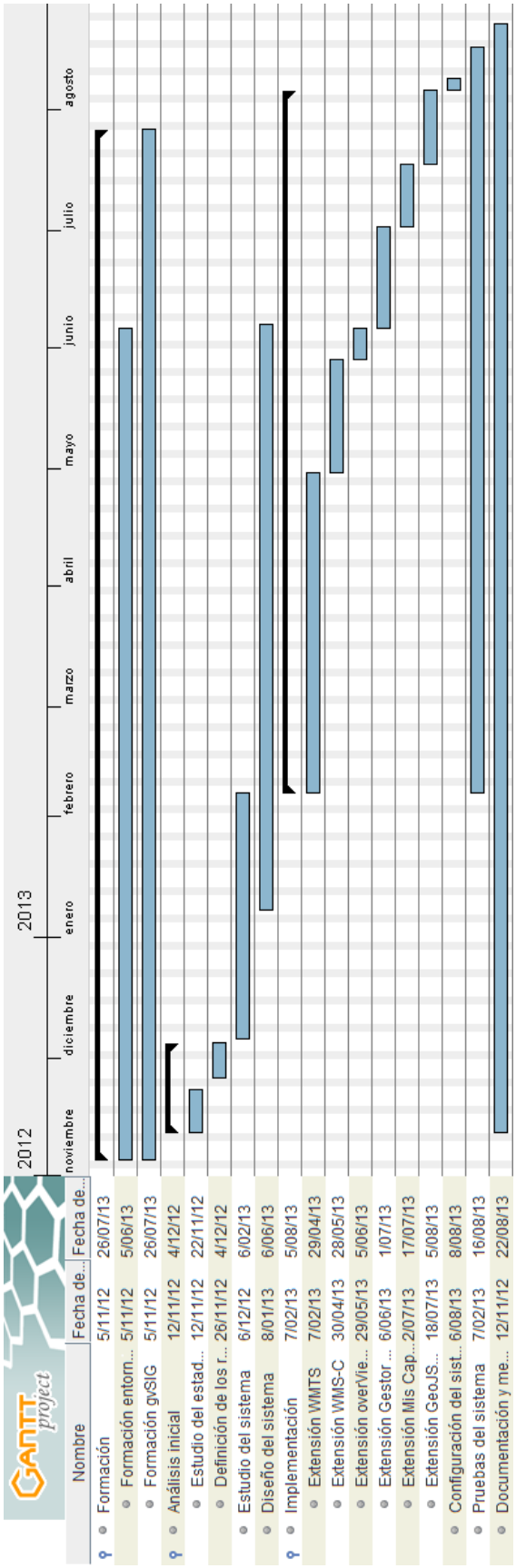

Figura 23 Diagrama de Gantt

A. 2. Herramientas utilizadas

A continuación se detallan las herramientas utilizadas para la realización del presente proyecto. Las herramientas han sido clasificadas según sus ámbitos de utilización.

- Desarrollo

  Eclipse: entorno de desarrollo.

  Java JDK 1.6: máquina virtual Java.

  Notepad ++: lectura de códigos fuente.

- Documentación del proyecto

  Adobe Acrobat 9 Pro: lectura de manuales y guías.

  Microsoft Word: documentación y memoria del proyecto.

  Microsoft Excel: control de esfuerzos y pruebas del sistema.

  Microsoft PowerPoint: presentación.

  Gantt Project: planificación de tareas.

- Gestión del proyecto y copias de seguridad

  Dropbox: copias de seguridad.

  SVN: control de versiones.

- Análisis y diseño

  UMLet: diagramas de clases.

  Microsoft Office Visio: diagramas.

  Adobe Photoshop: creación de iconos para los botones y capas.

ANEXO B
ESTADO DEL ARTE

Antes de comenzar con el proyecto, se realizó un estudio del arte actual. Es decir, las técnicas y programas existentes hoy en día relacionados con la temática del proyecto. Este estudio permitió aportar nuevos puntos de vista al proyecto así como nuevas ideas no planteadas en un inicio. El estudio comenzó conociéndose los diferentes productos SIG que hay en el mercado actualmente.

B. 1. Programas SIG en el mercado

Existen una gran cantidad de programas SIG en el mercado, desde software libre a comercial. A continuación se destacan unos pocos software de cada categoría.

Hay una gran cantidad de software SIG comercial, por lo que se han elegido los más reconocidos. Frente a la imposibilidad de probar el software comercial, se ha obtenido la siguiente información.

ArcGIS: el SIG de mayor difusión en la actualidad. Es un software propietario, no es gratuito y lo desarrolla la empresa *Environmental Systems Research Institute* (ESRI). Está disponible únicamente para Windows y Linux. A parte de la aplicación de escritorio, para la captura, edición, análisis, tratamiento, diseño, publicación e impresión de información geográfica, posee también ArcGIS Server, para la publicación y gestión web y ArcGIS Móvil para la captura y gestión de información en campo. Con ArcGIS móvil permiten aumentar la precisión de los puntos georreferenciados y mejorar la actualización de los datos. Es fácil de usar por personal de campo que no tiene por qué tener experiencia en SIG.

Manifold: únicamente disponible para entornos Windows. Manifold en su versión 7.00 soporta datos vectoriales y ráster, incluye SQL espacial y posee un servidor IMS integrado, entre otras características generales de los sistemas SIG.

IDRISI: únicamente disponible para entornos Windows. Es uno de los más completos, el cual no necesita pluggins para añadir funcionalidades.

En el apartado de software libre se pueden destacar los siguientes:

GRASS GIS: GRASS (*Geographic Resources Analysis Support System*) fue inicialmente desarrollado por el laboratorio de investigación del cuerpo de ingenieros del ejército de los Estados Unidos para la gestión del territorio y la gestión medioambiental. GRASS comenzó a difundirse en ámbitos educativos y de instituciones públicas y se desarrollaron numerosas aplicaciones alrededor de dicho sistema, hasta que en 1999 pasó a tener licencia del tipo GNU GPL. Fue entonces cuando el desarrollo estaba en manos de todas las personas interesadas. Al ser GRASS uno de los SIG con más tiempo de rodaje, el número de herramientas y utilidades que presentas es muy elevado.

Uno de los inconvenientes principales de GRASS es el hecho de que está diseñado para entornos UNIX, lo cual ha frenado su expansión hacia el público general y su uso se limita a centros universitarios y de investigación. Hoy en día existen versiones de GRASS que se pueden instalar en entornos Windows a través de emulación de Cygwin. Otro de los inconvenientes es que este software es demasiado complejo para personas sin conocimientos SIG.

Ventajas	Desventajas
• Solidez por los orígenes militares y la edad del proyecto. • Herramientas de análisis raster y potente modelado hidrológico. • Editor de topología.	• Interfaz no muy amigable. • Diseñado para entornos UNIX/Linux • Complejidad de uso.

Quantum GIS: es un SIG con una apariencia muy cuidada y con características muy interesantes tales como, soporte directo para edición en PostGIS, conexión con GRASS para tareas de edición de topología y un gran número de formatos soportados tanto vectoriales como de imagen.

Permite además la inclusión de plugins, actualmente se pueden encontrar un buen número de ellos para tareas tan interesantes como la conversión de archivos shape de ESRI a PostGIS o para conectarse a un GPS y mostrar su posición. Aunque Quantum

GIS esté programado en C++, existen versiones compiladas para varios sistemas operativos entre los que se encuentran Windows y Linux.

Ventajas	Desventajas
• Buena interfaz. • Buen soporte de formatos de datos. • Tiene integrada la edición de topología con GRASS.	• Para los nuevos usuarios, especialmente aquellos con poca experiencia en el uso de datos espaciales o bases de datos, el programa puede parecer intimidante.

gvSIG: como contrapartida al ArcGis, se encuentra gvSIG. gvSIG es una herramienta orientada al manejo de información geográfica. Se caracteriza por una interfaz amigable, siendo capaz de acceder a los formatos más usuales de forma ágil tanto ráster como vectoriales. Integrará en una vista datos tanto locales como remotos a través de un origen WMS. Incluye principalmente las aplicaciones *gvSIG Desktop* y *gvSIG Mobile*.

- *gvSIG Desktop* tiene las herramientas propias de un completo cliente SIG de escritorio.

- *gvSIG Mobile* ideal para proyectos de captura y actualización de datos en campo. Se caracteriza por disponer de una interfaz amigable, siendo capaz de acceder a los formatos más comunes y cuenta con un amplio número de herramientas SIG y GPS ideales para trabajar con información de naturaleza geográfica.

Estos programas SIG tienen integrados un gran abanico de funcionalidades, en la mayoría excesivas si lo que se busca es una herramienta cartográfica sencilla e intuitiva y orientadas a personal no experto en SIG.

Ventajas	Desventajas
• Software orientado al usuario final, tanto a nivel de interfaz de usuario como de funciones implementadas. • Soporte para los formatos más populares tanto vectoriales como de imágenes. • Funcionalidades previstas muy completas. • Totalmente en español.	• No permite enlazar tablas (JOIN).

B. 2. ¿Qué se busca?

Hoy en día Internet se ha convertido en una herramienta muy útil para todos los ciudadanos. Es por ello que muchos ayuntamientos y servicios públicos están creando servicios con información georreferenciada. El problema reside en que para generar estos contenidos hay que utilizar programas SIG, lo que implica tener conocimientos del mismo, cosa que no se suele dar.

Conociendo estas limitaciones, se busca realizar un software sencillo e intuitivo, cuyos usuarios no sean expertos en SIG. Como a cada usuario final le interesarán unas características determinadas, el programa debe permitir la configuración para los diferentes entornos donde se pueda utilizar.

A partir de estas premisas, se han seleccionado 6 características principales para el software resultado que permitirán decidir sobre que producto SIG actual se trabajará. A continuación se exponen las características y su importancia.

Licencia: la licencia es uno de los rasgos más importantes a la hora de seleccionar la plataforma sobre la que trabajar. Esto hace que el software no libre se descarte automáticamente ya que no puede estudiarse ni modificarse. Existen muchas licencias de software libre. La licencia GNU (*General Public License*) permite la modificación del software, obteniendo uno nuevo también bajo esta licencia. GNU además permite vender copias del programa. De hecho software libre no significa gratis, en algunos casos los programas libres son distribuidos gratuitamente, y en otras ocasiones por un precio muy alto. A menudo, el mismo programa se puede conseguir de ambos modos de fuentes distintas. El programa es libre a pesar del precio, porque los usuarios tienen libertad al usarlo.

Plugins: el concepto de plugin es muy importante, permite añadir funcionalidades a un programa.

Customización: la customización es el proceso por el cual el consumidor selecciona las preferencias del producto o contenidos de información, que desea que le sean suministrados. Es decir, sería la capacidad del programa de crear diferentes configuraciones del mismo con el objetivo de satisfacer unas necesidades específicas del cliente.

Fuentes de datos: las fuentes de datos en este caso son los tipos de datos que soporta el programa. Es importante que trabaje con el mayor número posible y que además estos sean estándares y no tipos propios.

Exportación: ya que son programas de edición y creación, al consumidor final le interesa que el programa le brinde la oportunidad de salvar su trabajo en los formatos principales y más utilizados en la industria.

Multiplataforma: al ser un programa de edición cartográfica, el destinatario principal son administraciones públicas. Las cuales pueden tener cualquier tipo de sistema operativo en sus equipos, es por ello importante crear un programa disponible para el mayor número de ellos.

Con estas 6 características principales se ha elaborado la siguiente tabla (ver Figura 24), la cual permite comparar los software SIG más importantes del mercado.

	Licencia	Plugins	Customización	Fuentes de datos	Exportación	Multiplataforma
ArcGIS	No libre	Si	Si	Servicios remotos: OGC (WMS, WFS, WCS, WFS-T, WPS) Geodatabases (personal geodatabase, file geodatabase, and ArcSDE geodatabase)	Sin información, al ser un software de pago.	Windows, Linux, Unix, entorno Web, móvil
GRASS GIS	Libre: GNU	Si	Si	Bases de datos: DBF, SQLite, mySQL, ODBC, PostgreSQL	Mapa raster, mapa vectorial, mapa 3D raster, tabla base de datos.	Windows, Linux, Unix, entorno Web
Quantum GIS	Libre: GNU	Si	Si	Servicios remotos: OGC (WMS, WFS, WCS, CAT, SFS, y GML)	Capas Shapefile, formatos de intercambio GPS (GPX), PostGIS Mapa jpg, png, bmp, ppm, tif, xbm, xpm	Windows, Linux, Unix, entorno Web
gvSIG	Libre: GNU	Si	Si	Servicios remotos: OGC (WMS, WFS, WCS, WFS-T, WPS) (en la versión 2.0 en desarrollo WMTS), ArcIMS, Ecwp Bases de datos y tablas: PostGIS, MySQL, ArcSDE, Oracle, JDBC, CSV	Capas Shapefile, Oracle Spatial, Geography Markuo Language (GML), Keyhole Markup Lenguage (KML), dxf (Autocad), PostGIS Mapa jpg, png, bmp, WMC (Web Map Context)	Windows, Mac, Linux, Unix, móvil

Figura 24 Comparativa entre ArcGIS, GRASS GIS, Quantum GIS, gvSIG

B. 3. ¿Por qué gvSIG?

gvSIG es software libre, lo que permite estudiarlo y modificarlo a nuestro libre albedrío. Como se ha visto, gvSIG es un software SIG de los más completos. Además es un proyecto vivo español, con una comunidad en aumento, aunque no llega al nivel de la plataforma ArcGIS, gvSIG está creciendo en su uso como una alternativa viable al SIG comercial. La asociación gvSIG publica casos de uso de gvSIG en diversos sectores y lugares geográficos, tanto de los productos oficiales – *gvSIG Desktop*, *gvSIG Mobile* – como desarrollos a medida.

Además la activa comunidad de desarrolladores trabaja arreglando bugs y extendiendo la funcionalidad de gvSIG, lo que consigue que sea un software actualizado y facilita su desarrollo.

Esta comunidad de gvSIG tiene como principal medio de comunicación las listas de correo (ver Figura 25). A través de estas los usuarios, desarrolladores y demás actores en el proyecto se comunican y resuelven las tareas del día a día del proyecto.

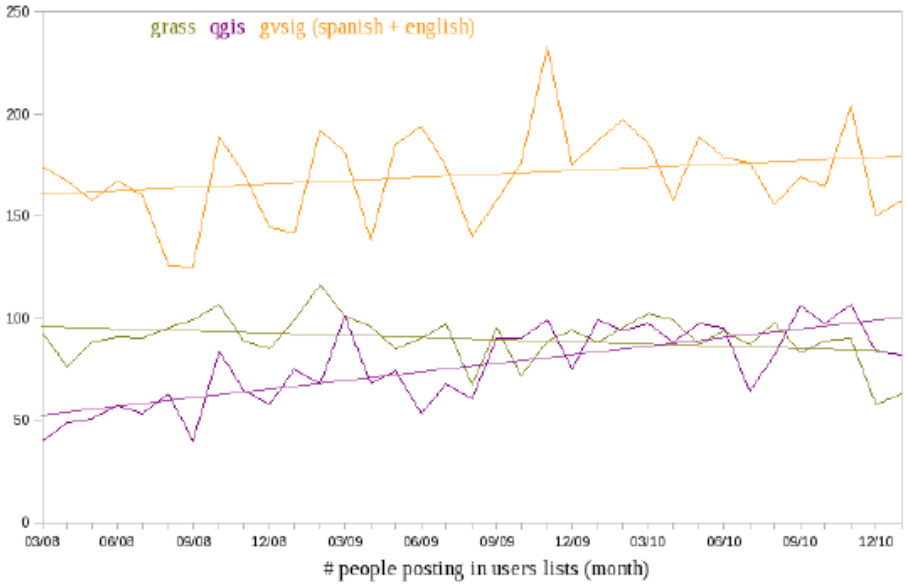

Figura 25 Gráfica de número personas que postean en las listas de distribución de los diferentes software SIG

gvSIG es el futuro. Cada vez es mayor el número de empresas que utilizan software SIG *Open Source* frente al comercial debido a entre otros al recorte de gastos por parte de las agencias y empresas. Ahí es donde gvSIG entra en juego, aportando la misma o mejor calidad que el software comercial en un software de código abierto.

B. 4. Nueva versión gvSIG en desarrollo

Como siempre, una plataforma como es gvSIG está en constante evolución y es por ello que van apareciendo nuevas versiones en desarrollo. Actualmente la última versión es gvSIG 1.12. Pero ya existe una nueva en desarrollo, gvSIG 2.0. Estas versiones en desarrollo no son estables y no se recomiendan utilizarlas aun, de este modo en este proyecto se utilizará la versión 1.12, pero vendrá bien echar un vistazo a alguna de las nuevas mejoras que aporta la versión 2.0 por si se decidiera más adelante incluirlas. Estas mejoras son las siguientes:

- Se ha rediseñado la forma en la que gvSIG maneja las fuentes de datos con el objetivo de mejorar tanto la fiabilidad como la modularidad, beneficiando así tanto a usuarios como a desarrolladores.
- Nuevo instalador que soporta instalación típica y personalizada. Mediante la instalación personalizada el usuario tiene control sobre los complementos que instala. Además, una vez instalado gvSIG pueden agregarse nuevos complementos.
- Algunos cambios en el interfaz de las herramientas de manejo de datos como:
    - ✓ Importación/exportación de ficheros.
    - ✓ Operaciones con tablas.
    - ✓ Nueva capa.
- Mejoras en el rendimiento de carga de capas. Ahora el usuario puede trabajar con la aplicación mientras se carga la capa.
- Soporte de WMTS (*Web Map Tiled Service*). El WMTS es un nuevo servicio estándar OGC que mejora al popular WMS gracias al manejo de imágenes tileadas.
- Caché de datos raster. Mejora el rendimiento en la visualización de datos raster.
- Soporte formato NETCDF (vectorial/ráster). NETCDF es un formato para datos científicos que soporta, entre otras cosas, datos multitemporales.
- Soporte de datos temporales. Se ha añadido una herramienta de filtrado temporal para ficheros que soportan la dimensión tiempo como NETCDF.
- Importar/exportar símbolos. Esta herramienta permitirá compartir símbolos entre usuarios.
- Entorno de scripting. Nueva herramienta que facilita la programación de scripts en los lenguajes Python (Jython), Groovy y Javascript.

ANEXO C
ANÁLISIS DEL SISTEMA

En este anexo se van a proceder a presentar con más detalle los requisitos del sistema a implementar. Comenzando con los requisitos propuestos por el cliente y su transformación a requisitos del sistema divididos en funcionales y no funcionales. También se analizará más a fondo la arquitectura de gvSIG.

C. 1. Análisis del problema

El sistema desarrollado en este proyecto tiene dos principales objetivos, el primero permitir la gestión de la información geoespacial de forma intuitiva y sencilla, facilitando la tarea al personal que no tiene conocimientos SIG. El segundo formar parte de la infraestructura IDEZar, permitiendo un flujo de trabajo completo desde la creación y edición de información georreferenciada hasta su publicación en visores avanzados.

Para conseguir estos objetivos se trabajó sobre el software gvSIG, ampliando su funcionalidad y diseñando un sistema adecuado para el entorno en el que se requería. Para ello, lo primero fue detallar los requisitos del sistema a partir de las especificaciones del cliente.

C. 2. Análisis de requisitos

El cliente aportó desde un inicio una serie de requisitos que debería cumplir el nuevo sistema, los cuales han sido recogidos a continuación. Estos requisitos permitieron definir los requisitos del sistema los cuales indican qué es lo que debe hacer el sistema y son los datos de entrada a la etapa de diseño.

C.2.1 Requisitos del cliente

RC-1 Cargado y guardado de capas de información vectorial de forma transparente para el usuario.

RC-2 Desarrollo e integración de herramientas de edición y análisis de cartografía intuitivas y orientadas a personal no experto en GIS.

RC-3 Exportación de capas de información en formatos que garanticen el intercambio de información vectorial.

RC-4 Soporte del concepto de estilo sobre las herramientas de exportación anteriores.

C.2.2. Requisitos del sistema

Requisitos funcionales

RF-1 El nuevo sistema debe aportar nuevas herramientas intuitivas y orientadas a personal no experto en SIG para la edición de la información vectorial.

RF-2 El nuevo sistema debe permitir la selección de estilos de dibujado para la edición de información vectorial.

RF-3 El nuevo sistema debe soportar servicios de mapas teselados.

RF-4 El nuevo sistema debe permitir el cargado y guardado de capas de información vectorial de forma transparente para el usuario, a partir de diferentes fuentes de datos.

RF-5 El nuevo sistema debe permitir la exportación de capas de información en formatos ligeros de intercambio de información vectorial como *GeoJSON*.

RF-6 El nuevo sistema debe soportar la exportación de capas vectoriales con un estilo asociado.

RF-7 El nuevo sistema debe soportar la integración con el flujo de gestión de IDEZar.

Requisitos no funcionales

RNF-1 Implementar el nuevo sistema sobre gvSIG.

RNF-2 El nuevo sistema debe funcionar sobre Windows

C. 3. Análisis de la arquitectura de gvSIG

La plataforma gvSIG está construida a modo de capas, cada una de las cuales define sus puntos de extensión. A su vez, cada extensión o plugin puede definir sus propios puntos de extensión. Este modelo de plugin, permite añadir gran variedad de funcionalidades a la plataforma base de gvSIG.

La arquitectura de gvSIG 1.12 (ver Figura 26) se puede abstraer hasta un nivel donde se encuentran 3 entidades principales, la interfaz de usuario (GUI), FMap y el modelo interno de datos (core). El esquema de la arquitectura de gvSIG quedaría de la siguiente forma:

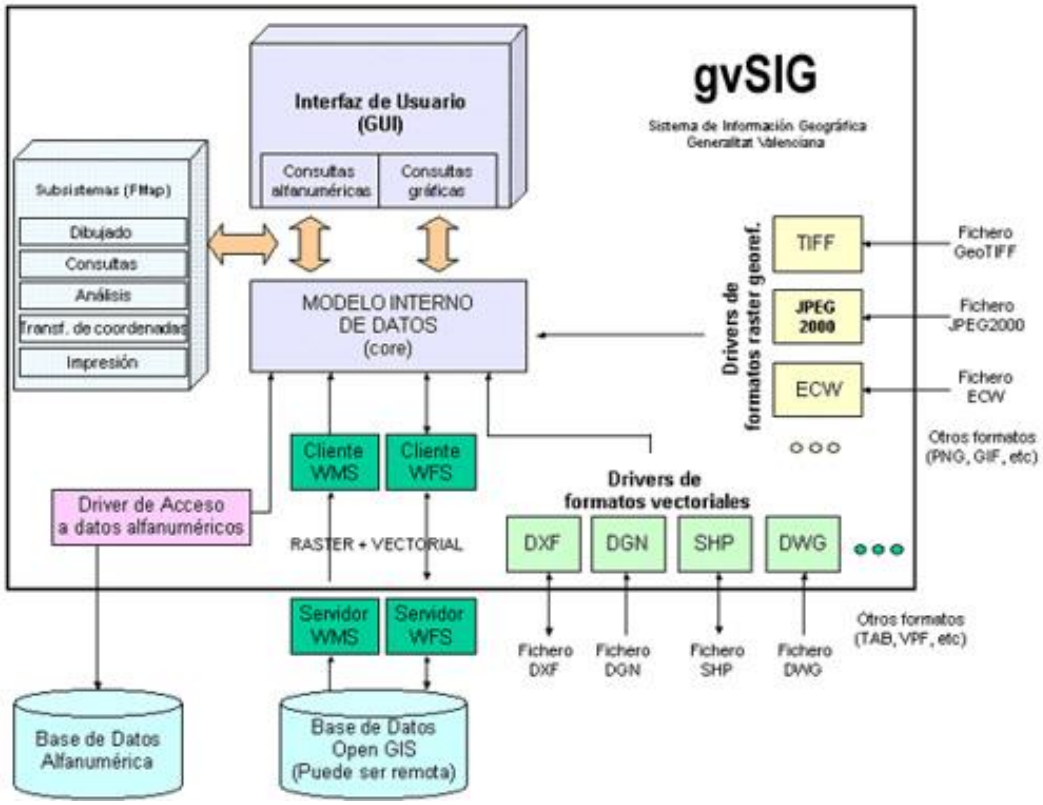

Figura 26 Arquitectura de gvSIG

La entidad de la interfaz de usuario (GUI) representa la parte visual de la aplicación y permite al usuario interactuar con los datos. Se compone de:

- Gestor gráfico Andami: es el cuerpo de la aplicación, todo lo demás son extensiones. Proporciona los métodos necesarios para que los plugins o extensiones puedan comunicarse tanto con la aplicación principal como entre ellos y soporta los métodos necesarios para gestionar el interface gráfico.

- Project: contiene los datos básicos del proyecto.
- Documents: son los distintos tipos de documentos que soporta la aplicación, Vistas, Tablas y Mapas. Este componente proporciona también los puntos de extensión necesarios para incluir nuevos tipo de documentos.
- View: representación gráfica de la cartografía.
- Layout: representación gráfica de una vista en un soporte apto para imprimir.
- Table: es la representación gráfica de los datos alfanuméricos.
- Layers: son el conjunto de capas que pueden insertarse en una vista de gvSIG

La entidad FMap es el motor de la aplicación. Incluye todas las clases necesarias para manejar objetos SIG, desde dibujar la cartografía hasta acceder a los datos. Se compone de un gestor de herramientas, capas y orígenes de datos. La entidad FMap se compone de:

- MapControl: se encarga de dibujar y mantener la herramienta actual, conoce todas las herramientas que existen en la aplicación.
- MapContex: es el contexto de la parte gráfica. Contiene los elementos necesarios para que el MapControl pueda realizar su labor.
- Behaivor: es un comportamiento de una herramienta. También dice cómo se comporta la herramienta gráficamente. Controla el dibujado de la herramienta y el iniciador de los eventos de la herramienta.
- Listeners: son los encargados de gestionar los eventos de las distintas herramientas.
- Layer: contiene las características de la capa y las herramientas necesarias para su gestión.
- Geometrías: son los distintos tipos de elementos gráficos que pueden ser representados dentro de una *layer*.
- DataSource y drivers: contiene los métodos necesarios para la gestión de los datos tanto gráficos como alfanuméricos.

Por último el modelo interno de datos (core), sirve de puente entre la aplicación y las fuentes de datos. Contiene las clases necesarias para acceder a los datos y escribirlos en una fuente. Los componentes de este subsistema son:

- RemoteService: contiene las herramientas necesarias para unificar el acceso a datos remotos.

- Drivers: gestiona los distintos tipos de datos soportados por gvSIG.
- DriverManager: proporciona la carga y el acceso a los drivers disponibles en la aplicación, tanto alfanuméricos como espaciales.
- WriterManager: proporciona la carga y el acceso a los *writers* disponibles en la aplicación, tanto alfanuméricos como espaciales.
- Writers: permite las operaciones de escritura sobre los distintos tipos de formatos soportados.
- VectorialSources: proporciona acceso a los datos con las geometrías.
- DataSources: proporciona acceso a los datos alfanuméricos.
- RasterSouces: proporciona acceso a los datos de tipo raster.

C. 4. División de las funcionalidades en extensiones

Como se ha explicado en el apartado anterior de la arquitectura de gvSIG, este está compuesto por extensiones que permiten añadir una gran cantidad de funcionalidades al sistema.

Al dividir las diferentes funcionalidades en extensiones independientes de gvSIG permite un diseño más conciso y una implementación más cuidada, ya que se crean módulos más pequeños lo que permiten un control de errores mucho mayor que con entidades muy grandes. Se enumeran a continuación las diferentes divisiones:

- Para la implementación de los clientes WMS-C y WMTS que permitieran a gvSIG conectarse a estos servicios remotos se diseñaron dos extensiones individuales, una para cada servicio.
- Para la importación y exportación en formato *GeoJSON* se diseñó otra extensión compuesta por tres drivers, dos de exportación y uno de importación.
- Para la edición y tratamiento de la información georreferenciada de forma sencilla e intuitiva, se diseñó una extensión que aglutinara diferentes herramientas que permitieran realizar esas funcionalidades de una forma sencilla.
- El cargado y guardado de capas preconfiguradas, de forma que el origen de la fuente de datos sea transparente para el usuario y evitando tener que volver a

configurarlas cada vez que se vayan a utilizar en un nuevo proyecto de gvSIG, se solucionó con el diseño de otra extensión.

- Con el objetivo de facilitar en mayor medida al usuario la edición, se diseñó otra extensión para el cargado de una vista en miniatura del mapa que estaba cargado en ese momento en la aplicación.

ANEXO D
DISEÑO E IMPLEMENTACIÓN DE LAS EXTENSIONES

Siguiendo el orden natural de todo desarrollo software, tras conocer los requisitos se pasa a la etapa de diseño e implementación. En este anexo se explicará esta etapa para cada una de las extensiones siguiendo el orden de desarrollo que se llevó acabo.

D. 1. Extensión del cliente WMTS

Esta es la primera extensión que se realizó y es por ello que fue la extensión que más tiempo costó desarrollar debido al desconocimiento inicial de los protocolos, de la arquitectura gvSIG y del desarrollo e integración de extensiones en gvSIG.

D.1.1 Introducción

WMTS es un estándar, definido en 2010 por el *Open Geospatial Consortium* (*OGC*), que produce mapas de datos georreferenciados, de forma dinámica a partir de información geográfica. Es el sucesor de WMS, el cual no era lo suficientemente eficiente en situaciones en las que el tiempo de respuesta y de pintado del mapa eran importantes. La mejora radica que mientras en el servicio WMS se realizaba una única petición al servidor indicando la extensión del mapa y sus coordenadas, calculando y renderizando de este modo en el servidor la imagen pedida y devolviéndosela al cliente. En WMTS ya existen unas imágenes prerrenderizadas para ciertos niveles de resolución, evitando de este modo generar una imagen nueva por cada petición. Además este servicio compone el mapa en pequeños tiles o teselas, los cuales son pedidos en paralelo para formar después el mapa reduciendo así el tiempo de descarga.

WMTS aporta un nuevo enfoque, mientras el WMS se centra en la renderización de mapas personalizados siendo una buena solución para datos dinámicos, el WMTS renuncia a la personalización de los mapas con el objetivo de obtener una mayor

escalabilidad, ofreciendo datos prerrenderizados donde la envolvente y las escalas han sido limitadas a un conjunto discreto de tiles que siguen una geometría de malla regular.

D.1.2. Cómo funciona el servicio

El estándar define tres recursos básicos: el documento de metadatos del servicio (capabilities), las teselas o tiles y el documento con la información de un punto sobre una tesela, así como tres operaciones básicas para obtener estos recursos: GetCapabilities, GetTile y GetFeatureInfo (opcional)

GetCapabilities: al igual que en el servicio WMS, está operación frente al servidor devuelve un fichero .xml que describe las capacidades del servicio, esto es, las URL de conexión, las peticiones que soporta y la descripción de las capas que ofrece, en que sistemas de referencia y con qué estilos de visualización.

GetTile: devuelve un fichero gráfico correspondiente a un tile, en el formato que le hayamos solicitado, normalmente .jpg, .gif o .png. El formato de la petición es el siguiente:

?REQUEST=GetTile&SERVICE=WMTS&Layer=*laCapaElegida*&Style=*default*&Format=*image/png*&TileMatrixSet=*EPSG:4326*&TileMatrix=*EPSG:4326:1*&TileRow=*1*&TileCol=*2*

Donde el texto en cursiva serían los parámetros elegidos a partir del fichero de *capabilities*. Se va a pasar a explicar cada parámetro.

Tras la etiqueta “Layer” debe aparecer el nombre de la capa seleccionada. La etiqueta “Style” indica el estilo de la capa elegida, cabe destacar que dentro de un *capabilities* de un servicio suele haber varias capas y cada una tiene unos valores de estilos, formatos de imagen, etc diferentes a las demás. “Format” señala en que formato de imagen se quiere que el servidor devuelva el tile. Cada *TileMatrixSet* contiene una o más matrices de tiles las cuales definen los tiles que están disponibles para el sistema de coordenadas elegido (ver Figura 27), es por ello que rellenando la etiqueta “TileMatrixSet” indica implícitamente que sistema de coordenadas se ha seleccionado. Con “TileMatrix” se informa que nivel de escala dentro de la matriz de tiles se selecciona.

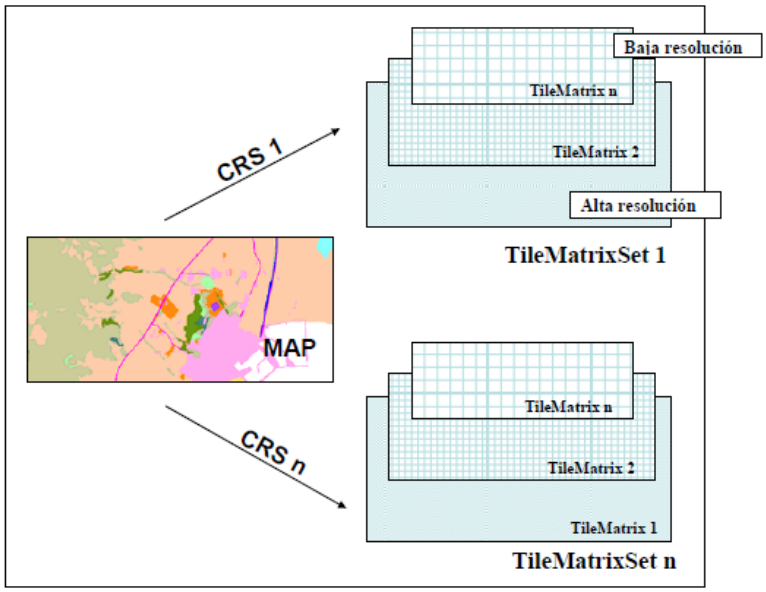

Figura 27 Diferentes TileMatrixSet para diferentes CRS

Por último las etiquetas "TileRow" y "TileCol" indican la posición del tile en número de fila y columna dentro de la extensión total del mapa definido en esa capa (ver Figura 28). Cada tesela de una matriz de teselas se identifica por el índice de columna (TileCol) y de fila (TileRow); estos índices tiene su origen 0,0 en la tesela izquierda y superior de la matriz y se incrementan hacia la derecha y hacia abajo respectivamente.

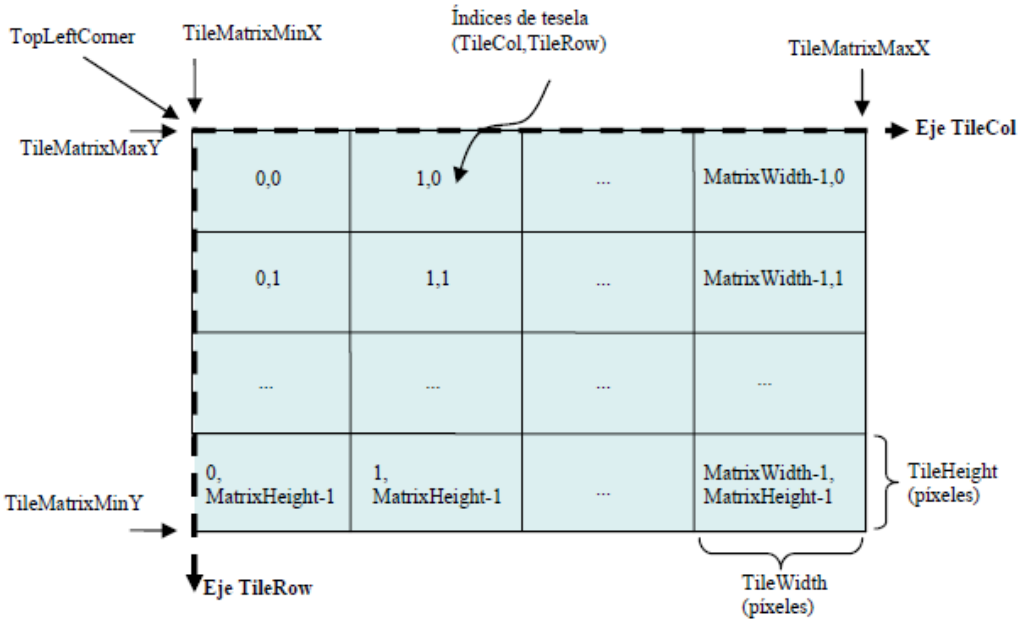

Figura 28 Área del mapa WMTS dividido en tiles

D.1.3. Estudio de la extensión WMS

El diseño de la extensión del cliente WMTS fue inspirado por la ya existente del cliente WMS. Por lo que lo primero que se hizo fue un estudio exhaustivo de la extensión de WMS.

La extensión WMS utiliza una librería llamada libRemoteClients la cual contiene todos los métodos necesarios para conectarse al servicio remoto, descargar el capabilities y parsearlo, devolviendo estructuras de datos que contienen la información del capabilities para su fácil uso.

La extensión WMS además de implementar el servicio de OGC WMS, también implementaba el servicio WMC (*Web Map Context*). Este servicio y sus clases necesarias se obviaron en el estudio de la extensión ya que no iban a ayudar a la implementación de las nuevas extensiones y además se simplificaba de este modo el estudio.

D.1.4. Estructura de la extensión

La extensión se compone principalmente de tres partes. La primera formada por la clase WMTSClientExtension que permite integrar la extensión en gvSIG. La segunda parte la componen una serie de clases encargadas del interfaz gráfico, aquí vienen incluidos los métodos que permiten al usuario conectarse a un servicio y seleccionar los parámetros deseados. Toda esta información obtenida a través del interfaz se pasa a la tercera parte, la encargada de la lógica de pintado que está implementada mayormente en la clase FLyrWMTS.

A continuación se muestra el diagrama de clases resumido de la parte de la lógica de pintado (ver Figura 29). Se pueden ver dos grandes clases, la mayor FLyrWMTS donde está toda la lógica y pintado y FMapWMTSDriver la cual implementa el puente entre la clase FLyrWMTS y la librería que se ha creado para esta extensión, libRemoteServicesExtended la cual se encarga de la conexión con el servicio remoto, proporcionando métodos que permiten invocar los de la librería de una forma transparente para que si en el futuro alguien quiera modificar el funcionamiento de la extensión no tenga que conocer la implementación de la librería.

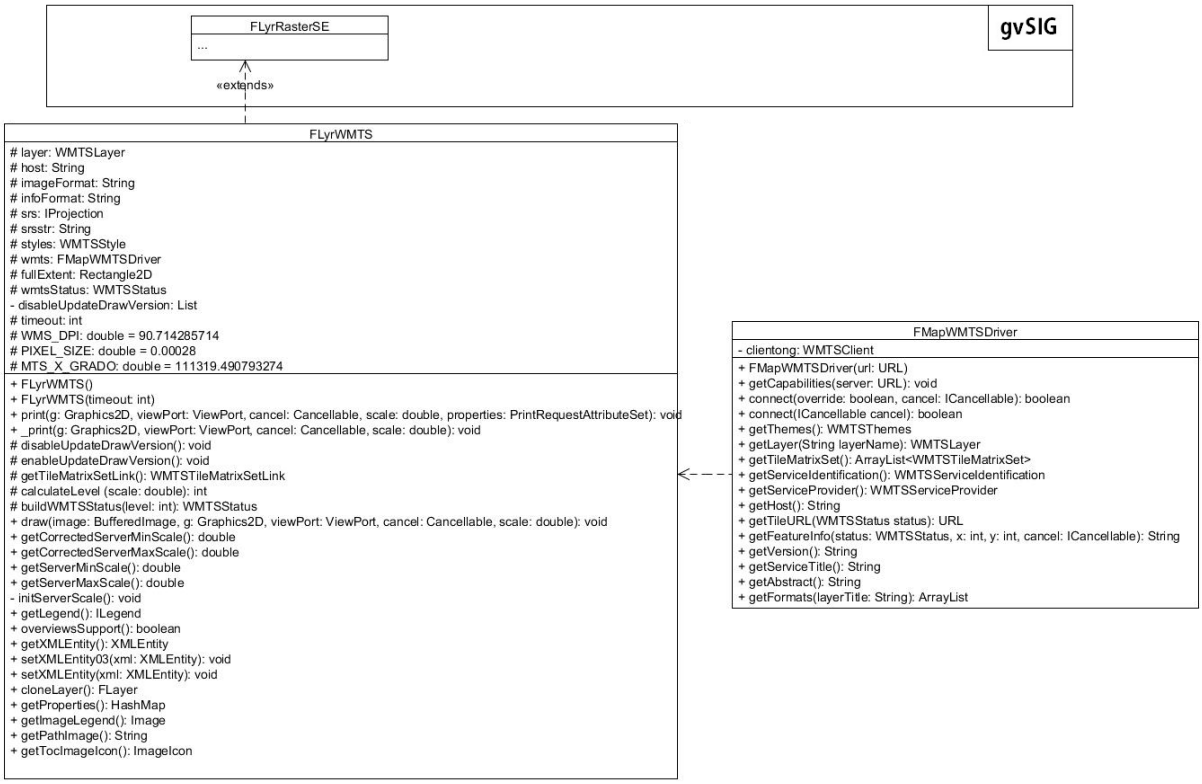

Figura 29 Diagrama de clases simplificado que representa la lógica de la extensión WMTS

D.1.5. La interfaz

La interfaz de esta extensión es similar a la que se puede encontrar en la extensión WMS, habiendo una conexión en el diseño facilitando así el uso al usuario. Se explica con las capturas de las diferentes pantallas por las que se pasa para seleccionar los diferentes parámetros de la extensión.

Al igual que para cargar una capa WMS, hay que pulsar sobre el botón de añadir capa abriéndose una pequeña ventana con diferentes pestañas con los diferentes medios a través de los que se puede cargar una capa. De este modo se ha añadido la pestaña WMTS como se puede observar en la Figura 30. Pulsando sobre la pestaña se muestra un campo de texto en el que se permite escribir el servidor al que se quiere acceder. Pulsando el botón Conectar se establece la conexión y en el recuadro grande en blanco apareceré entonces la descripción del servicio si los creadores del servicio han redactado alguna. Una vez conectado se puede pulsar el botón de Siguiente para avanzar en las ventanas y poder ir seleccionando los parámetros del servicio.

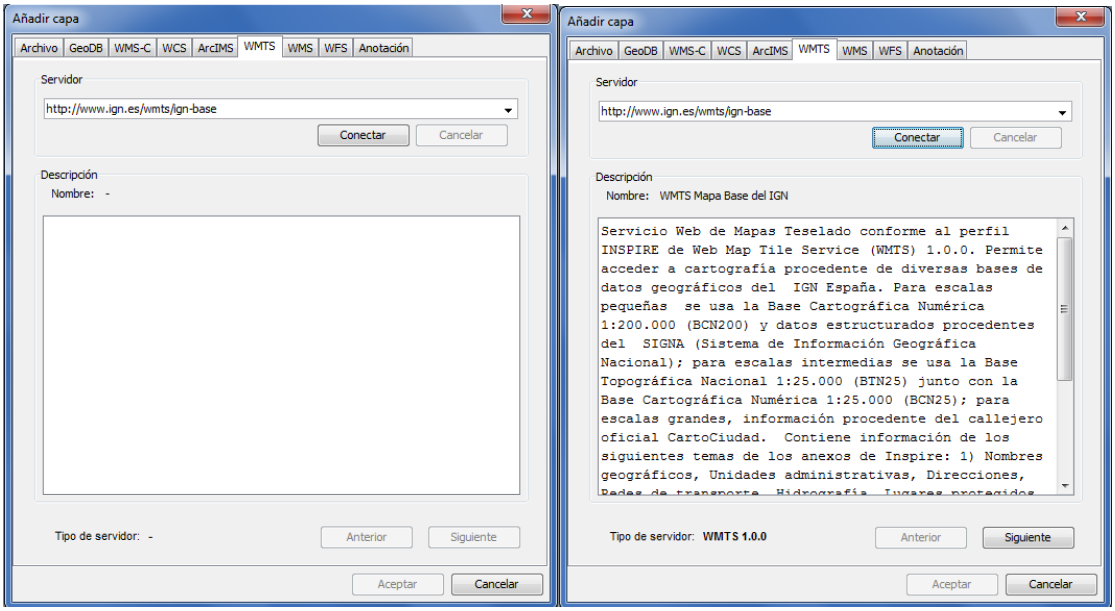

Figura 30 Pestaña de conexión a un servicio WMTS

Tras pulsar el botón siguiente aparecen una serie de pestañas nuevas. La primera Información, muestra la información del servicio (ver Figura 31).

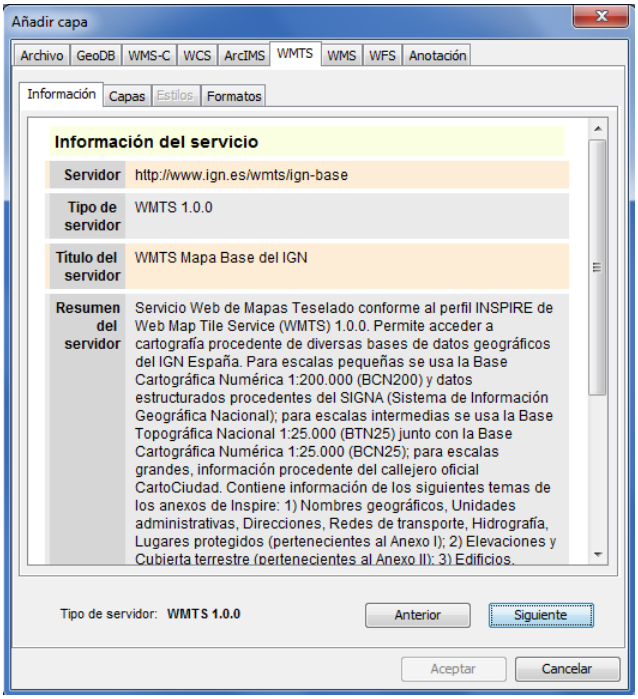

Figura 31 Pantalla que muestra la información del servicio WMTS

La segunda pestaña llamada Capas (ver Figura 32) muestra las capas disponibles en el servicio, permitiendo seleccionar la deseada con doble click sobre el nombre o pulsando el botón Añadir. Si se quiere cambiar la capa seleccionada basta con pulsar el botón Quitar y volver a coger una nueva.

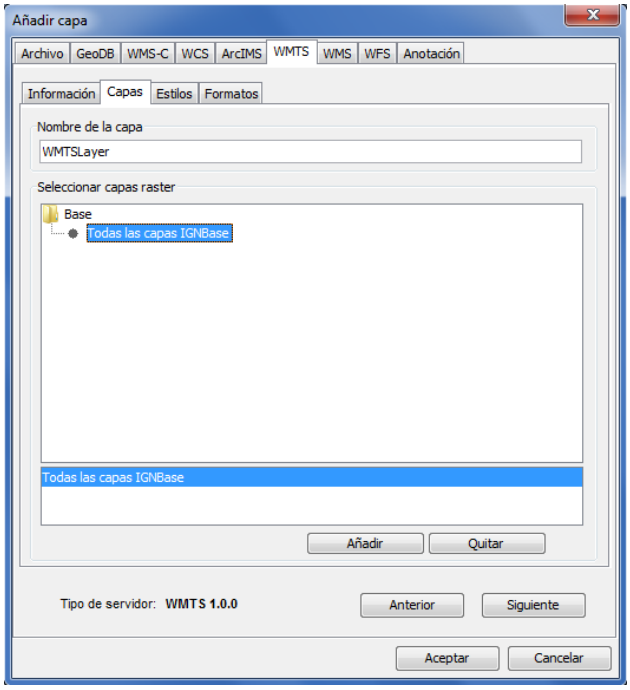

Figura 32 Pantalla que muestra las capas disponibles en el servicio

En la pestaña Estilos (ver Figura 33), aparecen los estilos disponibles para la capa seleccionada, en la mayoría de las ocasiones no suele haber estilos definidos y la pestaña aparece bloqueada o suele estar definido únicamente el estilo por defecto.

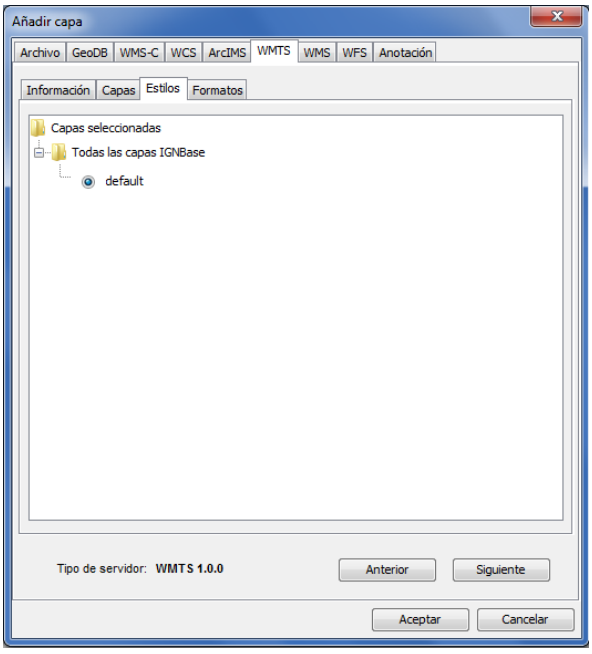

Figura 33 Pantalla que muestra los Estilos disponibles para una capa

Por último, la pestaña Formatos (ver Figura 34). En esta ocasión hay que elegir el formato de imagen, el SRS y el formato de texto. A diferencia de lo que sucederá en

WMS-C, no hay que seleccionar antes el SRS que el formato de imagen, ya que el servicio asegura que para todo SRS existen todos los formatos de imagen.

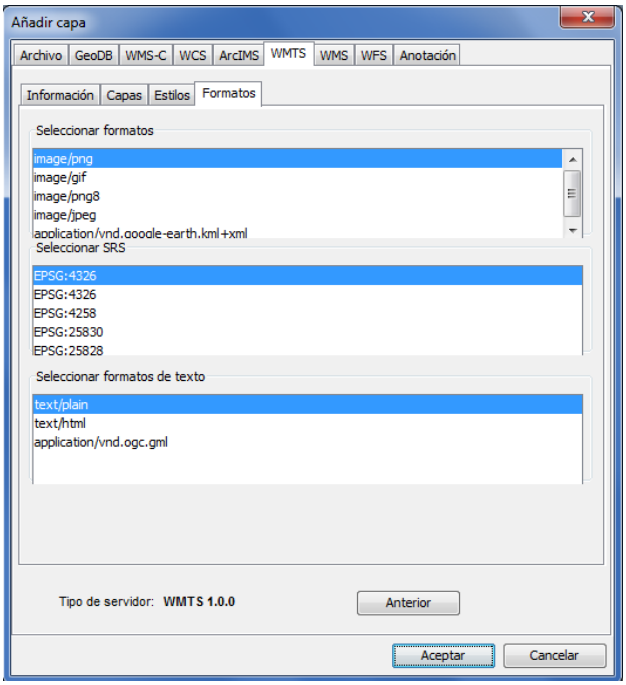

Figura 34 Pantalla para la selección de los formatos

D.1.6. La clase FlyrWMTS

Como en la extensión WMS la cual tenía la clase FlyrWMS encargada de calcular la extensión del mapa a pedir, realizar la petición y después pintarlo. En la nueva extensión se ha querido hacer algo parecido dando lugar a esta clase FlyrWMTS la cual es la encargada de realizar los cálculos oportunos para conocer qué tiles hay que descargar, descargarlos y pintarlos en su posición correcta.

Esta clase obtiene los datos que el usuario ha seleccionado a través del interfaz de usuario (capa, SRS, estilo, formato de imagen, formato de texto) de la clase WMTSParamsPanel.

gvSIG una vez se han seleccionado las opciones del servicio a través del interfaz y se le ha dado a aceptar, realiza una serie de llamadas a funciones. Una de las cuales es draw de esta clase FlyrWMTS. A partir de aquí se va a explicar cómo funciona esta función.

Esta función draw extiende a la función con el mismo nombre de la clase FLyrRasterSE y es la encargada de pintar la capa seleccionada. En este caso lo que se comprueba en la función primeramente es si la extensión del mapa a pintar intersecta la

extensión geográfica que muestra la pantalla, ya que si no es así no hay que molestarse en realizar ninguna petición al servidor porque por pantalla no aparecerá nada de todos modos.

Después se calcula el nivel de la *tileMatrix* a partir de la escala en la que se encuentra el visor en ese momento. WMTS al tener los tiles ya renderizados estos tienen ya unos niveles fijos de escala de los cuales habrá que escoger el más adecuado para la escala en la que se esté trabajando. La estrategia seguida, la cual se explicará más detalladamente en el siguiente apartado "decisiones tomadas", en este caso ha sido escoger la escala de la *tileMatrix* inmediatamente superior a la escala del visor, lo cual hará que más tarde al pintar el tile por pantalla haya que realizar una transformación de escala para adecuarle el tamaño.

Seguidamente se calcula el tamaño del tile (ancho y alto) en el sistema de medida establecido por el srs, pudiendo ser en metros o grados. Para ello se comprueba si es un sistema proyectado, entonces está en metros y las operaciones realizadas son:

```
widthMtsTile = (tileMatrix.getScaleDenominator() * tileMatrix.getTileWidth() * PIXEL_SIZE);

heightMtsTile = (tileMatrix.getScaleDenominator() * tileMatrix.getTileHeight() * PIXEL_SIZE);
```

En el caso de que no sea proyectado, serán grados y las operaciones realizadas:

```
widthMtsTile = (tileMatrix.getScaleDenominator() * tileMatrix.getTileWidth() * PIXEL_SIZE) / (MTS_X_GRADO);

heightMtsTile = (tileMatrix.getScaleDenominator() * tileMatrix.getTileHeight() * PIXEL_SIZE) / (MTS_X_GRADO);
```

Siendo la constante PIXEL_SIZE el tamaño de un píxel de la pantalla, lo cual se toma como estándar 0.00028 metros. Y la constante MTS_X_GRADO el número de metros que caben en un grado, de nuevo se toma como estándar el valor 111319.490793274.

Una vez calculado el ancho y alto del tile, se pasa a calcular el número de filas y columnas y qué números exactamente hay que pedir del *tileMatrix*. Calculando los tiles exactos que hay que pedir se ahorra mucho tiempo ya que no se tendrán que descargar tiles inservibles evitando las esperas que ocasionan. Primero se comprueba si la *tileMatrix* seleccionada tiene *limits* o no. Esto quiere decir que si tiene unos límites de máximos y mínimos de filas y columnas establecidos, por lo que en ese *tileMatrix* no se

encuentra el mapa con toda la extensión geográfica seleccionada, sino que solo una parte. En el caso de que no hay *limits*, se ayudará a la explicación con la Figura 35.

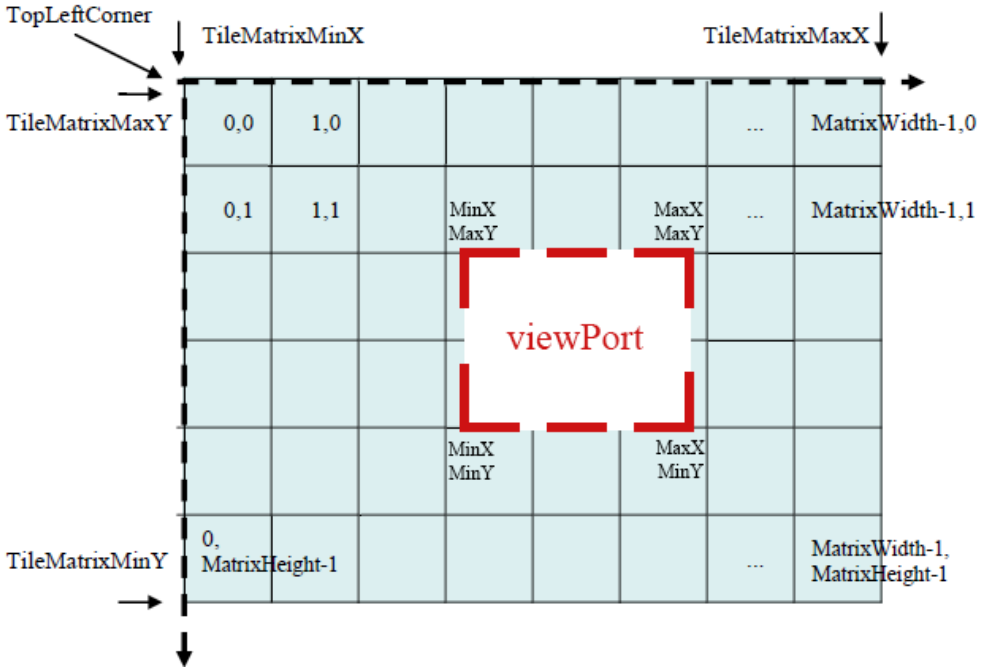

Figura 35 Extensión del viewPort indicando la parte del mapa que aparecerá por pantalla

El recuadro rojo señalado como *viewPort* representa lo que se ve por pantalla en gvSIG, mientras todo el conjunto de los cuadrados azules sería la extensión total del mapa. Cada cuadrado azul representa un tile. La extensión del *viewPort* no coincidirá casi nunca con tiles exactos, es decir que casi siempre mostrará por los laterales una parte del tile únicamente. En el caso de la imagen anterior habría que descargarse los tiles 3,2 4,2 5,2 3,3 4,3 y 5,3 de los cuales el 3,2 3,3 5,2 y 5,3 no se pintarán enteros por pantalla. En realidad para averiguar qué tiles son los que corresponden a la extensión del *viewPort* lo que se ha hecho es calcular la columna mínima (en este caso sería 3), la columna máxima (5), la fila mínima (2) y la fila máxima (3).

Para calcular la columna mínima: se resta la coordenada mínima de X del *viewPort* a la coordenada mínima de X del mapa. Así se conoce la distancia que existe por la izquierda del *viewPort* en el mapa. A este valor se le divide por el ancho del tile en unidades del mapa, obteniendo de este modo el número de tiles.

Para calcular la columna máxima: se resta la coordenada máxima de X del *viewPort* a la coordenada mínima de X del mapa. Y para pasar el valor obtenido en

unidades de mapas a números de tiles se divide entre el ancho del tile en unidades del mapa.

Para calcular la fila mínima: se resta la coordenada máxima Y del *viewPort* a la coordenada máxima Y del mapa. Así se conoce la distancia que existe por encima del *viewPort* en el mapa. De nuevo a este valor se le divide por el alto del tile en unidades del mapa, obteniendo de este modo el número de tiles.

Para calcular la fila máxima: se resta la coordenada mínima de Y del *viewPort* a la coordenada máxima de Y del mapa. Y para pasar el valor obtenido en unidades de mapas a números de tiles se divide entre el alto del tile en unidades del mapa.

Teniendo ya calculadas a partir de que filas y columnas hay que pedir y hasta cuales, se realiza un bucle donde se va rellenando la petición que se enviará al servidor para pedir el tile. También se calcula el desplazamiento en el eje X y en el eje Y que hay que aplicar a cada tile para indicar su sitio espacial correctamente. Esto se realiza conociendo la posición de la esquina superior izquierda del mapa, se suma el ancho del tile antes calculado multiplicado por el número de tiles que tiene a su izquierda para conocer el desplazamiento en X, y para el desplazamiento en Y se suma el alto del tile multiplicado por el número de filas de tiles que tiene por encima.

Para la obtención de los tiles se han utilizado hilos de ejecución para descargarlos en paralelo y reducir el tiempo de pintado. Esto se ha implementado con una clase que se ha llamado threadDescargaTile que implementa la clase Callable. Se ha optado por implementar la clase Callable en vez de la clase Thread debido a que se necesitaba que el hilo devolviera el fichero de imagen del tile para que una vez descargado todos, en la función draw se puedan pintar por pantalla. Es decir, el hilo de ejecución lo que hace es primero, descargar el tile utilizando la URL adecuada, redimensionarlo en memoria para adaptarlo a la escala actual en la que está gvSIG y devolverlo redimensionado junto a las coordenadas donde debe ser pintado. El pintado no se realiza en el hilo ya que el pintado en Java suele dar problemas si se realiza en paralelo y es por ello que se realiza después de que todos los hilos hayan acabado y devuelto su tile.

Con las descargas de los tiles aparece un problema. Qué ocurre si el servicio externo se bloquea y no devuelve el tile, la espera del tile nunca acabaría y no se

pintaría el mapa. Para solventarlo se ha puesto un *timeout* de 15 segundos, en los cuales si no se han descargado todos los tiles, la aplicación muestra un error indicando que se ha excedido el tiempo de pintado y se deshabilita la capa que se iba a pintar en gvSIG.

D.1.7. Decisiones tomadas en el diseño

Durante el diseño de la nueva extensión se tuvieron que evaluar diferentes soluciones a varios problemas y tomar la mejor. A continuación se expondrán las más relevantes.

Reescalar los tiles frente a permitir escalas fijas del mapa. Al tener cada WMTS unos niveles de escala ya definidos, los cuales es casi imposible que correspondan al nivel de escala sobre el que se esté trabajando en ese momento en gvSIG, se presentan las dos posibles soluciones mencionadas, descargar los tiles y reescalarlos linealmente para adecuarlos a la escala o únicamente permitir las mismas escalas en gvSIG que en el servicio WMTS. Esta segunda opción sería más eficiente en tiempo ya que no habría que aplicar deformaciones a las imágenes pero únicamente permitiría cargar únicamente una capa WMTS a la vez ya que cada servicio WMTS tiene niveles de escala diferentes definidos por su propietario. Además para fijar niveles de escalas fijas habría que modificar el núcleo de procesado de gvSIG, perdiendo de este modo la posibilidad de que fuera una extensión de gvSIG donde todo el core de la extensión no modifica la aplicación sino que únicamente se añade funcionalidad con la ventaja de poder instalarla y desinstalarla a merced del usuario. La solución adoptada fue reescalar los tiles porque va más acorde con la funcionalidad de gvSIG, además de que se puede de este modo mezclar capas de diferentes servicios WMTS.

Pedir al servicio WMTS los tiles del nivel de escala más cercano al nivel actual y que sea superior frente a que sea inferior. Si se piden los tiles con el nivel inmediatamente superior lo que ocurre es que hay más tiles que pedir al servicio, por lo que el pintado costará mayor tiempo, pero como se decidió reescalar los tiles, el nivel de detalle y resolución de las imágenes será mucho mayor que pidiendo el nivel inferior, el cual irá más rápido pero la imagen se pixelaría mucho, perdiendo la precisión que se requiere en este tipo de herramienta. También se podrían haber tomado estrategias mixtas, donde se elegiría el nivel más cercano por redondeo o por estrategias 15% eligiendo el nivel inferior 85% el superior. Tras probar todas las estrategias y viendo los

resultado se prefirió elegir siempre la escala superior para proporcionar la mejor calidad de imagen y precisión.

Calcular los tiles exactos que hay que pedir frente a ir comprobando tile por tile todos los disponibles si hay que pintarlos por pantalla. En este caso la segunda opción no tiene ventaja alguna sobre el primero ya que su tiempo de ejecución se va incrementando exponencialmente conforme se va bajando de nivel ya que se llegan a números estratosféricos de tiles. Por lo que calculando los tiles que van a aparecer por pantalla previamente se consigue que independientemente del nivel de escala el mapa cargue con la misma velocidad.

D. 2. Extensión del cliente WMS-C

Una vez terminada la extensión para la conexión a servicios WMTS se pasó al diseño y desarrollo de la extensión para soportar los servicios WMS-C. Gracias a todos los conocimientos obtenidos en la realización de la extensión anterior, permitieron agilizar en gran medida el proceso y ser mucho más eficientes en la creación de esta extensión.

D.2.1. Introducción

WMS-C (*Web Map Service–Cache*) es una adaptación del WMS editada por OSGeo, permitiendo de este modo que el servicio pueda pedir tiles añadiendo la descripción del conjunto de matrices de teselas a los metadatos del servicio WMS.

D.2.2. Cómo funciona el servicio

Al ser un servicio que reutiliza los recursos que proporciona WMS existen las tres mismas operaciones básicas para obtener recursos que este servicio: GetCapabilities, GetMap y GetFeatureInfo (opcional). Las dos importantes GetCapabilities para obtener el documento de metadatos del servicio y el GetMap utilizado para descargar los ficheros de imagen del mapa.

Al estar basado en WMS utiliza la operación GetMap como se ha mencionado, pero en este caso no descarga todo el mapa sino una tesela, por lo que habrá que realizar varias llamadas GetMap para conseguir al final la extensión del mapa deseado. El formato de la petición es el siguiente:

?REQUEST=*GetMap*&LAYERS=*laCapaElegida*&STYLES=*default*&SRS=*EPSG:23030*&BBOX=*665100.0515538651,4610900.077330798,671900.0773307976,4617719.103107731*&WIDTH=*256*&HEIGHT=*256*&FORMAT=*image/jpeg*

Donde el texto en cursiva serían los parámetros elegidos a partir del fichero de capabilities. Cada parámetro se debe terminar con '&' para separarlo del siguiente. Se va a pasar a explicar cada parámetro. Tras la etiqueta "Layers" debe aparecer el nombre de la capa seleccionada. La etiqueta "Styles" indica el estilo de la capa elegida, cabe destacar que dentro de un capabilities de un servicio suele haber varias capas y cada una tiene unos valores de estilos, formatos de imagen, etc diferentes a las demás. "SRS" (*Spatial Reference System*) es el sistema de referencia en que se solicita la capa. Cada servicio de mapas ofrece sus capas en distintos sistemas de referencia. Como los demás detalles, se pueden ver en el fichero de capacidades del servicio. Con "BBOX" se indican los valores de las coordenadas de las dos esquinas que definen el rectángulo que se solicita, expresados en el sistema de referencia que se ha indicado anteriormente en el parámetro SRS. En el caso del servicio WMS había que señalar la extensión del mapa a pedir, pero como en este caso hay que ir pidiéndolo por teselas el bbox tiene que corresponder exactamente con la extensión de la tesela prerrenderizada que tiene el servicio. "Width" y "Height" sirven para señalar el tamaño en píxeles de la imagen devuelta. "Format" señala el formato de imagen que se quiere solicitar.

D.2.3. Estructura de la extensión

La estructura de esta extensión es similar a la utilizada en el cliente WMTS, desarrollado previamente, ya que son servicios parecidos. De nuevo la extensión se compone principalmente de tres partes. La parte que permite la integración con gvSIG que está formada por la clase WMSCClientExtension. La parte encargada del interfaz gráfico, aquí vienen incluidos los clases que permiten al usuario conectarse a un servicio y seleccionar los parámetros deseados. Y la última parte encargada de la lógica de pintado implementada mayormente en la clase FLyrWMSC.

El diagrama de clases resumido de la parte de la lógica de pintado (ver Figura 36), es similar a la de la extensión anterior, principalmente cambian las funciones ya que son dos lógicas de pintado diferentes. Las dos grandes clases representadas son, FLyrWMSC que contiene la lógica y realiza el pintado del mapa y FMapWMSCDriver la cual implementa el puente entre la clase FLyrWMSC y la librería

libRemoteServicesExtended la cual se encarga de la conexión con el servicio remoto, proporcionando métodos que permitan invocar los de la librería de una forma transparente para que si en el futuro alguien quiera modificar el funcionamiento de la extensión no se tenga que empapar de la implementación de la librería.

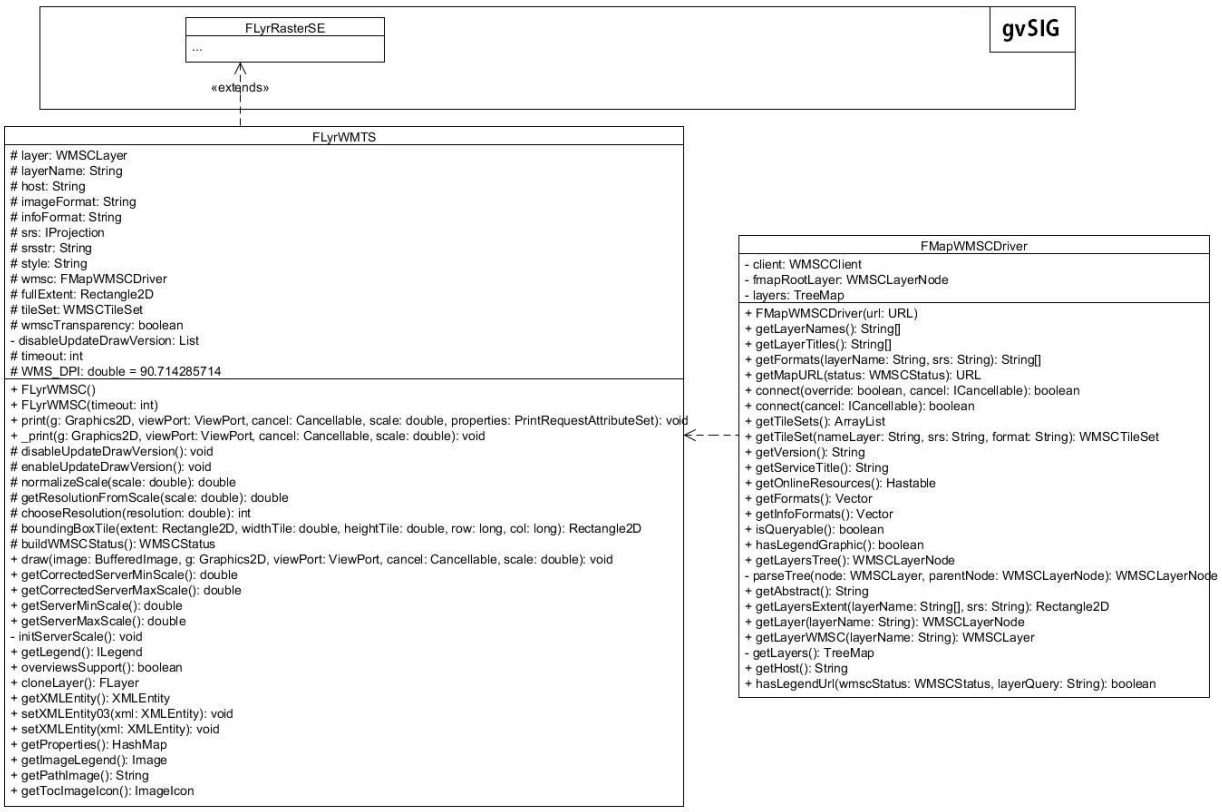

Figura 36 Diagrama de clases simplificado que representa la lógica de la extensión WMS-C

D.2.4. La interfaz

La interfaz de esta extensión es similar a la que se puede encontrar en la extensión WMS y en la nueva WMTS. Como en la anterior extensión se va a explicar a través de las capturas de las diferentes pantallas por las que se pasa para obtener los parámetros.

Al añadir capa, pulsando la pestaña WMS-C de la ventana emergente (ver Figura 37), aparece el menú para poder escribir la dirección del servicio al que se quiere conectar.

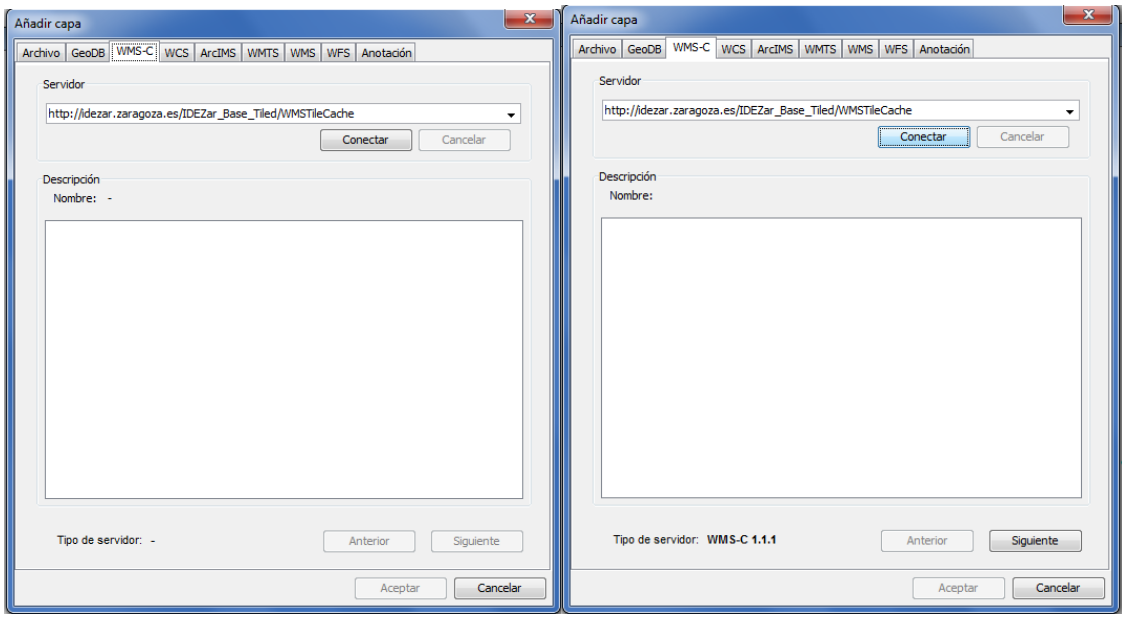

Figura 37 Pestaña de conexión a un servicio WMS-C

Una vez conectado al servicio, las pestañas que aparecen al pulsar el botón siguiente son las mismas que en el servicio WMTS. La primera la de Información (ver Figura 38), muestra la descripción del servicio y los valores que se van a ir seleccionando del servicio a través del interfaz.

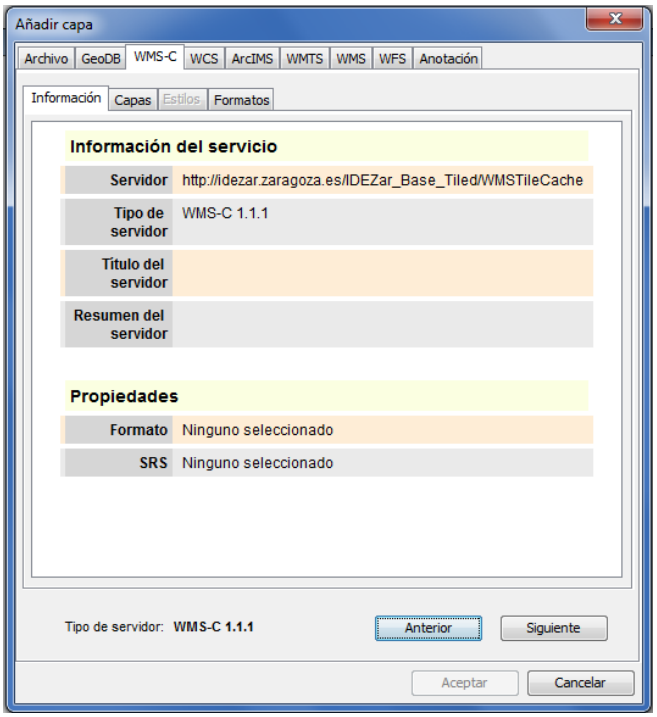

Figura 38 Pantalla que muestra la información del servicio WMS-C

En la pestaña Capas (ver Figura 39), se vuelven a mostrar las capas disponibles en el servicio para poder seleccionar la deseada.

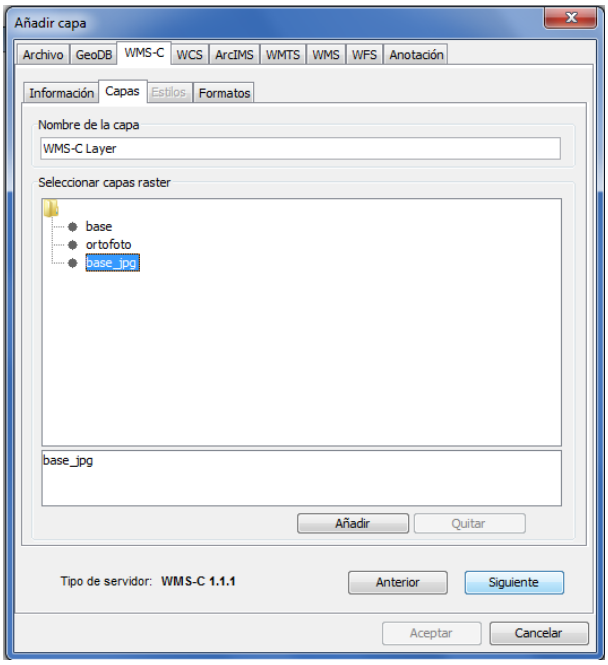

Figura 39 Pantalla que muestra las capas disponibles en el servicio

La pestaña Estilos solo está activa si la capa seleccionada tiene algún estilo definido, lo cual no suele suceder.

Por último la pestaña Formatos (ver Figura 40) permite seleccionar el SRS y el formato de imagen de la capa. Por cómo se define el servicio WMS-C en este caso hay que seleccionar primero el SRS del servicio y después aparecerán los formatos de imagen disponibles ya que puede que algunos formatos solo estén para algunos SRS al contrario de lo que sucede en WMS y WMTS.

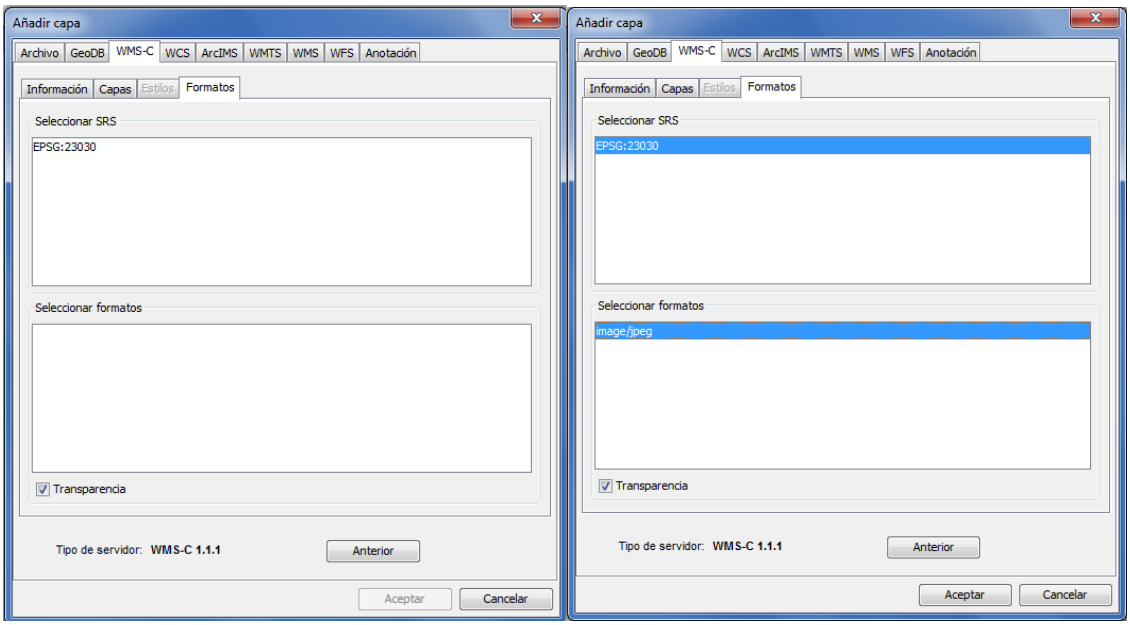

Figura 40 Pantalla para la selección de los formatos

D.2.5. La clase FlyrWMSC

Como se ha mencionado anteriormente, la estructura de esta extensión es similar a la extensión desarrollada WMTS, por lo que de nuevo existe una clase encargada de realizar los cálculos oportunos para conocer qué tiles hay que descargar, descargarlos y pintarlos en su posición correcta llamada en este caso FlyrWMSC.

Esta clase obtiene los datos que el usuario ha seleccionado a través del interfaz de usuario (capa, SRS, formato de imagen, estilo) de la clase WMSCParamsPanel.

gvSIG una vez se han seleccionado las opciones del servicio a través del interfaz y se le ha dado a aceptar, realiza una serie de llamadas a funciones. Una de las cuales es draw de esta clase FlyrWMSC. A partir de aquí se va a explicar cómo funciona esta función.

Esta función draw extiende a la función con el mismo nombre de la clase FLyrRasterSE y es la encargada de pintar la capa seleccionada. WMS-C proporciona diferentes resoluciones de imágenes, por lo que primero que se ha de hacer es a partir de la escala actual en la que se encuentra la vista en gvSIG se debe calcular a que resolución de imagen corresponde. Una vez ya se sabe la resolución que se va a pedir se obtiene el tamaño, ancho y alto, en píxeles de la tesela.

A continuación se ha planteado resolver el problema de que teselas pedir como en la extensión WMTS, se ha supuesto que el mapa está dividido en diversas filas y columnas (ver Figura 41). De este modo se calcula a partir de qué números de fila y columna y hasta cuáles hay que pedir. En este caso se empiezan a contar las teselas desde la esquina inferior izquierda y aumentando conforme se va subiendo y yendo hacia la derecha. Acotando la extensión total del mapa con los valores de min y max X e Y como se muestra en la siguiente imagen y acotando el trozo de mapa que se mostraría por pantalla que es representada por el recuadro rojo etiquetado como *viewPort* se calcularían las filas y las columnas de la siguiente forma.

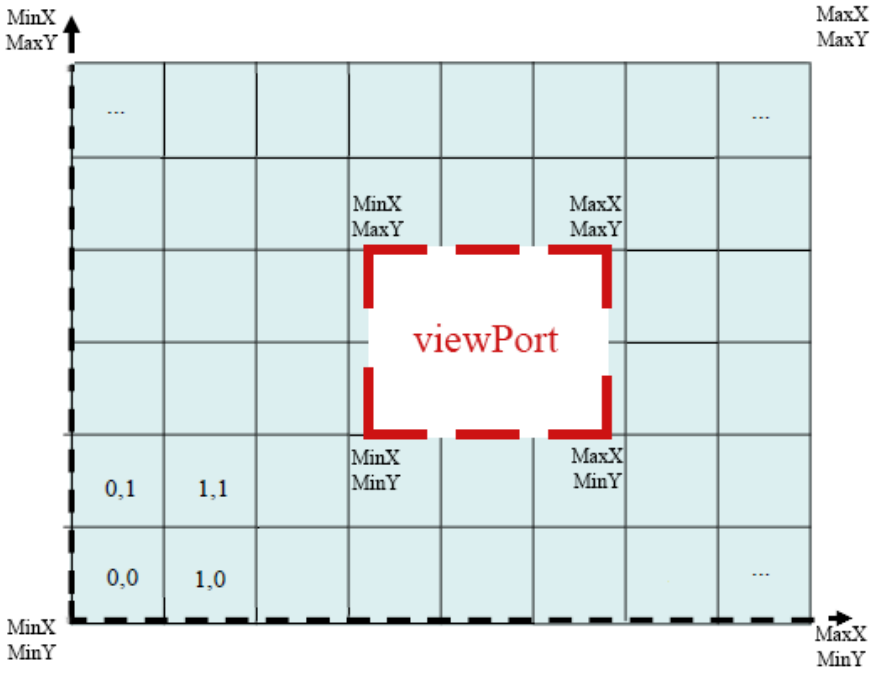

Figura 41 Extensión del viewPort indicando la parte del mapa que aparecerá por pantalla

La fila mínima: se resta la coordenada mínima en Y del *viewPort* a la coordenada mínima en Y del mapa. Y a este valor se le divide entre la altura en píxeles de la tesela calculada anteriormente.

La fila máxima: se resta la coordenada máxima en Y del *viewPort* a la coordenada mínima en Y del mapa. Y de nuevo se divide entre la altura de la tesela.

La columna mínima: se resta la coordenada mínima en X del *viewPort* a la coordenada mínima en X del mapa y se divide entre el ancho de la tesela en píxeles.

La columna máxima: se resta la coordenada máxima en X del *viewPort* a la coordenada mínima en X del mapa y una vez más se divide entre el ancho de la tesela.

Tras conocer las filas y las columnas se comienza el bucle para pedir los tiles. Antes de pedirlos en este caso hay que calcular el *bounding box* o extensión de cada uno ya que en este servicio hay que pedirlos como se ha mostrado anteriormente aprovechando la llamada getMap de WMS y no existen los parámetros ROW ni COL del getTile de WMTS. El *bounding box* indica las coordenadas de las cuatro esquinas del mapa o del tile en este caso y se ha calculado de la siguiente manera:

La esquina inferior izquierda en X es la suma de la coordenada mínima en X del mapa sumada con el producto del número de columna que le corresponde al tile por el ancho del tile.

La esquina inferior izquierda en Y es la suma de la coordenada mínima en Y del mapa sumada con el producto del número de fila que le corresponde al tile por la altura del tile.

Para obtener las otras dos esquinas únicamente hay que sumarles a los dos valores anteriores el ancho y el alto del tile respectivamente.

Una vez se tienen todos los parámetros necesarios ya para realizar las peticiones de los tiles, se sigue el mismo procedimiento explicado en WMTS para la descarga con hilos de ejecución para descargar los tiles en paralelo.

D. 3. Librería RemoteServicesExtended

En gvSIG 1.12 existe una librería llamada libRemoteServices que contiene todos los procedimientos necesarios para conectarse a los servicios remotos WCS, WFS y WMS así como para intercambiar datos con ellos. Para seguir la misma estructura en las extensiones de WMS-C y WMTS se creó la librería libRemoteServicesExtended. Esta decisión se tomó, frente a modificar la ya existente añadiendo las nuevas funcionalidades, debido a que no se quería modificar el gvSIG existente sino ampliar la funcionalidad. Si en el futuro alguien de la comunidad de gvSIG decide modificar la librería libRemoteServices, y el usuario que esté utilizando la nueva librería extendida para las extensiones WMS-C y WMTS decide actualizar la librería que tenía por la nueva modificación, reemplazaría la librería extendida por la original sin los métodos para las dos nuevas extensiones provocando que dejasen de funcionar.

La nueva librería principalmente contiene la lógica de parseo de los capabilities de las extensiones WMS-C y WMTS, además de proporcionar estructuras de datos para poder almacenar la información obtenida de ellos y pasársela a la extensión correspondiente.

D. 4. Extensión para la carga del Overview Map

Después de haber implementado los dos clientes de servicios de mapas teselados se pasó a la parte más gráfica de la implementación, donde se iban a diseñar y crear unas herramientas que facilitaran al usuario la edición y gestión de datos georreferenciados. Para entrar en materia con el aspecto más gráfico de gvSIG se comenzó con una extensión más sencilla, un *Overview Map*.

Un *Overview Map* es básicamente mostrar en miniatura la extensión completa de la información geográfica cargada en ese momento en el programa. Esta herramienta permitirá al usuario navegar rápidamente por el mapa, saltando de un punto a otro deseado a través de la miniatura.

D.4.1. Estructura de la extensión

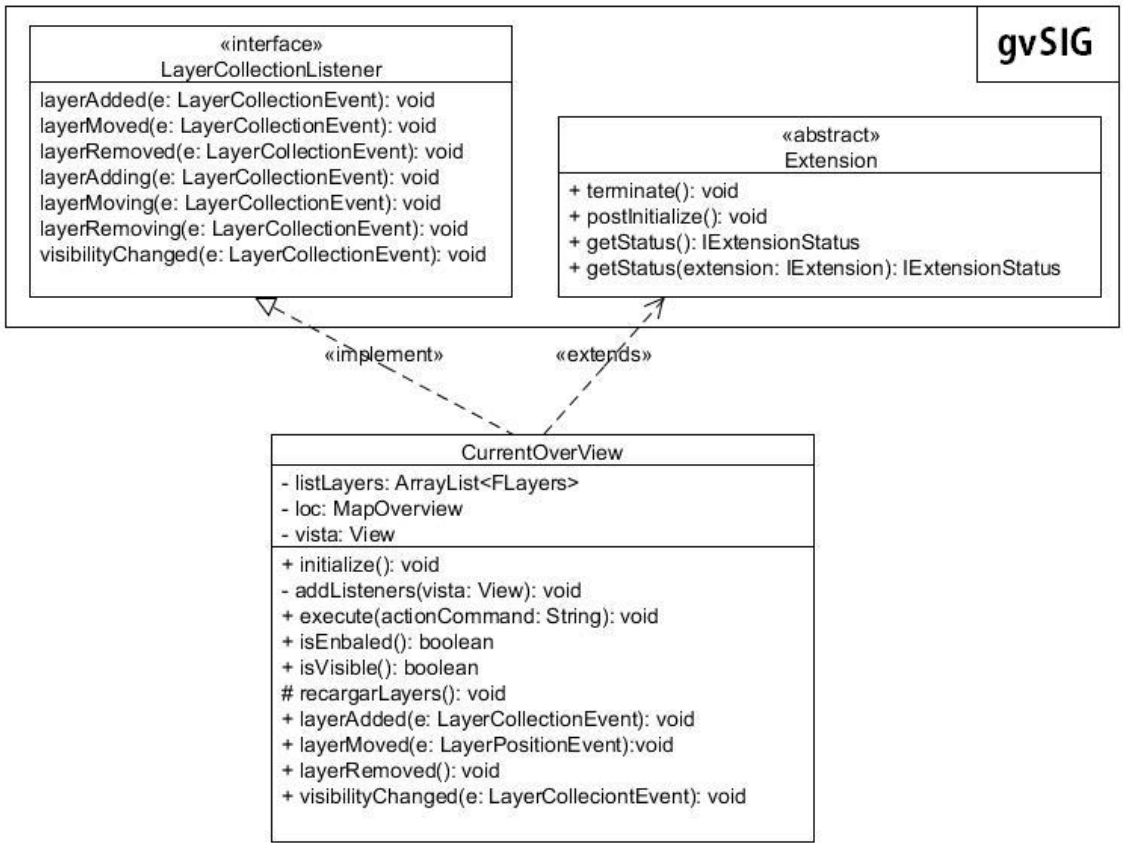

Figura 42 Diagrama de clases simplificado de la extensión Overview Map

Esta extensión está compuesta únicamente por una clase, CurrentOverView (ver Figura 42). Esta clase extiende la clase abstracta que aporta gvSIG Extension, para indicar que se trata de una extensión. Principalmente esta clase comprueba todas las

capas que están en el preciso momento en el que se ejecuta la extensión y pinta en miniatura la capa ráster que encuentra cargada en la última posición de la jerarquía.

La clase implementa además el interfaz LayerCollectionListener, el cual permite asociar listeners a las capas para conocer cuando se añade, elimina o mueve una capa y así poder comprobar en cada uno de los casos cuál de ellas es la capa ráster que se debe mostrar.

D.4.2. Diseño de la extensión

El primer diseño de esta extensión mostraba todas las capas cargadas en ese momento en la vista, pero se consideró poco eficiente ya que al ser de dimensiones reducidas, cuantas más capas se mostrasen menos información se apreciaba, perdiendo de ese modo su utilidad. La decisión tomada para solucionarlo fue simplemente mostrar la capa ráster cargada en el nivel más bajo, porque es la que contendrá más información ya que se suele poner abajo la capa ráster sobre la que se irán cuadrando las demás capas vectoriales.

D. 5. Extensión para la edición y gestión de datos vectoriales y su simbología

Seguidamente de implementar la extensión del *Overview Map*, teniendo un mayor conocimiento del desarrollo gráfico de gvSIG, se comenzó con el diseño y desarrollo de una de las extensiones claves del nuevo sistema para la edición y gestión de datos vectoriales y su simbología. La extensión consistió en una barra de edición colocada en el lateral derecho de gvSIG.

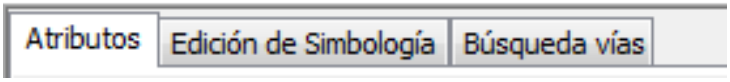

Figura 43 Pestañas de la herramienta de gestión de la información

Esta barra está formada por tres pestañas (ver Figura 43) para diferentes funcionalidades. La primera “Atributos” permite la edición rápida de las *features* de una capa vectorial. La segunda pestaña, “Edición de simbología” permite editar fácilmente la simbología asociada a una capa, es decir asociar colores, formas o imágenes a las

features de una capa. Por último la pestaña de "Búsqueda vías" permite buscar una dirección y devuelve una serie de resultados posibles, sobre los que poder pinchar centrando así el mapa sobre la dirección indicada.

Esta nueva herramienta facilita en gran medida realizar todas estas funciones de edición que no eran triviales ni sencillas de realizar en la distribución de gvSIG.

D.5.1. Estructura de la extensión

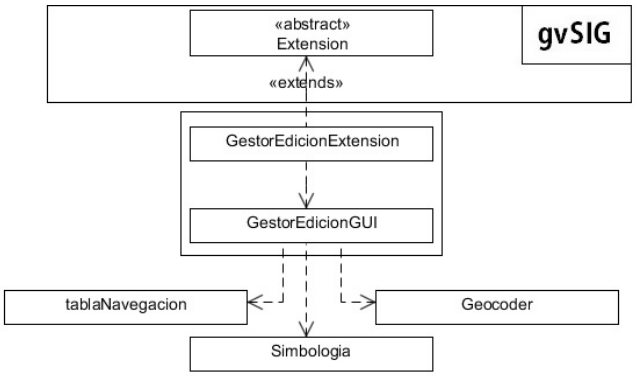

Figura 44 Diagrama de la estructura simplificada de la extensión

Como se observa en la Figura 44, la extensión está estructurada en cuatro partes. La principal que contiene la conexión de la extensión con gvSIG y gestiona el interfaz principal de la extensión la cual utiliza las otras tres partes, correspondientes a las tres pestañas ofrecidas "Atributos" (tablaNavegacion), "Edición de simbología" (Simbologia) y "Búsqueda vías" (Geocoder).

D.5.2. Interfaces

La interfaz de esta extensión ha variado mucho en el tiempo. Desde el principio se buscó una manera de hacerla lo más cómoda y sencilla para el usuario. En un inicio se concibió como una ventana emergente a la derecha de la vista del mapa. Esta solución fue la primera aproximación, ya que lo que se quería era una ventana anclada constante que no molestase al usuario durante la edición o simple uso del programa. Por lo que una ventana no era la mejor opción ya que cuando se empiezan a abrir ventanas que se pueden desplazar por pantalla, la interfaz acaba siendo un caos entorpeciendo el funcionamiento. De este modo se empezó a investigar otra manera para poder acoplar de algún modo la ventana a la vista de gvSIG de un modo similar a como estaba repartida la vista, con el TOC (*Table of Contents* o Tabla de contenidos) a la izquierda y el mapa a la derecha. No se podía hacer lo mismo e insertar ahí el nuevo panel de

edición ya que la vista está definida en el núcleo de gvSIG y para ello habría que modificarlo, perdiendo de ese modo el concepto plugin de aumentar funcionalidad.

La siguiente solución fue insertar la barra como un componente de interfaz de Java en el MapControl situándolo dentro de él centrado a la izquierda. De este modo se obtuvo la solución para el problema de la ventana, se tenía una serie de pestañas ancladas que se mostraban si se pulsaba sobre el botón de la extensión y se ocultaban pulsando de nuevo sobre el botón. Así la interfaz quedaba limpia y muy usable. El único problema era que al pintar el panel sobre el MapControl, se perdía parte del mapa por debajo, lo cual tampoco sería un gran problema exceptuando por el hecho de que se seguía considerando mapa lo que había por debajo del panel, por lo que por ejemplo al centrar el mapa sobre un punto, el punto no parecía que estuviese centrado ya que parte del mapa quedaba cubierto.

Para solucionar este problema, se siguió investigando, llegando a la conclusión de que también se le podían añadir componentes gráficos de Java a la Vista de gvSIG, así se añadió del mismo modo obteniendo el resultado deseado. Ahora cuando aparece el panel, el mapa se redimensiona mostrándolo a una escala superior y no se oculta parte de él.

D.5.3. Listeners

Los listeners en Java permiten escuchar eventos producidos por alguna clase y actuar al respecto. Para la implementación de esta extensión se implementaron las interfaces proporcionadas por gvSIG para reaccionar frente a eventos sucedidos sobre las capas. Los eventos sobre los que se actúa son los siguientes:

- activationChanged: es decir, cuando se selecciona una capa en el TOC (*Table of Contents* o Tabla de contenidos) se comprueba si la capa pasa a estar activa (seleccionada) y si es una capa vectorial, ya que las pestañas de “Atributos” y “Edición de simbología” solo aparecen activadas cuando hay una capa vectorial activa. Cuando se cambia la selección de una capa a otra, se recargan los datos mostrados en la “Atributos” mostrando los de la capa actualmente seleccionada.

- layerAdded: se ejecuta cuando una capa es añadida a la vista. Lo que se hace frente a este evento es añadirle a la capa añadida, el listener propio de edición, para que se pueda escuchar los eventos futuros de dicha capa.

- layerRemoved: para eliminar una capa está debe estar seleccionada. Al eliminarla se refresca la interfaz bloqueando de nuevo las pestañas de "Atributos" y de "Edición de la simbología" ya que no habrá ninguna capa seleccionada después de la eliminación.

D.5.4. Funcionamiento

A continuación se explican más detalladamente el funcionamiento cada una de las tres pestañas:

Pestaña Atributos

La pestaña "Atributos" muestra las *features* de una capa vectorial, permitiendo navegar sobre ellas y editarlas (ver Figura 45). Los datos de las *features* de la capa, están cargados en memoria en una estructura de datos de gvSIG llamada SelectableDataSource. De este modo se carga dicha estructura en la extensión y se van mostrando los valores de los diferentes campos de cada *feature* por pantalla.

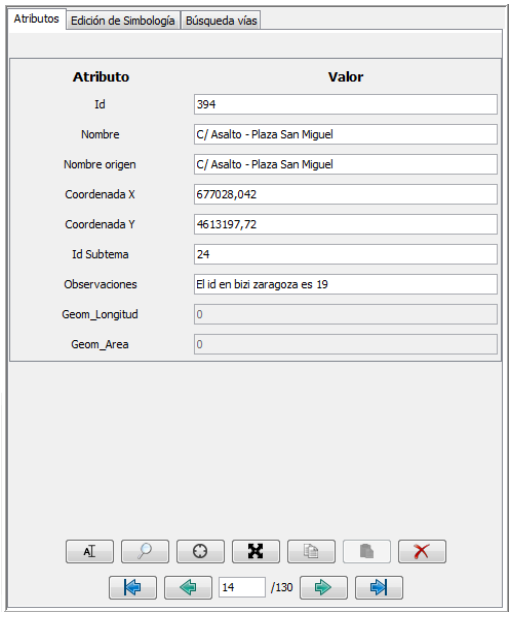

Figura 45 Pestaña de Atributos

Como se aprecia en la captura de la herramienta, en la parte superior se muestran los nombres de los atributos que tiene la capa y a su lado los valores de la *feature* seleccionada en ese momento.

Para poder modificar estos valores se optó por obligar a poner la capa en edición, ya que en un principio se permitía la edición de los valores en cualquier momento, pero se observó que el funcionamiento dejaba de ser intuitivo y se permitía

realizar una funcionalidad fuera de su ámbito, como es modificar datos fuera de la edición. La modificación de los datos se realiza en memoria y es al terminar la edición de la capa cuando los datos de la memoria son volcados sobre la fuente origen de los datos (*shapefile*, base de datos, etc)

En la parte inferior se encuentra el panel de botones, estos botones han sido los mínimos necesarios para poder facilitar en gran medida la edición, evitando botones y funcionalidades inútiles que sobrecargaran el interfaz. A continuación se van a explicar la función de cada botón y por qué se decidieron tales funcionalidades.

Comenzando por la primera fila de botones, el primer botón permite asignar un Alias al nombre del atributo, los ficheros *shapefiles* como las bases de datos solo permiten un número máximo de caracteres para sus atributos, esta funcionalidad permite asignar nombres más intuitivos a los atributos ayudando al usuario a la comprensión de lo que representan los atributos. El segundo botón corresponde a la búsqueda de *features*, permite buscar por el valor de un atributo específico, de este modo se puede acceder a una *feature* deseada rápidamente. El tercer botón permite centrar el mapa sobre la *feature* que está seleccionada en ese momento, manteniendo la escala del mapa actual. El cuarto botón permite centrar el mapa sobre la *features* que está seleccionada en ese momento haciendo zoom sobre ella. El quinto botón, que solo está habilitado cuando la capa está en modo edición, permite copiar los valores actuales de una *feature*. El sexto botón, permite pegar la *feature* copiada con el botón copiar, por lo que solo se habilitará cuando se haya copiado alguna *feature*. El último botón elimina la *feature* seleccionada.

La segunda fila de botones corresponde a los botones para la navegación entre *features*. El primero selecciona la primera *feature* y centra el mapa sobre ella. El segundo botón selecciona la *feature* anterior y centra el mapa sobre ella. El tercer botón hace lo mismo pero sobre la *feature* siguiente. Y el último botón selecciona la última *feature* y de nuevo centra el mapa sobre ella.

Por último se enumeran a continuación las decisiones más importantes de diseño que se llevaron a cabo en esta herramienta:

- La edición de los atributos de una *feature* solo se permite si la capa está en edición.

- El centrado sobre una *feature* solo se produce cuando se navega sobre ellos con las flechas, se deshabilitó esta funcionalidad cuando se cambiaba de una capa a otra porque dejaba de ser útil para el usuario y empezaba a ser engorroso.
- Al importar una capa vectorial, siempre se selecciona la primera *feature*, pero cuando se navega entre capas se recuerda la última *feature* sobre la que se ha estado en cada capa, permitiendo un uso más lineal.
- Al crear una *feature* nueva en una capa, la herramienta muestra el formulario vacío de esta nueva *feature*, permitiendo una edición rápida.
- Al eliminar una *feature*, se selecciona y muestra los valores de la *feature* anterior.

Pestaña de edición de simbología

La edición de la simbología en gvSIG no era trivial, las herramientas estaban dispersas en diferentes ventanas dificultando la tarea al usuario. Es por ello que se decidió agrupar las funcionalidades en una misma pestaña (ver Figura 46).

Dependiendo de si la capa a editar es de puntos, líneas o polígonos aparecen unas opciones u otras. En el caso de una capa de puntos se le puede asociar un color a los puntos o una imagen, cambiar el tamaño del punto y su ángulo, el ángulo es la inclinación de la imagen. Si la capa es de líneas se muestran las opciones para cambiar el color y la anchura del trazado. Por último si la capa contiene polígonos se permite la modificación del color de relleno y la anchura de su borde.

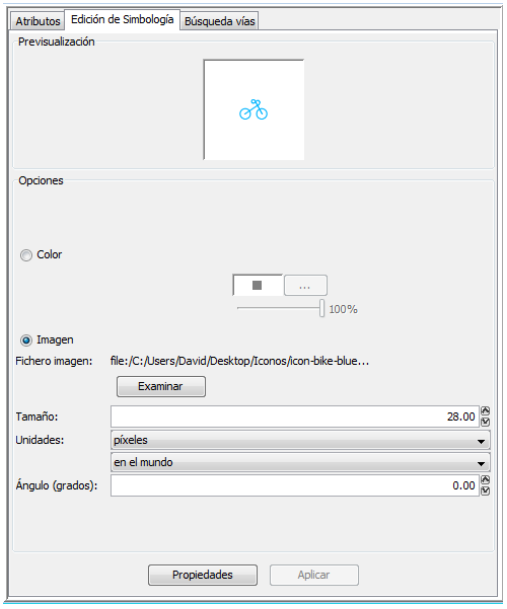

Figura 46 Pestaña de Edición de Simbología

Pestaña de Búsqueda de vías

Esta pestaña (ver Figura 47) permite buscar una dirección postal y centrar el mapa sobre ella utilizando un *geocoder*. Al centrar el mapa aparece el icono en el mapa indicando la localización exacta del lugar que se ha buscado. El *geocoder* utilizado es el que está implementado en IDEZar, pero la extensión se diseñó permitiendo incluir más en el futuro.

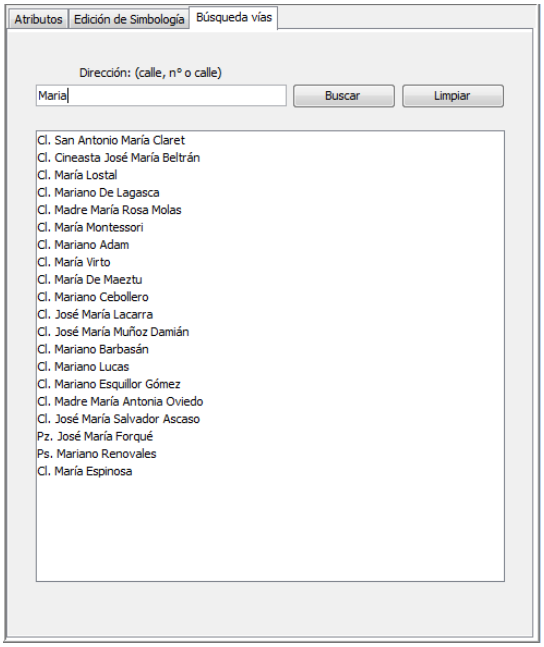

Figura 47 Pestaña de búsqueda de vías

Su funcionamiento reside en una conexión http al servicio SRW del *geocoder*, a donde se realizan peticiones a través del protocolo HTTP-GET. El usuario de la extensión escribe en el cuadro de texto la dirección que quiere buscar. Como se quiere una búsqueda que permita similitudes de cadenas de texto, se añade a cada cadena de texto que no sea dígito introducida por el usuario un * antes y después de la cadena, para que el servicio entienda que la cadena puede encontrarse precedida o seguida por cualquier otra cadena de texto.

El filtro de la consulta tiene la siguiente forma:

```
<Filter>
   <PropertyIsLike>
      <PropertyName>street</PropertyName>
      <Literal>”Aquí iría el texto del usuario”</Literal>
   </PropertyIsLike>
</Filter>
```

Y la consulta entera:

```
http://idezar.zaragoza.es/SRW/servlet/search/SRW?query=<Filter><PropertyIsLik
e><PropertyName>street</PropertyName><Literal>alfonso</Literal></PropertyIsLi
ke></Filter>&startRecord=1&maximumRecords=20&recordPacking=JSON
```

Donde cabe resaltar los parámetros, startRecord que indica el índice del primer recurso a devolver, maximumRecords que señala el número máximo de recursos que se devolverán y el recordPacking que informa del formato de la respuesta que devolverá el servicio. Los valores seleccionados para realizar las consultas desde la aplicación son startRecord=1 ya que se quiere que siempre empiecen los resultados devueltos por el que tenga más probabilidad de que fuera el que el usuario buscaba. MaximumRecords=20 se ha buscado un compromiso entre un número de respuestas suficientes para el usuario dándole un gran abanico de resultados posibles, pero sin sobresaturarlo de información, no es necesario que empiece a navegar por una gran cantidad de resultados posibles cuando en la gran mayoría de ocasiones será el primer resultado el buscado. Por último recordPacking=JSON, se ha decidido utilizar JSON ya que es un formato muy ligero y sencillo de utilizar.

El resultado de la consulta como se ha menciona se realiza en formato JSON. El cual tiene la siguiente estructura:

```
{
   "crs":{
      "properties":{
         "name":"urn:ogc:def:crs:EPSG::23030"
      },
      "type":"name"
   },
   "error":false,
   "features":[
      {
         "geometry":{
            "geometries":[
               {
                  "coordinates":[
                     677806.950073419,
                     4614091.4299955
                  ],
                  "properties":{
                     "numeroPolicia":""
                  },
                  "type":"Point"
               }
            ],
            "type":"GeometryCollection"
```

```
         },
         "properties":{
            "junta-municipal":"El Rabal",
            "junta-vecinal":"",
            "tipo-via":"calle",
            "tipo-via-abr":"Cl.",
            "title":"ALFONSO SOLANS SERRANO"
         },
         "type":"Feature"
      },   ],
   "numberOfRecords":14,
   "type":"FeatureCollection"
}
```


El crs indica el formato de proyección en el que se devuelven las coordenadas. Este valor se almacenará y se comparará con el valor de la proyección actual de la vista de gvSIG, ya que si son diferentes las proyecciones, las coordenadas devueltas por el servicio tendrán que ser reproyectadas, esto se explicará a continuación.

Después incluye una lista con la etiqueta "*features*" donde irán incluidos todos los resultados posibles. Cada resultado está compuesto por un campo "geometry" donde se almacena entre otras cosas las coordenadas que se utilizaran para centrar el mapa sobre dicho punto. Otro campo llamado "properties" que almacena el tipo de la vía, su nombre, los cuales son utilizados para mostrar los nombres de los resultados por pantalla, se concatena el tipo de la vía con el nombre.

El *geocoder* de IDEZar principalmente utiliza el sistema de coordendas EPSG: 23030, por lo que si se está trabajando en gvSIG sobre otro sistema de coordenadas, el resultado obtenido no tendría sentido. Es por ello que se comprueba que se está trabajando en dicho sistema de coordenadas, si no es así, se realiza una transformación de coordenadas de la proyección EPSG:23030 a la que se esté trabajando.

Las transformaciones de coordenadas entre diferentes sistemas de referencia no es algo trivial que se solucione con una simple cuenta. Es por ello que existen un gran número de librerías en el mercado que cumplen esa funcionalidad. gvSIG trae varias, pero aun utilizando estas librerías hay determinadas proyecciones en las que la transformación no se realiza con precisión, lo cual es un problema. Es por ello que existen rejillas de transformación que poseen valores para corregir la conversión, en concreto se ha utilizado la rejilla cuando las proyecciones origen y destino corresponde con alguna de las siguientes: EPSG:25829, EPSG:25830, EPSG:25831, EPSG:4258, EPSG:23029, EPSG:23030, EPSG:23031, EPSG:4230.

D. 6. Extensión para cargar y guardar capas independientemente del origen de datos

El objetivo de esta extensión es facilitar el trabajo al usuario, abstrayéndole de las configuraciones de las capas para las diferentes fuentes de datos. La extensión está diseñada principalmente para que un administrador guarde las capas preconfiguradas, evitando que el usuario estándar tenga que volver a configurarlas cada vez que vaya a usarlas. Siendo que esta función ahorra mucho tiempo de edición, también se ha facilitado que el usuario estándar pueda crear sus propias capas preconfiguradas.

D.6.1. Estructura de la extensión

Observando el diagrama de clases simplificado (ver Figura 48) se puede apreciar que de nuevo la clase principal de la extensión extiende de la clase Extension de gvSIG aportando así la capacidad de ser una extensión del mismo y no tener que modificar la estructura de gvSIG para implementar la nueva funcionalidad.

Principalmente, la clase MisCapas instancia las clases MisCapasTocMenuEntry y MisCapasWizard. Esta primera clase añade la opción “Añadir a Mis Capas” en el menú desplegable que aparece cuando se pulsa el segundo botón sobre una capa. La segunda clase permite añadir una nueva pestaña a la interfaz de añadir capa, esta nueva pestaña llamada “Mis Capas” muestra todas las capas preconfiguradas guardada y permite cargarlas o eliminarlas si fueron creadas por el usuario. Para poder añadir estas dos opciones a las interfaces la clase MisCapasTocMenuEntry tiene que extender la clase TocMenuEntry de gvSIG y la clase MisCapasWizard extiende la clase WizardPanel.

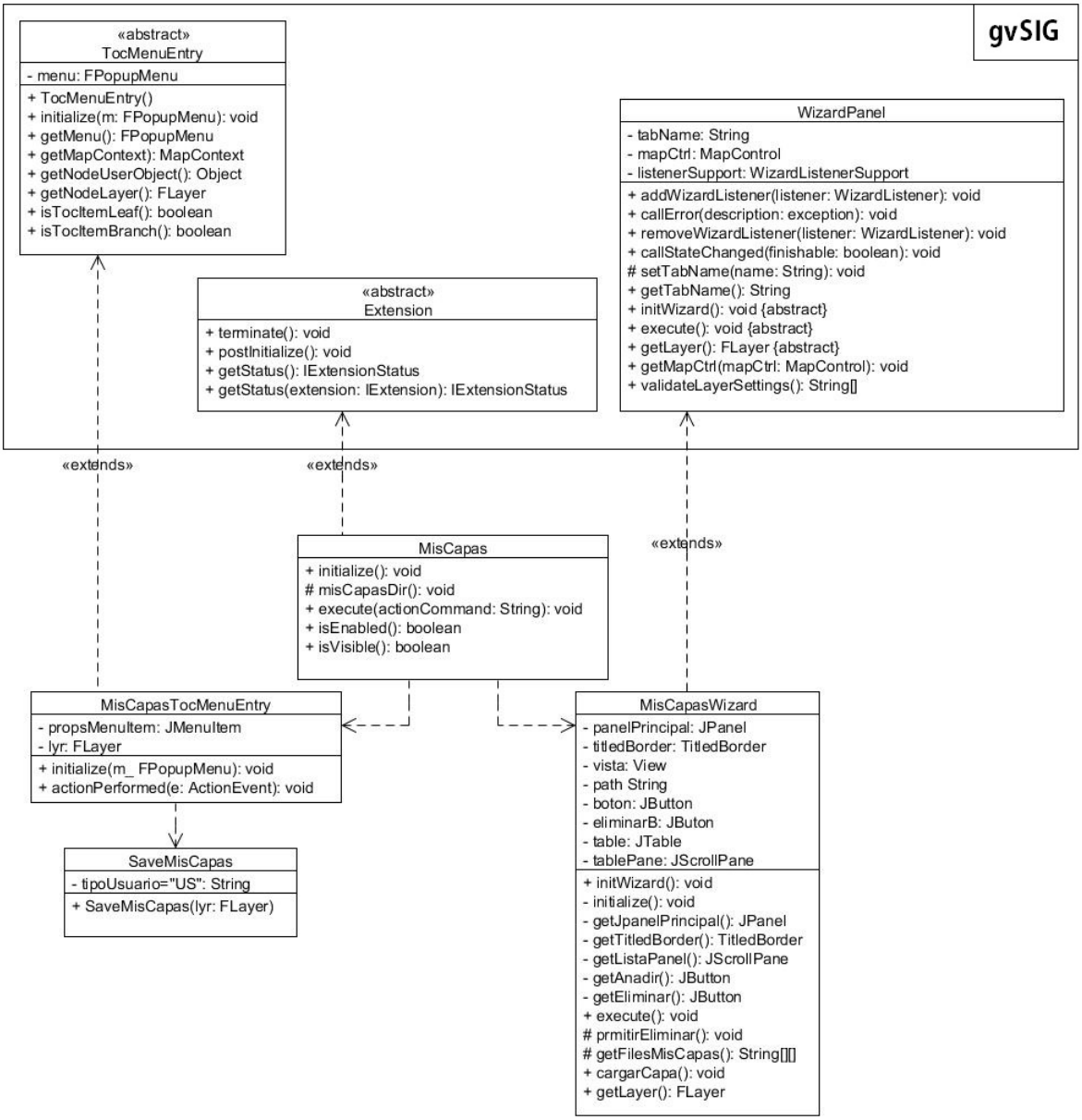

Figura 48 Diagrama de clases simplificado de la extensión Mis Capas

La clase SaveMisCapas que utiliza MisCapasTocMenuEntry, crea un fichero XML con el nombre que el usuario ha indicado y lo guarda en la carpeta "Mis Capas" del directorio de preferencias de gvSIG. En este fichero se almacenan todas las propiedades necesarias de la capa específica para poder recuperarla como la ruta a la fuente de datos, sus configuraciones, estilos de dibujado si tiene, etc.

Para cargarla la clase MisCapasWizard utiliza las funciones de la clase XMLEntity se gvSIG. Esta clase permite cargar y guardar capas en formato XML.

D.6.2. Tipos de usuarios

Hay dos tipos de usuarios de esta extensión, los administradores y los usuarios estándar del sistema.

El administrador con su herramienta crea capas preconfiguradas que distribuirá a los usuarios estándar con el objetivo de facilitar sus tareas. Estas capas no podrán ser eliminadas de la herramienta del usuario estándar.

El usuario estándar también puede crear capar preconfiguradas para poder facilitar su tarea, pero estas capas al ser propias de casa usuario pueden ser eliminadas por este.

Es por ello, que cada tipo de usuario tiene su propia herramienta "Mis Capas". La diferencia entre la del administrador y la del usuario estándar radica en el campo que almacena el tipo de usuario que ha creado la capa, con el objetivo de diferenciarlos.

D.6.3. Interfaces

Siendo que la aplicación permite añadir nuevas capas a "Mis Capas" y cargar las ya existentes se han diseñado dos interfaces, cada una solventando las diferentes funcionalidades.

La primera interfaz (ver Figura 49) permite guardar en la herramienta una capa que esté cargada en gvSIG. La interfaz consiste en una opción añadida en el cuadro de opciones que aparece en el TOC (*Table of Contents* o Tabla de contenidos) cuando se pulsa el segundo botón del ratón sobre una capa.

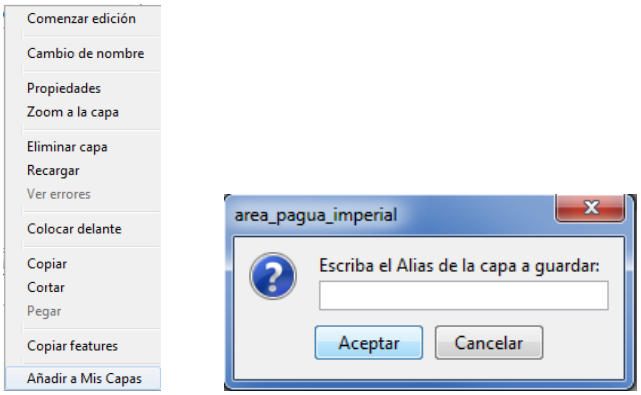

Figura 49 Menú de opciones sobre una capa y la ventana para introducir su Alias

Esta opción llamada "Añadir a Mis Capas" al ser pulsada abre un diálogo de texto para que el usuario introduzca un Alias a la capa. Cabe destacar que no se permite

introducir un nombre ya utilizado en una capa guardada, con el objetivo de evitar la eliminación o sobrescripción de capas ya creadas.

La segunda interfaz (ver Figura 50), permite cargar la capa deseada de todas las que tenga almacenada la herramienta. La interfaz consiste en una pestaña nueva en la ventana de añadir capa.

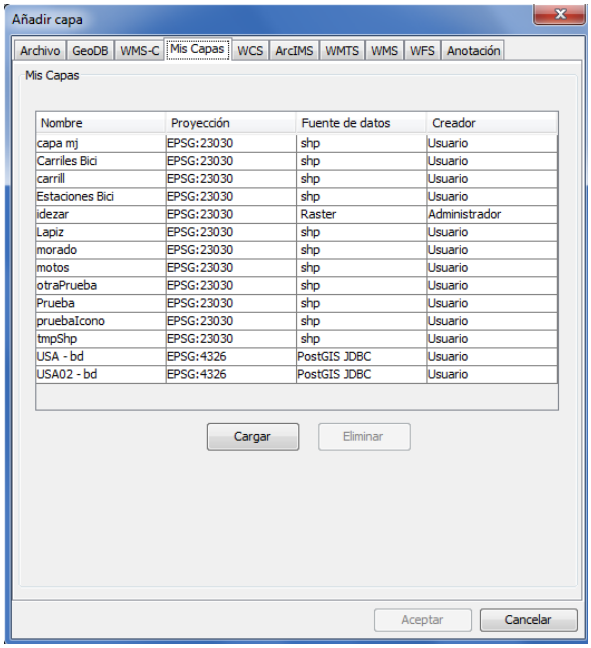

Figura 50 Interfaz gráfica que permite cargar una capa preconfigurada

En esta pestaña aparecerá la lista con todas las capas guardadas, las cuales se podrán añadir al presente proyecto de gvSIG pulsando doble click sobre ellas o seleccionándolas y pulsando el botón cargar. Además del Alias con el que se ha guardado la capa, la pestaña muestra la información de la proyección de cada capa, tipo de la fuente de datos y su creador.

D. 7. Extensión para la importación y exportación en formato ligero GeoJSON

GeoJSON es un formato de intercambio geoespacial basado en *JSON* (*JavaScript Object Notation*). *JSON* es un formato ligero para el intercambio de datos que se ha hecho popular gracias a su simplicidad frente a XML. Los servicios

integrados en IDEZar utilizan este formato para el intercambio de la información georreferenciada permitiendo su visualización en mapas interactivos.

Para afrontar la importación y exportación siguiendo el estilo utilizado en gvSIG para otros formatos ligeros, se implementaron dos drivers para la exportación y un driver para importar capas en formato *GeoJSON*.

D.7.1. Estructura de la extensión

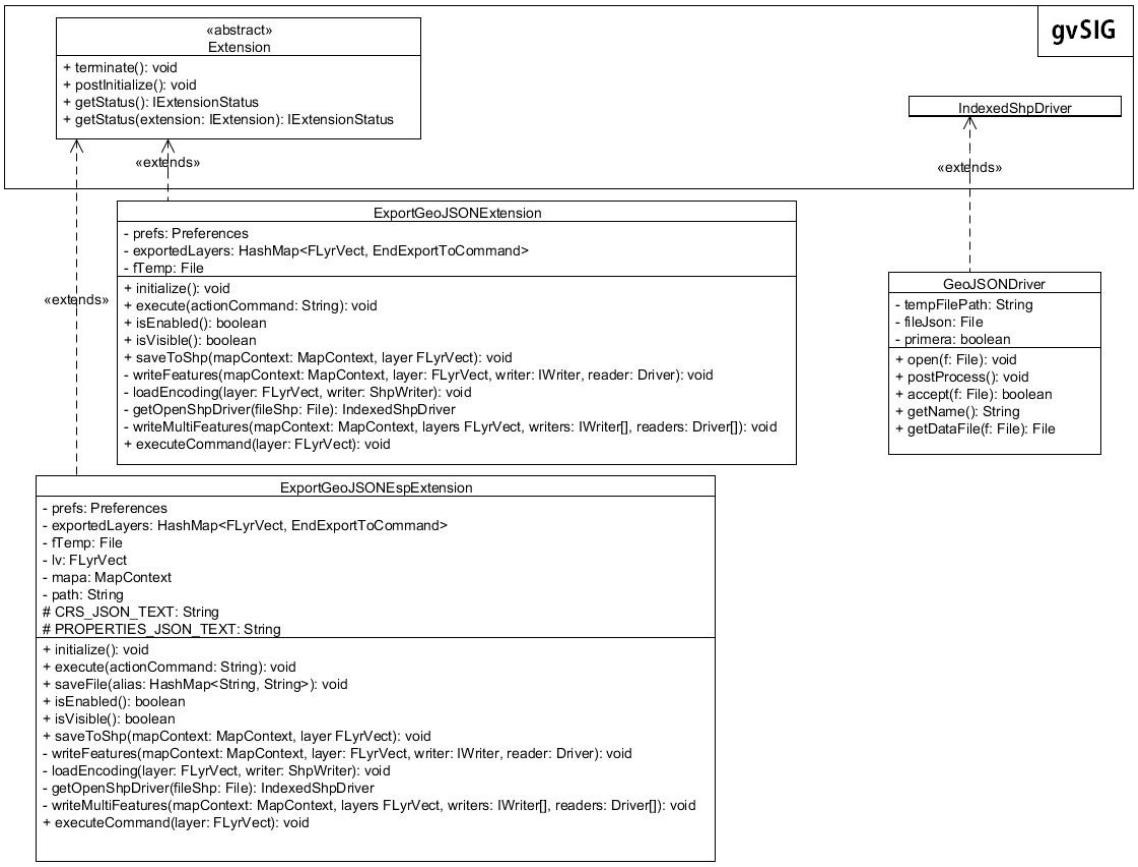

Figura 51 Diagrama de clases simplificado de la extensión GeoJSON

El anterior diagrama de clases (ver Figura 51) ha sido simplificado para hacerlo más comprensible, han sido obviadas las clases que permiten el tratamiento de datos SQL y del interfaz de la ventana emergente que aparece al exportar el *GeoJSON* con estilo, además de no desplegar la clase IndexedShpDriver (es el driver de cargado de *shapefiles*) de gvSIG.

Principalmente se tienen tres clases en esta extensión, que corresponden a cada una de las funcionalidades que aporta: exportación en *GeoJSON*, exportación en *GeoJSON* con un estilo de dibujado asociado y la importación en *GeoJSON*. Las clases que permiten la exportación extienden la clase Extensión de gvSIG, convirtiéndolas en

extensiones propias mientras la clase que permite la importación no necesita extender esta clase ya que su funcionalidad es usada directamente por gvSIG y necesita un formato específico como driver de lectura, ya que todos los drivers para la importación de ficheros en gvSIG deben tener las mismas funciones debido a que gvSIG no distingue entre los drivers, primero comprueba que driver utilizar y después llama a las mismas funciones independientemente del formato.

A causa de la decisión tomada de cómo permitir el tratamiento de *GeoJSON*, en los tres casos lo que se hace es que gvSIG interactúe con formatos *shapefiles*, es decir si se quiere cargar un *GeoJSON* se convierte a *shapefile* y se carga este en gvSIG. Y si se quiere exportar, primero se exporta en *shapefile* y después se convierte a *GeoJSON*. Estas operaciones de conversión se realizan gracias a las librerías OGR de GDAL. Estas librerías permiten la lectura y escritura de formatos de datos geoespaciales, muchas aplicaciones comerciales como GRASS GIS o Google Earth las utilizan para el acceso a datos geográficos.

D.7.2. Exportación

Se crearon dos drivers de exportación. El primer driver consiste en la exportación de la capa cargada en gvSIG tal cual en formato *GeoJSON*, es decir el fichero creado al exportar contiene las *features* de la capa origen con todos los atributos que poseían.

El primer driver para exportar se realizó de la siguiente manera. Como gvSIG permite exportar capas en formato *shapefile*, primero se exporta la capa en dicho formato en el directorio temporal del sistema. Una vez se tienen los ficheros *shapefiles*, a través de unas librerías que permiten convertir *shapefile* en *GeoJSON*, se realiza la transformación, guardando el fichero *GeoJSON* resultado en el directorio que había indicado el usuario de la aplicación. Es decir las transformaciones que se producen son transparentes al usuario por lo que él desconocerá que se produce una exportación en *shapefile* primero.

El segundo driver de exportación se desarrolló con el propósito de permitir una mayor integración de la nueva herramienta en el entorno IDEZar. Este driver permite la exportación de una capa en formato *GeoJSON* pero con un formato específico utilizado en la infraestructura IDEZar. De este modo se consigue una continuidad en el proceso

edición con la herramienta ya que una vez terminada la edición se pueda exportar la capa generada a un formato específico que permite directamente su consumo en diferentes aplicaciones y servicios.

El formato utilizado en IDEZar tiene la siguiente estructura:

```
{
    "crs": {
        "type": "EPSG",
        "properties": {
            "code": 23030
        }
    },
    "properties": {
        "title": "Motos",
        "icon": "",
        "link": "",
        "description": ""
    },
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "properties": {
                "title": "ff",
                "description": 15,
                "icon": "http://www.zaragoza.es/contenidos/iconos/agenda.png"
            },
            "geometry": {
                "type": "Point",
                "coordinates": [
                    676200.7538051647,
                    4613253.262332385
                ]
            }
        },…
}
```

Los cambios radican en la inclusión de dos objetos al inicio del *GeoJSON*, el objeto “crs” que indica la proyección geográfica de la capa y el objeto “propeties” que define las propiedades de la capa con los atributos *title* (título de la capa), *icon* (si se le asigna un icono por defecto a la capa), *link* y *description* (descripción de la capa). Después dentro de las propiedades de cada *feature*, en el formato de IDEZar debe tener unos atributos específicos. Los atributos que podrá tener son: *title* (el nombre asignado al punto), *link*, *description* (descripción de lo que representa el punto), *category* (categoría), date (fecha) y *priority* (prioridad). Cabe destacar que no es necesario que estén rellenos todos los atributos. En el caso de que la capa esté formada por puntos, es decir sea una capa de puntos, además de los atributos anteriores podrá llevar asociado otro atributo más llamado *icon* el cual guarda la ruta a la imagen que se utilizará para

representar el punto. Si la capa es de líneas o polígonos en vez del atributo *icon* llevarán un objeto llamado *style* que indicará el estilo de pintado de la línea o polígono. El objeto *style* está formado por los siguientes atributos: *strokeColor* (color del trazo), *strokeOpacity* (opacidad del trazo), *strokeWidth* (anchura del trazo), *fillColor* (color del relleno) y *fillOpacity* (opacidad del relleno). Estos dos últimos atributos solo tienen sentido en el caso de que la capa represente polígonos.

Para poder rellenar estos campos específicos al exportar una capa, se ha diseñado una ventana que permita relacionar cada atributo del *GeoJSON* de IDEZar con un atributo que posea la capa en gvSIG, asignando de este modo el valor del atributo de la capa al atributo necesario.

Si la capa es de puntos, la ventana tiene la siguiente forma (ver Figura 52):

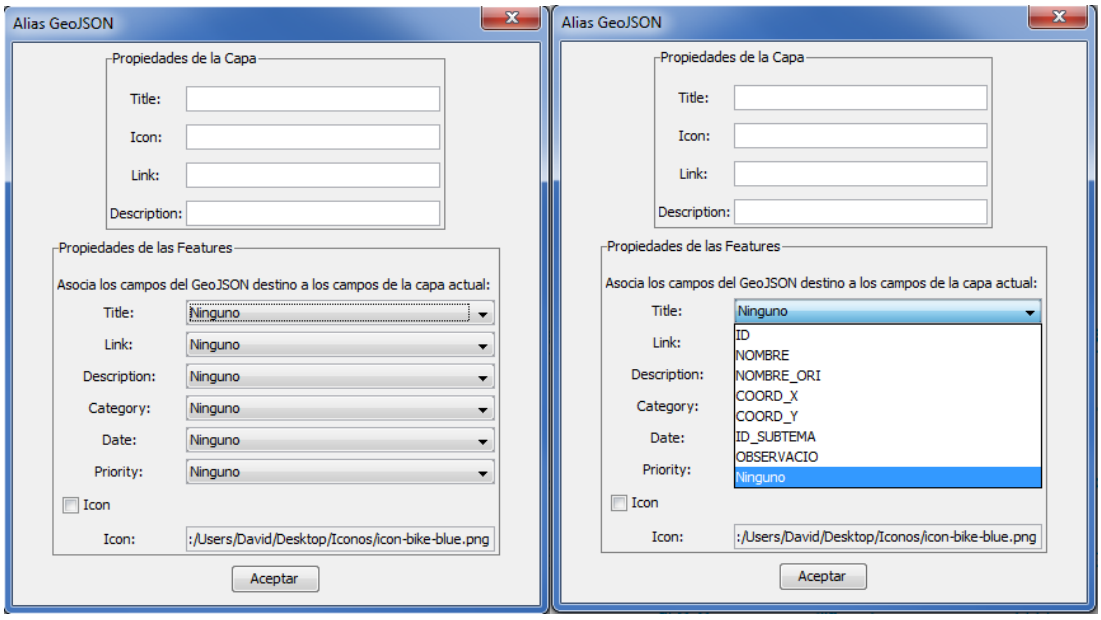

Figura 52 Ventanas para exportar una capa de puntos en GeoJSON con estilo

Arriba aparecen los campos de las propiedades de la capa. Si se deja algún atributo vacío en el *GeoJSON* seguirá apareciendo con valor vacío. Abajo aparecen todos los atributos que puede poseer una *feature*. Para poder asignarlos hay que desplegar la lista y seleccionar uno de los campos de la capa actual. Por defecto aparece el valor "Ninguno" seleccionado, el cual indica que no se le asocia a dicho atributo ningún campo y no aparecerá el atributo en el *GeoJSON* exportado. Por último al ser una capa de puntos aparece un campo de texto para la ruta del icono asociado a los puntos. Por defecto tendrá el valor de la ruta donde se encuentre el icono que tiene

asociada la capa en gvSIG. Para poder cambiarlo habrá que seleccionar el checkbox de *icon*.

Si la capa es de líneas o polígonos la ventana está definida del siguiente modo (ver Figura 53):

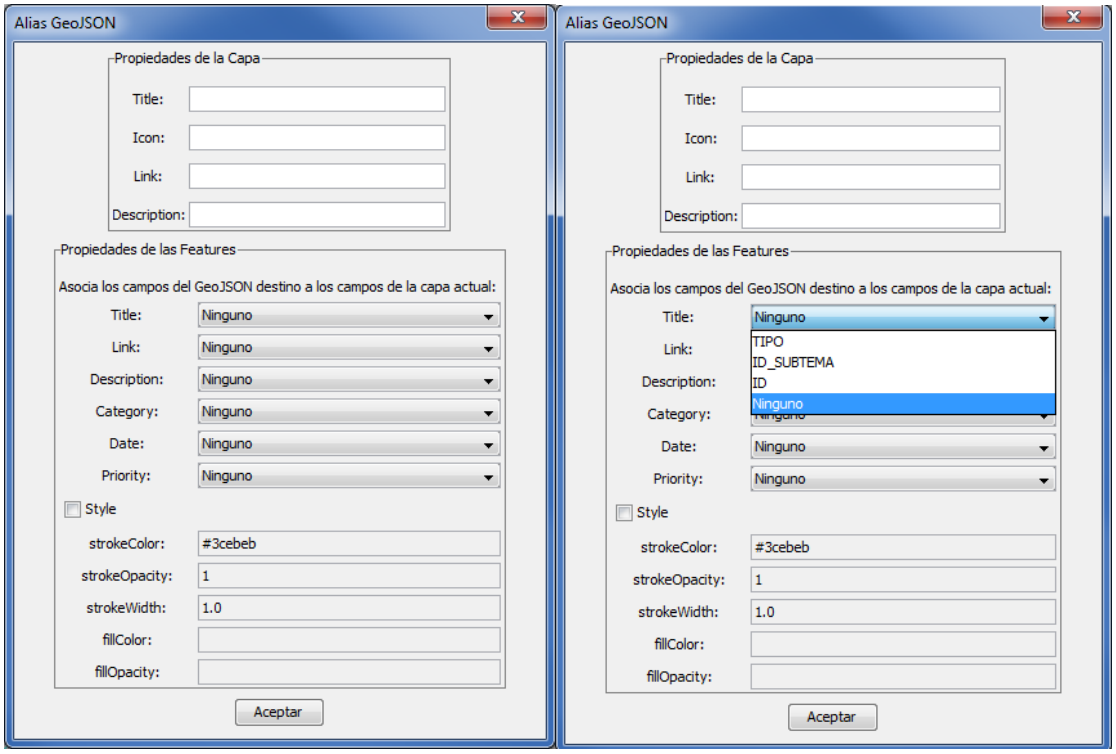

Figura 53 Ventanas para exportar una capa de líneas o polígonos en GeoJSON con estilo

El diseño es el mismo que en caso de la capa de puntos, únicamente cambia el atributo *icon* que desaparece en favor del objeto *style*. Los valores de los atributos del *style* se rellenan por defecto con los valores de dibujado de la capa en gvSIG. En el caso de ser una capa de líneas solo se rellenan los tres primeros campos y en las capas de polígonos los cinco atributos. Si se quieren modificar su valor habrá que seleccionar el checkbox.

D.7.3. Importación

El driver de importación permite cargar una capa en formato *GeoJSON*. En gvSIG para cargar una capa a partir de un fichero de formato ligero hay que ir a la ventana añadir capa y en la pestaña "Archivo" buscar la ruta del fichero a importar. gvSIG carga todos los drivers de importación que posee para saber si puede abrir el fichero seleccionado (ver Figura 54).

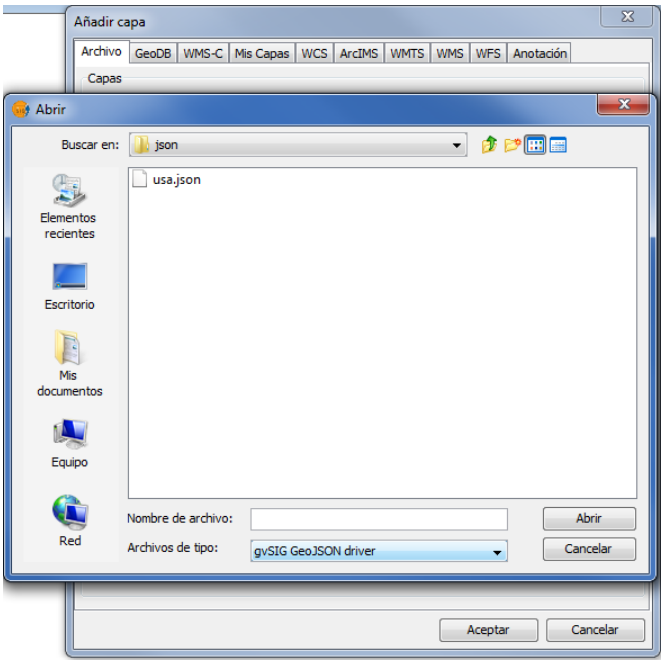

Figura 54 Importación de un GeoJSON a través de la pestaña Archivo

Para que gvSIG pueda cargar todos los drivers de importación y utilizarlos del mismo modo, estos deben tener la misma estructura interna implementando unas funcionas específicas. En este caso la solución adoptada no fue crear una estructura de datos nueva que permitiera leer el propio *GeoJSON* sino seguir utilizando la estructura implementada para los *shapefiles* en gvSIG. Es decir, se optó por la conversión de *GeoJSON* a *shapefile* de una forma encubierta para el usuario.

Pero al seguir necesitando las funciones específicas lo que se hizo fue extender la clase que permitía la importación de los *shapefiles*. De modo que solo se sobrescribirían las funciones que interactuasen con el *GeoJSON*, es decir las funciones *open(file)* encargada de abrir el fichero, *postProcess()* encargada del guardado de los cambios sobre la fuente original, *accept(file)* encargado de comprobar si es un fichero que hay que abrir con ese driver (comprueba si la extensión del fichero es .json) y *getDataFile(file)* encargada de la obtención del fichero de datos.

De esta forma la mecánica llevada a cabo para la importación de *GeoJSON* consiste en convertir el fichero importado en *shapefile* utilizando las GDAL de OGR. Este *shapefile* es importado a gvSIG utilizando el driver de *shapefiles* que ya posee. A partir de aquí la interacción del programa es con un *shapefile*, hasta el momento en el que hay que guardar los cambios realizados al fichero. Entones se vuelca sobre el *shapefile* y este convertido de nuevo en *GeoJSON* remplazando el fichero original.

ANEXO E
PRUEBAS FUNCIONALES

Para garantizar la calidad del sistema se diseñaron y realizaron las pruebas funcionales que se recogen en la siguiente tabla:

Id	Descripción/propósito	Precondiciones/acciones previas	Acciones de ejecución	Resultados esperados
1	Conectarse a un servicio WMTS	Iniciar la aplicación. Abrir una vista.	Pulsar el botón de añadir capa. Pestaña WMTS. Introducir la url de un servicio WMTS.	Permite navegar por diferentes ventanas donde se seleccionan los parámetros del mapa a pedir.
2	Conectarse a un servicio WMTS - No hay red	No hay red. Iniciar la aplicación. Abrir una vista.	Pulsar el botón de añadir capa. Pestaña WMTS. Introducir la url de un servicio WMTS.	Avisa de que no se puede conectar al servicio
3	Cargar un mapa a partir del servicio WMTS con la misma proyección que la vista actual.	Iniciar la aplicación. Abrir una vista. Conectarse a un servicio WMTS.	Seleccionar los parámetros del mapa a pedir.	Pinta el mapa WMTS correctamente.
4	Cargar un mapa a partir del servicio WMTS con la misma proyección que la vista actual. - No hay red	No hay red. Iniciar la aplicación. Abrir una vista. Conectarse a un servicio WMTS.	Seleccionar los parámetros del mapa a pedir.	Avisa de que se ha excedido el *timeout* establecido de 15 segundos para el pintado de la capa y la marca como no visible.
5	Conectarse a un servicio WMS-C	Iniciar la aplicación. Abrir una vista.	Pulsar el botón de añadir capa. Pestaña WMS-C. Introducir la url de un servicio WMS-C.	Permite navegar por diferentes ventanas donde se seleccionan los parámetros del mapa a pedir.
6	Conectarse a un servicio WMS-C - No hay red	No hay red. Iniciar la aplicación. Abrir una vista.	Pulsar el botón de añadir capa. Pestaña WMS-C. Introducir la url de un servicio WMS-C.	Avisa de que no se puede conectar al servicio

7	Cargar un mapa a partir del servicio WMS-C con la misma proyección que la vista actual.	Iniciar la aplicación. Abrir una vista. Conectarse a un servicio WMS-C.	Seleccionar los parámetros del mapa a pedir.	Pinta el mapa WMS-C correctamente.
8	Cargar un mapa a partir del servicio WMS-C con la misma proyección que la vista actual. - No hay red	No hay red. Iniciar la aplicación. Abrir una vista. Conectarse a un servicio WMS-C.	Seleccionar los parámetros del mapa a pedir.	Avisa de que se ha excedido el timeout establecido de 15 segundos para el pintado de la capa y la marca como no visible.
9	Cargar el *Overview Map* sin ninguna capa cargada	Iniciar la aplicación. Abrir una vista.	Pulsar el botón del *Overview Map*.	Carga *Overview Map* sin ningún mapa
10	Cargar el *Overview Map* con solos capas vectoriales cargadas	Iniciar la aplicación. Abrir una vista. Cargar algunas capas vectoriales.	Pulsar el botón del *Overview Map*.	Carga el *Overview Map* sin ningún mapa
11	Cargar el *Overview Map* con alguna capa ráster cargada.	Iniciar la aplicación. Abrir una vista. Cargar algunas capas ráster.	Pulsar el botón del *Overview Map*.	Carga el *Overview Map* con el último mapa ráster en la jerarquía.
12	Borrar la única capa ráster cargada, con el *Overview Map* cargado.	Iniciar la aplicación. Abrir una vista. Cargar una capa ráster. Abrir el *Overview Map*	Eliminar la capa ráster cargada.	El *Overview Map* no muestra ningún mapa.
13	Borrar la última capa ráster cargada existiendo más capas ráster cargadas, con el *Overview Map* cargado.	Iniciar la aplicación. Abrir una vista. Cargar varias capas ráster. Abrir el *Overview Map*	Eliminar la última capa ráster en la jerarquía cargada.	El *Overview Map* muestra la nueva última capa ráster de la jerarquía.
14	Reordenar la jerarquía de las capas ráster, con el *Overview Map* cargado.	Iniciar la aplicación. Abrir una vista. Cargar varias capas ráster. Abrir el *Overview Map*	Cambiar la posición en la jerarquía las capas ráster.	El *Overview Map* muestra la nueva última capa ráster de la jerarquía.
15	Abrir la herramienta de gestión de edición sin tener seleccionada ninguna capa vectorial	Iniciar la aplicación. Abrir una vista.	Pulsar el botón de gestión de edición.	Aparece la barra lateral, con las pestañas "Atributos" y "Edición de simbología" bloqueadas

16	Selección de la *feature* que se está editando en ese momento en la pestaña Atributos de la herramienta de gestión de edición.	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. Tener seleccionada la capa vectorial	Pulsar los botones de navegación (flechas) del gestor de edición.	Se seleccionan en el mapa
17	Desbloqueo de las pestañas "Atributos" y "Edición de simbología" cuando hay una capa vectorial seleccionada	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. No tener seleccionada ninguna capa vectorial	Seleccionar una capa vectorial.	Se desbloquean las pestañas y muestran la información de la capa seleccionada
18	Generar alias a los atributos.	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. Tener seleccionada la capa vectorial	Pulsar sobre el botón de alias y escribir los aliases de los atributos	En la pestaña atributos aparecen inmediatamente los nuevos nombres de los atributos.
19	Búsqueda de atributos con caracteres extraños	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. Tener seleccionada la capa vectorial	Pulsar sobre el botón de búsqueda. Buscar en un atributo de texto el carácter 'ñ'	Muestra todos los resultados que contienen el carácter 'ñ'
20	Centrar mapa sobre *feature* que se está editando.	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. Tener seleccionada la capa vectorial	Mover la posición del mapa con la herramienta desplazamiento. Pulsar el botón centrar de la pestaña "Atributos"	Se centra el mapa respecto a la *feature* que se está editando en ese momento en la pestaña "Atributos"
21	Centrar y hacer zoom mapa sobre *feature* que se está editando.	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. Tener seleccionada la capa vectorial	Mover la posición del mapa con la herramienta desplazamiento. Pulsar el botón zoom de la pestaña "Atributos"	Se centra y realiza zoom en el mapa respecto a la *feature* que se está editando en ese momento en la pestaña "Atributos"
22	Copiado y pegado de una *feature* en la misma capa.	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. Tener seleccionada la capa vectorial	Poner la capa vectorial en edición. Pulsar el botón copiar sobre una *feature*. Ir a otra *feature*. Pulsar el botón pegar.	La segunda *feature* tiene los mismos valores que tenía la primera *feature* en el momento de copiar los valores.

23	Copiado y pegado de una *feature* en distintas capas.	Iniciar la aplicación. Abrir una vista. Cargar dos capas vectoriales. Abrir la herramienta de gestión de edición. Tener seleccionada una capa vectorial	Poner la capa vectorial en edición. Pulsar el botón copiar sobre una *feature*. Ir a otra capa.	No deja pegar.
24	Añadir una *feature* a una capa	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. Tener seleccionada y en edición la capa vectorial.	Crear con las herramientas de dibujado una nueva *feature*.	Se incrementa el número de *features*. Aparecerá el formulario de la nueva *feature* en la pestaña Atributos.
25	Eliminar una *feature* de una capa	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. Tener seleccionada y en edición la capa vectorial.	Pulsar el botón de eliminar, cuando esté en la pestaña Atributos la *feature* que se quiere eliminar.	Se elimina, decrementa el número de *features*, y se muestra la siguiente *feature*.
26	Bloqueo de los campos de texto y los botones de copiar, pegar si no lo estaba y el de eliminar al terminar edición de la capa vectorial.	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. Tener seleccionada y en edición la capa vectorial.	Terminar edición de la capa.	Se bloquean los campos de texto, los botones de copiar, pegar y eliminar.
27	Asociar un icono a una capa vectorial de puntos.	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial de puntos. Abrir la herramienta de gestión de edición. Tener seleccionada la capa vectorial de puntos. Ir a la pestaña "Edición de simbología"	Seleccionar el radio botón de imagen. Seleccionar el fichero de imagen. Pulsar el botón aplicar.	Ahora los puntos están representados por la imagen seleccionada.
28	Incremento del tamaño del icono asociado a un punto cuando este está seleccionado.	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial de puntos que tiene asociada una imagen, o asociarle una imagen.	Seleccionar un punto de la capa.	El icono es ligeramente más grande que el icono del resto de puntos.
29	Volver a seleccionar el icono asociado a un punto, tras haber cambiado a la opción color.	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial de puntos que tiene asociada una imagen, o asociarle una imagen.	Selecciona la opción color. Aplicar. Seleccionar de nuevo la opción imagen. Aplicar	Vuelva a tener la misma imagen aplicada.

30	Aplicar tamaño a un punto, línea, polígono	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial. Abrir la herramienta de gestión de edición. Tener seleccionada la capa vectorial de puntos. Ir a la pestaña "edición de simbología"	Aumentar el tamaño con los botones o introduciendo el valor en el cuadro de texto. Pulsar el botón Aplicar.	Ha aumentado el tamaño del punto, línea o el borde del polígono.
31	Búsqueda en el *geocoder* de una dirección sin cargar ninguna capa.	Iniciar la aplicación. Abrir una vista. Abrir la herramienta de gestión de edición. Ir a la pestaña "búsqueda vías"	Introducir la dirección a buscar en el campo de texto y pulsar enter o el botón buscar	Aparece un aviso: "Debe existir al menos una capa para poder realizar la búsqueda"
32	Búsqueda en el *geocoder* de una dirección con caracteres "raros" como es "españa"	Iniciar la aplicación. Abrir una vista. Cargar una capa ráster. Abrir la herramienta de gestión de edición. Ir a la pestaña "búsqueda vías"	Introducir la dirección a buscar "españa" en el campo de texto y pulsar enter o el botón buscar	Aparece una lista de resultados.
33	Centrar resultado de una búsqueda del *geocoder*	Iniciar la aplicación. Abrir una vista. Cargar una capa ráster. Abrir la herramienta de gestión de edición. Ir a la pestaña "búsqueda vías". Realizar una búsqueda.	Pulsar sobre el resultado de la búsqueda deseado.	Centra el mapa y señala con un símbolo el lugar concreto de la dirección buscada.
34	Limpiar resultados de una búsqueda de *geocoder*.	Iniciar la aplicación. Abrir una vista. Cargar una capa ráster. Abrir la herramienta de gestión de edición. Ir a la pestaña "búsqueda vías". Realizar una búsqueda. Seleccionar un resultado.	Pulsar sobre el botón limpiar de la pestaña "búsqueda vías"	Borra lo que hubiera escrito en el cuadro de texto, los resultados y borra el icono del mapa que señala la dirección que había sido seleccionada.
35	Cambiar de capas con una búsqueda de *geocoder* realizada.	Iniciar la aplicación. Abrir una vista. Cargar dos capas. Abrir la herramienta de gestión de edición. Ir a la pestaña "búsqueda vías". Realizar una búsqueda. Seleccionar un	Cambiar de capa.	La pestaña "búsqueda vías" se mantiene con los mismos valores.
36	Añadir una capa a Mis Capas	Iniciar la aplicación. Abrir una vista. Cargar la capa que se quiera guardar en Mis Capas (vectorial o ráster)	Segundo botón sobre la capa. Añadir a Mis Capas. Se le da un Alias.	La nueva capa aparecerá en la lista de Mis Capas, donde se muestra: el nombre de la capa, proyección, fuente de datos y el creador

37	Eliminar una capa de Mis Capas creada por el usuario	Iniciar la aplicación. Abrir una vista.	Botón añadir capa. Pestaña "Mis Capas". Seleccionar la capa con creador "Usuario" a eliminar. Pulsar el botón eliminar.	La capa desaparece de la lista de capas de Mis Capas
38	Eliminar una capa de Mis Capas creada por el administrador	Iniciar la aplicación. Abrir una vista.	Botón añadir capa. Pestaña "Mis Capas". Seleccionar la capa con creador "Administrador" a eliminar. Pulsar el botón eliminar.	El botón de eliminar está bloqueado, impidiendo que se pueda eliminar
39	Cargar capa ráster a través de Mis Capas.	Iniciar la aplicación. Abrir una vista.	Botón añadir capa. Pestaña "Mis Capas". Doble click sobre la capa ráster.	Carga la capa correctamente.
40	Cargar capa vectorial de un *Shapefile* a través de Mis Capas.	Iniciar la aplicación. Abrir una vista.	Botón añadir capa. Pestaña "Mis Capas". Doble click sobre la capa con fuente de datos en un *Shapefile*.	Carga la capa correctamente.
41	Cargar capa vectorial de un fichero *GeoJSON* a través de Mis Capas.	Iniciar la aplicación. Abrir una vista.	Botón añadir capa. Pestaña "Mis Capas". Doble click sobre la capa con fuente de datos en un fichero *GeoJSON*.	Carga la capa correctamente.
42	Cargar capa vectorial de una BBDD a través de Mis Capas.	Iniciar la aplicación. Abrir una vista.	Botón añadir capa. Pestaña "Mis Capas". Doble click sobre la capa con fuente de datos en una BBDD.	Carga la capa correctamente.
43	Importación archivo *GeoJSON*	Iniciar la aplicación. Abrir una vista.	Botón añadir capa. Pestaña archivo. Buscar y seleccionar en el sistema el fichero *GeoJSON* a importar.	Se importa el fichero *GeoJSON*.
44	Modificar un *GeoJSON*	Iniciar la aplicación. Abrir una vista. Cargar un *GeoJSON*. Poner la capa en edición. Modificar algún valor.	Terminar edición de la capa. Guardar los cambios.	El *GeoJSON* origen amacena los nuevos valores.

45	Exportar una capa vectorial en formato *GeoJSON*	Iniciar la aplicación. Abrir una vista. Cargar una capa o crearla. Seleccionar la capa a exportar.	Capa/Exportar a/GeoJSON	Se exporta correctamente la capa a formato *GeoJSON*, siendo cada atributo de la capa una propiedad almacenada en *properties*.
46	Exportar una capa vectorial de puntos en formato *GeoJSON* con estilo, sin seleccionar ningún atributo.	Iniciar la aplicación. Abrir una vista. Cargar una capa de puntos o crearla. Seleccionar la capa a exportar.	Capa/Exportar a/GeoJSON con estilo. Rellenar los campos de la ventana emergente. No seleccionar ningún atributo.	Se exporta adecuadamente el *GeoJSON* con estilo.
47	Exportar una capa vectorial de puntos en formato *GeoJSON* seleccionando algún atributo	Iniciar la aplicación. Abrir una vista. Cargar una capa de puntos o crearla. Seleccionar la capa a exportar.	Capa/Exportar a/GeoJSON con estilo. Rellenar los campos de la ventana emergente. Seleccionar algún atributo.	Se exporta adecuadamente el *GeoJSON* con estilo.
48	Exportar una capa vectorial de líneas en formato *GeoJSON* sin seleccionar ningún atributo	Iniciar la aplicación. Abrir una vista. Cargar una capa de líneas o crearla. Seleccionar la capa a exportar.	Capa/Exportar a/GeoJSON con estilo. Rellenar los campos de la ventana emergente. No seleccionar ningún atributo.	Se exporta adecuadamente el *GeoJSON* con estilo.
49	Exportar una capa vectorial de líneas en formato *GeoJSON* con estilo, seleccionando algún atributo	Iniciar la aplicación. Abrir una vista. Cargar una capa de líneas o crearla. Seleccionar la capa a exportar.	Capa/Exportar a/GeoJSON con estilo. Rellenar los campos de la ventana emergente. Seleccionar algún atributo.	Se exporta adecuadamente el *GeoJSON* con estilo.
50	Exportar una capa vectorial de polígonos en formato *GeoJSON* con estilo, sin seleccionar ningún atributo	Iniciar la aplicación. Abrir una vista. Cargar una capa de polígonos o crearla. Seleccionar la capa a exportar.	Capa/Exportar a/GeoJSON con estilo. Rellenar los campos de la ventana emergente. No seleccionar ningún atributo.	Se exporta adecuadamente el *GeoJSON* con estilo.

51	Exportar una capa vectorial de polígonos en formato *GeoJSON* con estilo, seleccionando algún atributo	Iniciar la aplicación. Abrir una vista. Cargar una capa de polígonos o crearla. Seleccionar la capa a exportar.	Capa/Exportar a/GeoJSON con estilo. Rellenar los campos de la ventana emergente. Seleccionar algún atributo.	Se exporta adecuadamente el *GeoJSON* con estilo.
52	Exportar una capa vectorial de polígonos o líneas en formato *GeoJSON* con estilo, seleccionando algún atributo con el estilo definido en gvSIG.	Iniciar la aplicación. Abrir una vista. Cargar una capa de polígonos o crearla. Seleccionar la capa a exportar.	Capa/Exportar a/GeoJSON con estilo. Rellenar los campos de la ventana emergente.	Exporta adecuadamente el estilo.
53	Exportar una capa vectorial en formato *GeoJSON* con estilo, donde los nombres de los atributos contienen el carácter espacio.	Iniciar la aplicación. Abrir una vista. Cargar una capa vectorial o crearla con algún nombre de atributo que contenga espacios en blanco. Seleccionar la capa a exportar.	Capa/Exportar a/GeoJSON con estilo. Rellenar los campos de la ventana emergente.	Se exporta adecuadamente el *GeoJSON* con estilo.
54	Reproyección del *geocoder*.	Iniciar la aplicación. Abrir una vista con una proyección distinta a EPSG:23030. Cargar una capa. Abrir la herramienta de gestión de edición.	Ir a pestaña "búsqueda vías", buscar una dirección, seleccionar uno de los resultados.	El mapa se ha centrado correctamente en la dirección.

Realizando las pruebas se encontraron errores que fueron solucionados, a continuación se detallan:

- Id 2: Cuando no hay conexión de Internet no avisa que no se puede conectar al servicio WMTS, se queda esperando. SOLUCIONADO, ahora avisa con el mensaje: "Se ha producido un error al intentar conectar con el servicio."
- Id 6: Cuando no hay conexión de Internet no avisa que no se puede conectar al servicio WMS-C, se queda esperando. SOLUCIONADO, ahora avisa con el mensaje: "Se ha producido un error al intentar conectar con el servicio."
- Id 23: Al cambiar de una capa a otra, los botones copiar, pegar y eliminar se bloquean. SOLUCIONADO, ahora tiene el correcto funcionamiento.

- Id 44: Falló, saltó una excepción. SOLUCIONADO, había sido cambiada la ruta del ogr2ogr y no había sido cambiada en la ejecución de guardado.
- Id 52: El color en hexadecimal no era el adecuado cuando uno de los canales (RGB) era 0. SOLUCIONADO, si uno de los canales es 0 se inserta “00”.
- Id 53: Si el nombre del atributo de una campo contenía espacios en blanco daba error de sintaxis y no se exportaba. SOLUCIONADO, poniendo el nombre del campo entre comillado simple.

ANEXO F

MANUAL DEL DESARROLLADOR

F. 1. Introducción

A la hora de comenzar a desarrollar para gvSIG 1.12 surge la duda de cómo hacerlo, cómo configurar el entorno de trabajo, cómo comenzar una extensión nueva, etc. La documentación al respecto que se encuentra en la web es escasa y desactualizada la cual no aborda ni soluciona muchos de los problemas encontrados a la hora de la configuración del entorno, por ello para facilitar estas tareas a futuros desarrolladores se ha redactado el presente manual.

F. 2. Entorno de trabajo

El entorno de trabajo utilizado en este manual ha sido Eclipse Juno para Java en Windows 7.

El código fuente de gvSIG 1.12 se descarga desde el repositorio que tienen habilitado. Dentro del repositorio está disponible el trunk (versión en desarrollo pero no estable) y los tag (versiones anteriores y estables).

F. 3. Configurar el workspace

Lo primero es crear el workspace (ver Figura 55).

Figura 55 Ventana para la creación de un workspace en Eclipse

Nota: la ruta del workspace no debe contener espacios en blancos así que cuidado con establecer el workspace dentro de "Mis Documentos" o cualquier ruta así, porque luego el launcher puede no reconocer las clases y no se ejecutaría gvSIG.

Lo siguiente es establecer la codificación de los archivos a ISO-8859-1. Esto se cambia dentro de las preferencias (Window/Preferences...) de Eclipse, se expande "General" y únicamente hay que pinchar en Workspace (ver Figura 56).

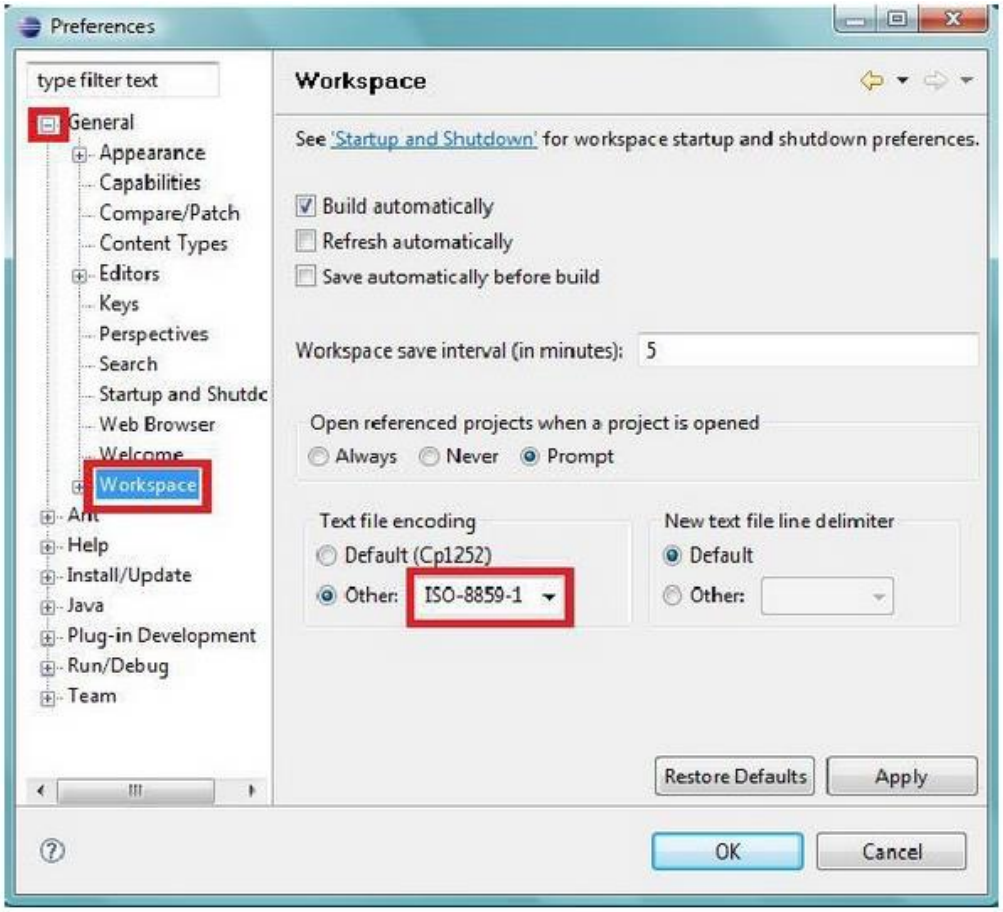

Figura 56 Ventana de preferencias en Eclipse

Lo siguiente que se debe configurar es la máquina virtual de JAVA, hay que utilizar Java JDK 1.6.

Para establecer la versión 1.6 de JAVA como la máquina virtual de Eclipse por defecto, hay que ir de nuevo a las Preferencias de Eclipse, expandir "Java", hacer click en "Installed JREs" y agregarla (botón add..) o asegurarse de que si está instalada junto con otras, ésta es la que usas (ver Figura 57).

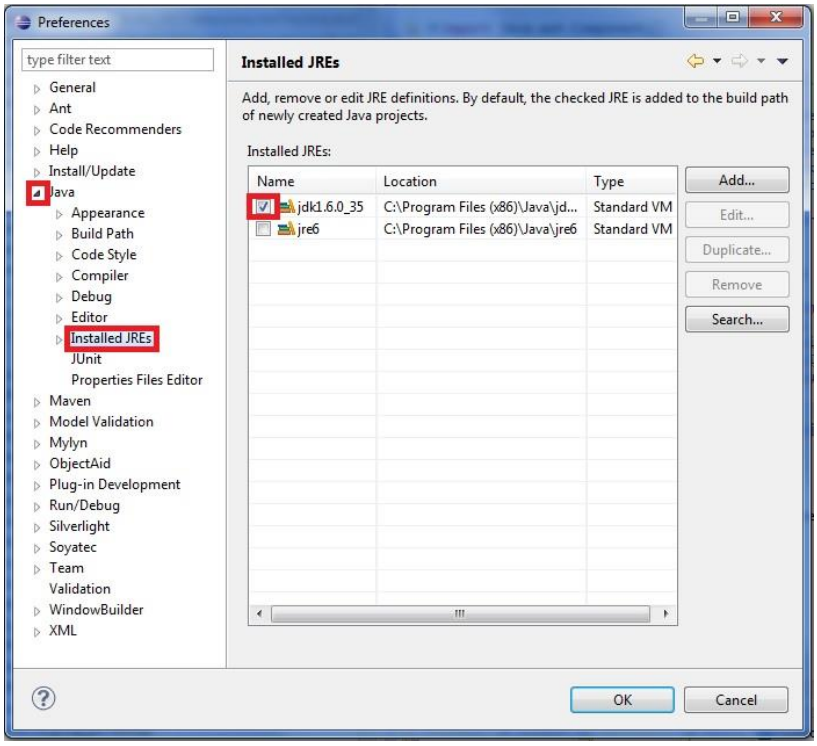

Figura 57 Selección de la máquina virtual Java en Eclipse

Nota: es imprescindible asegurarse de que la versión es JAVA 1.6 y que se instalan también las JAI y JAI IMAGE I/O sobre JDK 1.6. (ver Figura 58)

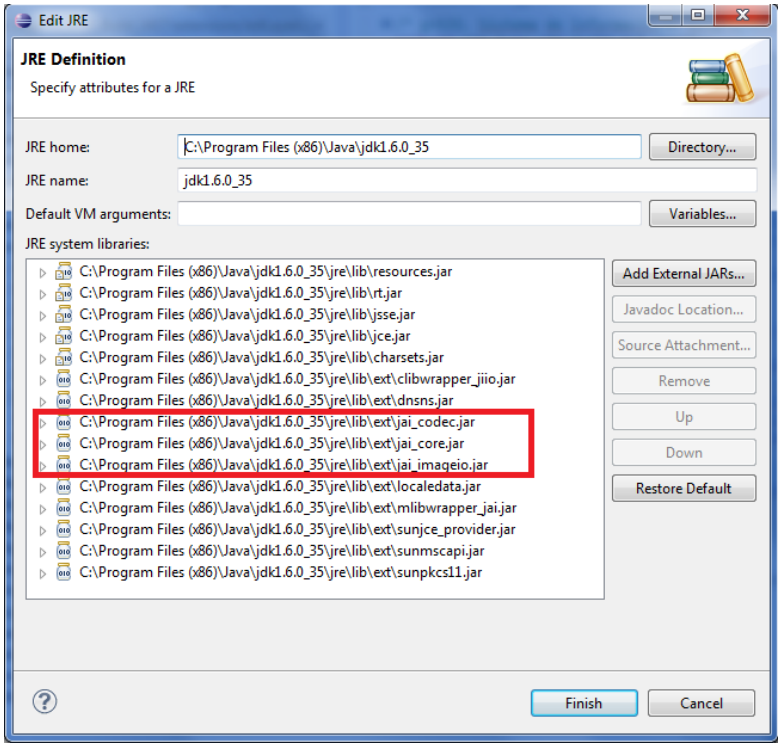

Figura 58 Comprobación de que están instaladas las JAI y JAI IMAGE I/O

Si no se han instalado estos 2 componentes gvSIG no va poder utilizar ciertos métodos con imágenes y según aparezca la interfaz de gvSIG 1.12 se va a cerrar,

impidiéndolo usar. Igual pasa si se ha descargado el JRE en vez de JDK, puede haber problemas a la hora de compilar ejecutando los build.xml porque haya llamadas que no reconozcan. Hay que tener cuidado en este paso e instalar todo y a la vez solo lo que se necesita.

F. 4. Conexión al repositorio SVN de gvSIG

Para conectarse al repositorio SVN de gvSIG hay que instalar en Eclipse el cliente Subclipse. Una vez instalado Subclipse, para poder descargarse un proyecto a local hay que ir al menú File » New Project y seleccionar "Checkout Projects from SVN" (ver Figura 59). En la siguiente ventana seleccionando la opción "Create a new repository location" (ver Figura 60) aparecerá un recuadro de texto para introducir la URL del repositorio, ahí se introducirá https://devel.gvsig.org/svn/gvsig-desktop esta es la dirección oficial del repositorio de gvSIG, existen otros repositorios externos que contienen otras extensiones como se verá a continuación.

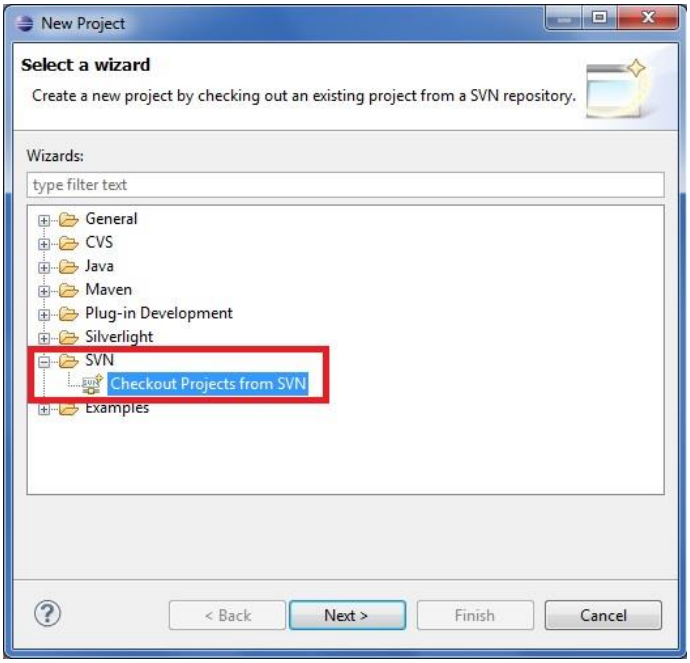

Figura 59 Conexión al repositorio

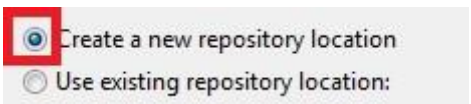

Figura 60 Opción de creación de un nuevo repositorio

F. 5. Estructura del repositorio SVN de gvSIG

Este repositorio tiene la típica estructura trunk/branches/tags. Teniendo, como se dijo anteriormente las versiones inestables o en desarrollo en el directorio trunk y las versiones estables en el directorio tags, en el directorio branches hay sobre todo copias de determinados proyectos de gvSIG.

Hay que expandir tags y localizar el último tag correspondiente a la versión con la que se va a trabajar (1.12). En este caso hay que expandir el v1_12_0_Build_1417. (ver Figura 61)

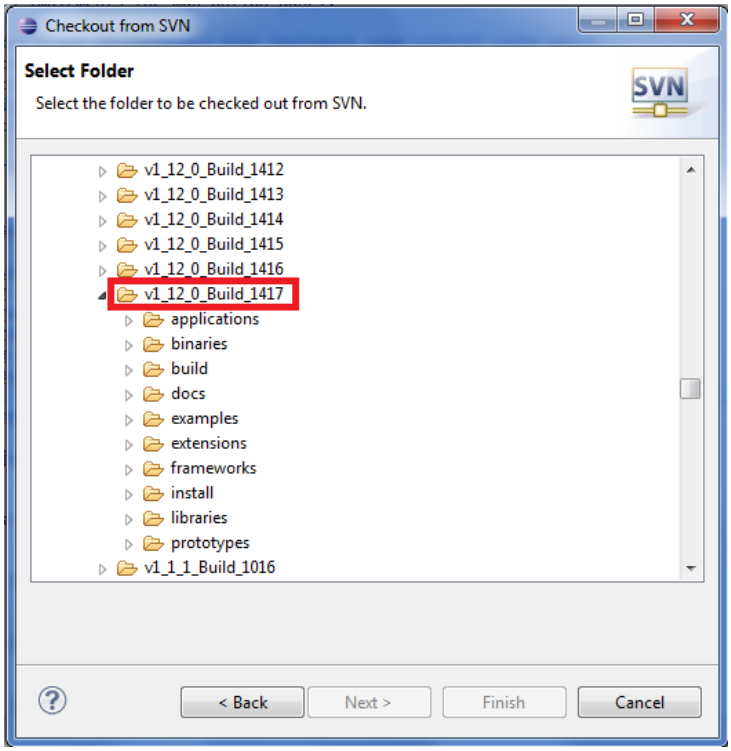

Figura 61 Tag correspondiente a gvSIG 1.12

Los directorios tag almacenan muchos proyectos de Eclipse. Los directorios de los que se obtendrán los proyectos son:

• applications (aplicaciones que funcionan sobre Andami)

• binaries (archivos .dll o .so)

• extensions (todas las extensiones de Andami)

• frameworks (Andami)

• libraries (todas las bibliotecas usadas por las extensiones, Andami yappgvSIG)

En la carpeta Install hay un fichero llamado gvsig_default_installation_projects.xml. Este indica que proyectos fueron empaquetados en la versión 1.12. Esa lista la usa el fichero de ant deploy.xml que está en install para crear los instalables.

Como se puede ver en el fichero, hay una variable que fija los que están en el repositorio principal y otra que fija los que están en otro. Los repositorios del resto de proyectos son:

* org.gvsig.consecutivenumber
Redmine: https://devel.gvsig.org/redmine/projects/gvsig-consecutive-numbers
svn: https://devel.gvsig.org/svn/gvsig-consecutive-numbers

* org.gvsig.chartlegend
Redmine: https://devel.gvsig.org/redmine/projects/gvsig-graphlegend
svn: https://devel.gvsig.org/svn/gvsig-graphlegend

* org.gvsig.copypastegeom
Redmine: https://devel.gvsig.org/redmine/projects/gvsig-copy-paste-geometries
svn: https://devel.gvsig.org/svn/gvsig-copy-paste-geometries

* org.gvsig.selectduplicates,
Redmine: https://devel.gvsig.org/redmine/projects/gvsig-select-duplicates
svn: https://devel.gvsig.org/svn/gvsig-select-duplicates

* org.gvsig.newgeoprocess,
Este está en join up
https://joinup.ec.europa.eu/software/gvsig-geoproces/description

* extNavTable
Repositorio: https://github.com/navtable/navtable

Sextante también va incluido pero no está en ese fichero porque recibe un trato especial: http://sextante.googlecode.com

F. 6. Descargar proyectos para ejecutar gvSIG 1.12

Lo primero de todo es desactivar la propiedad de Eclipse Build Automatically para descargar primero todo el código y luego hacer un build de él.

Si se sigue la mini-guía de gvSIG para descargar los proyectos mínimos para su funcionamiento citan los siguientes: _fwAndami, appgvSIG, binaries, libCorePlugin, libExceptions y libFMap, pero si se bajan únicamente estos cuando se compilaba con build-all daba error al existir dependencias sobre otros proyectos. Esto es así porque la guía estaba obsoleta. Al querer tener gvSIG con las mismas extensiones que la versión de distribución se descargaron las extensiones que indicaba el fichero gvsig_default_installation_projects.xml anteriormente comentado. Cabe destacar que en el repositorio hay más extensiones que las que indica, muchas de ellas obsoletas, otras a medio terminar y varias están duplicadas por lo que si se descargan todos los ficheros del repositorio gvSIG no funcionará al producirse dependencias con proyectos viejos y otros errores.

La metodología que se sigue para bajar cada proyecto es la misma así que a continuación se explica únicamente para el primer proyecto _fwAndami:

Hay que ir a la carpeta frameworks, hacer click derecho sobre el directorio _fwAndami y seleccionar la opción Checkout... y en la siguiente pantalla seleccionar Finish y esperar pacientemente a que finalice la descarga.

F. 7. Compilar los proyectos y solucionar errores

Una vez se hayan descargado todos los proyectos, se habilita la opción Build Automatically y se esperar a que se haga un build de todos los proyectos.

Tras esto, se compilan los proyectos a través de las tareas ant que contienen los diferentes proyectos. Si se quiere compilar toda la aplicación con una sola tarea ant, hay que ejecutar el fichero build.xml del proyecto appgvSIG.

Tras haber realizado todos los pasos explicados anteriormente, apareció aun así un error en la compilación:

```
Buildfile: D:\gvSIGDefinitivo\org.gvsig.newgeoprocess\build.xml
init:
     [echo]  Compiling geoprocessextension...
create-jar:
copy-data-files:

BUILD FAILED
D:\gvSIGDefinitivo\org.gvsig.newgeoprocess\build.xml:59:
D:\gvSIGDefinitivo\org.gvsig.newgeoprocess\lib does not exist.

Total time: 648 milliseconds
```

Para solucionarlo se modificó el build.xml de org.gvsig.newgeoprocess y comentado las siguientes líneas, ya que no existe ninguna carpeta lib en el proyecto:

```
<!--<copy todir="${dist}/lib">
     <fileset dir="lib/" includes="*"/>
</copy> -->
```

F. 8. Ejecutar gvSIG 1.12 en Eclipse

Una vez compilado únicamente queda ejecutarlo, para ello se necesita un archivo launcher que contiene una configuración correcta para que se pueda ejecutar gvSIG en Eclipse y bajo un Sistema Operativo Windows.

En caso de tener Eclipse abierto únicamente hay que reiniciarlo para que lo reconozca. Solo queda ejecutarlo desde Run y ya se puede abrir gvSIG 1.12. (ver Figura 62)

Figura 62 Run en Eclipse

F. 9. Anatomía de una extensión

Una vez se tiene el código de gvSIG con todas sus extensiones, es el momento de empezar el desarrollo de una nueva. Las extensiones son proyectos Java, un proyecto Java puede contener más de una extensión. Las partes fundamentales de un proyecto para que sea considerada una extensión son un fichero de configuración en XML y el código Java.

F.9.1. Fichero de configuración

El fichero de configuración llamado config.xml proporciona a gvSIG la información necesaria para que pueda cargar e integrar la extensión o extensiones en la interfaz gráfica. Define los elementos del interfaz de usuario como son las barras de herramientas y menús a la vez que los asocia con clases del código. También indica las diferentes extensiones que están implementadas en el proyecto.

Lo siguiente es un fichero de configuración de ejemplo, en el que únicamente hay una extensión:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plugin-config>
	<depends plugin-name="com.iver.cit.gvsig" />
	<libraries library-dir="."/>
	<extensions>
		<extension class-name=
			"edu.uoc.postgradosig. construcciones.Construcciones"
			description="Complemento que dibuja construcciones"
			active="true"
			priority="50">
			<menu text="Postgrado SIG/Construcciones"
			<action-command="MENU_CONSTRUCCIONES" />
			<tool-bar name="Postgrado SIG" position="2">
				<action-tool icon="seleccion.png"
				tooltip="Localizador" position="1"
				action-command="BOTON_LOCALIZADOR"/>
			</tool-bar>
		</extension>
	</extensions>
</plugin-config>
```

A continuación, se describe en detalle cada una de las etiquetas permitidas (ver Figura 63). No obstante, dentro de la distribución del código fuente de gvSIG, existe un fichero llamado *plugin-config.xsd*, situado en *_fwAndami/schemas*, donde se describen formalmente todas las etiquetas permitidas y el tipo de valores que aceptan.

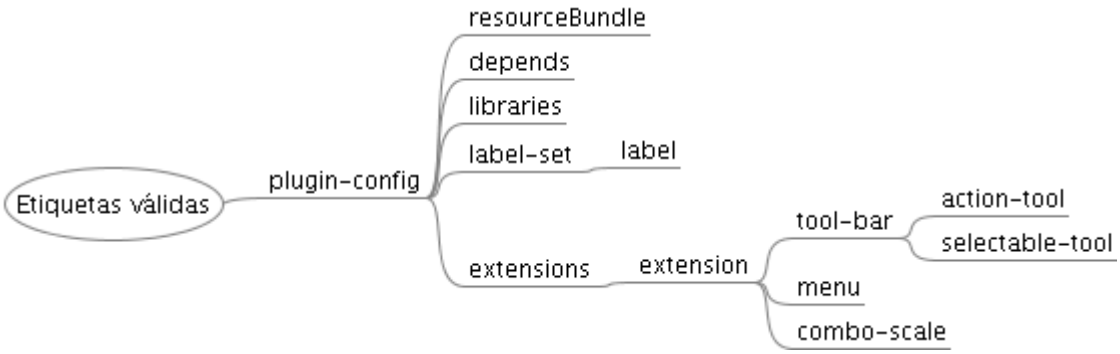

Figura 63 Jerarquía de etiquetas válidas en el fichero de configuración

La primera línea, describe la versión del lenguaje XML y su codificación. Después aparece la etiqueta <*plugin-config*> la cual marca el inicio y el fin del fichero de configuración.

- La etiqueta <*resourceBundle*> indica el nombre base que tendrán los ficheros de traducción. Por ejemplo, en el caso de que <*resourceBundle name="text"/*>, los ficheros de traducciones se llamarán *text.properties*,*text_en.properties*, etc.

- La etiqueta <*depends*> recoge todas las dependencias del nuevo complemento, a nivel de librerías, para que funciones correctamente, una extensión depende de otra cuando el código de la primera utiliza clases de la segunda.

- La etiqueta <*libraries*> especifica el directorio donde se encuentra el código del complemento.

- La etiqueta <*label-set*> indica el comienzo de un grupo de etiquetas para la barra de estado. Puede tener asociado el atributo *class-name* que indica la clase asociada al *label-set*. Estas etiquetas pueden contener la etiqueta <*label*>, la cual instala una caja de texto no editable en la barra de estado y posee los atributos *id* que indica el identificador de la etiqueta y el atributo *size* que almacena la anchura de la etiqueta en píxeles.

- Por último la etiqueta <*extensions*> engloba la lista de extensiones del proyecto.

Cada extensión está definida dentro de las etiquetas <*extensión*> y tienen los siguientes atributos:

- *class-name*: especifica el nombre de la clase que implementa la extensión.

- *description*: describe la funcionalidad que aporta la extensión. Esta etiqueta es únicamente informativa.

- *active*: indica si la extensión está activa. Dependiendo de si su valor es true o false gvSIG cargará o no la extensión.

- *priority*: establece la prioridad de carga de la extensión respecto al resto de extensiones declaradas en el fichero de configuración.

Si se quiere que la extensión aparezca en un menú, se cree una barra de herramientas para ella o simplemente se añada un botón a una barra de herramientas hay que utilizar las etiquetas <*menu*> o <*tool-bar*>.

La etiqueta <*menu*> hace referencia a los menús de gvSIG y está formada por los siguientes atributos:

- *action-command* especifica el identificador de la acción, permitiendo asociar varios menús a una sola clase Java.

- *text* establece la ubicación del menú, usando la barra "/" se pueden establecer diferentes niveles de menú.

- *position* indica la posición de la entrada de menú dentro del submenú en el que esté situada.

- *icon* icono que se mostrará dentro del botón.

- *tooltip* especifica el texto de ayuda que describe la herramienta al colocar el cursor encima del botón.

- *enable-Text* texto informativo de las condiciones que deben producirse para que el botón esté activo.

- *key* indica la combinación de teclas necesarias para lanzar este menú.

- *mmemonic* es la tecla que activará esta entrada de menú cuando el menú esté desplegado y tenga el foco.

- *is_separator* atributo que indica que deseamos añadir un separador en esta posición de la barra de menús.

La etiqueta <*tool-bar*> define una barra de herramientas y está formado por dos atributos obligatorios y uno opcional: *name* el cual especifica el nombre de la barra, *position* que indica la posición que ocupa la barra de herramientas entre las demás y por último los atributos opcionales *is-visible*, *que* determina si la *tool-bar* estará inicialmente visible u oculta si este atributo no aparece el valor por defecto es true, y *group*, que indica el grupo de *selectable-tools* al que pertenece y en el que solo puede estar seleccionado uno de los botones a la vez.

Para insertar un botón dentro de una barra de herramientas hay que introducir el elemento <*action-tool*> dentro de <*tool-bar*>. <*action-tool*> está compuesto por ocho atributos:

- *action-command* el cual tiene la misma funcionalidad que el atributo con el mismo nombre de la <*tool-bar*>
- *name* indica el nombre del botón.
- *position* señala la posición del botón dentro de la barra de herramientas.
- *icon* almacena la ruta a la imagen que usará el botón.
- *text* este atributo opcional almacena el texto a mostrar junto al icono.
- *tooltip* especifica el texto de ayuda que describe la herramienta al colocar el cursor encima del botón.
- *enable-text* texto informativo de las condiciones que deben producirse para que el botón esté activo.
- *last* deja un espacio extra a la derecha del icono. Se suele utilizar en el último icono de una barra de herramientas.

Si lo que se quiere introducir en la barra de herramientas es un botón que pueda estar pulsado o no hay que utilizar la etiqueta <*selectable-tool*>. Cuando en una barra de herramientas un *selectable-tool* está pulsado, los demás no lo están, es decir es restrictivo. Esta etiqueta tiene los mismos atributos que *action-tool*, las únicas diferencias residen en el atributo *action-command* que en este caso indica la herramienta seleccionada y en la inclusión del atributo *is-default* que determina qué herramienta está seleccionada inicialmente.

F.9.2. Código Java

Para que gvSIG pueda reconocer y cargar la extensión, la clase principal que se indica en el fichero de configuración debe extender a la clase *Extension* de gvSIG y sobrecargar sus funciones. En realidad la clase Java que extiende a *Extension* es la que actúa a modo de interfaz ente gvSIG y la extensión.

La clase *Extension* está compuesta por las siguientes funciones:

- void initialize(). Es la función que inicializa la extensión. Muchas veces está vacía o simplemente inicializa variables.
- boolean isEnabled(). Indica si la extensión está accesible en un momento dado.
- boolean isVisible(). Señala si los elementos de interfaz asociados a la extensión son visibles en un momento dado. gvSIG llama periódicamente a esta función respondiendo a los eventos de la interfaz gráfica. Es útil para las extensiones que solo deben estar activadas en situaciones específicas, como pueden ser herramientas de edición solo si la capa está en edición.
- void execute(String actionCommand). Es la función que se ejecuta cuando el usuario ha pulsado sobre alguno de los elementos del interfaz definidos para esta extensión.
- void postInitialize(). Se ejecuta después de haberse llamado a la función *initialize* de todas las extensiones. Es utilizada para realizar tareas de inicialización que necesiten interactuar con otras extensiones.
- void terminate(). Se la función que se llama al terminar la ejecución de gvSIG.

F. 10. Creación de una extensión con Eclipse

Una vez conocidas las partes de las que se compone un proyecto Java que implementa una o más extensiones se pasa a explicar paso a paso la creación en Eclipse de una.

- Primero se crea un proyecto nuevo en Eclipse.

- En este proyecto se crean las carpetas src, config e images. En src donde irá el código Java. En config irá el fichero de configuración y ficheros de configuración auxiliares si son necesarios. Y en la carpeta images se almacenaran las imágenes que utilice la extensión.

- Si el nuevo proyecto depende de librerías u otras extensiones es necesario indicar en Eclipse de cuáles depende. Esto se hace pinchando con el segundo botón en el proyecto y seleccionando *Properties*. Aparecerá una ventana donde habrá que seleccionar *Java Build Path* y añadir en la pestaña *Projects* los proyectos de los que depende.

- Para compilar el proyecto es necesario crear una tarea Ant con un fichero XML llamado build.xml. Un fichero de ejemplo útil para comenzar es el siguiente:

```
<project name="Generar extension en Andami" default="generate-without-source"
basedir=".">
   <description>
      Instala el plugin de ejemplo en Andami.
   </description>
  <!-- set global properties for this build -->
  <property name="src" location="src"/>
  <property name="build" location="bin"/>
  <property name="dist"  location="dist"/>
  <property name="plugin" value="com.iver.ejemplo"/>
  <property name="extension-dir" location="../_fwAndami/gvSIG/extensiones"/>

  <target name="init">
   <!-- Create the time stamp -->
   <tstamp/>
   <!-- Create the build directory structure used by compile -->
   <mkdir dir="${build}"/>
   <mkdir dir="${dist}"/>
         <!-- Creamos un fichero con el timeStamp para que lo lea el FPanelAbout -->
         <buildnumber/>
  </target>
  <target name="generate-without-source" description="generate the distribution without
the source file" >
   <!-- Create the distribution directory -->
   <mkdir dir="${dist}"/>
   <!-- Put everything in ${build} into the MyProject-${DSTAMP}.jar file -->
   <jar jarfile="${dist}/${plugin}.jar" basedir="${build}"/>
   <copy file="config/config.xml" todir="${dist}"/>
   <copy file="config/about.htm" todir="${dist}"/>
   <copy todir="${dist}">
         <fileset dir="." includes="text*.properties"/>
   </copy>
   <copy todir="${dist}/images">
         <fileset dir="images/" includes="*"/>
   </copy>
   <move todir="${extension-dir}/${plugin}/">
         <fileset dir="${dist}" includes="**/**"/>
   </move>
  </target>
</project>
```

ANEXO G

MANUAL DEL ADMINISTRADOR

Se ha redactado el siguiente manual con el objetivo de documentar y facilitar al administrador sus tareas de configuración de la nueva infraestructura desarrollada así como de las herramientas que la componen. De este modo se detalla a continuación cómo configurar gvSIG para habilitar únicamente las extensiones y barras de herramientas deseadas, evitando así una sobrecarga de información.

G. 1. Configuración a través del interfaz de gvSIG

gvSIG a través de su interfaz permite habilitar y deshabilitar extensiones. En el menú Ventana aparece la opción Preferencias (ver Figura 64).

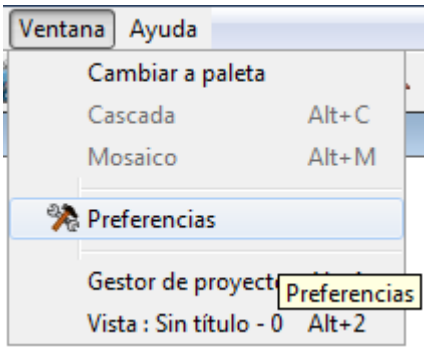

Figura 64 Opción Preferencias en el menú Ventana

Pulsando sobre esta opción se abre una ventana que permite configurar diferentes aspectos de gvSIG. Para la habilitación de extensiones hay que ir al árbol que aparece a la izquierda y pinchar sobre General/Extensiones, de este modo se desplegarán todas las extensiones que contiene gvSIG. Seleccionando la extensión deseada cargará su información en la parte derecha de la ventana junto con un checkbox que indicará si está activada o no. Cambiando la selección de este checkbox es como se habilita o deshabilita la extensión (ver Figura 65). Cabe destacar que los cambios serán visibles una vez se reinicie gvSIG.

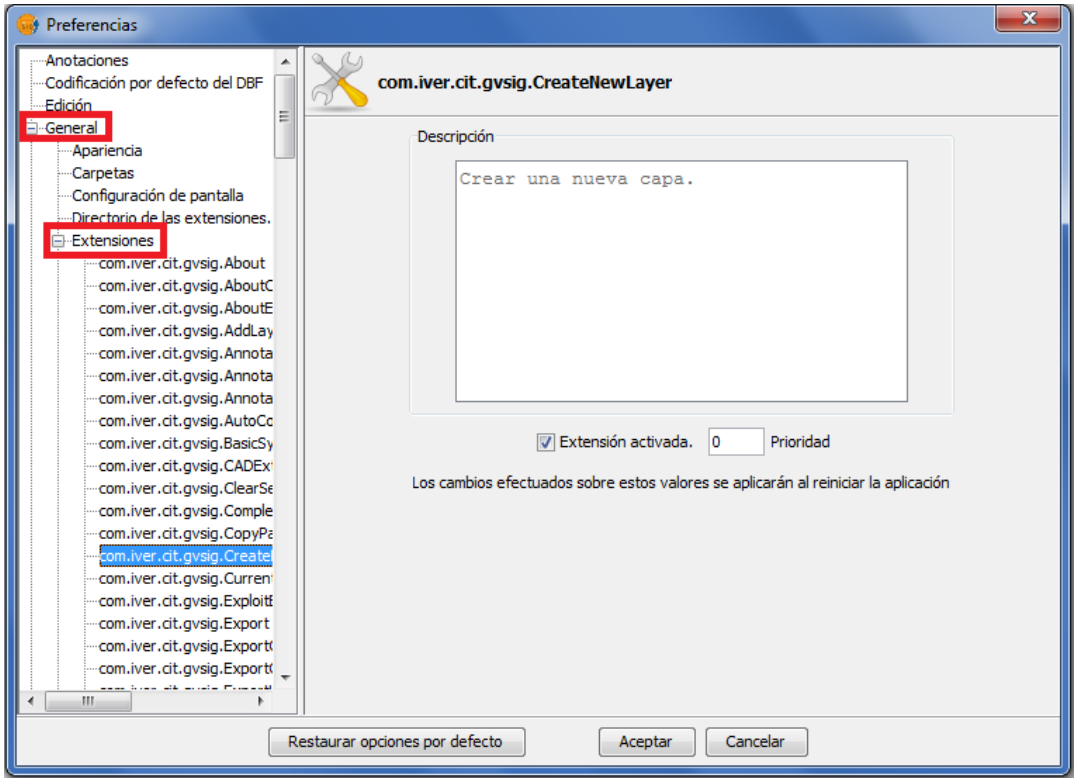

Figura 65 Ventana de Preferencias

Principalmente lo que hace esta ventana es atacar al fichero de configuraciones de cada extensión. La limitación que presenta hacerlo a través del interfaz es que no se pueden configurar las barras de herramientas, es decir tal vez se desee dejar la extensión habilitada pero que no aparezca un acceso en la barra de herramientas para simplificar la interfaz. Es por ello que está la segunda opción de configuración, a través de los ficheros de configuración.

También a tavés del interfaz se pueden ocultar o mostrar barras de herramientas en la opción del menú Ver/Barra de herramientas, los cambios realizados solo perduran durante la ejecución del programa, es decir la próxima vez que se cargue el programa volverán a aparecer todas las barras como antes. Para que desde un principio cargue únicamente las barras deseadas hay que hacerlo a través de los ficheros de configuración, como se explica en el siguiente apartado.

G. 2. Configuración a través de ficheros de configuración

Las extensiones de gvSIG están dotadas de un fichero XML de configuración llamado config.xml. Este fichero proporciona a gvSIG la información necesaria para que pueda cargar e integrar la extensión en la interfaz gráfica. Define los elementos del interfaz de usuario como son las barras de herramientas con la etiqueta <*tool-bar*> compuestas por botones definidos con <*action-tool*> *o* <*selectable-tool*> y los menús con <*menu*> a la vez que se asocian estos elementos gráficos con clases del código de la extensión.

Las barras de herramientas, <*tool-bar*> se componen de dos atributos obligatorios y uno opcional:

- *name* especifica el nombre de la barra.
- *position* indica la posición que ocupa la barra de herramientas entre las demás.
- *is-visible* (opcional) determina si la *tool-bar* estará inicialmente visible u oculta si este atributo no aparece el valor por defecto es true.

Para insertar un botón dentro de una barra de herramientas hay que introducir en el <*tool-bar*> el elemento <*action-tool*> si es un botón normal o <*selectable-tool*> *si* es un botón excluyente, es decir en la barra de herramientas solo puede estar pulsado uno de ellos. <*action-tool*> está compuesto por ocho atributos:

- *action-command* el cual tiene la misma funcionalidad que el atributo con el mismo nombre de la <*tool-bar*>
- *name* indica el nombre del botón.
- *position* señala la posición del botón dentro de la barra de herramientas.
- *icon* almacena la ruta a la imagen que usará el botón.
- *text* este atributo opcional almacena el texto a mostrar junto al icono.
- *tooltip* especifica el texto de ayuda que describe la herramienta al colocar el cursor encima del botón.

- *enable-text* texto informativo de las condiciones que deben producirse para que el botón esté activo.

- *last* deja un espacio extra a la derecha del icono. Se suele utilizar en el último icono de una barra de herramientas.

<*selectable-tool*> tiene los mismos atributos que <*action-tool*>, las únicas diferencias residen en el atributo *action-command* que en este caso indica la herramienta seleccionada y en la inclusión de *is-default*, que determina qué herramienta está seleccionada inicialmente, y *group*, que indica el grupo de *selectable-tools* al que pertenece y en el que solo puede estar seleccionado uno de los botones a la vez.

La etiqueta <*menu*> hace referencia a los menús de gvSIG y está formada por los siguientes atributos:

- *action-command* especifica el identificador de la acción, permitiendo asociar varios menús a una sola clase Java.

- *text* establece la ubicación del menú, usando la barra "/" se pueden establecer diferentes niveles de menú.

- *position* indica la posición de la entrada de menú dentro del submenú en el que esté situada.

- *icon* icono que se mostrará dentro del botón.

- *tooltip* especifica el texto de ayuda que describe la herramienta al colocar el cursor encima del botón.

- *enable-Text* texto informativo de las condiciones que deben producirse para que el botón esté activo.

- *key* indica la combinación de teclas necesarias para lanzar este menú.

- *mmemonic* es la tecla que activará esta entrada de menú cuando el menú esté desplegado y tenga el foco.

- *is_separator* atributo que indica que deseamos añadir un separador en esta posición de la barra de menús.

Conocidas las estructuras para las barras de herramientas y los menús y sabiendo además que cada extensión tiene en el fichero de configuración un atributo llamado *active* que indica si la extensión está habilitada o no su deshabilitación es inmediata. Si lo que se quiere es ocultar una barra de herramientas el atributo *is-visible* de los <*tool-bar*> lo permite, pudiendo recuperarla a través del interfaz de gvSIG en el menú Ver/Barra de herramientas. En el caso de eliminar un botón de una barra de herramientas hay que comentar o borrar dicho código enmarcado por <*action-tool*> o <*selectable-tool*>. Lo mismo ocurre con los menú, si se quiere que no aparezcan hay que comentar o borrar su código.

Siguiendo estos principios, se ha configurado la infraestructura deshabilitando extensiones cuya funcionalidad no era precisa en el entorno de ejecución, al igual que se han ocultado una gran cantidad de barras de herramientas simplificando la interfaz gráfica al usuario y evitando una sobreinformación al mismo.

G. 3. Configuración de Mis Capas

La herramienta Mis Capas también tiene una parte para el administrador. La herramienta Mis Capas permite crear accesos rápidos para cargar diferentes capas preconfiguradas de una manera rápida y sencilla abstrayendo el origen de la fuente de datos al usuario (ver Figura 66). Esta herramienta está principalmente pensada para que un administrador cree las capas preconfiguradas y las distribuya entre los usuarios del sistema con el objetivo de facilitar sus tareas. Además el usuario estándar también puede añadir sus propias capas. La principal diferencia entre las capas creadas por un administrador y un usuario es que las capas del administrador el usuario normal no puede eliminarlas mientras las que ha creado él para sí mismo sí.

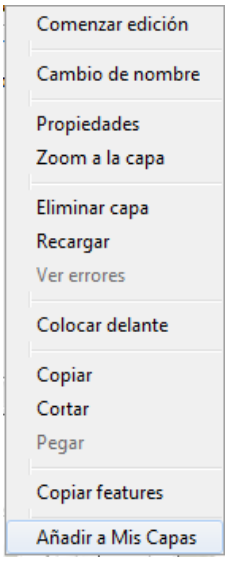

Figura 66 Menú de opciones sobre una capa

De este modo la herramienta de Mis Capas del administrador es diferente a la del usuario normal, ya que las capas que crea son capas admin. Otra manera que el administrador puede crear sus capas es utilizando la herramienta del usuario estándar y modificando el fichero creado, cambiando los dos primeros caracteres “US” por “AD”.

ANEXO H

MANUAL DE USUARIO

El presente manual ha sido elaborado para el usuario estándar del sistema, donde se explican la forma de uso de las nuevas herramientas. Para conocer el manejo del resto de herramientas de gvSIG 1.12 se puede acceder al manual de usuario en línea a través de la web de gvSIG[16].

H. 1. Gestor edición

Esta herramienta se activa pulsando sobre el icono  de la barra de herramientas. Aparece una barra en el lateral derecho con tres pestañas.

H.1.1. Pestaña Atributos

Permite editar los atributos asociados a una *feature*. Para poder modificar los valores hay que poner primero la capa en edición (ver Figura 67).

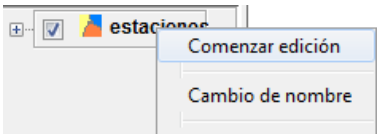

Figura 67 Comenzar edición

Una vez se haya terminado de modificar los valores y se quiera guardar los cambios hay que terminar la edición (ver Figura 68) e indicarlo en la ventana emergente que aparece (ver Figura 69).

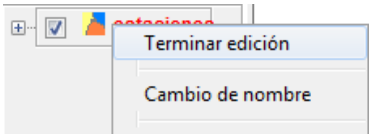

Figura 68 Terminar edición

[16] http://www.gvsig.org/web/projects/gvsig-desktop/docs/user/gvsig-desktop-1-12-manual-de-usuario

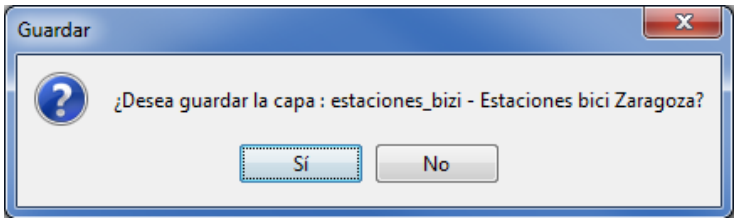

Figura 69 Ventana emergente para guardar la capa

Esta pestaña también contiene opciones para facilitar y agilizar la edición, estas se encuentran en el panel de botones de la parte inferior (ver Figura 70).

Figura 70 Panel de botones de la pestaña Atributos

El primer botón corresponde a la asignación de un Alias a los nombres de los atributos a través de una ventana emergente (ver Figura 71). De este modo si el nombre que se le había asignado a un atributo no es lo suficientemente descriptivo debido a restricciones de longitud de la fuente de datos origen se le puede cambiar en esta pestaña por otro sin restricciones.

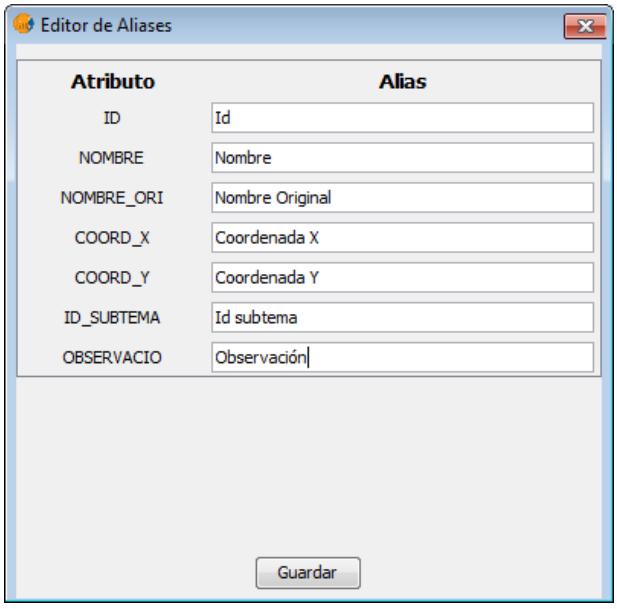

Figura 71 Ventana de edición de Alias

El segundo botón abre una ventana que permite realizar búsquedas de *features* seleccionando el atributo por el que buscar (ver Figura 72).

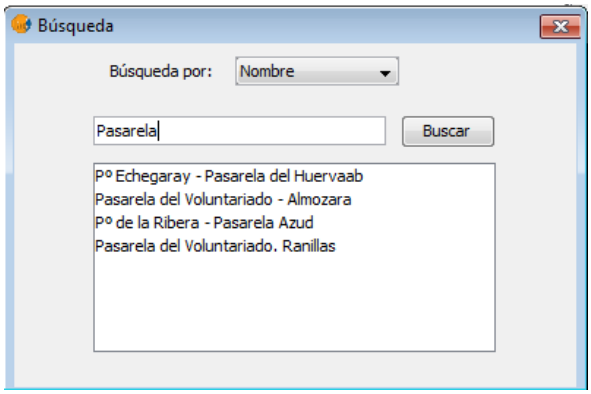

Figura 72 Ventana de búsqueda

El tercer botón sirve para centrar el mapa dejando la *feature* que se está editando en el centro del mismo.

El cuarto botón además de centrar el mapa en la *feature* realiza un zoom sobre ella.

El quinto botón permite copiar la información de una *feature* para poder pegarla sobre otra.

El sexto botón es el pegado de la información copiada de una *feature*.

El séptimo botón elimina la *feature*.

La segunda línea de botones está destinada a los botones de navegación entre *features*, de este modo las flechas de ambos laterales permiten ir a la primer y última *feature*, mientras las flechas centrales permiten retroceder o avanzar una *feature*. Por último la caja blanca que indica en que *feature* se está se puede introducir directamente el número de la *feature* a la que se quiere ir.

H.1.2. Pestaña Edición de Simbología

Esta pestaña está diseñada para editar el aspecto gráfico de la capa.

Si es una capa de puntos, la pestaña muestra la opción de cambiar el color del punto o asociarle una imagen. También el tamaño del punto y su ángulo de inclinación, esto último tiene sentido cuando es una imagen (ver Figura 73).

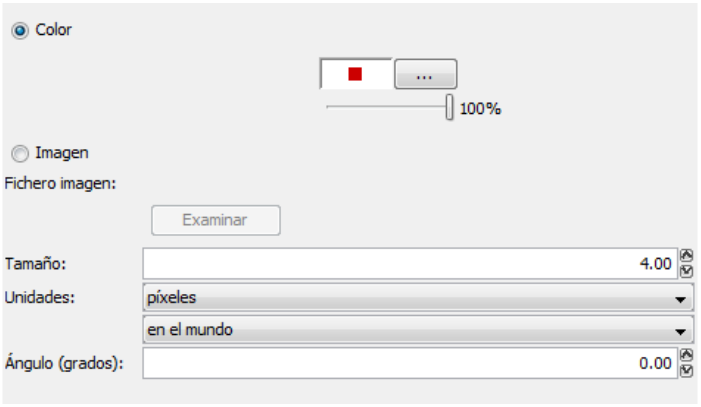

Figura 73 Panel de edición de la simbología de una capa de puntos

Si la capa es de líneas, muestra las opciones de cambiar el color y el ancho del trazado (ver Figura 74).

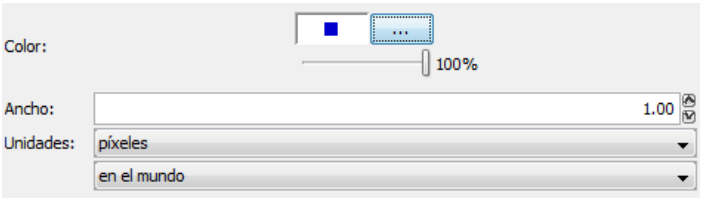

Figura 74 Panel de edición de la simbología de una capa de líneas

Si la capa es contiene polígonos, se puede modificar desde el interfaz el color de relleno y la anchura del borde (ver Figura 75).

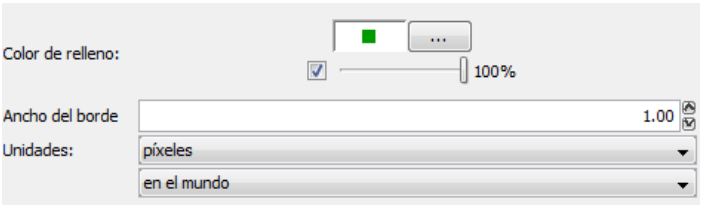

Figura 75 Panel de edición de la simbología de una capa de polígonos

Abajo en la pestaña hay dos botones, Propiedades y Aplicar. El botón Propiedades abre un diálogo para una edición más compleja de la simbología mientras el de Aplicar aplica los cambios realizados en la capa.

H.1.3. Pestaña Búsqueda vías

Esta tercera pestaña permite buscar direcciones postales para centrar el mapa en ellas (ver Figura 76). Hay un cuadro de búsqueda donde se introduce la dirección y en el parte de abajo aparecen los resultados posibles. Pinchando sobre estos resultados el mapa se centra en ellos.

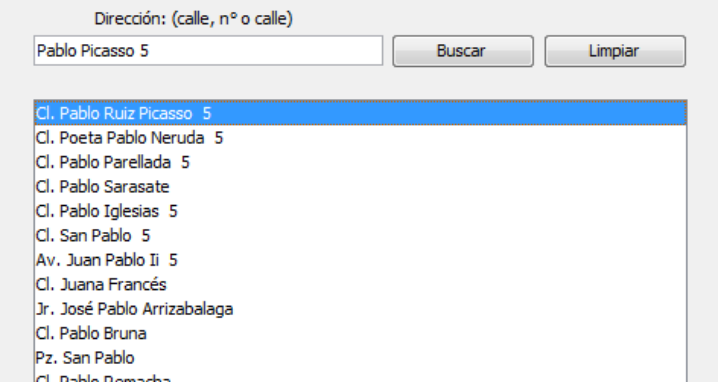

Figura 76 Panel de búsqueda de vías

H. 2. Overview Map

Esta herramienta se activa pulsando sobre el icono  . El mapa en miniatura aparece en la barra lateral izquierda, debajo de las capas cargadas.

Esta miniatura (ver Figura 77) enmarca en un recuadro rojo la parte del mapa que está siendo visualizado en pantalla. Para navegar por él se puede o generar un recuadro con el primer botón del ratón que será la extensión que se debe mostrar por pantalla o pinchar con el segundo botón en la zona a la que se quiere trasladar el recuadro existente, de este modo se continuará al mismo nivel de zoom.

Figura 77 Mapa en miniatura

H. 3. Mis Capas

Para añadir una capa a la lista de Mis Capas hay que pinchar sobre la capa con el segundo botón y seleccionar Añadir a Mis Capas (ver Figura 78).

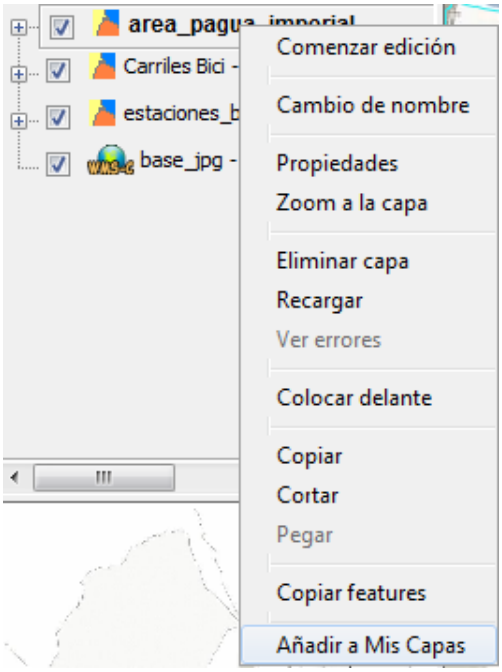

Figura 78 Menú de opciones sobre una capa

Hecho esto aparecerá una ventana (ver Figura 79) preguntando por un Alias al que se le asignará a la capa y por el que será después reconocida la capa en el interfaz de cargado de Mis Capas.

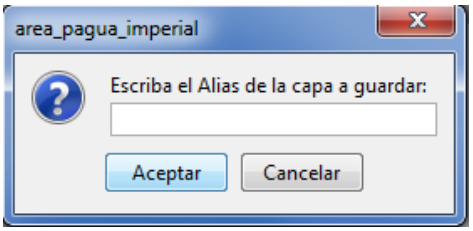

Figura 79 Ventana emergente para asociar un Alias a una capa

Importante: los nombres de los Alias son únicos, no se permiten dos capas con el mismo Alias.

Para cargar una capa de Mis Capas hay que ir a la pestaña Mis Capas (ver Figura 80) en la ventana de añadir capa y basta con seleccionar la capa y pulsar el botón Cargar. Si la capa tiene como creador Usuario también se podrá eliminar con el botón.

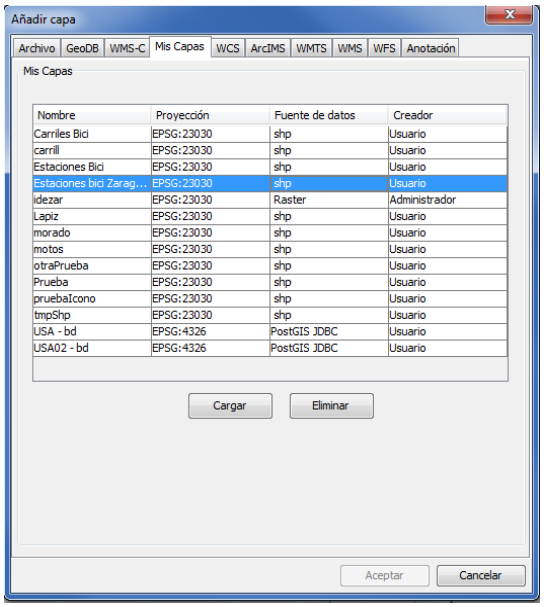

Figura 80 Ventana para cargar o eliminar una capa preconfigurada

H. 4. Servicio WMTS

Para cargar una capa de un servicio WMTS hay que ir a la pestaña WMTS (ver Figura 81) de la ventana de Añadir capa. Aparecerá una caja enmarcada en un recuadro que pone Servidor, es donde hay que introducir la URL del servicio. Para conectar se pulsa el botón con el mismo nombre. Una vez conectado aparecerá una descripción del servicio abajo y se habilitarán los botones Anterior y Siguiente.

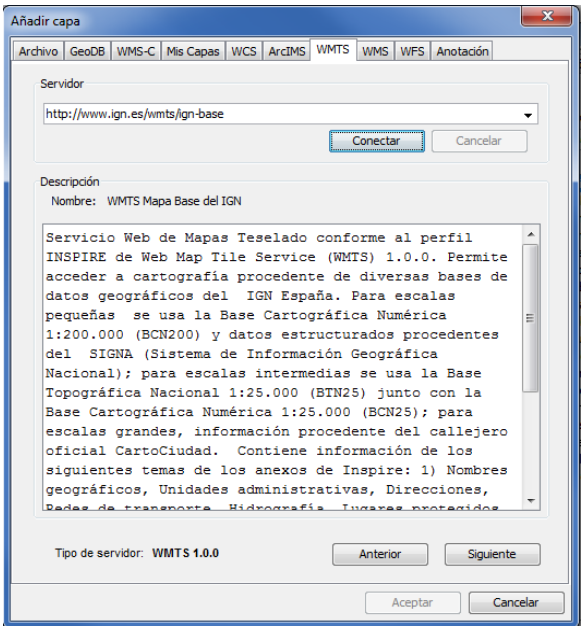

Figura 81 Pestaña de conexión a un servicio WMTS

Pulsando Siguiente se avanza a la primera pantalla de configuración (ver Figura 82), en la que únicamente se muestran las características del servicio.

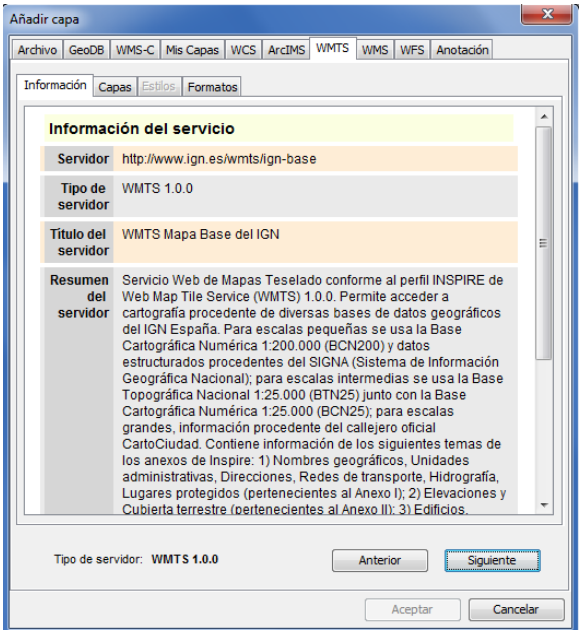

Figura 82 Pantalla que muestra la información del servicio WMTS

Avanzando de nuevo, se pasa a la pantalla donde se selecciona la capa del servicio que se quiere cargar (ver Figura 83).

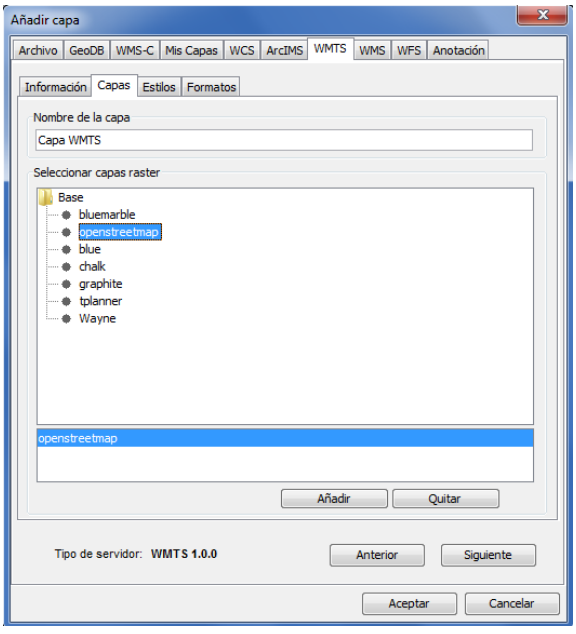

Figura 83 Pantalla que muestra las capas disponibles en el servicio

Después viene la pantalla para elegir el estilo (ver Figura 84), si la capa seleccionada los tiene.

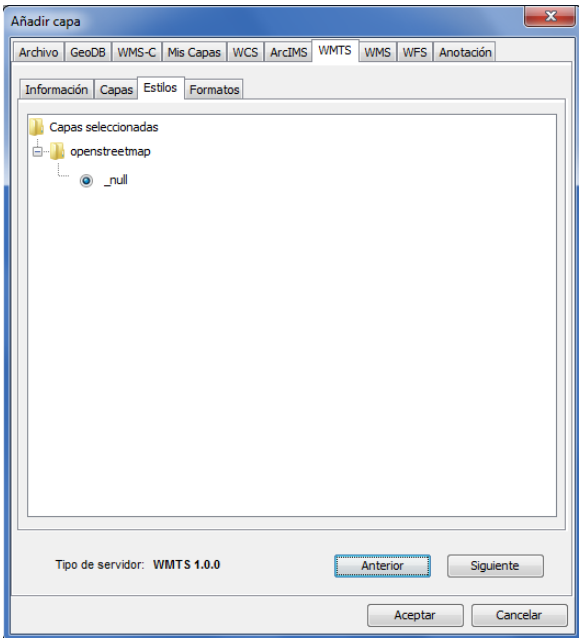

Figura 84 Pantalla que muestra los Estilos disponibles para una capa

Por último, la pantalla para seleccionar los formatos, SRS y formatos de texto (ver Figura 85).

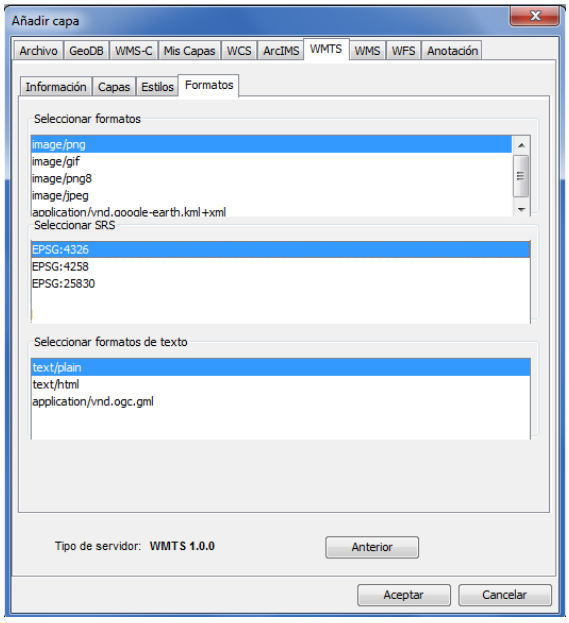

Figura 85 Pantalla para la selección de los formatos

Una vez seleccionados todos los parámetros deseados se pulsa el botón Aceptar y se carga la capa en gvSIG.

H. 5. Servicio WMS-C

Para cargar una capa de un servicio WMS-C es la misma mecánica que para el anterior servicio, cambiando que hay que ir a la pestaña WMS-C de la ventana de Añadir capa.

H. 6. Exportación en GeoJSON

Para exportar en *GeoJSON* una capa vectorial, esta tiene que estar seleccionada. Está opción se encuentra en el menú Capa/Exportar a…/GeoJSON (ver Figura 86)

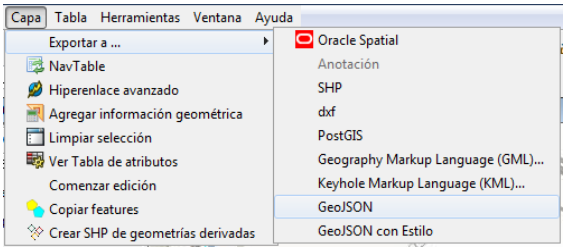

Figura 86 Opción para exportar a GeoJSON

H. 7. Exportación en GeoJSON con estilo

Para exportar en *GeoJSON* una capa vectorial con estilo, la capa tiene que estar seleccionada. Está opción se encuentra en el menú Capa/Exportar a…/GeoJSON con Estilo (ver Figura 87)

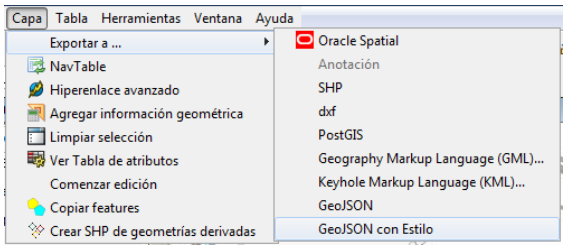

Figura 87 Opción para exportar a GeoJSON con estilo

Aparecerá una ventana para seleccionar la ubicación donde se guardará y seguidamente aparecerá el siguiente formulario (ver Figura 88) el cual hay que rellenar con los valores deseados.

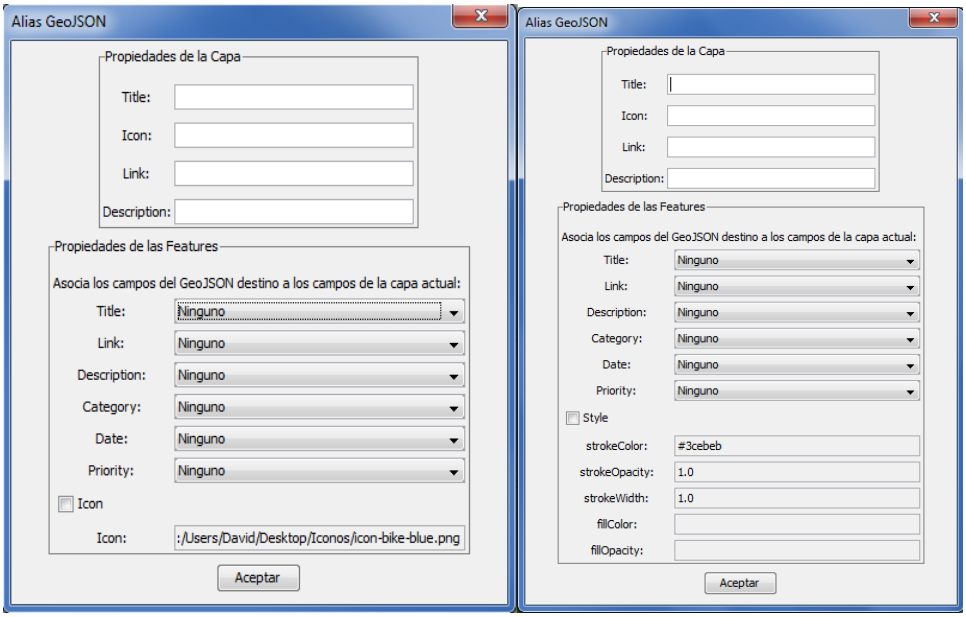

Figura 88 Ventanas para exportar GeoJSON con estilo (Izq. capa de puntos, Der. Capa de líneas)

La ventana de exportación está dividida en dos partes, en la parte superior aparecen las propiedades de la capa. En la parte inferior aparecen todos los atributos que pueden poseer una *feature*, en los *GeoJSON* con estilo estos atributos solo pueden ser *Title, Link, Description, Category, Date, Priority*, pudiendo no tener ninguno, estos atributos hay que mapearlos con los atributos que posee la capa para asociarles sus valores. Por último debe tener un atributo que indique el estilo. Si la capa a exportar es una capa de puntos el atributo asociado se llama *Icon* y almacena la ruta a la imagen. Si la capa es de líneas o polígonos tendrá el objeto Style el cual contiene los siguientes atributos; *strokeColor* (color del trazo), *strokeOpacity* (opacidad del trazo), *strokeWidth* (anchura del trazo), *fillColor* (color del relleno) y *fillOpacity* (opacidad del relleno). Estos dos últimos atributos solo tienen sentido en el caso de que la capa represente polígonos.

H. 8. Importación en GeoJSON

La importación de un fichero *GeoJSON* se hace a través del interfaz de Añadir Capa en la pestaña Archivo, del mismo modo que se cargan ficheros shapefile, DGN, KML…

III GLOSARIO, FIGURAS Y BIBLIOGRAFÍA

GLOSARIO

Bounding Box: habitualmente acortado a BBOX, es el área definida por dos longitudes y dos latitudes.

Capabilities: documento xml que contiene los metadatos de información de un servicio web de mapas.

CRS: *Coordinate Reference System*, en español sistema de referencia de coordenadas.

Driver: controlador que permite a un programa interactuar con archivos externos.

Feature: es un elemento básico de información geográfica que describe una entidad geográfica real o abstracta. Cada uno de estos elementos está compuesto de una serie de atributos que lo describen cuantitativamente y cualitativamente.

EPSG: *European Petroleum Survey Group*, era una organización científica con vínculos con la industria petrolera europea. Entre sus tareas se encontró la compilación y difusión del conjunto de parámetros geodésicos EPSG, la cual es una base de datos ampliamente usada que contiene sistemas de coordenadas, proyecciones geográficas, etc.

Geocoder: es un programa que permite encontrar las coordenadas geográficas asociadas a direcciones, códigos postales, puntos de interés…

GeoJSON: es un formato libre para codificar colecciones de features junto con sus atributos utilizando el formato JSON (*JavaScript Object Notation*).

GIF: es un formato de imagen cuyas siglas significan *Graphics Interchance Format*.

gvSIG: es un sistema de información geográfica formada por una aplicación de escritorio que permite capturar, almacenar, analizar y exportar cualquier tipo de información geográfica con el objetivo de solventar gestiones complejas y problemas de planificación.

IDE: son las siglas de Infraestructura de Datos Espaciales

IDEZar: es la Infraestructura de Datos Espaciales del Ayuntamiento de Zaragoza.

JPEG: estándar de comprensión y codificación de imágenes cuyas siglas significan *Joint Photographic Experts Group*.

OGC: *Open Geospatial Consortium*, es una organización internacional encargada de elaborar e implementar estándares abiertos de contenidos y servicios geospaciales.

Open Source: Código abierto, se refiere al software de libre redistribución, cuyo código fuente es accesible y se puede modificar dando lugar a obras derivadas.

OSGeo: *Open Source Geospatial Foundation*, es una organización no gubernamental sin ánimo de lucro cuya misión es la de promover el desarrollo colaborativo de tecnologías y datos geospaciales de libre uso.

Overview Map: vista en miniatura del mapa, la cual permite orientar al usuario mostrándole la localización que está viendo en el momento puesta en su contexto geográfico.

Plugin: o extensión, es un componente software que permite añadir una funcionalidad específica a una aplicación ya existente.

PNG: es un formato de imagen con compresión sin pérdida, cuyas siglas significan *Portable Network Graphics*.

Renderizar: proceso informático por el que se genera una imagen a partir de unos cálculos.

Shapefile: es un popular formato de datos vectoriales geospaciales desarrollado y regulado por ESRI como una especificación abierta para la interoperabilidad de datos entre SIG. Representan espacialmente: puntos, líneas y polígonos, cada uno de los cuales tiene atributos que lo describen.

SIG: *Sistema de Información Geográfica*.

SQL: es un lenguaje declarativo de acceso a bases de datos relacionales cuyas siglas significan *Structured Query Language*.

SRS: *Spatial Reference System*, siglas en ingles de Sistema de Referencia Espacial.

SRW: *Search/Retrieve Web Service*, es un servicio web para búsquedas y recuperación de información.

Tile: o tesela es una pequeña imagen parte de una imagen mayor.

TileMatrix: matrices de teselas.

TOC: es la abreviación de *Table of Contents* o tabla de contenidos. En gvSIG se la llama TOC a la barra que aparece a la izquierda de la vista y que contiene la jerarquía de las capas cargadas.

viewPort: es el recuadro que delimita el área visible de un mapa. Se define por las coordenadas geográficas de sus esquinas inferior izquierda y superior derecha.

WMS: *Web Map Service*, es un servicio que genera mapas de datos georreferenciados espacialmente de forma dinámica.

WMS-C: *Web Map Service-Caching*, es una recomendación formulada por OSGeo que permite realizar peticiones de teselas utilizando el estándar WMS.

WMTS: *Web Map Tile Service*, es un estándar de OGC que sirve teselas de mapa prerenderizadas

XML: *eXtensible Markup Language*, es un formato de codificación de documentos utilizado principalmente para el intercambio de información de una manera fácil.

ÍNDICE DE FIGURAS

BIBLIOGRAFÍA

[1] IAAA, Guía introducción a los SIG. (Guía interna del grupo)

[2] Razones por las que utilizar gvSIG
http://mappinggis.com/2012/08/22/cinco-razones-para-comenzar-a-usar-vsig/#lightbox/2/

[3] Colección de los mensajes de la lista de distribución para desarrolladores de gvSIG
http://osgeo-org.1560.x6.nabble.com/gvSIG-desarrolladores-f4163512.html

[4] Manual para desarrolladores de gvSIG
https://gvsig.org/web/docdev/manual-para-desarrolladores-gvsig

[5] Jeff de la Beaujardiere, OpenGIS® Web Map Server Implementation Specification

[6] Joan Masó, Keith Pomakis and Núria Julià, OpenGIS® Web Map Tile Service Implementation Standard

[7] Sintaxis de GDAL
http://www.gdal.org/ogr2ogr.html

[8] Sintaxis SQL en GDAL para añadir campos y valores
http://gis.stackexchange.com/questions/59705/gdal-sql-syntax-to-add-field-an-put-values

[9] El operador like en gvSIG
http://gvsig-argentina.org.ar/curso/SQL-El_operador_LIKE_en_gvSIG.pdf

[10] GeosLab, Manual de usuario del servicio SRW de IDEZar. (Guía interna de la empresa)

[11] Personalizar gvSIG
http://osgeo-org.1560.x6.nabble.com/Personalizar-menues-y-herramientas-de-gvSIG-td4996714.html

[12] Preguntas y respuestas sobre programación
http://stackoverflow.com

Nota: Las direcciones de Internet que aparecen en esta bibliografía están revisadas y accedidas a fecha 23/08/2013.