UNIVERSITAT OBERTA DE CATALUNYA

Ingeniería en Informática

Integración de JASPA en gvSIG.

Memoria de proyecto.

Alumno: Ildefonso Junquero Martín-Arroyo

Dirigido por: Jesús Manuel de Diego Alarcón

Profesor responsable: Antoni Pérez-Navarro

Resumen

Este documento describe los trabajos realizados por el alumno como proyecto fin de carrera del área de sistemas de información geográfica en la Universitat Oberta de Catalunya para la titulación de Ingeniería Informática.

En el documento se analizan conceptos básicos de los sistemas de información geográfica, para posteriormente centrarse principalmente en el estándar Simple Feature Access que define la forma de almacenar información espacial en bases de datos. Posteriormente se analizan las tecnologías usadas: JASPA como extensión que permite almacenar información geográfica en una base de datos no espacial, y gvSIG como aplicativo que reúne, en el marco de una interfaz única y amigable, un gran conjunto de funcionalidad orientada a la explotación de información geográfica almacenada en diferentes formatos, entre ellos los formatos de bases de datos espaciales. JASPA actualmente soporta dos bases de datos: H2 y PostgreSQL.

Tras una primera descripción de las herramientas usadas, se detalla el trabajo software realizado, el cual ha consistido en el desarrollo de un driver JASPA para gvSIG. Para dicho desarrollo se ha seguido un ciclo de vida en cascada consistente en la recogida y análisis de los requisitos, diseño del software, codificación del programa informático y ejecución de un plan de pruebas. Las ficha con el resumen de la ejecución del plan de pruebas se encuentran en el Anexo I. Resumen de resultado de pruebas, de este mismo documento.

El desarrollo del driver dota a gvSIG de la posibilidad de acceder a información espacial almacenada en una base de datos ligera H2 con extensión JASPA, lo cual permitirá el intercambio entre usuarios de información geográfica que sigue un modelo relacional, no soportado por los formatos tradicionales basados en archivos como ESRI Shapefile, y logrando reducir las necesidades de administración en comparación con las bases de datos espaciales corporativas.

Índice

Ilustraciones

Tablas

1. Introducción

El presente documento constituye la memoria del proyecto fin de carrera de la titulación de Ingeniería en Informática de la Universitat Oberta de Catalunya, ligado a los sistemas de información geográfica (SIG). El objetivo particular de este proyecto es la integración de JASPA para H2 en gvSIG Desktop 2.0.

H2[1] es un motor de bases de datos de código libre, que posee la capacidad tanto de poder ser embebido en aplicaciones Java como de ser ejecutado en modo cliente-servidor.

JASPA[2] (JAvaSPAtial) es una extensión espacial para sistemas de bases de datos relacionales, entre los que se encuentra H2. JASPA implementa el estándar "Simple Feature Access" (SFA) de OpenGIS Consortium[3] (OGC) y el estándar ISO/IEC 13249-3 "SQL/MM – Part 3: Spatial"[4].

gvSIG[5]es un proyecto de desarrollo de sistemas de información geográfica en software libre, que incluye principalmente las aplicaciones gvSIG Desktop y gvSIG Mobile. gvSIG Desktop fue el primer aplicativo que se desarrolló, por lo que se conoce simplemente como gvSIG.

La versión 2.0 de gvSIG Desktop (en la actualidad en versión 2.0 Alfa) incorpora como nueva característica la posibilidad de añadir a los ya existentes, nuevos proveedores de datos geográficos, como complementos del aplicativo que pueden añadirse a ella sin necesidad de modificar el código fuente de la misma. Basándose en esta nueva característica, se ha desarrollado un driver que permite el acceso a la información geográfica almacenada en H2 a través de la extensión JASPA.

A continuación se realiza una descripción de la secuencia de desarrollo del proyecto dividida en capítulos. En el capítulo 2 se realiza un estudio previo de las características de la información espacial, para posteriormente, en el capítulo 3, analizar la aplicación de dichas características al caso concreto de JASPA y gvSIG.

Una vez realizado el estudio previo, se pasa a las fases del desarrollo software siguiendo un ciclo en cascada donde cada capítulo es una fase del desarrollo; en el capítulo 4 se encuentra descrita la fase de análisis de requisitos, en el capítulo 5 la fase de diseño, en el capítulo 6 la fase de desarrollo del código fuente, y en el

---

[1] http://www.h2database.com

[2] http://jaspa.upv.es/ ó https://joinup.ec.europa.eu/software/jaspa/description

[3] http://www.opengeospatial.org/standards/sfs

[4] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=53698

[5] http://es.wikipedia.org/wiki/GvSIG

capítulo 7 la fase de pruebas. Finalmente, en el capítulo 8 se realiza una reflexión acerca de las conclusiones finales del proyecto y se establecen futuras líneas de desarrollo para la continuación del mismo.

2. Características de la información espacial

Para empezar a analizar las características de la información espacial resulta interesante reflexionar acerca de la problemática a la que se enfrenta.

En general, los datos que manejamos en nuestra vida tienen una componente espacial, que de manera simplista se puede definir como el ámbito geográfico asociado a una cierta entidad, siendo este ámbito geográfico tan simple como una localización puntual, o tan complejo como un conjunto de áreas n-dimensionales.

Para comunicar la información espacial sin ambigüedades, será necesario definir una forma de almacenar, manejar e interpretar esa información de tal forma que nos intercambiar datos de localización entre las personas, y como no, entre los sistemas automáticos que hacen uso de ella.

La información geográfica puede ser manejada en dos formatos diferentes denominados raster y vectorial. El formato vectorial utiliza descripciones de formas geométricas (principalmente puntos, polilíneas y polígonos) para el almacenamiento de la información, mientras que el formato raster, utiliza una imagen (mapa de bits). En la Ilustración 1: Cartografía Raster y Vectorial, se muestra las dos formas de almacenar la información de la realidad.

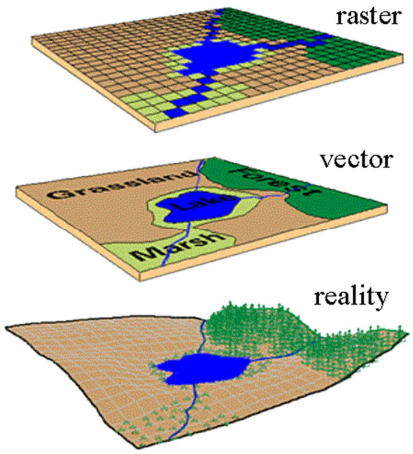

Ilustración 1: Cartografía Raster y Vectorial

Antes de entrar en mayor profundidad acerca de las características de la información espacial, es conveniente realizar algunas definiciones básicas que se encontrarán frecuentemente a lo largo del documento.

A continuación se definen algunos conceptos básicos, descritos con detalle en (Botella Plana), imprescindibles para entender el documento:

- Elemento geográfico o entidad geográfica: abreviado como elemento y siendo traducción al español del término inglés “feature”; se refiere a la abstracción de un fenómeno del mundo real realizada mediante la agrupación de sus características espaciales y no espaciales. Ejemplos de elementos geográficos podrías ser: un lago, una carretera, un árbol, etc.

- Atributo de un elemento (geográfico): característica del elemento que tiene asociados un nombre, un tipo de dato y un dominio de valores. Todo atributo tiene un valor perteneciente al dominio de valores del mismo. Las geometrías asociadas a los elementos no son más que un tipo de atributo del mismo (atributos espaciales).

  Tomemos como ejemplo el caso de una carretera. La entidad geográfica carretera se compondría de la información espacial, definida como el conjunto ordenado de posiciones geográficas por el que pasa la carretera, y de las características no espaciales, que se definirán como un listado de atributos, y sus correspondientes valores, característicos de la carretera, como pueden ser: nombre de la carretera = A49, tipo de carretera = autovía, número de carriles = 3, ancho de la calzada (m) = 15, etc.

- Elemento simple: elemento cuyos atributos espaciales pueden describirse en trozos de líneas, ya sean uniones rectas de puntos o interpolación curva de conjuntos de puntos. La interpolación se usa en curvas y superficies, las cuales, debido a su naturaleza están constituidas por un conjunto infinito de puntos, y por lo tanto, no pueden ser representado por una lista exhaustiva de puntos.

- Envolvente (en inglés Envelope): también conocido como "bounding box", define el rectángulo geométrico mínimo que incluye un conjunto de elementos geográficos.

- Sistema de referencia espacial, también conocido por sus siglas del inglés Spatial Reference System (SRS) o Coordinate Reference System (CRS): es un sistema de coordenadas que se utiliza para localizar entidades geográficas. Un sistema de referencia espacial define una proyección de mapa específica, así como las transformaciones entre diferentes sistemas de referencia espaciales. Uno de los estándares de sistemas de información geográfica más comúnmente aceptados, el estándar Simple Feature Access del OpenGIS Consortium, define que los SRS deben ser descritos por un texto bien formateado y es soportado ampliamente por los sistemas de información geográfica. Los SRS se pueden referenciar por un identificador único numérico denominado SRID, entre los que se incluyen los códigos EPSG definidos por la "International Association of Oil and Gas Producers". (Anónimo, Wikipedia)

- Códigos EPSG: El "European Petroleum Survey Group" o EPSG (1986 – 2005) fue una organización científica vinculada a la industria del petróleo europea. Estaba formada por especialistas que trabajaban en el campo de la geodesia, la topografía y la cartografía aplicadas en relación con la exploración petrolífera. EPSG compiló y difundió el conjunto de parámetros geodésicos EPSG, una base de datos ampliamente usada que contiene elipsoides, datums, sistemas de coordenadas, proyecciones cartográficas, etc. Las tareas previamente desempeñadas por la EPSG son retomadas en 2005 por la International Association of Oil and Gas Producers Surveying and Positioning Committee (OGP). (Anónimo, Wikipedia)

Como acabamos de ver, la información espacial presenta características particulares que hacen necesaria la existencia de sistemas especializados capaces de almacenar y manipular dicha información. Son los SIG los que ofrecen las herramientas necesarias para almacenar, consultar, manipular, y visualizar los elementos geográficos, produciendo diferentes resultados en función de la necesidad del usuario de dicho sistema. Ejemplos de estos diferentes resultados pueden ser la impresión de un mapa de carreteras, el cálculo del área de una región agrupado por actividades realizadas en dicho área, cálculo de rutas óptimas entre dos lugares, etc.

En los últimos años se han creado muchos SIG diferentes, con la particularidad de que muchos de ellos ofrecían un paquete completo de herramientas (bases de datos espaciales, aplicaciones que manejaban y visualizaban los datos, algoritmos) pero basados en formatos propietarios que no permitían su uso por sistemas de otros fabricantes.

Para salvar este obstáculo, se constituyó el Open Geospatial Consortium[6] (OGC), cuyo fin es la definición de estándares abiertos e interoperables dentro de los SIG y de la World Wide Web. Persigue acuerdos entre las diferentes empresas del sector que posibiliten la interoperación de sus sistemas y facilitar el intercambio de la información geográfica en beneficio de los usuarios

Uno de los trabajos más importantes del OGC es la definición del "Simple Feature Access" [7] (SFA), que es un estándar para el almacenamiento y acceso a la información espacial en bases de datos SQL. Sin este estándar, probablemente cada base de datos implementaría el almacenamiento de la información de una manera diferente e incompatible.

Un estándar equivalente al SFA, y que también es ampliamente usado, es el estándar internacional "ISO/IEC 13249-3 SQL/MM Part 3: Spatial" el cual define como almacenar, recuperar y procesar datos espaciales usando SQL y define un conjunto de funciones para convertir, comparar y en generar manipular este tipo de información.

El estándar SFA que propone el OGC necesita en su versión más ampliamente usada de ciertas características particulares en las bases de datos, como por ejemplo la posibilidad de definición de tipos de datos de usuario, que finalmente hace que las implementaciones concretas del estándar dependan de las capacidades del motor de base de datos; sin embargo, una parte central es compartida por todas las bases de datos espaciales que implementan el estándar.

Este estándar no solo define la forma de almacenar dicha información en bases de datos SQL, sino que define también las estructuras de datos que los programas pueden utilizar para implementar las geometrías en memoria. Estas estructuras no

[6] www.opengeospatial.org

[7] Obsérvese que "Simple Feature Access", se traduciría como "acceso a elementos geográficos simples".

obligan a los programas a almacenar sus datos en estas estructuras, sino más bien definen cómo se deben pasar dichos datos cuando se interconectan distintos programas o incluso cuando un programa hace uso de una librería geométrica. Por ejemplo, si una librería geométrica implementa el cálculo del área de un polígono, esperará que los datos del polígono sobre el que se quiere calcular el área se le pasen con una determinada estructura, que puede diferir de la estructura de datos con la que el programa los almacena, por lo tanto el programa tendrá que convertir de su estructura de almacenamiento interna a la esperada por la librería geométrica. Si cada librería geométrica espera los datos con un formato diferente, el programa que las utilice tendrá que hacer conversiones para cada librería, multiplicando el esfuerzo. OGC define en el documento "Part 1" unas estructuras de datos que pueden ser utilizadas por librerías y programas para el intercambio de información geométrica, por lo tanto se requiere menor esfuerzo de integración de librerías si se utiliza dicho estándar.

El estándar "Simple Feature Access" está dividido en dos partes

- Parte 1 define un modelo de datos, común a los distintos almacenamientos, que representa a las entidades para ser usado por las aplicaciones que se utilizan los almacenamientos de datos de elementos geográficos (en nuestro caso bases de datos SQL) para el acceso a los datos.

- Parte 2 define cómo se ha de realizar una implementación del modelo abstracto de datos, definido en la parte 1, en una base de datos SQL, y define dos posibilidades: utilizando tipos de datos SQL estándar y utilizando tipos de datos definidos por el usuario si la base de datos lo permite.

La mayoría de las bases de datos geográficas sigue de alguna forma este estándar, y JASPA, como extensión espacial de la base de datos H2, también, por lo tanto es importante conocer el estándar para comprender la arquitectura interna de JASPA.

2.1. El estándar "Simple Feature Access - Part 1: Common Architecture"

Esta parte 1 del SFA describe el modelo de objetos geográficos que se utiliza de forma común por los sistemas de almacenamiento y manejo de información geográfica, independientemente de la manera de almacenar dicha información.

La modelización más básica de las características espaciales de las entidades se representa en el diagrama de clases UML de la Ilustración 2: Jerarquía de clases geométricas:

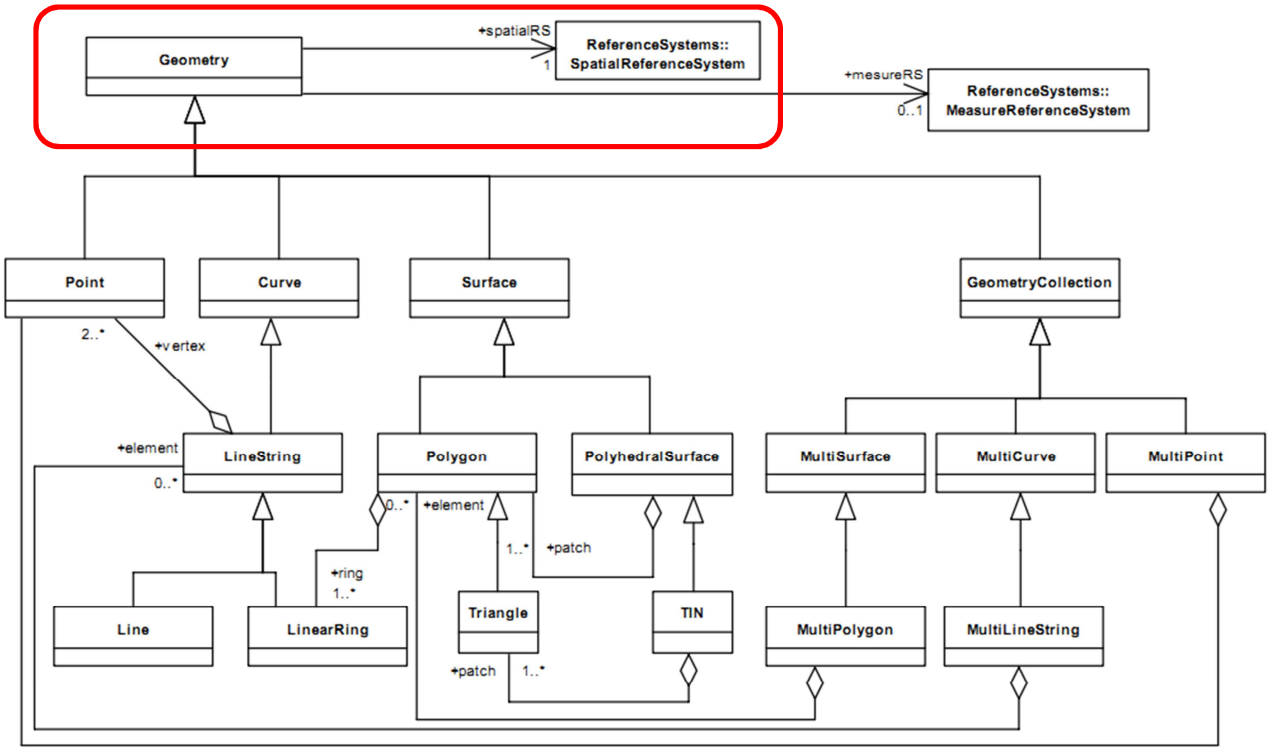

Ilustración 2: Jerarquía de clases geométricas

La clase raíz es la clase Geometry, que es una clase abstracta, no instanciable y que tiene asociado un sistema de referencia espacial (SpatialRS), que heredan todas las clases que derivan de Geometry y que describe el espacio de coordenadas en el cual está definido el objeto geométrico.

En la Ilustración 3: Clasificación principal de geometrías, aparece resaltada la clasificación de la geometría en tres categorías principales: Point, Curve y Surface.

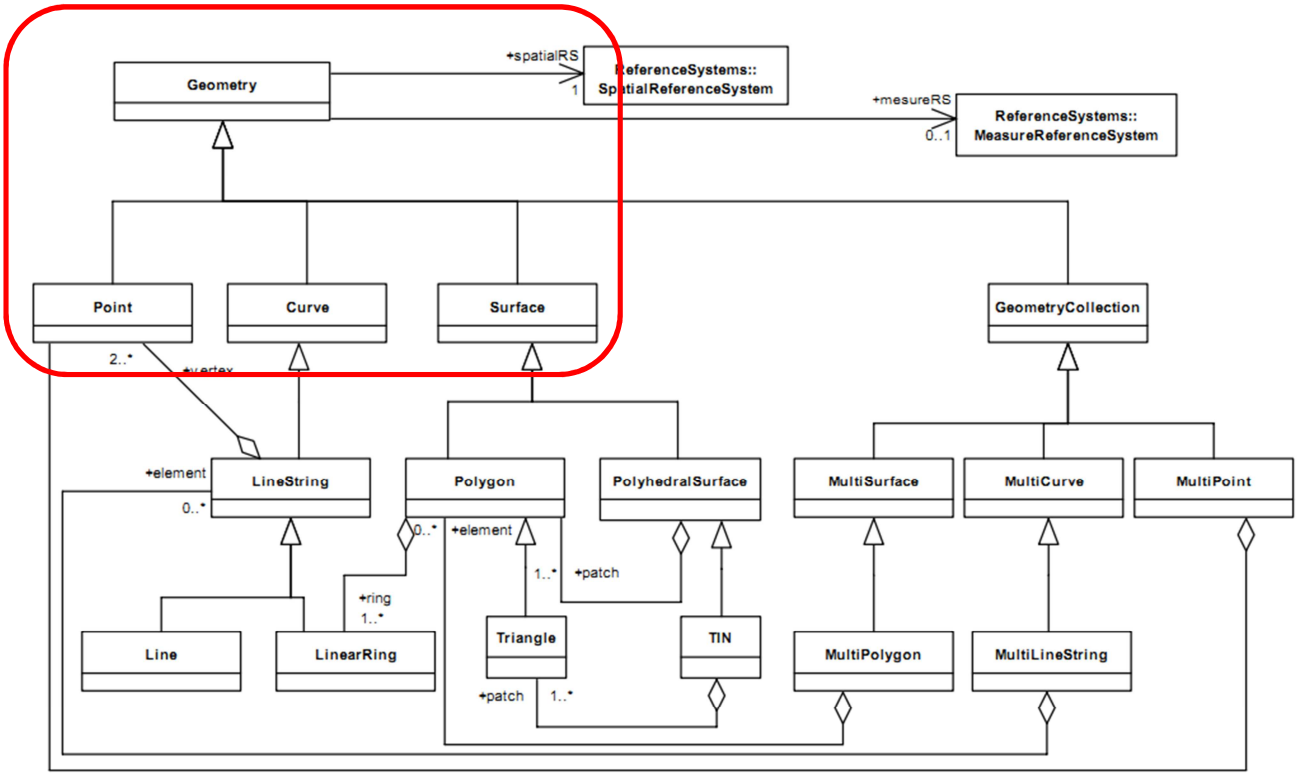

Ilustración 3: Clasificación principal de geometrías

La representación gráfica, mostrada en la Ilustración 4: Representación gráfica de los tipos geométricos, de cada uno de los tipos descritos ayudan a entender mejor el concepto que se encuentra tras cada una de las definiciones.

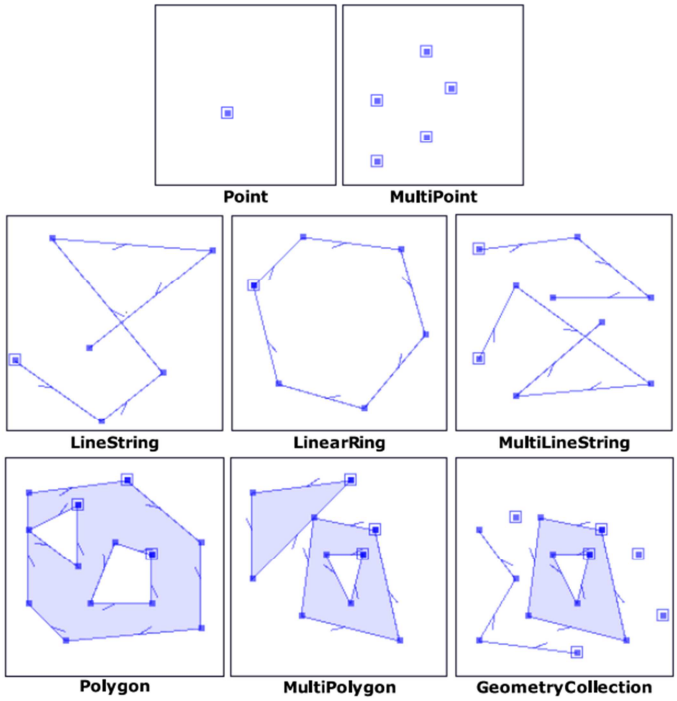

Ilustración 4: Representación gráfica de los tipos geométricos

El objeto punto se define de manera elemental: es una única posición geográfica que determina unívocamente una localización. Los puntos son elementos de dimensión 0.

El objeto curva, de dimensión 1, se compone como una secuencia ordenada de objetos punto. Los objetos curva, según se muestra en la Ilustración 5: Clase Curve y derivadas, se dividen en distintos tipos dependiendo de la interpolación usada. En la especificación de OGC sólo se define la interpolación lineal, dando lugar al subtipo LineString (Segmentos de línea encadenados, conocido también como Polilínea).

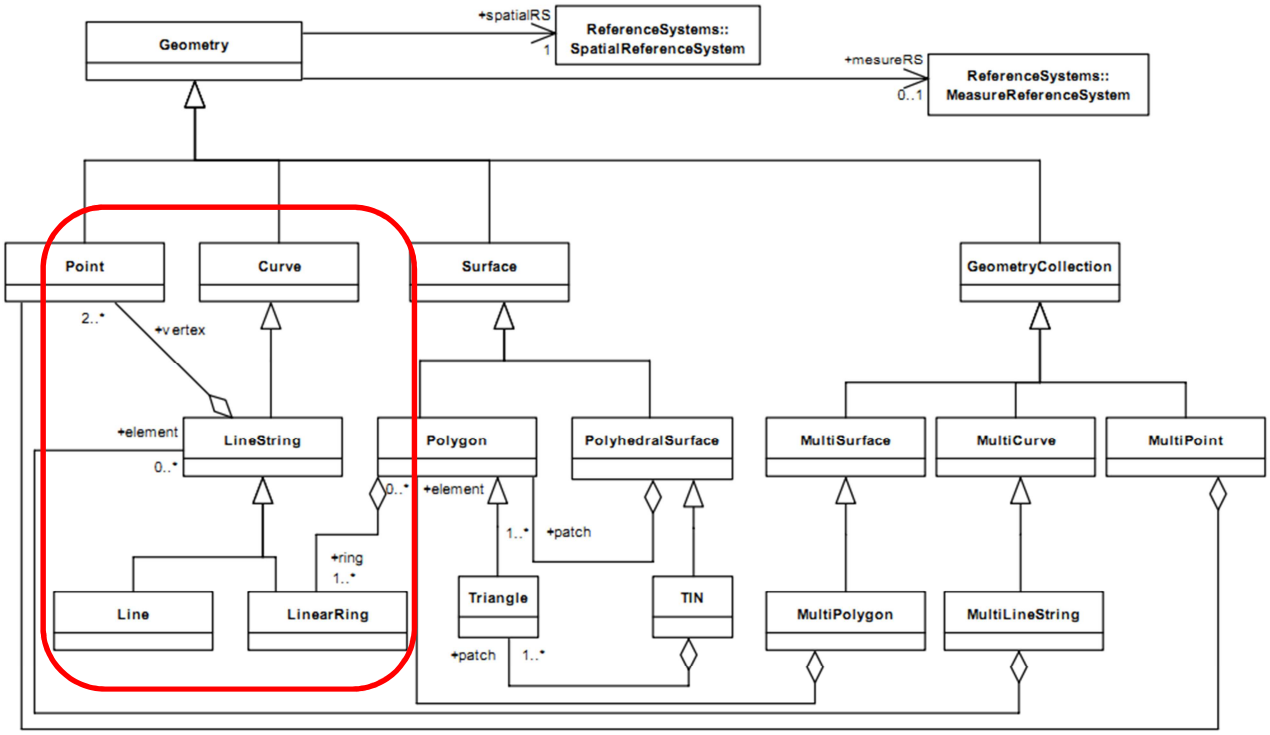

Ilustración 5: Clase Curve y derivadas

La subclase Line, es una especialización de la clase LineString que tiene sólo dos puntos (equivaldría al concepto de segmento).

La subclase LinearString es una especialización de LineString que tiene como características que: a) es cerrada: el último punto coincide con el primero y b) los segmentos de la polilínea no se cortan entre ellos. Sería equivalente al concepto de polilínea simple y cerrada.

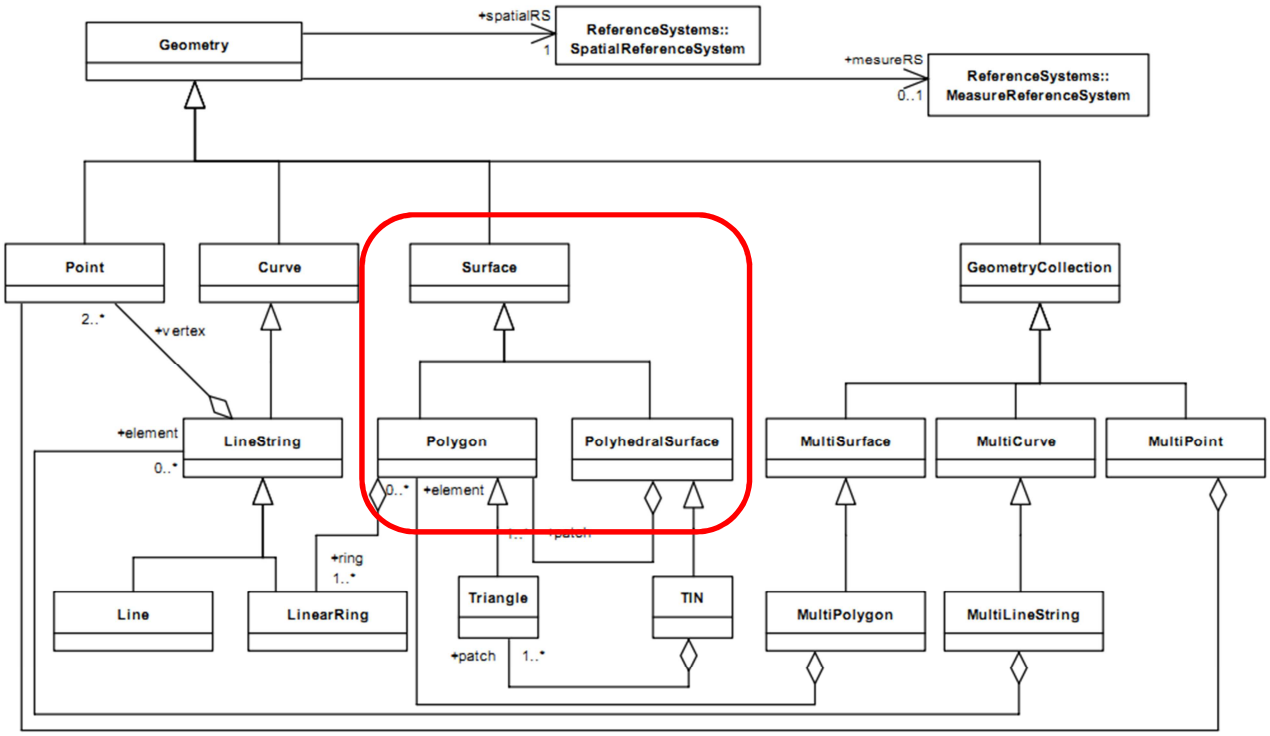

Ilustración 6: Clase superficie y derivadas

La clase Surface, de dimensión 2, que se encuentra remarcada en la Ilustración 6: Clase superficie y derivadas, es un área geométrica cerrada que puede tener uno o varios "agujeros". Los objetos de tipo Surface se dividen en Polygon y PolyhedralSurface. Los polígonos (Polygon) son la forma más simple de presentación formada por varios objetos LinearString donde uno de ellos representa el área básica y los demás los agujeros de esa área. La superficie poliédrica (PolyhedralSurface) la forma uno o más polígonos unidos por sus vértices comunes.

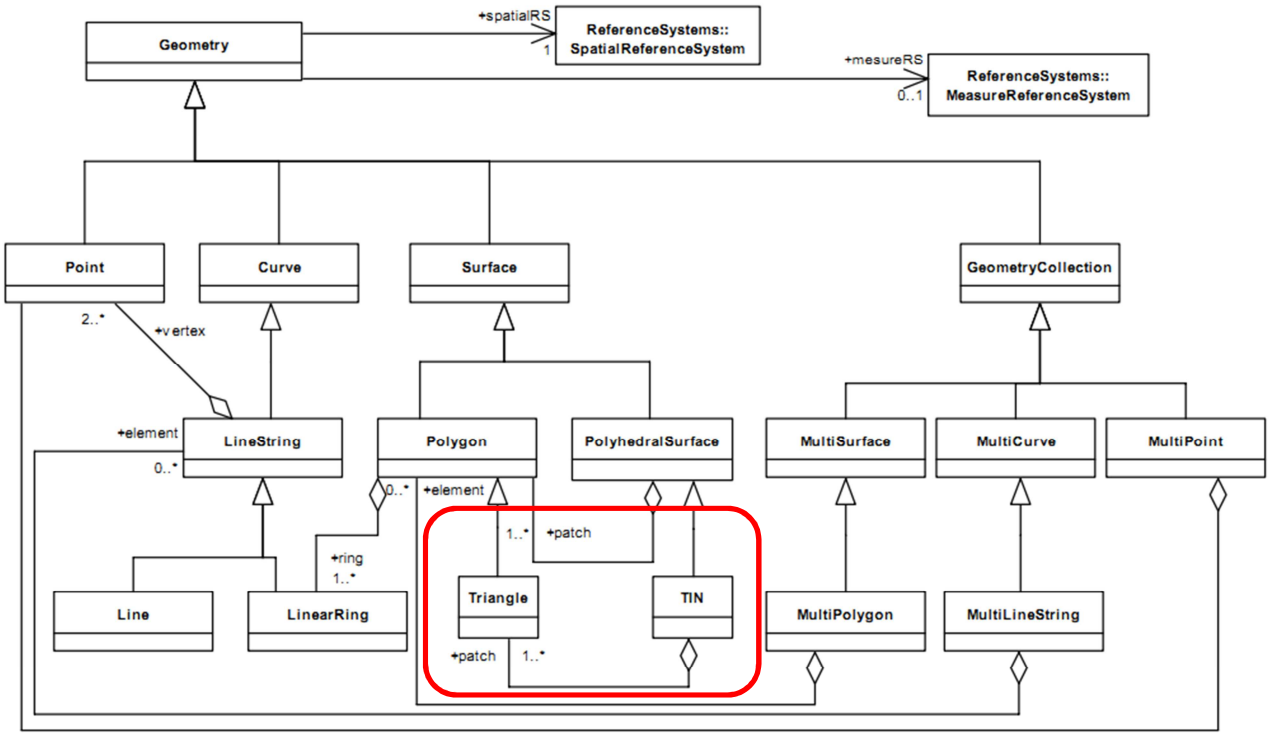

Ilustración 7: Clases Triangle y TIN

Las clases Triangle y TIN se muestran en la Ilustración 7: Clases Triangle y TIN.

Un triángulo se define como un tipo especial de polígono con sólo tres vértices todos distintos y sin agujeros.

Un TIN (del inglés Triangulated Irregular Network – Red irregular triangulada) consiste en una superficie poliédrica compuesta exclusivamente por áreas triangulares. Esta es la forma habitual en la que se presentan las vistas de alambre de los modelos 3D, tal y como se muestra en la Ilustración 8: Modelo de rejilla triangular (TIN).

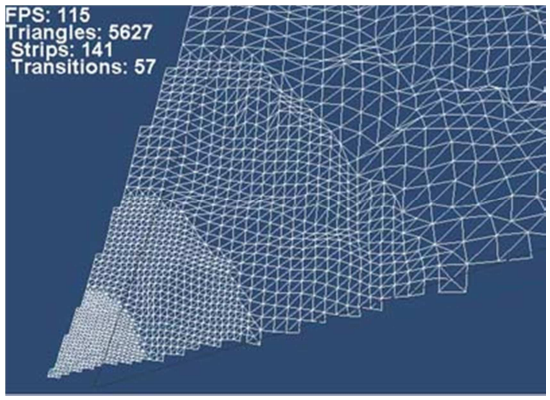

Ilustración 8: Modelo de rejilla triangular (TIN)

Aparte de estos elementos simples, y según se muestra en la Ilustración 9: Colecciones de geometrías, se definen, para cada uno de los tipos básicos Point, Curve

y Surface, colecciones de elementos de estos tipos, que no son más que conjuntos no ordenados de elementos simples. De forma general, un GeometryCollection contendría un conjunto de cualquier tipo de las geometrías descritas previamente, mientras que las clases MultiPoint, MultiLineString y MultiPolygon contendrían conjuntos de geometrías exclusivamente de tipo Point, LineString y Polígono respectivamente. MultiCurve y MultiSurface son clases que generalizan la colección para manejo de Curvas y Superficies.

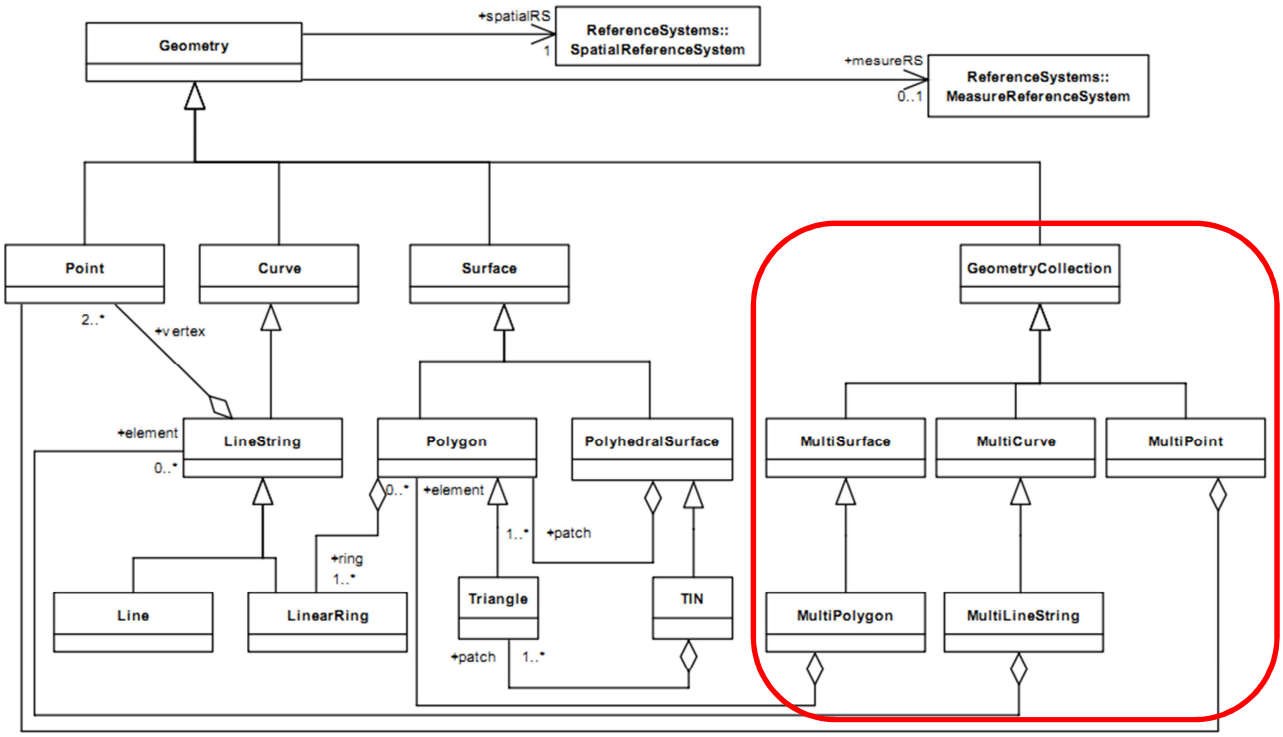

Ilustración 9: Colecciones de geometrías

2.2. El estándar "Simple Feature Access - Part 2: SQL Option"

La norma SFA – Part 2 SQL Option establece que para ajustarse a la norma, un aplicativo deberá cumplir uno de los tres siguientes requisitos además de los componentes apropiados relacionados en la Parte 1:

a) La implementación de las tablas de elementos geográficos estará basada en tipos de datos SQL predefinidos:

a.1) usando tipos de datos SQL numéricos para el almacenamiento y acceso a través de SQL/CLI de las geometrías.

a.2) usando tipos de datos SQL binarios para el almacenamiento y acceso a través de SQL/CLI de las geometrías.

b) La implementación de las tablas de elementos geográficos se realizará con tipos de datos geométricos SQL que soporten el acceso a través de SQL/CLI tanto textual como binario.

La implementación de JASPA, como veremos más adelante, implementa el almacenamiento usando tipos de datos VARBINARY como tipo de dato geométrico para el almacenamiento y acceso a las geometrías, por lo tanto, cumple el estándar definido en b). A continuación, se analizará la definición de la opción b) en este mismo documento, aconsejando al lector, en caso de que lo considere de interés, profundizar en las otras dos opciones a través de la consulta en el propio estándar.

2.2.1. Esquema de datos usando tipos de datos geométricos

Como se ha indicado anteriormente, cuando el esquema de datos SQL de una base de datos que soporta tipos de datos geométricos, no hacen falta las tablas de geometrías, puesto que cada columna de una tabla de datos de elementos geográficos (Feature Table) puede almacenar la geometría indicando que el tipo de dato de la columna es de tipo geométrico.

El esquema de datos definido para el almacenamiento con este tipo de datos se muestra en la Ilustración 10: Elementos geográficos con tipos de datos geométricos.

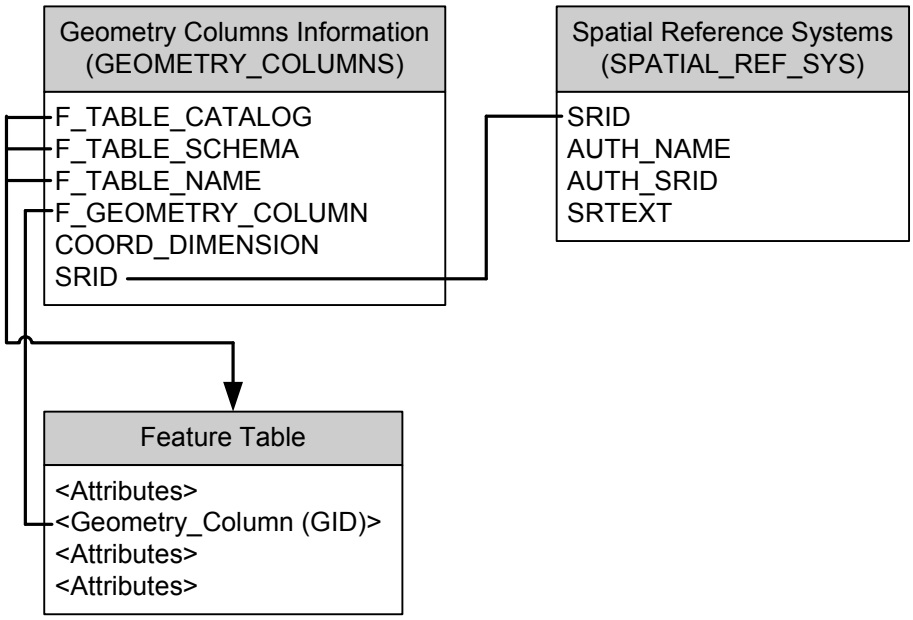

Ilustración 10: Elementos geográficos con tipos de datos geométricos

Para poder aplicar este modelo de datos, se requiere que la base de datos soporte tipos de datos no estándar definidos por el usuario o la posibilidad de almacenamiento de la información geográfica en un tipo de datos conocido como BLOB (Binary Large OBject).

Cuando se utilizan datos tipo BLOB, para identificar aquellas columnas que tienen un

tipo de datos geométrico el estándar propone que haya unas tablas que almacenen los metadatos de información espacial de dichas columnas. Entre estas tablas está GEOMETRY_COLUMNS que define para cada columna de datos espaciales de cada tabla, las propiedades que caracterizan a las geometrías almacenadas en dicha columna:

a) La identidad (catálogo, esquema y nombre) de la tabla que almacena las entidades geométricas de la cual la columna geométrica es miembro.

b) El nombre de la columna que almacena la geometría.

c) El identificador del sistema de referencia espacial usado por las geometrías almacenadas en la columna.

d) La dimensión de la columna que almacena la geometría.

En este esquema de datos, se utiliza una tabla por cada tipo de elemento geográfico donde se almacenan sus atributos no espaciales con tipos de datos estándar, y sus atributos espaciales con un tipo de datos binario.

Para representar de manera inequívoca un objeto geométrico, un sistema de referencia espacial o transformaciones entre sistemas de referencia, el OGC ha definido un formato de representación binaria de las geometrías conocido como WKB (Well Known Binary for Geometry). El formato WKB se muestra de manera habitual como cadenas hexadecimales, donde el primer byte indica si los datos se representan en formato big-endian o little-endian[8], los siguientes 4 bytes representan un entero que define el tipo de geometría y los siguientes definen las coordenadas de las geometrías, que pueden ser XY, XYZ, XYM o XYZM.

Además del formato WKB, resulta de utilidad una representación textual de las geometrías de tal manera que sean, al menos aquellas geometrías sencillas, humanamente comprensibles con un simple vistazo. Para ello también se define el formato WKT (Well Known Text).

Por ejemplo, un punto de coordenadas x=2, y=4, se representaría en WKT como POINT (2.0 4.0) y en formato WKB como la cadena hexadecimal 000000000140000000000000004010000000000000.

Aunque el formato WKT es inteligible a simple vista por un lector humano que conozca la sintaxis, su problema es que necesita mayor espacio de almacenamiento y para procesar la información se necesitan algoritmos más complejos, por contra, el formato WKB ocupa menos espacio, y es más fácil de procesar por programas informáticos.

De la misma forma que se almacena la información espacial de los elementos,

[8] Se refiere al alineamiento de los datos en los registros de la CPU, que puede producir incompatibilidades cuando procesadores de diferentes alineamientos comparten información.

también se almacena su sistema de referencia espacial asociado, y para la definición de dichos sistemas de referencia también se define un formato WKT for SRS, equivalente al formato para geometrías pero aplicado a los sistemas de referencia espacial.

Adicionalmente, se definen dos formatos adicionales: EWKT y EWKB que añaden el SRID en la geometría. Un ejemplo de EWKT podría ser

"SRID=25830;POINT(712243 6653327 77.1)"

El estándar SQL/MM (derivado del estándar SFA-SQL), define los tipos de datos geométricos y las funciones que operan sobre ellos para las bases de datos que soportan tipos de datos geométricos o tipos de datos definidos por el usuario.

Existen bibliotecas de código que facilitan la manipulación de las geometrías almacenadas en formato WKB y funciones de transformación entre formato WKT y WKB. Una de las más conocidas es la "Java Topology Suite"[9]

Java Topology Suite (JTS) es una biblioteca de implementación libre realizada en Java para el manejo de datos espaciales. JTS tiene un conjunto completo de funciones de procesamiento de geometrías que puede usarse para la implementación del estándar SQL/MM.

JTS posee las siguientes características principales:

- Sigue la especificación "Simple Features Specification for SQL" del OGC.
- Ofrece una implementación robusta, consistente y complete de los algoritmos espaciales fundamentales en 2D.
- Es una biblioteca 100% Java, de código abierto (licencia LGPL) y de probada estabilidad y rendimiento.
- Usada en gran cantidad de productos de manejo de información espacial, incluido JASPA.

Esta biblioteca realiza la definición de un conjunto de clases que permiten realizar ciertas operaciones geométricas sobre los datos espaciales, como por ejemplo convertir de un WKB a textual o viceversa, realizar operaciones entre dos geometrías (unión, diferencia, intersección...), etc.

[9] http://www.vividsolutions.com/jts/JTSHome.htm

3. Estudio de H2, JASPA y gvSIG

Una vez definidos los conceptos básicos asociados a la información espacial y descritos de manera resumida los estándares usados por los productos que se utilizarán para el proyecto, se procede a describir las características particulares de cada uno de ellos: H2, JASPA y gvSIG.

3.1. H2

H2 es un motor de base de datos implementada en Java, que puede funcionar en modo cliente/servidor o embebido en un aplicativo java.

El funcionamiento cliente/servidor es el tradicional de los motores de bases de datos relacionales, en el cual, una instancia del motor de datos se puede comunicar simultáneamente con diferentes clientes de dichos datos, transmitiéndose la información entre ellos a través de un driver de comunicaciones, que en el caso de aplicaciones java sigue el estándar Java Database Connectivity (JDBC).

El modo de funcionamiento embebido se consigue uniendo en una única el cliente de la base de datos, y el motor de la misma, eliminando la comunicación extra-procesos puesto que tanto cliente como servidor se ejecutan en el mismo espacio de memoria del aplicativo.

H2 en concreto permite incluso que la información que procesa, se mantenga en memoria o en disco, siendo la opción de almacenamiento en memoria la más rápida, pero tiene el inconveniente el límite de la capacidad de la memoria RAM que es muy inferior a las capacidades de los discos.

H2 posee una API basada en JDBC, y ofrece un excepcional rendimiento y bajo consumo de recursos a la vez que una administración muy sencilla.

La principal carencia de H2, teniendo en cuenta el objetivo de este trabajo, es que carece de soporte de datos espaciales en su implementación nativa. Para paliar esta carencia, se ha desarrollado una extensión espacial denominada JASPA.

3.2. JASPA

JASPA (Java SPAtial) como se ha indicado anteriormente es una extensión espacial escrita en Java para motores de bases de datos relacionales que implementa los estándares mencionados Simple Feature Access de OGC y el estándar ISO SQL/MM.

JASPA actualmente soporta los motores de bases de datos relacionales PostgreSQL[10] y

[10] http://www.postgresql.org.es

H2, y tiene previsto soportar otros motores de bases de datos, entre ellos HSQLDB[11] (HyperSQL DataBase). Para el almacenamiento de la información espacial utiliza la implementación definida por el OGC para SQL utilizando tipos de datos geométricos que ha sido descrita anteriormente en este documento.

3.2.1. Implementación del estándar SFA para SQL de OGC

En la Ilustración 10: Elementos geográficos con tipos de datos geométricos, se mostró la implementación de referencia basada en las tablas GEOMETRY_COLUMNS y SPATIAL_REF_SYS que están asociadas con una tabla por cada tipo de elemento geográfico (FeatureType).

Para verificar que JASPA sigue este esquema de datos para el almacenamiento de los datos espaciales, se puede acceder a través de la consola web de administración de H2 y consultar su esquema de datos.

En la Tabla 1 Modelo de datos de JASPA sobre H2, se muestra el resultado de la consulta "select * from GEOMETRY_COLUMNS" de una base de datos JASPA y dos tablas con una columna geométrica cada una. Se observa como el modelo de datos de esta tabla sigue la norma del OGC.

F_TABLE_ CATALOG	F_TABLE_ SCHEMA	F_TABLE_NAME	F_GEOMETRY_ COLUMN	COORD_ DIMENSION	SRID	TYPE
	PUBLIC	CARRETERAS	GEOM	2	-1	MULTILINESTRING
	PUBLIC	River	GEOM	2	4326	MULTILINESTRING

Tabla 1 Modelo de datos de JASPA sobre H2

También se observa en la Ilustración 11: Sistemas de referencia espaciales en JASPA, que la tabla SPATIAL_REF_SYS tiene la estructura recomendada por el estándar OGC, y que utiliza el formato WKT para la descripción de cada sistema de referencia:

[11] http://hsqldb.org

SELECT * FROM JASPA.SPATIAL_REF_SYS;

SRID	AUTH_NAME	AUTH_SRID	SRTEXT
2000	EPSG	2000	PROJCS["Anguilla 1957 / British West Indies Grid", GEOGCS["Anguilla 1957", DATUM["Anguilla 1957", SPHEROID["Clarke 1880 (RGS)", 6378249.145, 293.465, AUTHORITY["EPSG","7012"]], AUTHORITY["EPSG","6600"]], PRIMEM["Greenwich", 0.0, AUTHORITY["EPSG","8901"]], UNIT["degree", 0.0174532925199433295], AXIS["Geodetic longitude", EAST], AXIS["Geodetic latitude", NORTH], AUTHORITY["EPSG","4600"]], PROJECTION["Transverse_Mercator"], PARAMETER["central_meridian", -62.0], PARAMETER["latitude_of_origin", 0.0], PARAMETER["scale_factor", 0.9995], PARAMETER["false_easting", 400000.0], PARAMETER["false_northing", 0.0], UNIT["m", 1.0], AXIS["Easting", EAST], AXIS["Northing", NORTH], AUTHORITY["EPSG","2000"]]
2001	EPSG	2001	PROJCS["Antigua 1943 / British West Indies Grid", GEOGCS["Antigua 1943", DATUM["Antigua 1943", SPHEROID["Clarke 1880 (RGS)", 6378249.145, 293.465, AUTHORITY["EPSG","7012"]], TOWGS84[-255.0, -15.0, 71.0, 0.0, 0.0, 0.0, 0.0], AUTHORITY["EPSG","6601"]], PRIMEM["Greenwich", 0.0, AUTHORITY["EPSG","8901"]], UNIT["degree", 0.0174532925199433295], AXIS["Geodetic longitude", EAST], AXIS["Geodetic latitude", NORTH], AUTHORITY["EPSG","4601"]], PROJECTION["Transverse_Mercator"], PARAMETER["central_meridian", -62.0], PARAMETER["latitude_of_origin", 0.0], PARAMETER["scale_factor", 0.9995],

Ilustración 11: Sistemas de referencia espaciales en JASPA

3.2.2. Implementación del estándar SFA Common del OGC

JASPA soporta varios tipos de geometrías, que se muestran en la Ilustración 12: Geometrías soportadas por JASPA. Aquellas con fondo azul no se pueden crear, sino que solo sirven para agrupar características comunes de los geometrías que derivan de ellas, quedando por lo tanto 7 tipos de geometrías directamente usables: Point, LineString, Polygon, GeometryCollection, MultiPoint, MultiLineString, y MultiPolygon. Todas ellas siguen la definición del estándar de OGC.

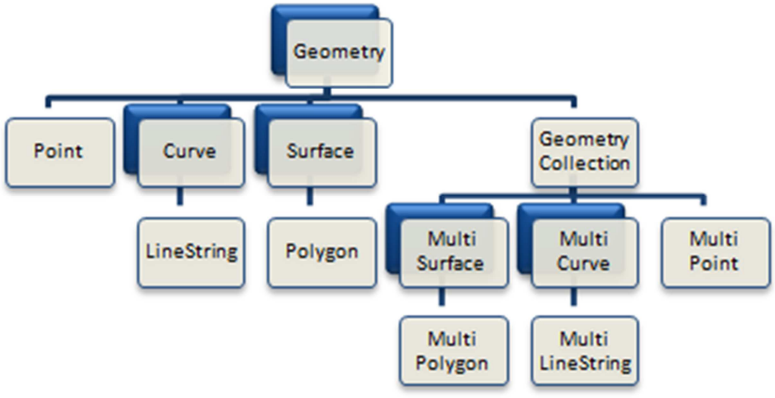

Ilustración 12: Geometrías soportadas por JASPA

El estándar SQL-MM añade los tipos de geometría CircularString, CompoundCurve y

CurvePolygon, sin embargo JASPA no soporta estos nuevos tipos.

JASPA, debido a las diferentes capacidades y características de cada motor de base de datos que soporta, posee algunas diferencias para cada implementación, reflejándose en dos diferentes denominaciones del producto:

- JASPA4PG (JASPA for PostgreSQL)
- JASPA4H2 (JASPA for H2)

Las diferencias entre JASPA4PG y JASPA4H2 se encuentran descritas en su manual[12]. Estas diferencias se reflejan en la sintaxis de las sentencias SQL de consulta y manipulación de datos, por lo que es conveniente tenerlas en cuenta:

- Cuando el nombre de una tabla o columna no se entrecomilla, H2 la convierte a mayúsculas y PostgreSQL la convierte a minúsculas.

- Las conversiones de tipos solo son soportados por PostgreSQL, así que solo están soportados por JASPA4PG, si bien, se recomienda evitar su uso puesto que no están soportados en todas las bases de datos.

- La forma de crear arreglos (en inglés array) es completamente diferente en PostgreSQL que en H2, y por lo tanto, también diferente en JASPA4PG y JASPA4H2.

- Las funciones agregadas se han renombrado en JASPA4H2 añadiendo el sufijo "Agg" al nombre de función normal.

- Debido a que H2 no permite tener funciones con el mismo número de parámetros pero de distintos tipos, la implementación JASPA4H2 ha renombrado algunas de esas funciones.

- Las funciones "difference" y "expand", al existir previamente en la base de datos H2, se han renombrado añadiendo el prefijo "ST_".

Los índices espaciales se utilizan para acelerar las operaciones de consultas donde hay operaciones espaciales involucradas. Actualmente, los índices espaciales solo están soportados en la implementación de JASPA para PostgreSQL.

JASPA ofrece documentación de un taller de trabajo, disponible en inglés y en español, a modo de guía de autoaprendizaje que ayuda enormemente en la comprensión del producto. A lo largo del taller de trabajo, se mantienen duplicados los ejemplos con la sintaxis de JASPA4PG y JASPA4H2.[13]

JASPA, en su implementación JASPA4PG, tiene un alto grado de compatibilidad con

[12] http://jaspa.upv.es/jaspa/v0.2.0/manual/html

[13] http://jaspa.upv.es/blog/documentation/

PostGIS[14], que es una extensión espacial de la base de datos de fuente abierta PostgreSQL. PostGIS es un referente en los SIG, y JASPA, gracias a su compatibilidad con PostGIS, ofrece en la versión 0.2, drivers de datos para tres SIG de fuente abierta: gvSIG 1.11, Kosmo GIS[15] 2.0.1 y MapServer[16]. Estos drivers son pequeñas adaptaciones de los drivers originales para PostGIS, pero ajustados a la sintaxis y características específicas de JASPA.

3.3. gvSIG Desktop

gvSIG Desktop es un Sistema de Información Geográfica (SIG), esto es, un aplicativo de escritorio diseñada para capturar, almacenar, manipular, analizar y desplegar en todas sus formas, la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión.

Este aplicativo es capaz de acceder a los formatos de datos más comunes, tanto vectoriales como raster, y cuenta con un amplio número de herramientas para trabajar con información de naturaleza geográfica.

gvSIG soporta una gran cantidad de formatos de datos, cuyo listado se puede consultar en su página web, pero de entre los cuales merece la pena destacar los mostrados en la Tabla 2: Resumen de tipos de formatos soportados por gvSIG Desktop.

Origen	Tipo	Nombre	Descripción
Archivo	Raster	TIFF	Imagen raster
Archivo	Vectorial	SHP	Shapefile. Formato de datos propietario de la compañía ESRI[17].
Archivo	Vectorial	GML	Geography Markup Language. Formato definido por el estándar GML[18] de OGC.
Archivo	Vectorial	KML	Keyhole Markup Language. Formato definido

[14] http://postgis.refractions.net

[15] http://www.opengis.es

[16] http://mapserver.org

[17] http://www.esri.com

[18] http://www.opengeospatial.org/standards/gml

			inicialmente por Google y adoptado posteriormente por el OGC[19].
Servicio Web	Raster	WMS	Web Map Service. Estándar definido por el OGC para ofrecer mapas en formato raster mediante un servicio web[20].
Servicio Web	Vectorial	WFS	Web Feature Service. Estándar definido por el OGC para ofrecer mapas en formato vectorial mediante un servicio web[21].
BBDD	Vectorial	PostGIS	Driver para acceso a la base de datos espacial de fuente abierta PostGIS.

Tabla 2: Resumen de tipos de formatos soportados por gvSIG Desktop

Con esta cantidad de formatos soportados por gvSIG, no es de extrañar que el modelo de geometrías de gvSIG no siga con total exactitud el estándar OGC, sin embargo, esto no es obstáculo para que gvSIG pueda soportar un conjunto de geometrías más que suficientes para la mayoría de los SIG.

3.3.1. Modelo de geometrías de gvSIG

Para analizar cómo encaja gvSIG con los modelos de datos estudiados hasta ahora, a continuación se describirán las geometrías soportadas por gvSIG y se identificará la correspondencia entre ellas y las soportadas por JASPA, que son las definidas en el estándar de OGC.

En gvSIG 2.0 se implementa un modelo de geometrías que pretende seguir la recomendación ISO 19107:2003 Geographic information -- Spatial schema[22].

En la Ilustración 13: Jerarquía del modelo de geometrías de gvSIG 2.0[23], se puede ver la clase Geometry y todas las clases de las que hereda, y las que heredan de ella. Se

[19] http://www.opengeospatial.org/standards/kml

[20] http://www.opengeospatial.org/standards/wms

[21] www.opengeospatial.org/standards/wfs

[22] www.iso.org/iso/catalogue_detail.htm?csnumber=26012

[23] http://www.gvsig.org/web/projects/gvsig-desktop/docs/devel/org.gvsig.fmap.geom/2.0.0/el-modelo-de-geometrias-1/el-modelo-de-geometrias

observa una herencia de la clase java.awt.Shape que se mantiene por motivos de compatibilidad hacia atrás, pero es posible que en futuras versiones desaparezca.

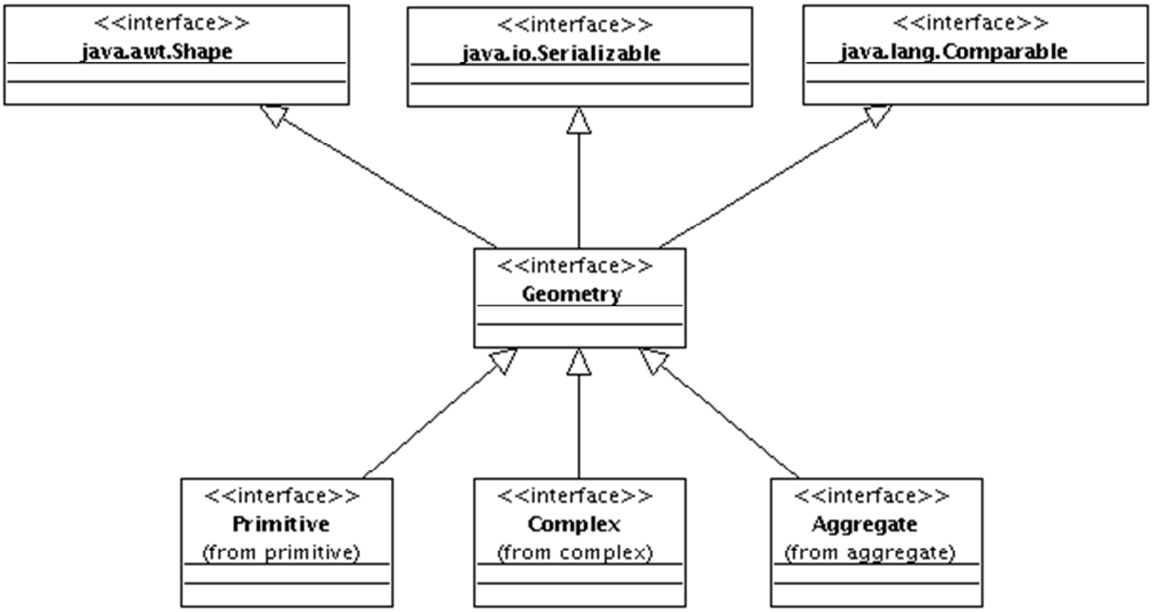

Ilustración 13: Jerarquía del modelo de geometrías de gvSIG 2.0

Las geometrías se clasifican en tres tipos:

- Primitivas (que heredan de Primitive): es el tipo de geometría más básico. Los puntos, líneas y polígonos son ejemplos de geometrías primitivas.
- Complejas (que heredan de Complex): son geometrías compuestas por un conjunto de primitivas. En la versión gvSIG 2.0, no se soportan las geometrías complejas.
- Múltiples (que heredan de Aggregate): son geometrías formadas por la unión de varias geometrías primitivas o complejas.

Todas las geometrías existentes deberán implementar una de estas tres interfaces, en función del tipo de geometría que sea. La explicación detallada de cada uno de ellas se puede encontrar la especificación ISO19107.

Si se realiza un análisis de las imágenes presentadas en la Ilustración 14: Diagrama de geometrías primitivas de gvSIG[24], e Ilustración 15: Diagrama de geometrías de tipo múltiple de gvSIG[25], y se comparan con la Ilustración 3: Clasificación principal de

[24] http://www.gvsig.org/web/projects/gvsig-desktop/docs/devel/org.gvsig.fmap.geom/2.0.0/el-modelo-de-geometrias-1/las-primitivas

[25] http://www.gvsig.org/web/projects/gvsig-desktop/docs/devel/org.gvsig.fmap.geom/2.0.0/el-modelo-de-geometrias-1/geometrias-multiples

geometrías, y con la Ilustración 12: Geometrías soportadas por JASPA, se puede establecer la relación entre los tipos geométricos definidos en el estándar "Simple Feature Access - Part 1: Common Architecture", los soportados por JASPA, y los soportados por gvSIG.

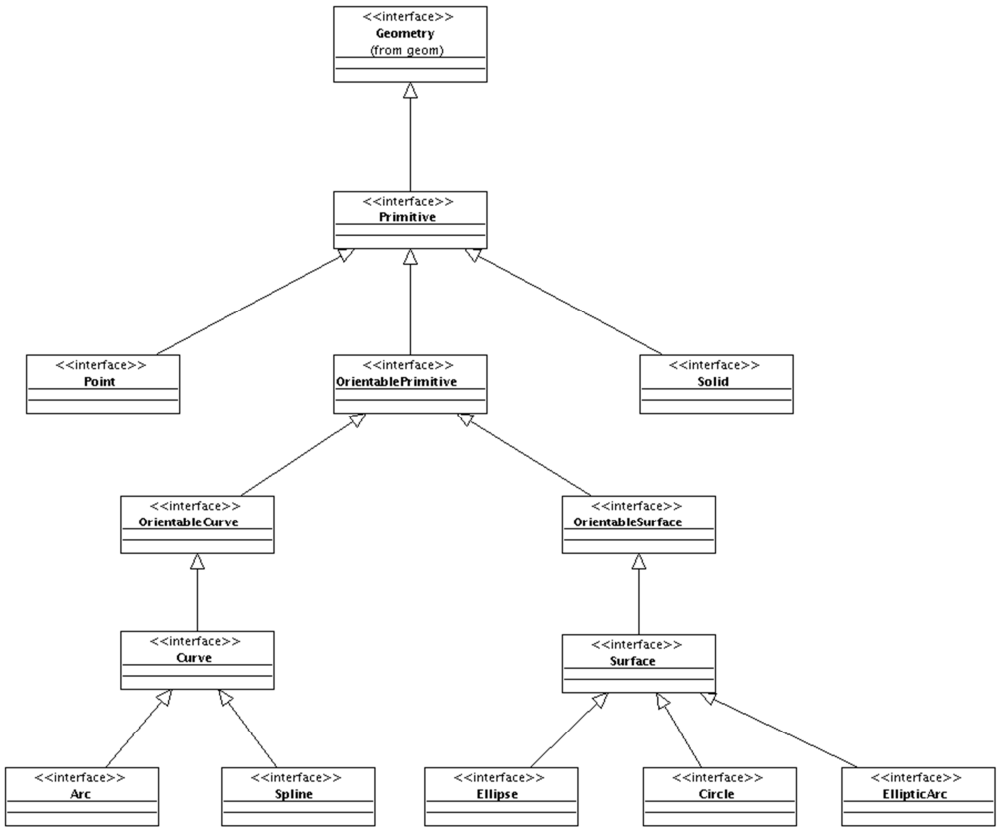

Ilustración 14: Diagrama de geometrías primitivas de gvSIG

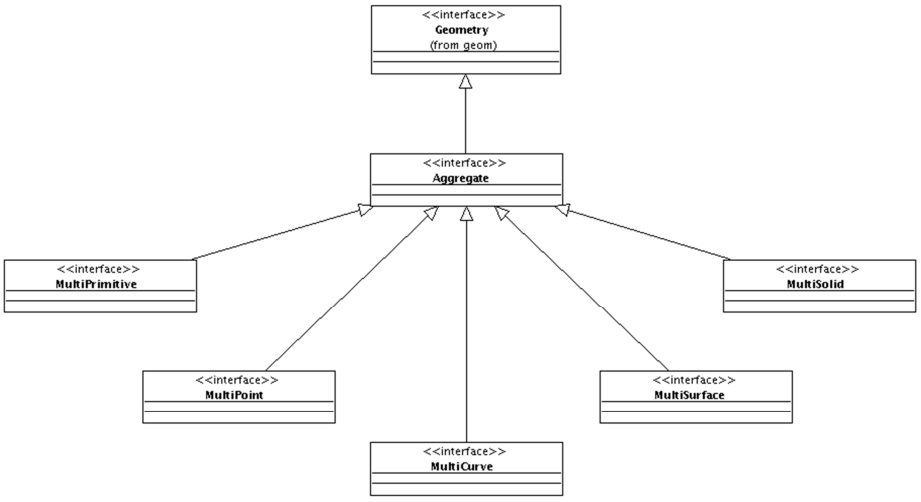

Ilustración 15: Diagrama de geometrías de tipo múltiple de gvSIG

Tras realizar el mencionado análisis, se puede obtener la equivalencia de tipos de geometrías que se muestra en la Tabla 3: Correspondencias entre tipos de geometrías, en la que se observa que una buena parte de las geometrías definidas en el estándar son soportadas directamente por JASPA y que existe una equivalencia directa en gvSIG, lo cual permitirá la conversión de todas las geometrías almacenadas JASPA en elementos gráficos que se podrán mostrar en gvSIG.

Merece la pena comentar que uno de los errores reportados durante el desarrollo de este trabajo en el bugtracker de gvSIG (el #559[26]) está relacionado con esta tabla de equivalencia, puesto convierte de LineString a Surface, y la conversión correcta es de LineString a Curve. Este error es el que finalmente produce que aunque las geometrías se pueden editar, a la hora de salvarlas en JASPA se produce una excepción indicando que el tipo de geometría que se intenta almacenar no es compatible con el definido en la columna donde se ha de guardar.

JASPA 0.2	gvSIG 2.0	Simple Feature Access	Imagen
Point	Point	Point	
LineString	Curve	LineString	
Polygon	Surface	Polygon	
MultiPoint	MultiPoint	MultiPoint	
MultiLineString	MultiCurve	MultiLineString	
MultiPolygon	MultiSurface	MultiPolygon	

Tabla 3: Correspondencias entre tipos de geometrías

[26] https://devel.gvsig.org/redmine/issues/559

4. Análisis de requisitos

Para la elaboración del listado de requisitos se ha partido del enunciado del proyecto, y se ha completado añadiendo ciertas características que si bien no están explícitamente solicitadas en el enunciado se ha considerado que están implícitas por el hecho de que el producto final debe cumplir con ciertas características básicas de los SIG.

En la Tabla 4: Listado de requisitos, se enumeran todos los requisitos identificados, y se distinguen en la columna Tipo, lo requisitos "explícitos" obtenidos del enunciado y los "implícitos", que son derivados de un requisito explícito. La numeración se ha realizado de tal forma que los requisitos implícitos se consideran como sub-requisitos de un requisito explícito. Así mismo, dentro del tipo, se define si el requisito es funcional o técnico.

Nº	Tipo	Requisito	Comentarios
1	Explícito Técnico	El driver debe ser compatible con la versión 2 de gvSIG	gvSIG 2.0 se encuentra en desarrollo por lo que es posible que contenga ciertos errores que afecten al desarrollo del driver, y que no sean solucionados hasta después de la finalización del PFC.
1.1	Implícito Técnico	Verificar que el driver funciona sobre una instalación de gvSIG 2.0 obtenida a través de su página de descarga.	Si durante el desarrollo del driver se identifican errores o modificaciones necesarias para su implementación, deben ser notificadas para su corrección en el código fuente de gvSIG y realizar las adaptaciones necesarias para que funcione con la versión disponible.
2	Explícito Técnico	El driver debe ser compatible con la versión 0.2 de JASPA	
3	Explícito Técnico	El driver debe dar acceso a la información almacenada en una base de datos H2	JASPA 0.2 tiene realmente dos implementaciones JASPA4PG y JASPA4H2, por lo que el driver ha de ser

			compatible con JASPA2H2
3.1	Implícito Funcional	El driver debe permitir visualizar las geometrías de los elementos	
3.2	Implícito Funcional	El driver debe permitir acceder a la información no geométrica de los elementos.	
3.3	Implícito Funcional	El driver debe permitir la edición de elementos geométricos (features)	
3.4	Implícito Funcional	El driver debe soportar diferentes sistemas de referencias de coordenadas	
4	Explícito Funcional	El driver debe disponer de un manual de instalación	
5	Explícito Funcional	El driver debe disponer de un paquete de instalación	
6	Explícito Funcional	El driver debe disponer de un manual de usuario	
7	Explícito Técnico	El código fuente debe estar convenientemente documentado	
8	Implícito Funcional	El driver debe permitir la creación de nuevas tablas con geometrías en la base de datos	Este es un requisito implícito en el requisito 3, pero que no es usado habitualmente. De hecho, hay que activar las características avanzadas de gvSIG para poder probarlo.

Tabla 4: Listado de requisitos

5. Diseño software

Antes de profundizar en el diseño del driver propiamente dicho, es imprescindible conocer la infraestructura de clases de gvSIG sobre la cual se ha realizado el desarrollo. Una vez descrita dicha infraestructura, se analizarán los detalles específicos de la implementación del driver, que como se verá más adelante, está formado por dos paquetes java: el complemento y el driver propiamente dicho.

5.1. Introducción a la arquitectura gvSIG

Para el desarrollo del driver, gvSIG ofrece ciertas funcionalidades que facilitan el desarrollo y define las interfaces que ha de cumplir el driver para que la plataforma pueda interactuar con él.

La información detallada de la arquitectura y facilidades que ofrece gvSIG está disponible en un apartado de la web de gvSIG destinado a desarrolladores[27]. La información que se incorpora a continuación es un resumen de dicho apartado focalizado exclusivamente en la parte de interés para el desarrollo del driver JASPA.

A continuación se verá cómo, siguiendo la recomendación de gvSIG, el desarrollo del nuevo complemento se ha de dividido en dos módulos. El primero módulo contiene las clases encargadas del registro e inicialización del complemento, y el segundo las clases que implementan realmente la librería que contiene el driver de acceso a JASPA.

Para la integración de estos dos módulos en gvSIG, se dispone de un framework[28] que actúa como constructor e inicializador de toda la estructura del aplicativo. A ese framework se denomina Andami, y a los complementos que se dan de alta, se denominan, dentro de Andami, extensiones.

Es tal la flexibilidad de Andami, que prácticamente todo el aplicativo gvSIG no es más que un conjunto de complementos que colaboran entre sí coordinados por este framework.

En la Ilustración 16: Diagrama general de bloques de gvSIG[29], se observa cómo gvSIG (aplicación) se compone del Framework Andami, y un conjunto de complementos que utilizan diferentes librerías.

---

[27] http://www.gvsig.org/web/projects/gvsig-desktop/docs/devel/developers_quick_start/2-0.0

[28] Se ha optado por dejar el término en inglés porque no existe una traducción comúnmente aceptada. Para una definición más extensa, consultar en http://es.wikipedia.org/wiki/Framework

[29] http://www.gvsig.org/web/docdev/docs/v1_0/gvsig/andami/andami.img/bloques-gvsig-general.png/view

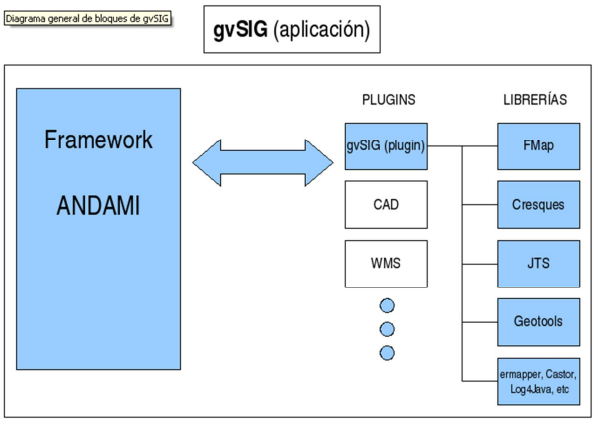

Ilustración 16: Diagrama general de bloques de gvSIG

Algunas de las librerías son comunes a varios complementos, y en consecuencia, se manejan dependencias entre módulos y librerías.

La diversidad de módulos es muy grande: unos actúan como bibliotecas de clases, otros están especializados en la presentación de los datos, otros en el acceso a los datos, etc. La combinación de todos ellos genera finalmente el producto gvSIG.

Pasemos por lo tanto a estudiar los tres principales elementos que se necesitan para el diseño del driver: framework Andami, complementos y librerías.

5.1.1. El framework Andami

Andami, tal y como se ha presentado anteriormente, es la herramienta que utiliza gvSIG para componer el aplicativo. Se puede observar en la Ilustración 17: Diagrama de componentes de Andami[30], que se compone de cuatro módulos:

- Gestión de ventanas
- Gestión de interfaz gráfica
- Gestión de complementos (plugins) y extensiones
- Servicios a los plugins.

La gestión de ventanas permite controlar la aparición de las mismas, su número, su ordenación, etc.

La gestión de interfaz gráfica permite controlar los menús, barras de herramientas y barra de estado.

Los servicios a los complementos ofrecen cierta funcionalidad de uso habitual por parte de los complementos, pero que en el caso del driver JASPA, no se utilizan, y por

[30] http://www.gvsig.org/web/docdev/docs/v1_0/gvsig/andami

lo tanto no se profundizará en ellos.

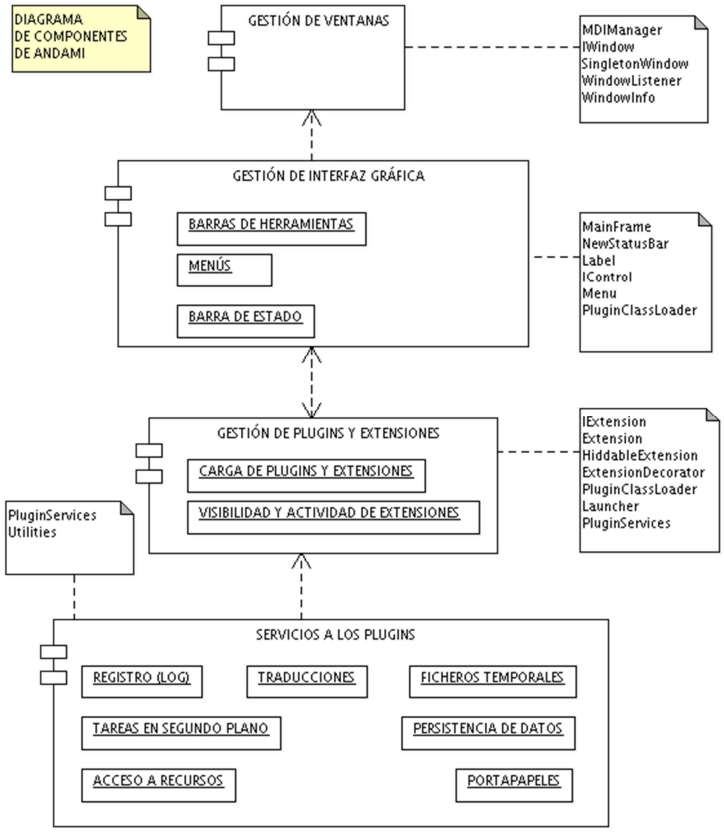

Ilustración 17: Diagrama de componentes de Andami

Finalmente, para el desarrollo del driver JASPA objeto de este proyecto, la gestión de complementos y extensiones es el único módulo de Andami que se ha de analizar con detenimiento, puesto que en él se encuentran las claves de la integración del driver JASPA.

5.1.1.1. Complementos y extensiones

Como se mencionó anteriormente, en una instalación de gvSIG, encontraremos una carpeta gvSIG/extensiones. Esta carpeta contiene los distintos complementos del aplicativo. Un complemento no es más que un conjunto de clases Java que añaden nuevas funcionalidades a la aplicación.

Si inspeccionamos el mencionado directorio gvSIG/extensiones, observaremos que existe un complemento llamado com.iver.core, que contiene libCorePlugin (el skin de Andami) y un complemento llamado com.iver.cit.gvsig, que contiene el complemento gvSIG (el complemento principal de la aplicación). Dependiendo de la instalación de gvSIG que se haya realizado, se pueden localizar otros complementos como com.iver.cit.gvsig.cad (funcionalidades de edición de cartografía), org.gvsig.scripting (que añade las capacidades de scripting), com.iver.cit.gvsig.wms (cliente de WMS), etc.

Las nuevas funcionalidades añadidas por los complementos, deben conectarse de alguna forma con el resto del aplicativo. Este punto de conexión lo aportan las extensiones, que son clases Java que implementan la interfaz IExtension, y que hacen de puente entre la funcionalidad ya existente y la nueva funcionalidad aportada por el complemento. Por lo tanto, cada complemento puede contener una o más extensiones funcionales en gvSIG.

En la Ilustración 18: La interfaz IExtension, se muestra de forma esquemática las clases que intervienen en la activación de una funcionalidad mediante la implementación del interface IExtension.

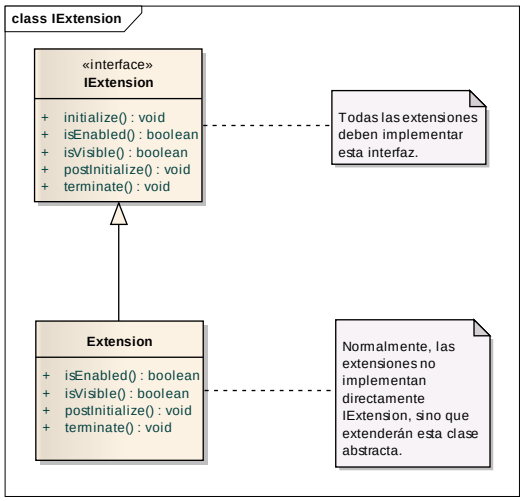

Ilustración 18: La interfaz IExtension

Todas las extensiones han de implementar el interfaz IExtension, del cual tienen interés para el proyecto los siguientes métodos:

- public void initialize() Una vez se leen y ordenan todas las extensiones presentes en los complementos, se llama a este método para cada una de ellas, de forma secuencial siguiendo el orden definido en la prioridad de las extensiones.
- public void postInitialize() Una vez se han inicializado todas las extensiones, se llama a este método para cada una de ellas, siguiendo el mismo orden de antes.
- public void terminate() Al salir de la aplicación, se llama a este método para cada una de las extensiones, siguiendo un orden secuencial inverso al que se siguió en la inicialización (es decir, la última extensión que se inicializó es la primera que se finalizará, y la primera extensión que se inicializó será la última en ser finalizada).
- public boolean isEnabled() Los elementos de interfaz de usuario (botones, menús, controles de la barra de estado) asociados a esta extensión estarán activados (o desactivados) dependiendo del valor devuelto por este método. Cuando un botón está desactivado, aparece en gris claro y no es posible pulsar

sobre él. En la implementación de la extensión de JASPA, este método devuelve false.
- public boolean isVisible() Los elementos de interfaz de usuario (botones, menús, controles de la barra de estado) asociados a esta extensión estarán visibles dependiendo del valor devuelto por este método. Cuando un botón está invisible, no se mostrará independientemente del valor de isEnabled().En la implementación de la extensión de JASPA, este método devuelve false.

gvSIG, ofrece una clase abstracta Extension, que ya implementa IExtension. Esta clase permite que si nuestra extensión hereda de ella, a futuro, si se modifica el interfaz IExtension, no será necesario modificar nuestra extensión puesto que la implementación la haría la clase padre.

En el caso del driver JASPA, la clase que implementa el interfaz IExtension, se denomina JASPAExtension y hereda de la clase abstracta Extension. Esta clase, lo único que hace es indicar que no tiene elementos visibles y que no tiene elementos habilitados, puesto que, como ya veremos más adelante, donde se realiza la verdadera inicialización del driver es en la clase Default

5.1.1.2. Estructura de un complemento

Como se ha mencionado anteriormente, cada complemento está contenido en una subcarpeta, y dentro de esta, existen dos ficheros de interés:

- config.xml: describe dónde encontrar las clases de las que depende el módulo, qué opciones de menú, barras de herramientas y botones añade el complemento al aplicativo. El contenido del archivo config.xml del complemento JASPA es el siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<plugin-config>
      <depends plugin-name="org.gvsig.app" />
      <depends plugin-name="org.gvsig.geodb"/>
      <resourceBundle name="text"/>
      <libraries library-dir="lib"/>
      <extensions>
            <extension class-name=
           "org.gvsig.fmap.dal.db.store.jaspa.app.extension.JASPAExtension"
                  description="Support to access JASPA databases"
                  active="true">
            </extension>
      </extensions>
</plugin-config>
```

- package.info: este archivo, aunque es opcional, se suele encontrar en la carpeta, puesto que es el que informa sobre la versión, nombre, identificador o descripción del complemento. Con esta información gvSIG nos permite identificarlo para poder actualizarlo, desinstalarlo o reinstalarlo. El contenido del archivo package.info del complemento JASPA es el siguiente:

```
#Tue Jun 05 20:45:51 CEST 2012
owner=UOC
code=org.gvsig.fmap.dal.db.store.jaspa.app.extension
java-version=j1_6
official=false
type=plugin
version=1.0.0-SNAPSHOT-1
state=devel
operating-system=all
dependencies=
sources-url=
web-url=http\://www.uoc.edu
architecture=all
model-version=1.0.1
categories=
description=Extension of the JASPA project
buildNumber=1
gvSIG-version=2.0.0
name=org.gvsig.fmap.dal.db.store.jaspa.app.extension
```

5.1.2. Las librerías de gvSIG

Como hemos visto anteriormente, los tres principales elementos que se necesitan para el diseño del driver son: el framework Andami, que ha sido descrito anteriormente junto con los complementos que son el segundo elemento, y para completar el análisis se pasará a describir las librerías.

Las librerías se definen de forma general como piezas de código que aportan una funcionalidad, la cual se activa mediante un mecanismo de inicialización que se describe a continuación.

5.1.2.1. Inicialización de librerías

En gvSIG existe una API para la inicialización y configuración de librerías similar a la usada para la inicialización de complementos.

El API es muy sencillo, consiste en una clase interfaz Library que define dos métodos muy parecidos a los del interfaz IExtension de Andami: initialize y postInitialize. Por conveniencia, existe una clase abstracta AbstractLibrary equivalente a la clase abstracta Extension que facilita el desarrollo con una implementación por defecto de los métodos initialize y postInitialize.

Se describe las relaciones entre las clases en la Ilustración 19: Diagrama de clases para inicialización de librerías

La implementación de estos métodos realiza ciertas acciones de registro, y cuando finalizan, llaman a los métodos doInitialize y doPostInitialize respectivamente, que son los métodos que se redefinen en las clases heredadas para realizar efectivamente la inicialización.

Estas acciones de registro que realiza la clase AbstractLibrary están relacionadas con el control de la ejecución para que una librería se inicialice sólo una vez dentro aunque se invoque varias veces a sus métodos initialize/postInitialize, garantizando que los métodos doInitialize y doPostInitialize se llamarán una sola vez.

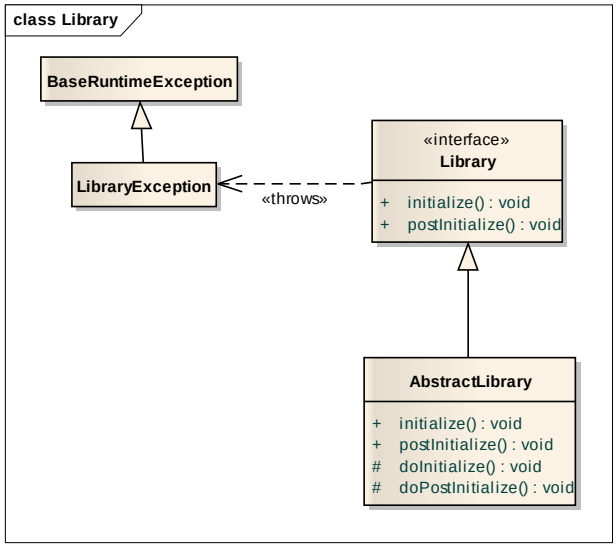

Ilustración 19: Diagrama de clases para inicialización de librerías

El orden de inicialización de librerías y extensiones en gvSIG es el siguiente:

1. Llamada a los métodos initialize de todas las librerías.
2. Llamada a los métodos initialize de todas las extensiones.
3. Llamada a los métodos postInitialize de todas las librerías.
4. Llamada a los métodos postInitialize de todas las extensiones.

La lista de librerías a inicializar en cada complemento se obtiene de los meta-datos de las librerías que lo componen. gvSIG busca estos meta-datos en una subcarpeta de nombre "services" que se encuentra en la carpeta META-INF de la librería según se muestra en la Ilustración 20: Meta-datos para registro de librerías

Ilustración 20: Meta-datos para registro de librerías

Este archivo contiene exclusivamente el nombre completo de las clases que hay que instanciar e inicializar. Por ejemplo, en el caso de la librería para JASPA, el contenido del archivo services es el siguiente:

```
org.gvsig.fmap.dal.db.store.jaspa.impl.JASPALibraryImpl
```

Para poder instanciar las librerías, gvSIG utiliza una característica especial de Java denominada "Reflection", que permite al código en ejecución, entre otras cosas, crear un objeto de una clase a partir exclusivamente del nombre de la misma.

Si bien la implementación de este mecanismo de inicialización puede parecer compleja, para el desarrollador todo esto es automático, y gvSIG garantiza que los métodos de la clase que implementa el interface Library serán llamados convenientemente.

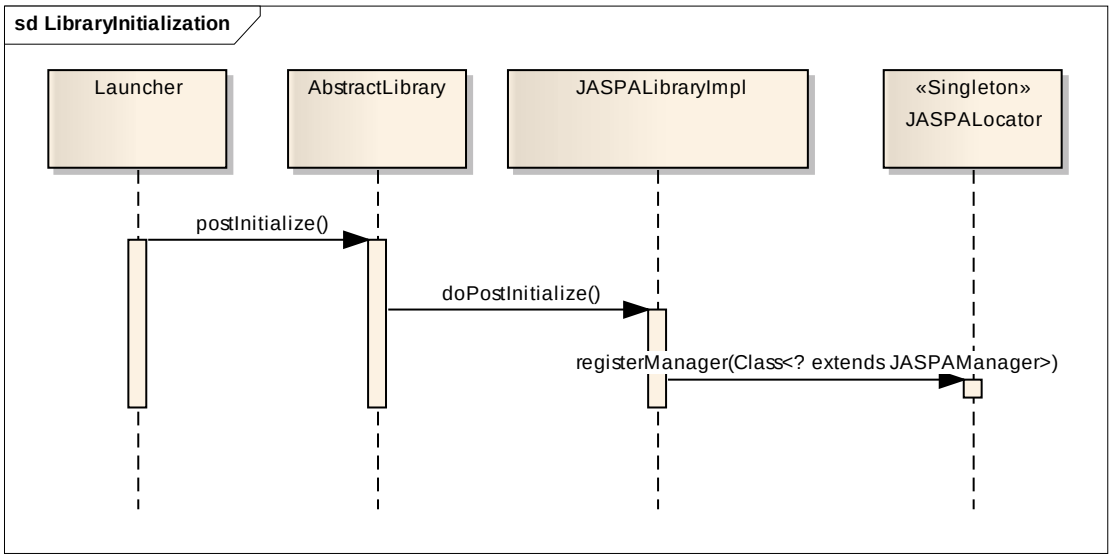

Ilustración 21: Diagrama de secuencia de inicialización del driver JASPA

La secuencia de inicialización del complemento se describe en la Ilustración 21: Diagrama de secuencia de inicialización del driver JASPA.

En la operación doInitialize de nuestra implementación de la librería, se utiliza la clase singleton JASPALocator, que es el punto de entrada para registrar y posteriormente localizar la clase JASPAManager, que permitirá obtener las clases que implementan las interfaces JASPAServerExplorer y JASPAStoreProviderWriter.

De toda la librería tools, son destacables tres conceptos básicos que se utilizan en el desarrollo de drivers, puesto que se utilizan para separar de forma estricta el API de la implementación:

- Utilidades para la separación entre API e implementación (registro de API, registro de implementaciones para las API, registro de proveedores de servicios SPI o localización de una implementación de un API)

- Managers. Se trata de componentes que aportan una funcionalidad al aplicativo, haciendo las veces de factoría para los distintos servicios que ofrece la librería, así como de configuración de estos. Actúan a modo de singleton, ya que normalmente en el aplicativo solo existe una instancia para cada uno de estos componentes. Toda librería tendrá uno a más Managers según la lógica de esta.

- Locators. Un Locator es un objeto que implementa el patrón de diseño Locator mostrado en la Ilustración 22: Patrón Locator y que proporciona referencias a componentes, servicios u objetos en general de un aplicativo. En el caso que nos ocupa, permite registrar implementaciones de managers para una librería dada mediante un nombre, así como recuperar una implementación de manager de un API en concreto a partir del nombre dado.

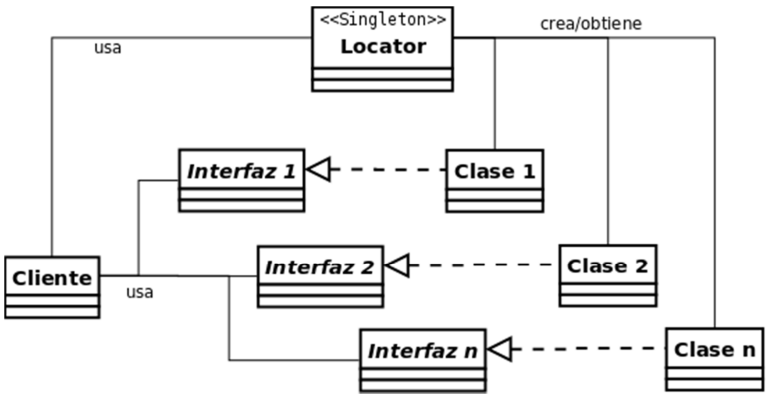

Ilustración 22: Patrón Locator

Este patrón permite separar cómo se gestiona o instancia un objeto, de la propia implementación del objeto. Además, esta independencia también aplica a los clientes del objeto, que no tienen que conocer cómo deben acceder o instanciar una referencia a un objeto, delegando esta tarea en el Locator.

De esta forma, podemos sustituir la opción de tener un Singleton por cada objeto del aplicativo o sistema, por un Locator. Aunque para la implementación de un Locator se suele emplear el patrón Singleton.

Dependiendo del tamaño y estructura del aplicativo, se suele tener un único Locator para todo, uno por cada sistema o librería, etc.

Hay que destacar que el patrón Locator no es un patrón de creación, como los de tipo Factory, sino que sólo se centra en la obtención de referencias. La creación de las instancias que devolverá el Locator queda fuera del patrón.

Para obtener una información más detallada del uso del Locator en gvSIG, se recomienda la lectura del apartado "org.gvsig.tools.locator"[31] de la web de gvSIG.

Uno de los casos de uso más habituales será el registro de implementaciones en los Locator. Cada librería que emplee el mecanismo del Locator deberá implementar un Library para su API, y otro para su implementación (ver ejemplo en org.gvsig.tools.locator).

[31] http://www.gvsig.org/web/projects/gvsig-desktop/docs/devel/org.gvsig.tools/2.1.0/org-gvsig-tools-locator

5.1.3. Biblioteca de acceso a datos (DAL)

La descripción detallada de la biblioteca usada por gvSIG para el acceso a los datos geográficos se puede encontrar en la web de gvSIG[32], pero con la finalidad de entender el diseño software del proyecto, se muestra a continuación un resumen adaptado al proyecto, puesto que la explicación detallada escapa del objetivo de este documento.

5.1.3.1. Introducción a la DAL

En la versión 2 de gvSIG se ha desarrollado una nueva biblioteca[33] de acceso a datos, denominada DAL por sus siglas del inglés Data Access Library, la cual ha dotado a gvSIG de una capa de abstracción que permite a los consumidores de datos del aplicativo, operar de forma homogénea con diferentes fuentes y formatos de datos.

Con esta característica, nuevos proveedores de datos como el desarrollado en el ámbito de este proyecto, pueden integrarse en gvSIG sin necesidad de modificar el código fuente del mismo.

Esta independencia se consigue a través de la definición de una API de acceso a datos que es usada por los consumidores sin conocer de su implementación. Posteriormente, los distintos proveedores de datos, realizan diferentes implementaciones de dicha API, donde cada implementación depende del origen de datos con el que se conectan, pero que los consumidores no necesitan conocer.

En la Ilustración 23: Contexto de operación de la capa de acceso a datos (DAL)[34], se muestran los componentes que forman parte de la librería de acceso a datos:

- La librería de acceso a datos
- Los distintos almacenes de datos
- Las operaciones, manejo de leyendas específicos de algunos formatos.

Íntimamente relacionados con estos componentes estarían las librerías de geometrías, y las librerías de manejo de DWG y DXF.

---

[32] http://www.gvsig.org/web/projects/gvsig-desktop/docs/devel/org.gvsig.fmap.dal

[33] En la documentación de gvSIG se denomina librería, pero por no mezclar con el concepto de la clase Library descrito anteriormente, se ha optado por usar el nombre biblioteca, con el que también se conoce comúnmente a un conjunto de clases java.

[34] http://www.gvsig.org/web/projects/gvsig-desktop/docs/devel/org.gvsig.fmap.dal/2.0.0/contexto

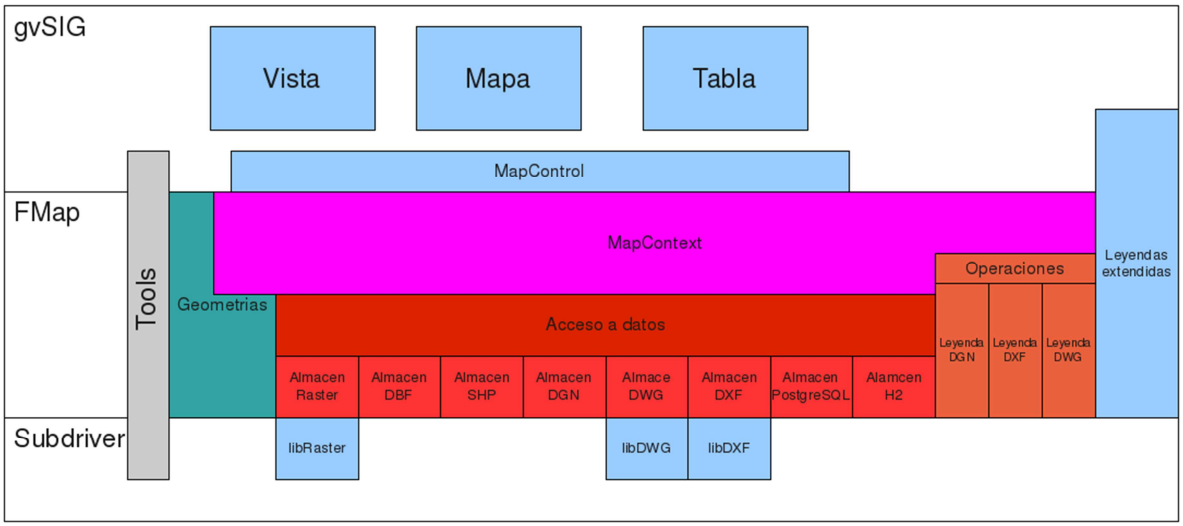

Ilustración 23: Contexto de operación de la capa de acceso a datos (DAL)

5.1.3.2. Descripción general de la arquitectura de DAL

El acceso a datos de gvSIG contiene dos grandes conjuntos de interfaces:

El primero describe API para consumidores de datos, es decir, el interface que ofrece la capa de acceso a datos para que, independientemente del proveedor de datos específico con el que se conecte, un consumidor de datos pueda obtener información de la fuente de datos.

El segundo describe la SPI (Service Provider Interface), que es la estructura de clases que un nuevo proveedor de datos debe extender, para que la librería de acceso a datos pueda exponer los servicios del nuevo proveedor a los consumidores de datos.

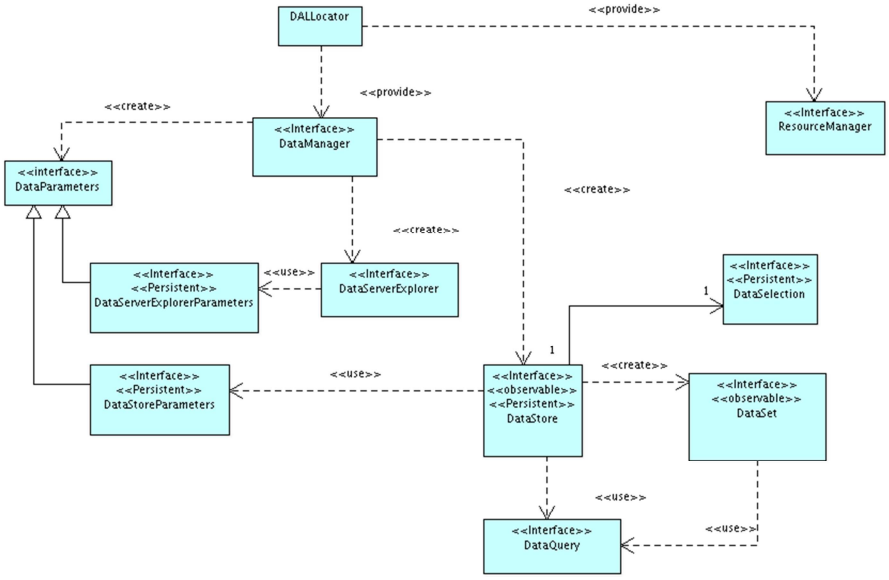

Ilustración 24: API genérica de acceso a datos de la DAL

Como se indicó en el apartado Características de la información espacial de este documento, la información geográfica puede ser manejada en dos formatos diferentes: raster y vectorial.

El API genérico de la DAL, define una capa de acceso independiente de si el formato de los datos es raster o vectorial. A partir de esta primera capa de abstracción que se muestra en la

Ilustración 24: API genérica de acceso a datos de la DAL[35], aparecen dos grandes especializaciones de ella. Por un lado una especialización que nos da acceso a datos de tipo tabular, bien sean de tipo alfanumérico o vectorial y por otro, una especialización que permite el acceso a datos raster.

Las clases principales que componen el diagrama son:

- DataManager. Se trata de la factoría que nos da acceso al API de acceso a datos. A partir de él podemos explorar los almacenes de datos que nos suministra un servidor o servicio dado, así como acceder a los almacenes de datos contenidos en el servidor. DataManager es el punto de entrada a todo el modelo de objetos.

- DataServerExplorer. Nos permite obtener la lista de almacenes disponibles en un servidor o servicio dado. Así podemos pedirle que nos dé la lista de tablas de una base de datos o de ficheros susceptibles de ser tratados como almacenes, por ejemplo archivos SHP, DBF o DXF, que existan en una capeta dada. A partir de la información que nos suministre sobre cada almacén, podremos crear el DataStore adecuado para acceder a sus datos.

- DataExplorerParameters. Representa el juego de parámetros que necesita el DataServerExplorer para poder acceder al servidor o servicio y poder consular los almacenes de datos que éste sirve.

- DataStore. Representa un almacén de datos. Un fichero SHP o una tabla de una base de datos, y dispone de mecanismos para acceder a sus datos, su estructura y tipo, así como en qué forma se pueden modificar éstos.

- DataStoreParameters. Representa el conjunto de parámetros que se necesitan para poder acceder al almacén de datos. Por ejemplo, si estamos accediendo a un fichero DBF, contendrá la ruta al fichero, si estamos accediendo a una base de datos JDBC, contendrá información tal como el nombre o la dirección IP del host donde se encuentra la base de datos, el puerto TCP de acceso, el nombre de la base de datos, el usuario y la contraseña para autenticación, etc.

- DataQuery. Representa el conjunto de valores que conforman las condiciones

[35] http://www.gvsig.org/web/projects/gvsig-desktop/docs/devel/org.gvsig.fmap.dal/2.0.0/descripcion-general-de-la-arquitectura

en las que se basará el DataSet para recuperar y devolver los datos.

- DataSet. Representa un conjunto de datos del almacén de datos. El DataStore contiene mecanismos para acceder a los datos, permitiendo aplicar filtros u órdenes a éstos, así como información contextual que pueda ser útil a la hora de decidir cómo ha de realizarse de forma óptima la recuperación de éstos desde el almacén.

- DALLocator. Se trata del localizador de la librería. Nos proporciona los servicios de localización del DataManager disponible en la librería. En general su uso se basa en obtener el DALLocator a partir del nombre de la librería, y una vez localizado, obtener de éste su DataManager.

Con este conjunto de clases, se obtienen dos servicios de gran utilidad:

- Acceso a la descripción interna de un servidor de datos.

  La clase DataServerExplorer permite determinar qué almacenes de datos hay en un proveedor de datos, así como crear nuevos almacenes. Los principales tipos de servidores soportados en gvSIG son:

  - Almacenamiento de información espacial en archivos. Esta funcionalidad está soportada por la clase FilesystemServerExplorer.
  - Sistemas de bases de datos que soportan conexión JDBC. Esta funcionalidad está implementada en la clase JDBCServerExplorer, y como veremos a continuación, es la clase que se ha usado en el proyecto para la implementación del driver.
  - Sistemas remotos basados en WMS/WCS.
  - Sistemas remotos basados en WFS/WFS-T.

- Acceso a los datos de un almacén de datos.

  La clase DataStore es la que permite a los consumidores el acceso a datos geográficos contenidos en almacenes de datos de diversos tipos de manera homogénea. Los almacenes de datos soportados por gvSIG son:

  - Archivos DBF/SHP
  - Archivos DXF
  - Archivos GML
  - Bases de datos JDBC
  - Otros

Dado que el tipo de datos que debe manejar el proveedor de datos a implementar para el proyecto es de tipo vectorial, se describe a continuación la especialización de tipos de datos tabulares y vectoriales, asumiendo que los datos vectoriales se almacenan en tipos de datos geométricos de los datos tabulares según recomienda el estándar del OGC estudiado en los apartados El estándar “Simple Feature Access - Part 1: Common Architecture” y El estándar “Simple Feature Access - Part 2: SQL Option”.

5.1.3.3. Acceso a datos tabulares

Partiendo de la arquitectura general del API de acceso a datos tratada anteriormente, pasaremos a analizar cómo se realiza la especialización para el acceso a datos tabulares.

Dentro del acceso a datos tabulares se incluye tanto el acceso a datos de carácter alfanumérico como vectorial, y en ambos casos, las entidades pueden tener varios atributos geométricos y varios atributos no geométricos.

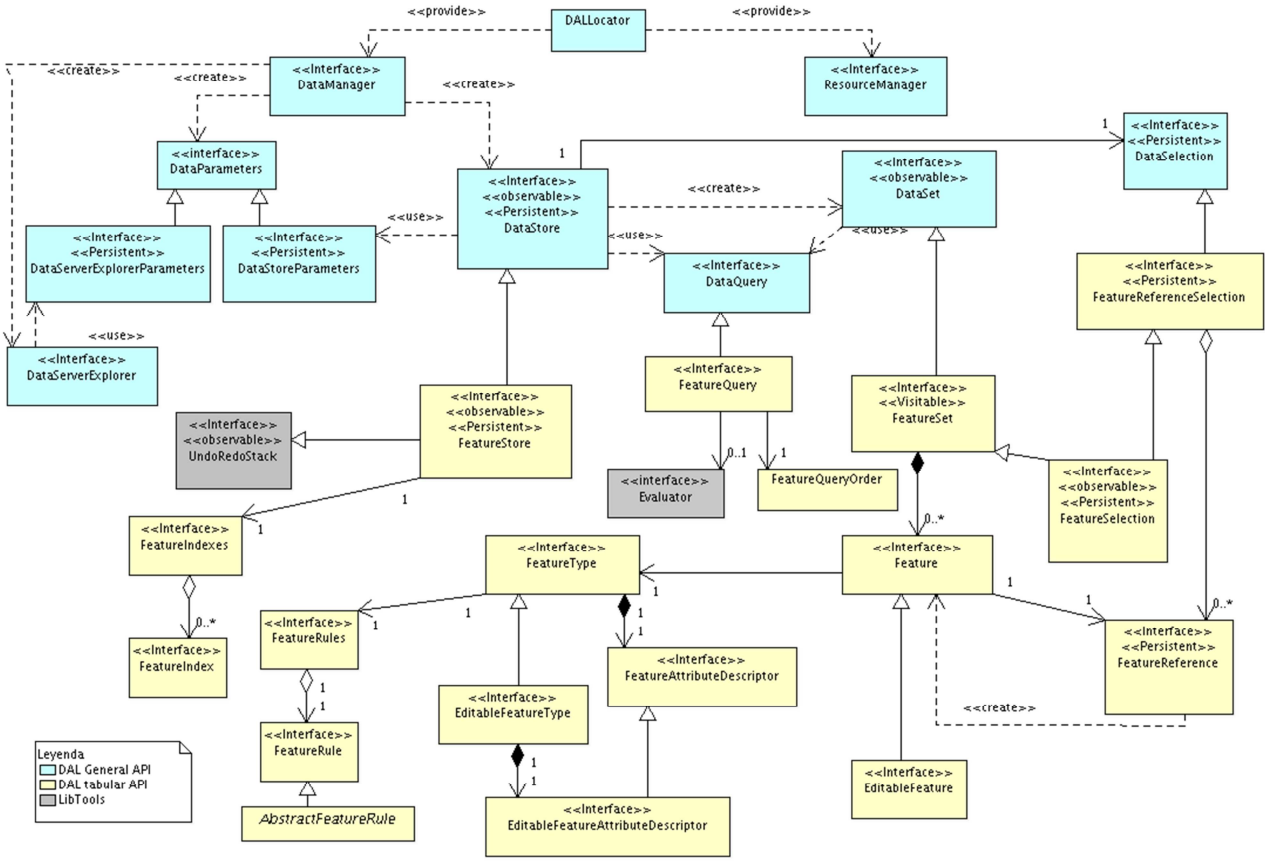

Ilustración 25: Diagrama de clases de acceso a datos tabulares

Los recuadros con fondo en fondo amarillo que se muestran en la Ilustración 25: Diagrama de clases de acceso a datos tabulares, la cual ha sido obtenida de la web www.gvsig.org[36], ilustran la especialización de las clases generales al caso de acceso a datos tabulares.

[36] http://www.gvsig.org/web/projects/gvsig-desktop/docs/devel/org.gvsig.fmap.dal/2.0.0/featurestore/descripcion-general

Las clases que especializan a las generales para el caso de datos tabulares comienzan por la palabra Feature (en español "fenómeno") para manifestar su relación con las entidades vectoriales en lugar de Data que es un concepto más genérico. Las principales clases del diagrama son:

- FeatureStore, como especialización de DataStore. Añade funcionalidades propias del acceso a datos alfanuméricos y vectoriales. Conoce de fenómenos, cómo consultar sus valores o su estructura, así como de acciones específicas para acceder a ellos.
- FeatureQuery, como especialización de DataQuery. Contiene información relevante sobre la definición de filtros y su ordenación que hace uso del conocimiento de estar trabajando sobre datos alfanuméricos y vectoriales.
- FeatureSet, como especialización de DataSet. Conoce los fenómenos y es capaz de iterar sobre ellos.
- Feature. Aparece como contenedor de un fenómeno, permitiendo acceder a la información de éste.
- FeatureType. Aparece como el contenedor de la estructura de un fenómeno. Qué atributos tiene o de qué tipo son.

Estas clases descritas hasta ahora no son más que la definición de una API, es decir, la definición del interfaz que deben cumplir aquellas clases que pretendan ofrecer a los consumidores de datos de tipo tabular existentes en gvSIG. En función del almacén de datos con el que interactuará el driver específico, se crean nuevas clases que implementen dicho interfaz y le provean de la funcionalidad adaptada al almacén de datos.

5.2. Diseño del acceso a datos JASPA

Como se ha analizado en el apartado Introducción a la arquitectura gvSIG, hay dos módulos a desarrollar para integrar un nuevo proveedor de datos JASPA a gvSIG y finalmente hay que empaquetar dichos módulos en un instalador. A continuación se describe con mayor detalle estas actividades.

Por un lado es necesario implementar el nuevo proveedor de datos JASPA que cumpla con la API que los consumidores de datos vectoriales esperan. Este proveedor de datos debe exponer su funcionalidad a través de una nueva librería que deberá de heredar de la clase AbstractLibrary para registrarse e inicializarse. Este registro permitirá a gvSIG instanciar las clases que permiten el acceso a JASPA solamente a través de su nombre, sin necesidad de conocer su implementación, puesto que gvSIG solo necesita conocer el API que implementan.

Por otro lado, será necesario desarrollar un complemento que se deberá instalar en gvSIG y que permitirá al framework Andami identificar e inicializar la librería JASPA.

Por último, una vez desarrollados el driver y el complemento, y tras instalarlo en una versión de gvSIG de desarrollo, es posible crear un paquete de instalación para que el nuevo complemento pueda ser instalado fácilmente por los usuarios finales mediante el gestor de paquetes de gvSIG.

Antes de comenzar a analizar el detalle del diseño del complemento y de la librería, se describirán un par de elementos de gvSIG que se pueden crear a través de su interfaz de usuario y que serán básicos para entender la explicación del diseño que se mostrará a continuación.

Los elementos en concreto son la vista y la tabla.

- Vista: una vista es una representación gráfica de un conjunto de geometrías accesibles a través de un proveedor de datos geográficos, por lo tanto, para visualizar y modificar los datos geográficos, el punto de entrada principal será una vista.

- Tabla: una tabla es una representación tabular de un conjunto de datos, que no tienen por qué ser geográficos. Dado que JASPA no es más que una extensión geográfica de una base de datos, y que de forma subyacente, se encuentra una base de datos tradicional, el driver deberá permitir también el acceso a esos datos de manera tabular.

5.2.1. Diseño del driver JASPA

Una decisión fundamental a tomar en el diseño del driver JASPA es si implementar los interfaces directamente, o apoyarse en una implementación de referencia ya existente. Una correcta decisión del punto de partida puede dar lugar a una implementación rápida y exitosa o justo lo contrario.

gvSIG dispone de una implementación de un proveedor de datos tabulares basada en un driver genérico JDBC. Dado que JASPA funciona sobre una base de datos que soporta conexiones JDBC, se ha tomado la decisión de diseño de que el driver JASPA extienda y especialice la implementación básica JDBC, añadiendo las características espaciales de JASPA y las particularidades del motor de bases de datos H2.

La implementación del interface DataServerExplorer especializado para servidores JDBC se realiza en la clase JDBCServerExplorer, si bien no se implementa directamente, sino a través de dos clases intermedias: DBServerExplorer y AbstractDBServerExplorer. La definición de estas clases intermedias no es de utilidad para entender el diseño, por lo que nos centraremos en cómo extender a partir de JDBCServerExplorer. En la Ilustración 26: Diagrama de clases de JASPAServerExplorer, se muestra cómo es la cadena de herencia de estas clases.

Los parámetros de conexión con el servidor de base de datos los obtiene de la clase JASPAServerExplorerParameters, la cual se pasa como parámetro en el constructor de la clase JASPAServerExplorer para que cuando se inicie la conexión, se pueda obtener de dicha clase los parámetros necesarios para una conexión JDBC: servidor, puerto TCP, usuario, contraseña, nombre de la base de datos, etc.

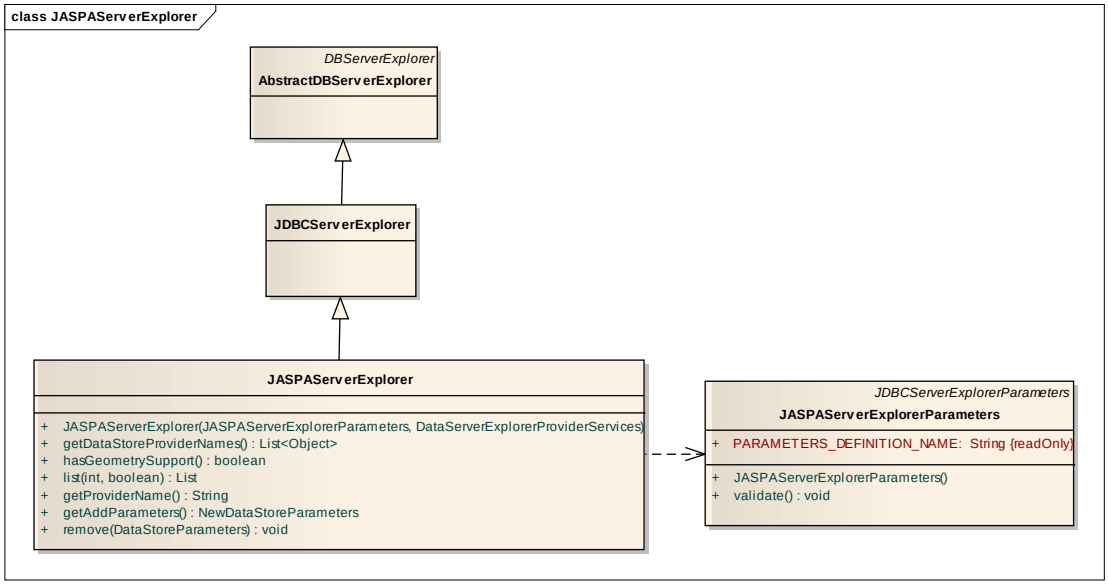

Ilustración 26: Diagrama de clases de JASPAServerExplorer

Una vez conectada al servidor, la clase puede obtener la siguiente información:

- hasGeometrySupport: indica si la base de datos solo es una base de datos relacional o soporta geometrías. En el caso de JASPA, devuelve siempre true indicando que si soporta geometrías.
- canAdd: indica si la base de datos permite añadir nuevos datos en sus almacenes de datos. En el caso de JASPA, heredado de JDBC, devuelve false si la base de datos está en modo solo lectura.
- list: obtiene una lista con todos los almacenes de datos accesibles en la base de datos.

Una vez definidos todos los parámetros de conexión, se pueden obtener a través del método list todos los almacenes de datos (tablas) disponibles en el servidor JASPA. Por cada almacén de dato, se obtiene una descripción completa de la información geográfica que contiene.

Para el acceso a los datos almacenados en JDBC, se encuentra disponible la clase JDBCStoreProviderWriter, que implementa el interface FeatureStoreProvider mediante el cual se crean los FeatureStore usados por los consumidores de datos. Para el driver JASPA, se ha optado por heredar de la clase JDBCStoreProviderWriter ya que facilitará enormemente el desarrollo.

Las relaciones entre las clases que se acaban de describir, se muestran en la Ilustración 27: Diagrama de clases de JASPAStoreProviderWriter.

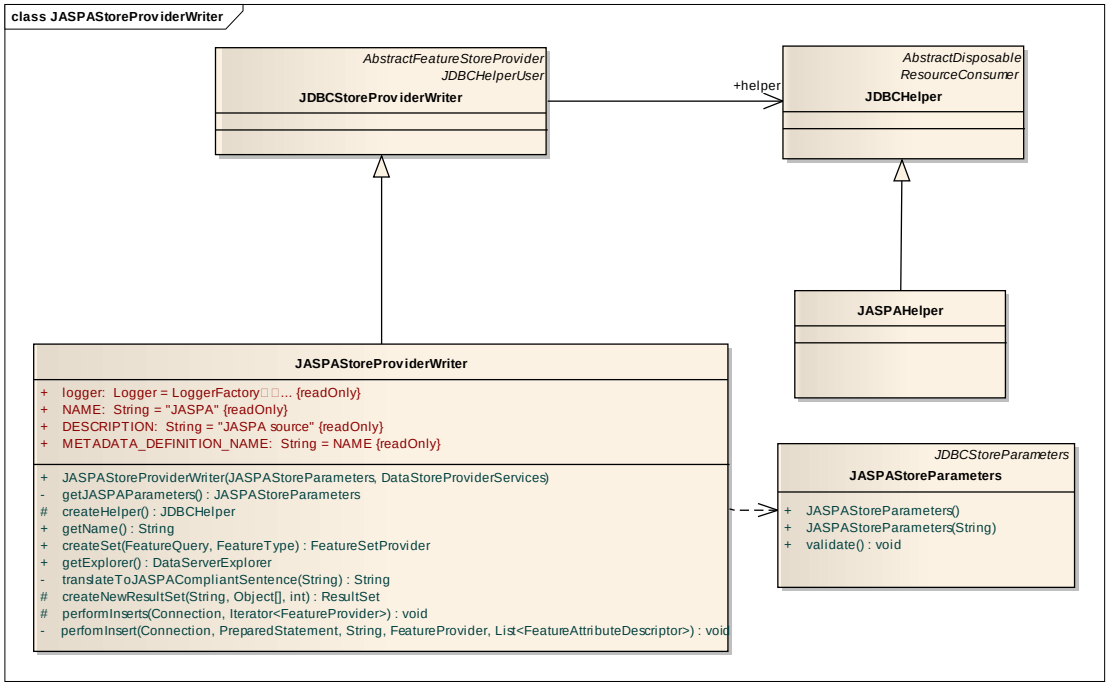

Ilustración 27: Diagrama de clases de JASPAStoreProviderWriter

Es importante resaltar que cuando gvSIG llama a alguno de los métodos de AbstractFeatureStoreProvider redefinidos en JASPAStoreProviderWriter, éstos no acceden directamente a la base de datos con JDBC, sino que usa una clase de utilidad JASPAHelper que simplifica el acceso a los datos de la base de datos, puesto que para acceder a los mismos es necesario codificar una secuencia de llamadas a rutinas de las librerías de JDBC que no son triviales y que la clase JASPAHelper encapsula en llamadas más simples, gestionando internamente las desconexiones, errores de sintaxis, errores de acceso, etc., generando excepciones en caso de que algo vaya mal. Esta clase JASPAHelper es una especialización de la clase JDBCHelper que se encargará de tratar los aspectos específicos de la base de datos JASPA respecto a la generalidad de una base de datos JDBC.

En la clase JASPAStoreProviderWriter se realiza una acción de gran importancia para el driver, que es la sustitución de sentencias SQL genéricas, por sentencias SQL compatibles con JASPA. Esta sustitución se realiza en el método translateToJASPACompliantSentence. En este método se utilizan expresiones regulares para identificar si la sentencia SQL es una sentencia genérica de gvSIG con la particularidad de forzar un CRS determinado. Cuando se identifica tal condición, se sustituye por la consulta correcta para el proveedor JASPA. En el foro de desarrolladores de gvSIG se puede seguir el hilo de comentarios al respecto[37]

[37] http://osgeo-org.1560.n6.nabble.com/Duda-acerca-del-tratamiento-de-los-CRS-td4694794.html

5.2.2. Diseño de la librería JASPA

La librería JASPA es la encargada de registrar e inicializar el nuevo proveedor de datos implementado mediante el driver JASPA.

En el apartado Las librerías de gvSIG, se describió cómo gvSIG implementa la carga e inicialización de librerías, y a continuación se muestra cómo está diseñada la librería JASPA siguiendo las recomendaciones del capítulo mencionado.

La clase encargada del registro e inicialización se denomina JASPALibraryImpl, la cual hereda, según se ha recomendado, de la clase AbstractLibrary, y se han sobrecargado los métodos doInitialize y doPostInitialize.

En el método doInitialize, se realiza el registro del manager en el Locator, para poder ser recuperado posteriormente a través del nombre. El manager que utiliza es el manager por defecto DefaultDataManager. No es necesaria ninguna especialización.

En el método doPostInitialize es donde se registran las clases que conforman el driver JASPA y las clases que almacenan los parámetros de conexión a los proveedores de datos.

En consecuencia, una vez finalizado el método doPostInitialize, para que un consumidor de datos pueda usar el driver, solo necesita conocer el nombre del Locator, puesto que a partir de su nombre, se puede obtener su única instancia, ya que JASPALocator es una clase singleton.

Con esta instancia del JASPALocator, se puede obtener su FeatureManager y con él, ya se pueden crear objetos de las clases que implementan para JASPA los interfaces FeatureServerExplorer, FeatureStore, etc.

En concreto, el consumidor de datos puede crear una instancia del JASPAServerExplorer y con ella podrá consultar los almacenes de datos disponibles en el servidor, así como información característica de cada almacén de datos, como por ejemplo, las columnas que almacenan geometrías, su clave primara, etc.

Con los datos obtenidos del JASPAServerExplorer, o si se conocen los parámetros del FeatureStore, a partir de un DataStoreParameters, se puede crear a partir del FeatureManager una instancia del FeatureStore y a partir de éste realizar consultas contra ese almacén o modificar sus datos. Una consulta devolverá un *FeatureSet* y a partir de él podrá acceder a los datos.

5.2.3. Diseño del complemento

El diseño del complemento es extremadamente simple, puesto que al no tener ningún tipo de interacción con el usuario, lo único que ha de hacer es heredar de la implementación de referencia Extension según se describe en el apartado Complementos y extensiones, y sobrecargar los métodos isEnabled y isVisible para devolver false indicando que no hay que mostrar nada en la barra de menú.

En la Ilustración 28: Diseño de clases del complemento JASPA para gvSIG, se muestra la relación entre las clases Extension y JASPAExtension.

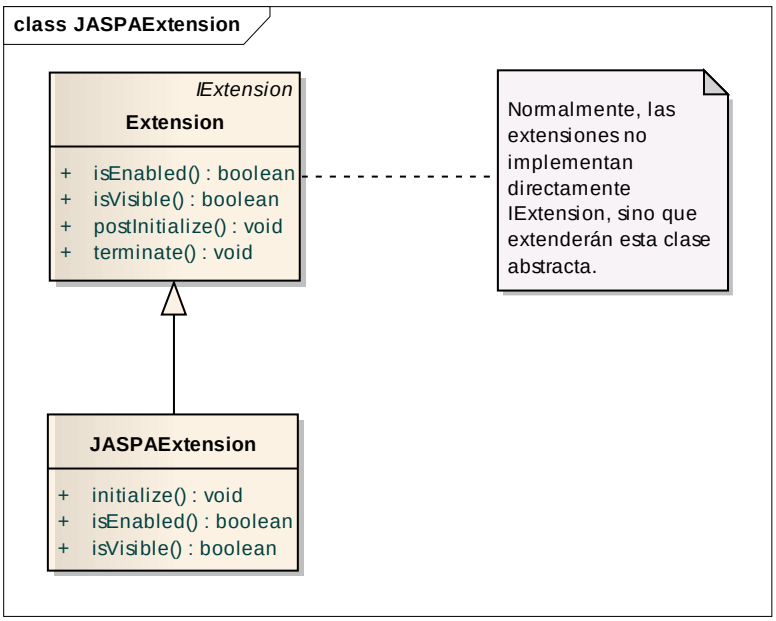

Ilustración 28: Diseño de clases del complemento JASPA para gvSIG

6. Desarrollo

A continuación se describen los aspectos más importantes relacionados con la actividad de desarrollo software del proyecto.

6.1. Entorno de desarrollo

Para el desarrollo de gvSIG (producto base y complementos), es necesario instalar y configurar un entorno de desarrollo basado en Eclipse[38] y sobre el que, para facilitar su uso, se instala un complemento de Maven[39].

Maven es una herramienta de software para la gestión y construcción de proyectos Java que facilita para la construcción de un proyecto software, la gestión de las dependencias de las bibliotecas que usa y que se basa en la definición de unos objetivos del desarrollo, y un conjunto de pasos y herramientas a usar en cada paso, para la consecución de dichos objetivos, entre los cuales, se encuentran los objetivos básicos de compilación y empaquetado.

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma. Este entorno de desarrollo, aunque actualmente soporta diferentes compiladores, nación como el un IDE para desarrollo java, y éste lenguaje sigue siendo el lenguaje de programación por excelencia usado en Eclipse y el usado para el desarrollo de gvSIG.

Para facilitar el desarrollo de complementos, el propio gvSIG dispone de uno de ellos que una vez instalado, y según se muestra en la Ilustración 29: Opción de menú de creación de complementos (Plugins), permite generar proyectos que sirven como plantilla la construcción del nuevo complemento sin necesidad de compilar el código fuente de gvSIG.

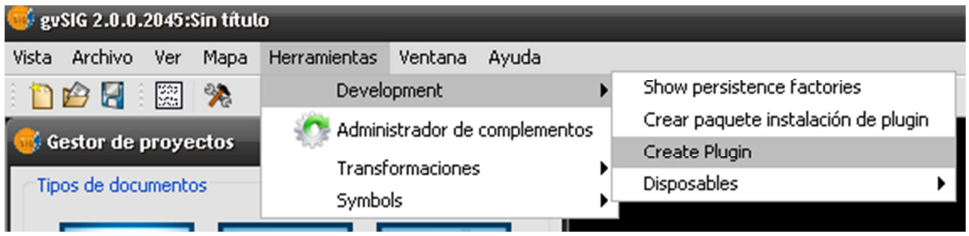

Ilustración 29: Opción de menú de creación de complementos (Plugins)

En el asistente de dicho complemento, se seleccionan ciertos parámetros básicos, según se muestra en la

Ilustración 30: Asistente de creación de proyecto para complemento, a partir de los

[38] www.eclipse.org

[39] www.maven.org

cuales, se genera, en base a una plantilla, dos proyectos para Maven: un primer proyecto que contiene las clases que componen la lógica del nuevo proveedor de datos y las clases que permiten el registro y activación del nuevo complemento en gvSIG, y que dependen del proyecto del proveedor de datos.

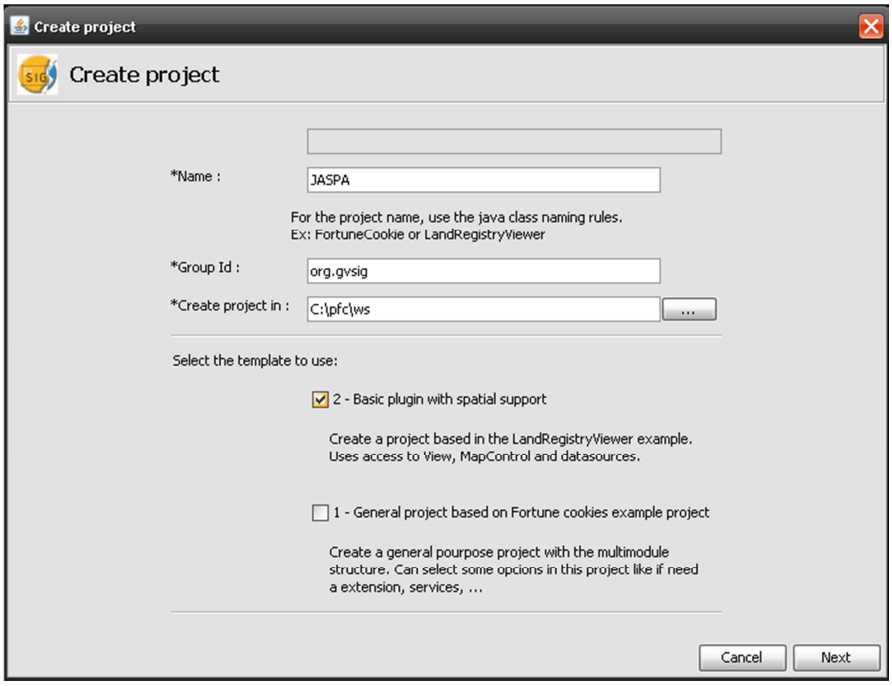

Ilustración 30: Asistente de creación de proyecto para complemento

Una vez creados los proyectos Maven, solo queda personalizarlos en función del módulo a implementar.

Para el despliegue del complemento de acceso a datos JASPA, se han de realizar dos pasos: la construcción del proyecto del driver, y la construcción del proyecto del complemento. Este último depende del driver, por lo que la construcción ha de seguir el orden descrito.

En la instalación del complemento, que se realiza con el comando mvn install, se genera la biblioteca del mismo, se descarga del repositorio la última versión del driver, lo empaqueta todo siguiendo el formato de complementos de gvSIG, y lo copia a la carpeta gvSIG/extensiones de la instalación de gvSIG que ha debido realizarse previamente en el equipo.

Al finalizar este paso, ya se puede ejecutar gvSIG con el driver JASPA ya disponible para su uso.

6.2. Código fuente del desarrollo

El desarrollo final se ha materializado en tres paquetes software:

- org.gvsig.fmap.dal.db.store.jaspa.app.extension: contiene el código fuente del complemento desarrollado.

- org.gvsig.fmap.dal.db.store.jaspa.lib.api: contiene el código fuente de la API de la librería.

- org.gvsig.fmap.dal.db.store.jaspa.lib.impl: contiene el código fuente de la implementación de la librería.

A continuación se muestran algunas capturas de pantalla del desarrollo ya en funcionamiento.

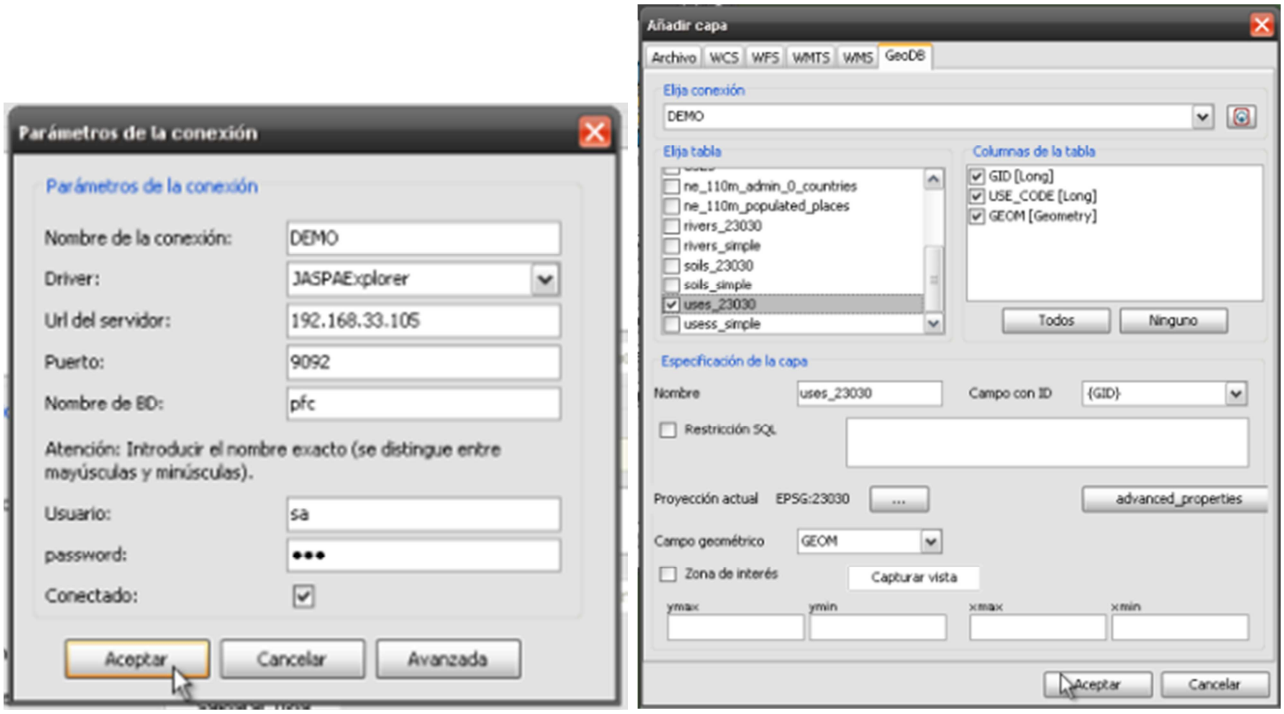

Ilustración 31: Edición de parámetros de conexión y selección de capa

Ilustración 32: Edición de geometrías con JASPA

7. Plan de pruebas

Partiendo del listado de requisitos presentados en la Tabla 4: Listado de requisitos, se ha diseñado un plan de pruebas que aseguren la calidad de la implementación del desarrollo y permitan comprobar el cumplimiento de los requisitos del proyecto. Para la realización del procedimiento de pruebas descrito en el plan se dispone de una hoja de registro de pruebas, en la cual se anotará por cada prueba si ha sido superada o no. Existe en la hoja de registro de pruebas una columna en la cual se pueden anotar observaciones, las cuales se utilizan para dejar comentarios de porqué una prueba que ha fallado se ha considerado superada o porqué una prueba que ha fallado se ha considerado satisfactoria.

Un ejemplo de salvedad sería que hubiese un error en la descripción de la prueba a pasar que fuese detectado durante la realización de la misma. En ese caso, se realizaría la prueba correctamente y se dejaría registro del error de la descripción.

Otro ejemplo de salvedad sería que la prueba fuese satisfactoria, pero por ejemplo al realizar otra operación sobre el aplicativo, éste dejara de funcionar correctamente. En ese caso, en función de la gravedad del error, el responsable de pasar la prueba tendría la potestad de decidir si la prueba se considera conforme o no, pero en cualquier caso, deberá anotar la salvedad.

7.1. Descripción del entorno de pruebas

El entorno para llevar a cabo el plan de pruebas consiste en un equipo informático con los siguientes paquetes software:

- Sistema operativo Ubuntu 11.04 (NattyNarwhal)[40]
- gvSIG Desktop 2.0build 2046.
- JASPA 0.2.[41]

El sistema operativo Ubuntu 11.04 ha sido seleccionado por ser una versión reciente y estable, sobre la cual se ha comprobado previamente que gvSIG se puede instalar fácilmente.

El software gvSIG Desktop 2.0 se encuentra disponible en la página de descargas de gvsig.org[42]. En dicha página hay descargas disponibles para Linux y para Windows con y sin pre-requisitos de instalación. Para la preparación del entorno de pruebas se recomienda el uso del instalador que incluye pre-requisitos de instalación con la finalidad de garantizar que el entorno es apto para la ejecución de gvSIG.

---

[40] http://releases.ubuntu.com/11.04/

[41] http://jaspa.upv.es/blog/downloads/

[42] http://www.gvsig.org/web/projects/gvsig-desktop/official/gvsig-2.0/descargas

El software JASPA 0.2 está disponible para los sistemas operativos Linux y Windows de 32 y 64 bit. Para cada sistema operativo hay dos versiones, una que se instala sobre PostgreSQL y otra que se instala sobre la base de datos H2. La versión que utiliza la base de datos H2 ya incluye la propia base de datos, por lo que no es necesario instalar dicho motor de base de datos independientemente.

Las características técnicas mínimas del equipo para la realización de las pruebas serán las definidas en la Tabla 5: Características Hardware:

CPU: 2GHz
Memoria: 1GB
Disco duro: 40GB
Resolución de pantalla: 1024x768.
Tarjeta de red, unidad USB, CD-ROM o algún medio que permita la instalación de las aplicaciones necesarias.

Tabla 5: Características Hardware

Se recomienda el uso de una máquina virtual en lugar de una máquina física para la realización de las pruebas. Es conveniente realizar una copia de dicha máquina virtual antes de comenzar la realización de las pruebas, de tal forma que si hubiese que repetir las pruebas, no se tenga que volver a preparar todo el entorno.

Las pruebas se realizarán con un usuario sin permisos de administrativos para garantizar que el driver no necesita de permisos especiales para su correcto funcionamiento.

Para una correcta realización de las pruebas es necesario disponer de un conjunto de datos que deben estar precargados en la base de datos con la que se realizarán las pruebas.

El conjunto de datos a cargar deberá contener varios tipos de geometrías y sistemas de referencia de coordenadas.

7.2. Cualificación para la realización de las pruebas

Para llevar a cabo el procedimiento de pruebas es necesario estar familiarizado con los conceptos y productos sobre los que se trabajará. El procedimiento de pruebas se ha diseñado indicando los pasos a seguir, pero confiando en la capacidad del responsable de las pruebas de elegir la mejor forma de realizarla y decidir si el producto pasa la prueba sin o con salvedades, en cuyo caso, habrá de anotarlas en la hoja de registro de prueba.

A efectos prácticos, esto quiere decir que si, por ejemplo, para conectar con la base de datos hay un menú, una combinación de teclas, una tecla de función, un acceso rápido o cualquier otra forma de acceder a esa funcionalidad, en el procedimiento no se indicará cómo acceder a ella, sino que el responsable de las pruebas, bajo su propio criterio, decidirá la mejor manera de hacerlo, y que el criterio puede variar cada vez que se repite una prueba.

7.3. Preparación de las pruebas

Antes de comenzar a realizar las pruebas, es necesario preparar y verificar el entorno de pruebas. Para ello, se verificarán los siguientes puntos:

1. El equipo sobre el que se prueba cumple con los requisitos especificados en la Tabla 5: Características Hardware.
2. Comprobar que la base de datos JASPA está en ejecución, y ejecutar en caso contrario.
3. Comprobar que el programa gvSIG 2.0 ejecuta correctamente.
4. Comprobar que se dispone de los paquetes software necesarios para la realización de las pruebas.
5. Comprobar que se dispone del material necesario para realizar el registro de las pruebas.

Para la realización de las pruebas es necesario contar con un conjunto de datos cargados en la base de datos que se encuentran en la Tabla 6: Orígenes de datos de prueba.

Nombre del archivo	Origen de los datos	SRID
RIVERS	JASPA[43]	-1
SOILS	JASPA	-1
USES	JASPA	-1
110m-admin-1-states-provinces-lines-shp	www.naturalearthdata.com[44]	**EPSG**:4326
110m-admin-1-states-provinces-poly-shp	www.naturalearthdata.com	**EPSG**:4326
ne_110m_admin_0_countries	www.naturalearthdata.com	**EPSG**:4326

[43] El instalador de JASPA dispone en la subcarpeta doc/data un conjunto de datos de prueba.

[44] http://www.naturalearthdata.com/downloads/110m-cultural-vectors/

ne_110m_populated_places	www.naturalearthdata.com	**EPSG**:4326

Tabla 6: Orígenes de datos de prueba

Para la carga de dichos datos se debe seguir el manual de usuario de JASPA. En concreto, los comandos usados para generar los archivos sql que realizan la carga de los datos de NaturalEarthData.com son:

```
shp2jaspah2 –w –s 4326 –S –g GEOM 110m-admin-1-states-provinces-lines-shp.shp
110m-admin-1-states-provinces-lines > 110m-admin-1-states-provinces-lines.sql

shp2jaspah2 –w –s 4326 –S –g GEOM 110m-admin-1-states-provinces-poly-shp.shp
110m-admin-1-states-provinces-poly > 110m-admin-1-states-provinces-poly.sql

shp2jaspah2 –w –s 4326 –S –g GEOM ne_110m_admin_0_countries.shp
ne_110m_admin_0_countries > ne_110m_admin_0_countries.sql

shp2jaspah2 –w –s 4326 –S –g GEOM ne_110m_populated_places.shp
ne_110m_populated_places  > ne_110m_populated_places.sql
```

Una vez generados los archivos sql, solo hay que cargarlos usando el comando

```
h2runscript –url jdbc:h2:tcp://localhost/pfc -user sa -password 123 -script
<sql_file>
```

para cada uno de los archivos con extensión sql generados.

7.4. Procedimiento

El procedimiento de pruebas ha sido diseñado para que la finalización de una prueba deje preparado el entorno para el comienzo de la prueba siguiente, de tal manera que la realización de las mismas no puede ser aleatoria, siendo obligatorio realizarlas en orden numérico creciente.

Un ejemplo ilustrativo es que hay una prueba que verifica el buen funcionamiento del instalador del paquete software a probar, y que si no se realiza o falla, las pruebas que cuentan con que el driver se encuentre correctamente instalado, no se podrán realizar.

7.4.1. Hoja de registro de pruebas

Por cada prueba del procedimiento, se dispondrá de una hoja de registro de pruebas según se ilustra en la Tabla 9 Hoja de registro de pruebas, recogida en el Anexo I y desarrollada en el Anexo II.

Esta hoja de registro permitirá recoger a medida que se realiza la prueba, si los pasos que componen la prueba se superan sin ningún comentario, o si por el contrario, durante la realización de alguno de los pasos es necesario realizar algún tipo de observación, o incluso dar el paso, y por lo tanto la prueba, por incorrecta.

7.4.2. Hoja de resumen de resultados

Tras la finalización de cada prueba, se recogerá en una hoja de resumen, el resultado final de la prueba realizada, donde se registrará la versión usada del driver, y la fecha en la que se realiza.

El resultado de cada prueba que compone el plan de pruebas puede ser OK si la prueba se considera pasada correctamente o No OK, si se considera que la prueba no es satisfactoria. En cualquiera de los dos casos, se indicará si hay observaciones en las pruebas realizadas. Las observaciones se registrarán en cada hoja de registro de prueba.

En la Tabla 7: Tabla resumen resultado de pruebas, se muestra el resumen final de las pruebas realizadas al driver donde se observa que, debido a los problemas de gvSIG, existen limitaciones en el funcionamiento del driver. La versión completa de esta tabla se puede encontrar en el Anexo I. Resumen de resultado de pruebas.

Nº Prueba	Nombre de la prueba	Resultado OK / No OK	Obs. Si / No
1	Verificar que el driver funciona sobre una instalación de gvSIG 2.0 obtenida a través de su página de descarga.	OK	No
2	Comprobar que la versión de gvSIG es la correcta	OK	No
3	Comprobar que la versión de JASPA es la correcta	OK	No
4	Realizar la instalación del driver siguiendo el manual de instalación. Verificar que existe el instalable y que se ha usado en la instalación. Verificar que existe un manual de usuario.	OK	No
5	Comprobar el acceso de lectura a la información del esquema de la base de datos	OK	Si
6	Comprobar que se visualizan tablas con geometrías de forma gráfica y tabular	OK	No
7	Comprobar que se puede acceder y modificar la información no geométrica de la base de datos.	OK	No
8	Comprobar que se pueden crear y modificar geometrías	OK	Si
9	Comprobar que se soportan varios CRS.	OK	Si
10	Comprobar que se pueden crear nuevas tablas en la base de datos.	OK	Si

Tabla 7: Tabla resumen resultado de pruebas

8. Conclusiones y recomendaciones

A continuación se recogen una serie de conclusiones obtenidas tras la realización del proyecto junto con algunos comentarios personales relativos a las características y dificultades que se han ido encontrando a lo largo de la realización del proyecto.

La primera conclusión y más importante de todas es que el proyecto ha conseguido alcanzar todos los objetivos inicialmente establecidos y con todos los requisitos establecidos según se puede observar en la tabla resumen de las pruebas que se encuentra en el Anexo I. Resumen de resultado de pruebas, en el cual se observa que todas han pasado satisfactoriamente teniendo en cuenta el estado de inestabilidad de gvSIG 2.0, que es lo primero que aparece al ejecutar el programa según se muestra en la Ilustración 33: Aviso de inestabilidad de gvSIG.

Ilustración 33: Aviso de inestabilidad de gvSIG

La integración de JASPA en gvSIG finalmente permite el acceso a la información almacenada en la base de datos H2 con la extensión JASPA, y añade a gvSIG la capacidad de mostrar gráficamente dicha información, realizando todo tipo de procesamiento, como la división en categorías con diferentes atributos gráficos, y la presentación con diferentes sistemas de coordenadas según se muestra en la Ilustración 34: Captura de pantalla de gvSIG presentando datos de JASPA.

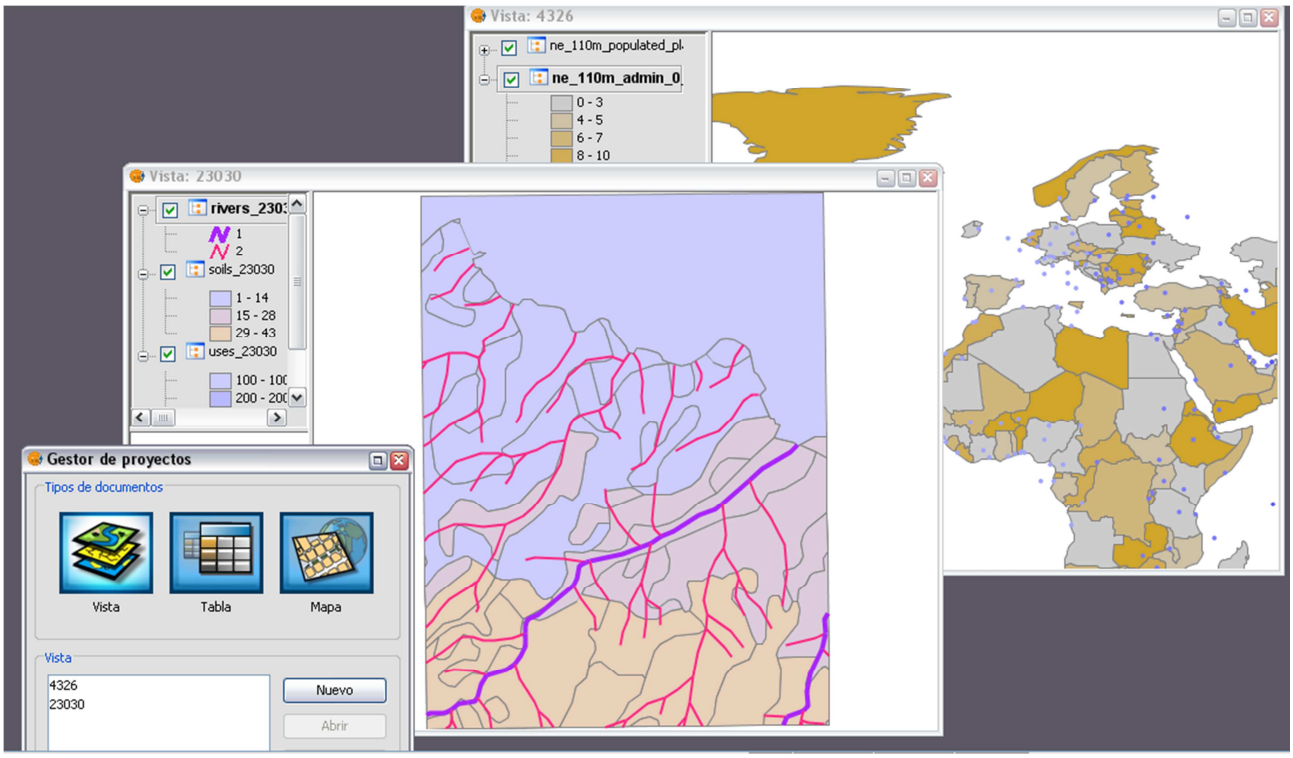

Ilustración 34: Captura de pantalla de gvSIG presentando datos de JASPA

La tarea no ha sido sencilla, puesto que para el desarrollo del proyecto se han encontrado varias dificultades originadas por el hecho de que la versión de gvSIG sobre la que se había de desarrollar el proyecto se encuentra en un estado de desarrollo poco avanzado, por lo cual contiene gran cantidad de errores que impiden el correcto funcionamiento de gvSIG.

Esta situación ha dificultado el desarrollo, puesto que en varias ocasiones se han presentado dudas de si el mal funcionamiento observado era debido a la implementación del proyecto o a errores en gvSIG. La resolución de estas dudas no ha sido tarea sencilla, ya que para identificar el origen del error se ha tenido que compilar y depurar el código fuente original de gvSIG hasta encontrar, en algunos casos, el origen del error en el código del producto y no en el driver en desarrollo.

Todos los errores detectados han sido reportados convenientemente, y en algún caso, se ha incluso sugerido el código que implementa la solución, el cual ha sido realizado y validado en la copia con la que se ha realizado el desarrollo.

Es ahora responsabilidad de los desarrolladores de gvSIG la resolución de los errores y la incorporación del código sugerido, o uno alternativo, al código fuente del aplicativo.

Si bien estos errores son del producto, hasta que se corrijan, el driver no dispondrá de toda la funcionalidad implementada. De los errores reportados en el bugtracker de

gvSIG, hay dos que tienen una especial repercusión:

> #559[45], mencionado anteriormente en el documento en el apartado Modelo de geometrías de gvSIG, ya que impide la edición de geometrías de tipo agregado y geometrías tipo polilínea (LINESTRING). Para este bug, durante la fase de desarrollo se implementó un parche con el que se realizaron pruebas de edición de geometrías que resultaron satisfactorias.
>
> #719[46], este error hace que el usuario haya de tener la precaución de configurar el CRS de la vista igual al de los datos de las capas a mostrar. Si bien, el driver funciona con datos que tienen diferentes CRS.

Otra dificultad encontrada ha sido la poca documentación existente de gvSIG que facilite a los desarrolladores la creación de nuevos proveedores de datos. La documentación disponible define con mucho más detalle la API que tienen los consumidores de datos que la que tienen los proveedores, por lo que para profundizar en dicha información, ha sido necesario realizar, a partir del código fuente de gvSIG, una ingeniería inversa de cómo se implementa un nuevo proveedor de datos.

No todo en el desarrollo del proyecto ha sido dificultades, puesto que afortunadamente, las carencias de documentación se han suplido con la disponibilidad del código fuente del producto, que ha permitido analizar el origen de los problemas.

Además, en gvSIG 2.0 ya se dispone de un proveedor de datos de PostGIS, el cual es muy parecido en su funcionalidad a lo que debe realizar el proveedor de datos JASPA, siendo de mucha utilidad para identificar las API a implementar.

Otra fuente de información de gran utilidad ha sido el foro para desarrolladores[47] de gvSIG. En este foro no solo se puede encontrar problemas y soluciones, sino que los propios desarrolladores de gvSIG ofrecen su colaboración de una manera rápida y eficaz.

Si este mismo proyecto se hubiese tenido que realizar con un producto de código cerrado, difícilmente se hubiesen conseguido los objetivos en el tiempo estipulado.

Otro aspecto que ha facilitado enormemente el desarrollo es la disponibilidad de un complemento especial para desarrolladores, que en base a la definición de las características a implementar, solamente con la introducción de unos datos básicos, es capaz de generar partiendo de una plantilla la estructura de clases que implementan una extensión. Estas clases ya disponen del código fuente que realiza el registro e inicialización de la extensión en gvSIG.

---

[45] https://devel.gvsig.org/redmine/issues/559

[46] https://devel.gvsig.org/redmine/issues/719

[47] http://osgeo-org.1560.n6.nabble.com/gvSIG-desarrolladores-f4163512.html

Para la elaboración del procedimiento de pruebas, y dado que el enunciado del proyecto no describe de manera precisa la funcionalidad a implementar, se ha optado por estudiar la funcionalidad ofrecida por gvSIG para acceso a datos tabulares, ya sean geográficos o no, y definir un conjunto de pruebas que valide la implementación del driver, no solo como medio de consulta de los datos almacenados en una base de datos JASPA, sino también la funcionalidad relacionada con la modificación y creación de nuevos datos en la base de datos.

8.1. Recomendaciones para futuros desarrollos

Como se analizó en el capítulo dedicado a JASPA, su versión 0.2 soporta las bases de datos PostgreSQL y H2.

El objetivo de este proyecto era la implementación de un driver para JASPA en su versión H2, sin embargo, dado que existen pocas diferencias en sus variantes PostgreSQL y H2, sería factible hacer que el driver desarrollado fuese compatible con las dos bases de datos, siendo incluso posible identificar automáticamente la versión de JASPA usada de manera que para el usuario sería transparente. Esta posibilidad se ha tenido en cuenta durante el desarrollo denominando al driver de forma genérica JASPA en lugar de JASPA4H2.

También, una vez se encuentre disponible una versión de gvSIG 2.0 estable y con los errores reportados ya corregidos, sería interesante validar de nuevo la implementación realizada, sobre todo en lo concerniente a cambios de CRS, puesto que es el punto funcional que ha sido validado en menor profundidad.

Un posible desarrollo a futuro, no ligado directamente con el driver desarrollado, sino con JASPA, puede ser la posibilidad de ampliar la funcionalidad de JASPA para que, además de soportar formatos vectoriales, admita formatos raster. Probablemente esto llevaría no solo a la necesidad de realizar modificaciones sobre el driver para que soportara el almacenamiento de geometrías raster, sino también sobre gvSIG.

Otra posible mejora en JASPA sería la inclusión de índices espaciales cuando se usa la base de datos H2, lo cual mejoraría enormemente el rendimiento de ciertas operaciones sobre geometrías.

Con esta mejora, en gvSIG se podría plantear también la posibilidad de realizar las modificaciones oportunas para que ciertas operaciones espaciales como pueden ser la unión o intersección de geometrías, o la localización de elementos dentro de un área, se realizaran en JASPA en lugar de en gvSIG. Con esta modificación se podría mejorar el rendimiento ya que no sería necesaria la transferencia de información geométrica de todos los elementos a unir, sino que desde JASPA se enviaría solo el resultado de la unión.

Esta modificación en gvSIG probablemente conllevara un cambio en la API de tal manera que le fuese posible identificar si el servidor del que obtiene la información es capaz de realizar dichas operaciones geométricas, para que cuando fuera posible, sustituir las operaciones locales por operaciones en el servidor. La implementación de

dichas consultas en el driver sería sencilla puesto que no sería más que una llamada a una sentencia SQL bien construida que devolvería como resultado una geometría, y esto es una funcionalidad que ya implementa el driver.

9. Anexo I. Resumen de resultado de pruebas

Resumen de resultado de pruebas **Driver JASPA para gvSIG 2.0** **Versión del driver: __1.0.0-devel___**		**Fecha:** **Página:** **__1_ de __1_**	
Nº Prueba	**Nombre de la prueba**	**Resultado** OK / No OK	**Obs.** Si / No
1	Verificar que el driver funciona sobre una instalación de gvSIG 2.0 obtenida a través de su página de descarga.	OK	No
2	Comprobar que la versión de gvSIG es la correcta	OK	No
3	Comprobar que la versión de JASPA es la correcta	OK	No
4	Realizar la instalación del driver siguiendo el manual de instalación. Verificar que existe el instalable y que se ha usado en la instalación. Verificar que existe un manual de usuario.	OK	No
5	Comprobar el acceso de lectura a la información del esquema de la base de datos	OK	Si
6	Comprobar que se visualizan tablas con geometrías de forma gráfica y tabular	OK	No
7	Comprobar que se puede acceder y modificar la información no geométrica de la base de datos.	OK	No
8	Comprobar que se pueden crear y modificar geometrías	OK	Si
9	Comprobar que se soportan varios CRS.	OK	Si
10	Comprobar que se pueden crear nuevas tablas en la base de datos.	OK	Si

Tabla 8. Hoja resumen de resultado de pruebas

10. Anexo II. Formato de hoja de registro de pruebas

Hoja de registro de pruebas **Driver JASPA para gvSIG 2.0** **Número de prueba: _____ Requisito: _____**		**Fecha:**	
Prueba:		**Resultado**	
Paso	**Comentarios**	**Conforme**	**No Conforme**
Observaciones:			

Tabla 9 Hoja de registro de pruebas

11. Anexo III. Hojas de registro de pruebas

<table>
<tr><td colspan="2">Hoja de registro de pruebas
Driver JASPA para gvSIG 2.0
Número de prueba: __1___ Requisito: __1.1___</td><td colspan="2">Fecha: 23 Mayo
Página:
__1_ de __1_</td></tr>
<tr><td colspan="2">Prueba: Verificar que el driver funciona sobre una instalación de gvSIG 2.0 obtenida a través de su página de descarga.</td><td colspan="2">Resultado</td></tr>
<tr><td>Paso</td><td>Comentarios</td><td>Conforme</td><td>No Conforme</td></tr>
<tr><td>1</td><td>Descargar gvSIG Desktop 2.0.0 build 2046</td><td>SI</td><td></td></tr>
<tr><td>2</td><td>Instalar gvSIG Desktop 2.0.0 build 2046 para Linux</td><td>SI</td><td></td></tr>
<tr><td>3</td><td>Instalar gvSIG Desktop 2.0.0 build 2046 para Windows</td><td>SI [1]</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4">Observaciones:
[1] En el momento de realizar la instalación, no fue posible instalar la construcción 2046, por lo que se realizaron las pruebas con la 2045.</td></tr>
</table>

Tabla 10: Hoja de registro de prueba 1

Hoja de registro de pruebas **Driver JASPA para gvSIG 2.0** **Número de prueba: __2___ Requisito: __1___**		**Fecha: 23 Mayo** **Página:** **__1_ de __1_**	
Prueba: Comprobar que la versión de gvSIG es la correcta		**Resultado**	
Paso	**Comentarios**	**Conforme**	**No Conforme**
1	Ejecutar gvSIG y verificar en el "Splash Screen" que aparece la versión 2.0 build 2046.	SI	
2	Ir al menú Ayuda | Acerca de y comprobar que en el diálogo aparece la versión de gvSIG desktop 2.0 o superior	SI	
3	Comprobar que el usuario con el que se ha lanzado la aplicación es un usuario sin permisos administrativos.	SI	
Observaciones:			

Tabla 11: Hoja de registro de prueba 2

Hoja de registro de pruebas **Driver JASPA para gvSIG 2.0** **Número de prueba: __3___ Requisito: __2___**		**Fecha: 23 Mayo** **Página:** **__1_ de __1_**	
Prueba: Comprobar que la versión de JASPA es la correcta		**Resultado**	
Paso	**Comentarios**	**Conforme**	**No Conforme**
1	Ejecutar JASPA según las instrucciones del producto (lanzar el archivo h2).	SI	
2	Ejecutar la consola de JASPA según las instrucciones del producto (lanzar el archivo h2console) y conectar a la base de datos H2.	SI	
3	Ejecutar el comando de consulta de versión de JASPA: select Jaspa_Full_Version() y verificar que es la versión 0.2 o superior y que el entorno de ejecución Java es igual o superior a Java 1.5.	SI	

Observaciones:

El resultado de la ejecución del comando Jaspa_Full_Versión ha sido:

JASPA (JAva SPAtial) For H2=”0.2.0” JTS=”1.11.0 for jaspa” GeoTools=”2.7.2” JRE=”1.6.0_25”

Tabla 12: Hoja de registro de prueba 3

<table>
<tr><td colspan="3">Hoja de registro de pruebas
Driver JASPA para gvSIG 2.0
Número de prueba: __4___ Requisito: _4,5,6 y 7_____</td><td>Fecha: 23 Mayo
Página:
__1_ de __1_</td></tr>
<tr><td colspan="3">Prueba: Realizar la instalación del driver siguiendo el manual de instalación. Verificar que existe el instalable y que se ha usado en la instalación. Verificar que existe un manual de usuario. Verificar que el código fuente se encuentra documentado.</td><td>Resultado</td></tr>
<tr><td>Paso</td><td>Comentarios</td><td>Conforme</td><td>No Conforme</td></tr>
<tr><td>1</td><td>Verificar que el manual de instalación hace uso de un paquete de instalación y que está disponible.</td><td>SI</td><td></td></tr>
<tr><td>2</td><td>Seguir el manual de instalación y comprobar que los pasos indicados funcionan correctamente.</td><td>SI</td><td></td></tr>
<tr><td>3</td><td>Verificar que existe un manual de usuario que explica el uso de la librería</td><td>SI</td><td></td></tr>
<tr><td>4</td><td>Verificar que el código fuente se encuentra documentado</td><td>SI</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="4">Observaciones:</td></tr>
</table>

Tabla 13: Hoja de registro de prueba 4

<table>
<tr><td colspan="2">Hoja de registro de pruebas
Driver JASPA para gvSIG 2.0
Número de prueba: __5___ Requisito: __3___</td><td colspan="2">Fecha: 23 Mayo
Página:
__1_ de __1_</td></tr>
<tr><td colspan="2">Prueba: Comprobar el acceso de lectura a la información del esquema de la base de datos</td><td colspan="2">Resultado</td></tr>
<tr><td>Paso</td><td>Comentarios</td><td>Conforme</td><td>No Conforme</td></tr>
<tr><td>1</td><td>Realizar una carga de datos de prueba con los archivos shp que se encuentran en el directorio doc/data de la instalación de jaspa4h2 según instrucciones del manual de JASPA.

shp2jaspah2 –w –s 23030 –S –g GEOM uses.shp uses_23030 > uses_23030.sql

shp2jaspah2 –w –s 23030 –S –g GEOM soils.shp soils_23030 > soils_23030.sql

shp2jaspah2 –w –s 23030 –S –g GEOM rivers.shp rivers_23030 > rivers_23030.sql

h2runscript -url jdbc:h2:tcp://localhost/pfc -user sa -password 123 -script uses_23030.sql

h2runscript -url jdbc:h2:tcp://localhost/pfc -user sa -password 123 -script soils_23030.sql

h2runscript -url jdbc:h2:tcp://localhost/pfc -user sa -password 123 -script rivers_23030.sql

Realizar la carga de datos de prueba con los archivos shp de naturalearthdata.com

shp2jaspah2 –w –s 4326 –S –g GEOM 110m-admin-1-states-provinces-lines-shp.shp 110m-admin-1-states-provinces-lines > 110m-admin-1-states-provinces-lines.sql

shp2jaspah2 –w –s 4326 –S –g GEOM 110m-admin-1-states-provinces-poly-shp.shp 110m-admin-1-states-provinces-poly > 110m-admin-1-states-provinces-poly.sql

shp2jaspah2 –w –s 4326 –S –g GEOM ne_110m_admin_0_countries.shp ne_110m_admin_0_countries > ne_110m_admin_0_countries.sql

shp2jaspah2 –w –s 4326 –S –g GEOM ne_110m_populated_places.shp ne_110m_populated_places > ne_110m_populated_places.sql

h2runscript –url jdbc:h2:tcp://localhost/pfc -user sa -</td><td>SI</td><td></td></tr>
</table>

	`password 123 -script 110m-admin-1-states-provinces-lines.sql`
	
	`h2runscript –url jdbc:h2:tcp://localhost/pfc -user sa -password 123 –script 110m-admin-1-states-provinces-poly.sql`
	
	`h2runscript –url jdbc:h2:tcp://localhost/pfc -user sa -password 123 –script ne_110m_admin_0_countries.sql`
	
	`h2runscript –url jdbc:h2:tcp://localhost/pfc -user sa -password 123 –script ne_110m_populated_places.sql`		

2	Lanzar la aplicación gvSIG desktop. Crear una vista y añadir una capa de tipo GeoDB por cada tabla geográfica del conjunto de datos de prueba.	SI	
3	Crear una conexión de a través del driver "JASPAExplorer", introduciendo los datos de conexión incorrectos y verificar que la conexión no se produce. Repetirlo con distinto conjunto de datos incorrecto. Repetirlo con datos de conexión correctos y verificar que la conexión se establece correctamente.	SI	
4	Verificar que aparecen todas las tablas con información geográfica, y que no aparecen las tablas de metadatos ni aquellas tablas que no disponen de información geográfica.	SI	Ver [1]

Observaciones:

Los parámetros de la conexión usados son:

- Nombre de la conexión PFC
- Driver: JASPAExplorer
- Url del servidor: localhost
- Puerto: 9092
- Nombre de BD: pfc
- User: sa
- Password: 123
- Conectado: seleccionado

[1] Aparecen todas las tablas de la base de datos, no solo aquellas que tienen información geográfica. Esto es debido a una funcionalidad pendiente en gvSIG 2.0 y que se encuentra reportada en la web de gvSIG con el bug id #707:

https://devel.gvsig.org/redmine/issues/707

Se considera por lo tanto la prueba correcta.

Tabla 14: Hoja de registro de prueba 5

Hoja de registro de pruebas **Driver JASPA para gvSIG 2.0** **Número de prueba: __6___ Requisito: __3.1___**		**Fecha: 23 Mayo** **Página:** **__1_ de __1_**	
Prueba: Comprobar que se visualizan tablas con geometrías de forma gráfica y tabular		**Resultado**	
Paso	**Comentarios**	**Conforme**	**No Conforme**
1	Por cada tabla con geometrías, crear una capa.	SI	
2	Comprobar que se muestran en la vista todas las capas seleccionadas y se ven los objetos geométricos almacenados en la base de datos.	SI	
3	Comprobar que se cambian las propiedades gráficas (color, grosor, tipo de línea) de la capa y se actualiza en la vista.	SI	
4	Hacer zoom de distintos niveles y comprobar que se muestran correctamente.	SI	
5	Seleccionar una de las capas de la geometría y elegir la opción de menú Capa | Ver tabla de atributos. Comprobar que se muestran los datos de manera tabular y que se puede consultar toda la tabla con la barra de desplazamiento lateral.	SI	
Observaciones:			

Tabla 15: Hoja de registro de prueba 6

Hoja de registro de pruebas **Driver JASPA para gvSIG 2.0** **Número de prueba: __7___ Requisito: __3.2___**		**Fecha: 23 Mayo** **Página:** **__1_ de __1_**	
Prueba: Comprobar que se puede acceder y modificar la información no geométrica de las tablas de la base de datos		**Resultado**	
Paso	**Comentarios**	**Conforme**	**No Conforme**
1	Añadir una conexión a datos tabulares y conectar con la base de datos PFC mediante el driver JASPA.	SI	
2	Comprobar que aparecen las tablas de la base de datos y seleccionar una tabla de la lista.	SI	
3	Comprobar que la tabla tiene más de 100 elementos y que se pueden visualizar todos usando la barra de desplazamiento vertical.	SI	
4	Pasar al modo edición con la opción de menú Tabla | Comenzar edición.	SI	
5	Modificar algunos datos de la tabla aleatoriamente anotando los datos modificados.	SI	
6	Almacenar los datos con la opción de menú Tabla | Finalizar edición y aceptar los cambios.	SI	
7	Comprobar en la consola de H2 que los datos se han modificado correctamente.	SI	
Observaciones: Cambios realizados sobre la tabla RIVERS: GID=26, RIVER_CODE=2 (antes 1), GID=106, RIVER_CODE=1 (antes 2) GID = 1, RIVER_CODE=1 (antes 2)			

Tabla 16: Hoja de registro de prueba 7

Hoja de registro de pruebas **Driver JASPA para gvSIG 2.0** **Número de prueba: __8___ Requisito: __3.3___**		**Fecha: 23 Mayo** **Página:** **__1_ de __1_**	
Prueba: Comprobar que se pueden crear y modificar geometrías.		**Resultado**	
Paso	**Comentarios**	**Conforme**	**No Conforme**
1	Elegir una de las capas de las vistas creadas en la prueba 3.1-1 y pasar a modo edición con la opción de menú Capa | Comenzar edición.	SI	
2	Añadir elementos geométricos sobre la capa en edición.	SI	
3	Guardar los elementos geométricos con la opción de menú Capa | Finalizar edición y confirmar que se desean almacenar los cambios.	SI	
4	Repetir los pasos 2, 3 y 4 para diferentes capas.	SI [1]	
5	Comprobar que los objetos salvados se encuentran en la base de datos reiniciando la aplicación y volviendo a mostrar las capas editadas.	SI [1]	

Observaciones:

[1] Esta prueba se ha dado por válida por entenderse de que los errores detectados y que se describen a continuación, son origen del programa gvSIG y no del driver.

a. No se pueden editar geometrías de tipo agregadas, es decir (MULTIPOINT, MULTILINESTRING, MULTIPOLYGON). Debido a este punto, se ha tenido que generar un nuevo conjunto de datos de prueba, puesto que los archivos SQL que vienen con la instalación de JASPA (uses, soils y rivers), utilizan estos tipos. Para generar geometrías simples se ha usado la opción –S del programa shp2jaspah2, que fuerza al uso de geometrías simples. Una vez realizado el cambio, las pruebas de edición fueron correctas.

b. No se permite la edición de geometrías donde el SRID de la base de datos es diferente al de la vista. Debido a un problema en gvSIG, no es posible usar un SRID diferente en vista y capas.

c. Las geometrías de tipo LINESTRING no pueden editarse. Ver la información en el bug reportado en https://devel.gvsig.org/redmine/issues/559.

d. Cuando se termina una edición y posteriormente se intenta iniciar una nueva edición, el programa genera una excepción y es necesario reiniciarlo para poder volver a editar.

Tabla 17: Hoja de registro de prueba 8

Hoja de registro de pruebas **Driver JASPA para gvSIG 2.0** **Número de prueba: __9___ Requisito: _3.4____**		**Fecha: 23 Mayo** **Página:** **__1_ de __1_**	
Prueba: Comprobar que se soportan varios CRS.		**Resultado**	
Paso	**Comentarios**	**Conforme**	**No Conforme**
1	Crear una vista y seleccionar el mismo CRS que la base de datos.	SI	
2	Añadir una capa de la base de datos con el mismo CRS de la vista.	SI	
Observaciones: Las pruebas realizadas han sido con la limitación de usar en vista y capa el mismo CRS.			

Tabla 18: Hoja de registro de prueba 9

Hoja de registro de pruebas **Driver JASPA para gvSIG 2.0** **Número de prueba: __10___ Requisito: __8___**		**Fecha: 23 Mayo** **Página:** **__1_ de __1_**	
Prueba: Comprobar que se pueden crear nuevas tablas en la base de datos.		**Resultado**	
Paso	**Comentarios**	**Conforme**	**No Conforme**
1	Habilitar en las preferencias de la aplicación la posibilidad de creación avanzada de nueva capa. Y elegir la opción de menú Vista | Nueva Capa	SI	

2	Proceder a la creación de una nueva capa siguiendo el manual de usuario. Con la siguiente estructura:
	`Campo – Tipo – Longitud – Tipo – Subtipo - CRS – PK – NULL`
	`GID Integer * * * * Si Si`
	`Nombre String 20 * * * No No`
	`Valor Integer * * * * No No`
	`GEOM Geometry * Surface GEOM2D ¿ No No`
		SI	

3	Finalizar la introducción de datos para la creación de la capa	SI	
4	Añadir nuevos elementos a la tabla con gvSIG	SI [1]	
Observaciones: Debido de nuevo a errores en gvSIG, la creación de nuevas capas tiene limitaciones y es necesario seguir estrictamente las instrucciones del manual de usuario. [1] La edición de elementos es complicada. Hay que sacar la vista tabular de la tabla con la opción de menú Tabla | Ver tabla de atributos e introducir los identificadores únicos GID mientras se edita. Además, algunas veces saltan excepciones de punteros null que si se vuelve a intentar funciona.			

Tabla 19: Hoja de registro de prueba 10

12. Bibliografía

Anónimo, W. (s.f.). Obtenido de Wikipedia: http://es.wikipedia.org/wiki/European_Petroleum_Survey_Group

Anónimo, W. (s.f.). Obtenido de Wikipedia: http://en.wikipedia.org/wiki/Spatial_reference_system

Botella Plana, A. (s.f.). *Base de datos geográficos. Almacenes de datos geográficos.* UOC - P07/89036/02929.

Herring, J. R. (2006). *OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture.* Open Geospatial Consortium Inc.

Herring, J. R. (2006). *OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option.* Open Geospatial Consortium Inc.

Knut, S. (s.f.). *SQL/MM Spatial: The Standard to Manage Spatial Data in Relational Database Systems.* ISO.

13. Glosario

API: Application Programming Interface. Conjunto

EWKB: (Extended Well Known Binary) Representación extendida de WKB donde la geometría incluye el SRID.

EWKT: (Extended Well Known Text) Representación extendida de WKT donde la geometría incluye el SRID.

gvSIG: es un aplicativo de sistemas de información geográfica en software libre desarrollado principalmente en el lenguaje de programación Java.

JASPA: (JAva SPAtial) es una extensión especial para sistemas de bases de datos relacionales que implementa los estándares "Simple Feature Access for SQL" y "SQL/MM" de OpenGIS.

Java: Lenguaje de programación orientado a objetos. La plataforma gvSIG se ha desarrollado en este lenguaje.

JDBC: (Java DataBaseConnection) es un estándar que define la forma de acceder a bases de datos relacionales desde Java.

JTS: (Java Topology Suite) librería espacial de código abierto desarrollada en Java que pone a disposición del programador objetos con herramientas de topología para tareas de análisis espacial.

OGC: Open Geospatial Consortium (www.opengeospatial.org)

SPI: Service Provider Interface

SQL: (Structured Query Language) Lenguaje de consulta estructurado para consulta a bases de datos relacionales.

SHP: (ESRI Shapefile) Formato de archivo informático para el almacenamiento de datos espaciales.

SRID: (Spatial Reference System Identifier) Valor que identifica sin ambigüedad un sistema de referencia.

UOC: Universitat Oberta de Catalunya

WKB: (Well Known Binary) Representación en formato binario de una geometría.

WKT: (Well Known Text) Representación en formato textual de una geometría.

XML: Extensible Markup Language