Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Geoportal WEB IDE (Infraestructura de Datos Espaciales)

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Patricia Mascarell Gregori

Tutor: José Albaladejo Meroño

Cotutor: Houcine Hassan

2013/2014

Resumen

El objetivo del presente estudio es por una parte la definición teórica de todos los componentes de una Infraestructura de Datos Espaciales y por otra la puesta en práctica de implantación de una Infraestructura de Datos Espaciales de ejemplo empleando herramientas de software libre. En este estudio se incluye un detallado proceso de implantación que trata desde la transformación de datos, instalación de aplicaciones y empleo de las mismas.

Palabras clave: Geoportal, IDE, Infraestructura, Datos, Espacial, geoservicios.

Abstract

The aim of this study is the theoretical definition of all the components of a Spatial Data Infrastructure and its implementation using free software tools. This study includes a detailed implementation process that comes from the transformation of data, application installation and usage.

Key words: SDI, Spatial, Data, Infraestructure, geowebservices.

Tabla de contenidos

Índice de tablas

1 Introducción

En la actualidad, el uso de tecnología SIG (Sistema de Información Geográfica) es la solución para muchos problemas que requieren, cada vez más, el acceso a varios tipos de información geográfica o de distribución espacial. La tecnología SIG permite almacenar y manipular información usando geografía, analizar patrones, relaciones y tendencias en la información con el fin de contribuir a la toma de mejores decisiones. Además, el uso de esta tecnología facilita la realización de ciertos trabajos y da solución a muchos problemas que nos ocupan hoy en día en muchos campos de la vida.

Actualmente, los avances en la tecnología y el uso extendido de los smartphones, tabletas, GPS, etc., ha provocado que cualquier usuario sea capaz de consultar información geográfica. Este hecho es posible gracias a la existencia de aplicaciones que gestionan servicios y datos para que de forma sencilla sean capaces de obtener respuesta a necesidades y/o curiosidades.

Con este trabajo de final de grado, se pretende explicar cómo se puede obtener un cliente de consulta de datos espaciales, a partir de herramientas libres de licencias y que siguen estándares libres, definidos y aprobados por organizaciones y gobiernos.

En el presente trabajo, se hablará de los estándares que han facilitado que hoy en día exista una comunidad de usuarios que trabaja para avanzar en el campo de los SIG. También se describirá la arquitectura, tanto física como lógica, necesaria para publicar geoservicios que serán utilizados para realizar consultas y visualizar mapas desde el cliente web que se ha desarrollado para ello, de aquí en adelante Geoportal WEB.

Por último, se explicará detalladamente cómo se ha realizado la instalación del software base que es necesario para el buen funcionamiento y el desarrollo del Geoportal WEB.

2 Objetivos

El presente estudio tiene varios objetivos. El principal de todos ellos es la definición teórica del término Infraestructura de Datos Espacial (IDE) y describir todos los componentes que intervienen en ella.

El siguiente punto a alcanzar será la explicación de los pasos a seguir para la puesta en práctica e implantación de una IDE, utilizando herramientas de consulta y publicación de servicios cartográficos que cumplen con los estándares definidos por la OGC (Open Geospatial Consortium).

Por último, se verá cómo desarrollar paso a paso un Geoportal WEB utilizando software libre que cumple con los estándares de interoperabilidad IDE.

Ante todo, el propósito final será demostrar que una herramienta IDE es una de las mejores soluciones que existen hoy en día para el uso de información cartográfica distribuida. Además, gracias a la existencia de herramientas de software libre y de una gran comunidad de usuarios y desarrolladores, se puede afirmar que optar por una solución IDE está al alcance de cualquier tipo de usuario hoy en día.

3 Competencias Grado Superior de Ingeniería Informática

En el desarrollo de este trabajo de final de Grado, se ha potenciado ciertas competencias, las cuales, algunas son propias del grado Superior de Ingeniería Informática, y otras son competencias adquiridas a lo largo de mi experiencia profesional en este campo.

Competencia
Diseñar y desarrollar proyectos informáticos.
Gestión y ejecución de proyectos de investigación desarrollo e innovación en el ámbito de esta ingeniería.
Planificación, proyecto, dirección, ejecución y gestión de procesos y productos de aplicación en la sociedad de la información en el ámbito geomático.
Sintetizar de forma crítica información proveniente de fuentes diversas.
Planificar eficientemente el trabajo.
Comunicarse de forma oral y escrita en su lengua nativa.
Aportar soluciones creativas en la resolución de problemas y resolver con acierto problemas multidisciplinares.
Aprender autónomamente identificando necesidades, procedimientos y evaluando el propio aprendizaje.
Trabajar aplicando criterios de calidad.
Compromiso ético en el trabajo.
Conocimientos y gestión en equipos multidisciplinares de infraestructuras de datos espaciales (IDE).

Tabla 1: Competencias

4 IDE. ¿Qué es?

Las infraestructuras de datos espaciales han supuesto un cambio radical en el funcionamiento de los Sistemas de Información Geográfica, que han pasado de ser sistemas monolíticos aislados en una organización a ser sistemas modulares basados en servicios web operables entre distintos organismos.

Se puede decir que el germen de lo que hoy en día se entiende como infraestructura de datos espaciales (a partir de ahora IDE) se gestó en la conferencia de las Naciones Unidas sobre medio ambiente y desarrollo, celebrada en Río de Janeiro en el año 1992. Es ahí donde por primera vez se menciona como crítica la importancia de la información geográfica, ya que se considera que tal información es un elemento de apoyo crítico en la toma de decisiones y en el tratamiento de problemas en la inmensa mayoría de los campos (medio ambiente, urbanismo, desarrollo, cooperación internacional, gestión de catástrofes, etc.). Además, esta información no sólo es útil a escala regional o nacional, sino también a escala global. Por lo tanto, se observó la necesidad de encontrar un medio para conocer y compartir la información geográfica disponible en todos los organismos de forma sencilla.

Hasta el momento, cada organización generaba información geográfica de acuerdo con sus necesidades, ya fueran administraciones públicas, empresas u otras organizaciones, y las iniciativas para compartirla eran limitadas. Esto provocaba que la información y los esfuerzos humanos, materiales y económicos se duplicaran constantemente. Para evitarlo, se define el concepto de IDE, entendida como la agregación de conjuntos de datos espaciales y servicios de datos espaciales, metadatos, servicios y tecnologías de red, acuerdos sobre puesta en común, acceso y utilización, y mecanismos, procesos y procedimientos de coordinación y seguimiento establecidos entre los organismos (Directiva/2007/2/CE). Los conceptos básicos de esta definición son los siguientes:

- **Información contenida**: una IDE es más que un conjunto de datos espaciales que se ofrecen para ser utilizados por más usuarios que los concebidos originalmente. Una IDE debe contener también servicios para descubrir qué datos hay disponibles, servicios para acceder a la información geográfica y a la cartografía, y finalmente metadatos que describan los conjuntos de datos y los servicios disponibles.
- **Utilización de estándares**: los servicios y tecnologías utilizados en una IDE deben basarse en estándares aceptados internacionalmente para permitir la interoperabilidad técnica entre los sistemas de la IDE.
- **Establecimiento de acuerdos**: el aspecto más importante de la IDE es el establecimiento de acuerdos entre los actores para utilizar tecnologías y servicios compatibles y para unificar los modelos de datos de cada nodo de la IDE, de forma que se permita la combinación de los conjuntos de datos espaciales y la interacción de los servicios sin intervención manual repetitiva y que el resultado sea coherente (Directiva/2007/2/CE). Estos acuerdos también disminuyen los costes que conlleva la integración de la información proveniente de diversas fuentes, y eliminan la necesidad de desarrollar paralelamente instrumentos para el descubrimiento, intercambio y explotación de datos espaciales.

Como se verá en detalle en el capítulo 5 de este documento, INSPIRE (Directiva/2007/2/CE) es la directiva europea que se ha establecido como marco de políticas, disposiciones institucionales, tecnologías, datos y personal de todas las IDEs en la Unión Europea. INSPIRE dicta una serie de normas que son de cumplimiento obligado para las IDEs europeas; asimismo, cada país miembro puede establecer normas más restrictivas para su ámbito siempre que no entren en conflicto con INSPIRE.

4.1 Componentes

En la figura siguiente se muestran los componentes que forman parte de una IDE.

Figura 1: Componentes de una IDE

- **Datos**. Antes de la aparición de la IDE los datos se encontraban dispersos y fragmentados en distintas administraciones y/o empresas, lo que provocaba dos problemas: por un lado, se duplicaban esfuerzos en la captura y mantenimiento de la información geográfica, y por otro lado, era complicado encontrar cartografía apropiada para un trabajo porque había que solicitarla a distintos organismos, siguiendo distintos trámites burocráticos, y había que comprobar que la información obtenida fuera coherente.

  Las IDEs evitan estos problemas mediante la aplicación de dos técnicas. En primer lugar, se obliga a capturar y mantener la información geográfica una única vez allí donde puede hacerse de modo más efectivo. De esta forma, la cartografía que se utiliza siempre es la mejor para cada caso y el coste de captura y mantenimiento se minimiza. En segundo lugar, las IDEs obligan a compartir la información geográfica mediante servicios de datos basados en estándares, lo que permite que el acceso a la información se pueda hacer de

forma interoperable utilizando herramientas informáticas de análisis y visualización.

Otro pilar fundamental de las IDEs son los metadatos, imprescindibles para conocer siempre qué cartografía se está usando y el único medio del que se dispondrá para poder seleccionar qué cartografía se ajusta mejor a cada caso de uso.

- **Políticas**. El apoyo político es fundamental para el desarrollo correcto de las IDEs, ya que el mayor esfuerzo de captura y mantenimiento de información geográfica se realiza en al sector público. Por ello, las políticas que determine el sector público con respecto a su mantenimiento, recolección y uso son las que mayor impacto tienen sobre la IDEs. Sin embargo, no se puede olvidar el sector privado, ya que es uno de los más interesados en el uso de la información que proporciona una IDE.

- **Estándares**. Es indispensable para el buen funcionamiento de una IDE que sea interoperable, tanto desde el punto de vista técnico como semántico. La interoperabilidad técnica consiste en que los participantes en el proceso de comunicación utilicen los mismos lenguajes, lo que implica llegar a acuerdos para la utilización de los mismos formatos y servicios de datos basados en estándares aceptados internacionalmente. La interoperabilidad semántica consiste en que la información compartida sea coherente en cuanto a significado, lo que implica que los organismos lleguen a acuerdos respecto a los modelos conceptuales de la información compartida. A lo largo de éste módulo, veremos qué organizaciones internacionales son las encargadas de dictar los estándares requeridos para el intercambio de información geográfica, así como las principales especificaciones que ya están en uso y disponibles para su utilización.

- **Redes accesibles**. Un sistema de información que ignore la gran importancia que tiene Internet hoy en día está condenado al fracaso. En el campo de las IDEs es necesario que haya redes accesibles a los usuarios y que estas redes soporten el tráfico producido cuando un usuario realiza una petición a un servidor y éste le responde con la información geográfica solicitada.

- **Usuarios.** Uno de los principales pilares dentro de una IDE son los usuarios, ya que sólo si una IDE cubre sus necesidades éstos aceptarán esta nueva forma de trabajar, con lo cual las IDEs se mantendrán y evolucionarán. Por lo tanto, es importante conocer quiénes son los usuarios potenciales de cada una de las IDEs y las necesidades que tendrán para poder definir los roles e identificar y evitar conflictos de interés entre usuarios. Todo ello permitirá conseguir el máximo nivel de satisfacción de los usuarios de la IDE.

4.2 Niveles

Una IDE está formada por un conjunto de nodos, cada uno de ellos administrado por un organismo responsable. Los nodos de una IDE se encuadran en distintos niveles y tienen distintos tipos de relaciones, en función del tipo de organismo responsable del nodo. La figura siguiente muestra de forma esquemática estos niveles y las relaciones que se establecen entre ellos.

Figura 2: Niveles de una IDE

En primer lugar, se puede clasificar el nodo IDE en función de la posición en la jerarquía administrativa que ocupe el organismo responsable del nodo. En la figura se dividen los nodos en siete niveles, desde la IDE global que está en el nivel superior hasta las IDEs locales y las IDEs corporativas en el nivel inferior. Parece claro que el detalle de la información geográfica crece según se desciende en esta jerarquía, de la misma manera que el área geográfica cubierta por la IDE también disminuye. A modo de ejemplo, la IDE de España cubre todo el territorio de España pero no puede recoger la información a una escala más detallada que 1:25000. La IDE de Galicia, comunidad autónoma de España, cubre un territorio menor, pero puede recoger la información a una escala de 1:5000. Finalmente, la IDE de un municipio de Galicia cubre un territorio mucho más pequeño, pero puede recoger la información con mucho más detalle.

Entre los nodos de la IDE hay dos tipos de relaciones: las relaciones horizontales y las relaciones verticales. Las relaciones horizontales son las que se producen entre nodos de la IDE del mismo nivel. Un ejemplo claro de este tipo de relación es la que se produce para compartir información fronteriza, de manera que los datos sean continuos y coherentes. Por otra parte, las relaciones verticales son las que se producen entre nodos de la IDE de distinto nivel. Un ejemplo de este tipo de relación es el caso en el que un nodo de nivel superior consulta un nodo de nivel inferior para responder una

consulta que no puede responder con su propia información, o cuando un nodo de nivel superior recopila información de los nodos inferiores para agregarla y generar información nueva. Otro ejemplo de este tipo de relación ocurre cuando un nodo de nivel superior impone requisitos a los nodos inferiores para hacer que su información o su modo de funcionamiento sea coherente.

4.3 Actores

Los actores involucrados en el funcionamiento de una IDE son varios, tal y como se puede observar en la figura siguiente.

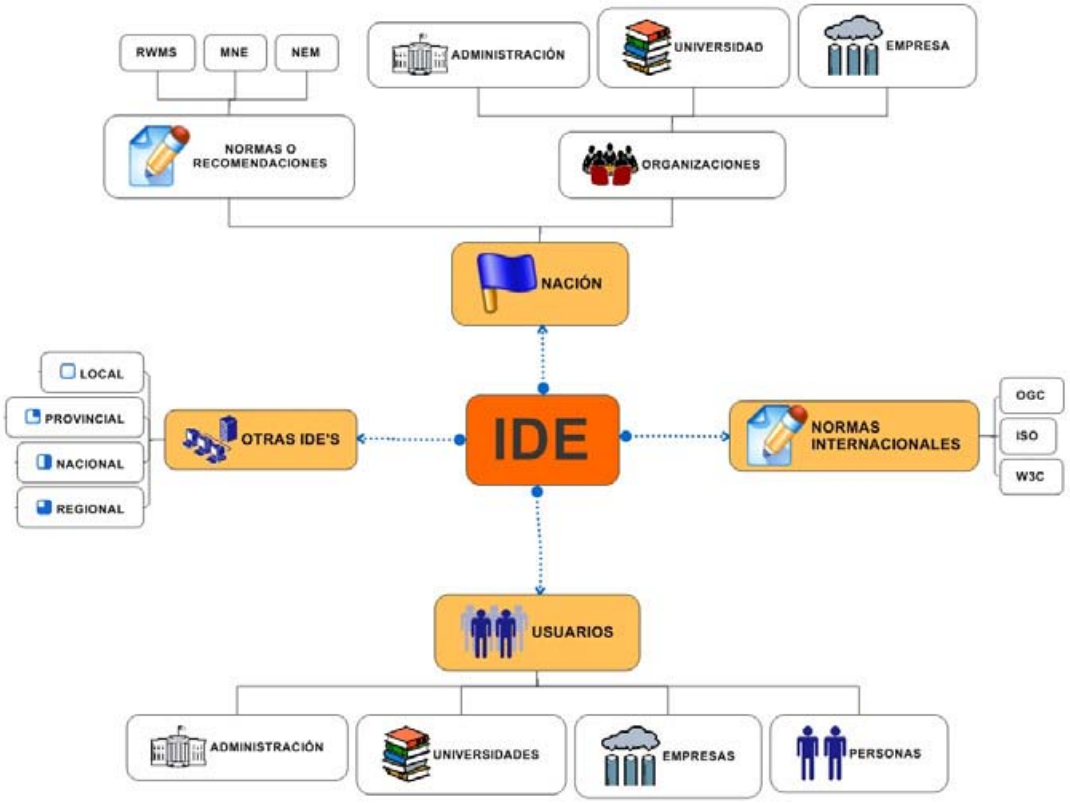


Figura 3: Actores de una IDE

En primer lugar, se encuentran los usuarios de la IDE, que son los que determinarán su éxito o su fracaso. Entre los usuarios podemos considerar distintos tipos: administraciones públicas que necesitan la información para realizar sus tareas (por ejemplo, las áreas de urbanismo de los municipios o los organismos de gestión de las cuencas hidrográficas), empresas que pueden usar la información para sus negocios (por ejemplo, empresas de elaboración de planes urbanísticos), universidades y centros de investigación, o personas individuales. La IDE debe recoger todos estos tipos de usuarios, ya que el éxito de la misma radica en saber cubrir las necesidades del mayor número posible de usuarios.

Otro tipo de actor involucrado en una IDE son los organismos internacionales de creación de estándares, en concreto la Organización de Estandarización Internacional, la ISO, y sus miembros nacionales, el *Open Geospatial Consortium* (OGC) y el *World Wide Web Consortium* (W3C). Estos organismos definen las normas y estándares que formarán la base tecnológica de la IDE y que permitirá su interoperabilidad.

A continuación encontramos al responsable particular de cada nodo IDE. Este responsable puede ser una administración pública, una empresa, una universidad, un centro tecnológico u otro tipo de organismo que se hace responsable de la administración del nodo IDE y de establecer los estándares que se deben cumplir dentro de su ámbito, así como las normas o recomendaciones adicionales. A modo de ejemplo, para la IDE de España el responsable es el Instituto Geográfico Nacional, para la IDE de Galicia el responsable es el Sistema de Información Territorial de Galicia y para la IDE de la provincia de A Coruña el responsable es la Diputación de A Coruña.

Finalmente, el resto de nodos de la IDE también son actores de una IDE, ya que se debe alcanzar la coherencia de la información evitando la duplicidad de esfuerzos.

5 Acuerdos internacionales

¿Qué acuerdos internacionales se han establecido para poner en marcha las IDEs? ¿Qué normativas ha aprobado la Unión Europea para construir la IDE de Europa? ¿Qué acuerdos se han alcanzado en España para construir la IDE de España? ¿Qué organismos internacionales definen estándares referentes a información geográfica?

El elemento más importante de las IDEs, más allá de la información, de los servicios y de las tecnologías, son los acuerdos alcanzados entre los organismos para utilizar tecnologías y modelos de datos compatibles y coherentes. En este punto se describen estos acuerdos, tanto a escala global, como continental y nacional. Además, se describen también los organismos que definen estándares que afectan a la información geográfica.

5.1 Inspire

INSPIRE (*Infraestructure for Spatial Information in Europe*) es una iniciativa de la Comisión Europea cuyo funcionamiento se recoge en la Directiva 2007/2/CE del Parlamento Europeo y del Consejo, de 14 de marzo de 2007, publicada en el Diario Oficial de la UE (DOUE) el 25 de Abril de 2007, que tiene como objetivo la creación de una Infraestructura de Datos Espaciales en Europa. La Directiva establece los objetivos, y los Estados miembros tendrán dos años desde su publicación para ajustar sus respectivas legislaciones y procedimientos administrativos nacionales.

INSPIRE ha sido desarrollada en colaboración con Estados miembro y países en estado de adhesión con el propósito de hacer disponible información geográfica relevante, concertada y de calidad, de forma que se permita la formulación, implementación, monitorización y evaluación de las políticas de impacto o de dimensión territorial, de la Comunidad Europea.

INSPIRE es una iniciativa legal que establece estándares y protocolos de tipo técnico, aspectos organizativos y de coordinación, políticas sobre la información que incluye el acceso a los datos y la creación y mantenimiento de información espacial.

INSPIRE es el primer paso de una amplia iniciativa multilateral que inicialmente dirigirá su interés sobre la información espacial necesaria para políticas medioambientales y que estará disponible para satisfacer las necesidades prácticas de otras áreas, tales como la agricultura y el transporte.

5.1.1 Objetivos

INSPIRE intenta conseguir fuentes armonizadas de IG (Información Geográfica) para dar soporte a la formulación, implementación y evaluación de políticas comunitarias como medio ambiente, etc.

5.1.2 Desarrollo de Inspire

Grupo de Expertos:

- 2 por Estado Miembro (uno Medio Ambiente, otro IG (Agencia Cartográfica Nacional))
- Observadores: Organizaciones europeas de MA e IG (*EuroGeographics*, EUROGI, EEA, WWF, EPRO...)
- Expertos por España:
  - Roberto Vallejo Bombín Dirección General de Biodiversidad. Ministerio Medio Ambiente.
  - Sebastián Mas Mayoral. Instituto Geográfico Nacional. Ministerio de Fomento.

5.1.3 Directiva de Inspire

¿Qué comprende?

- Datos espaciales (IG) y sus Metadatos
  - Referentes al territorio UE, en formato digital, del sector público, comprendidos en los temas especificados en los Anexos I, II y III de la directiva.
  - Disposiciones para datos aportados por terceros pero siempre sujetos al consentimiento de ellos.
- Servicios sobre los datos espaciales anteriores

Metadatos

Los estados miembros son los encargados de la creación de metadatos para los conjuntos y servicios de datos espaciales y de su actualización

- Los metadatos incluirán información sobre:
  - Conformidad de los conjuntos de datos.
  - Derechos de utilización.
  - Calidad y validez de los datos.
  - Autoridades públicas responsables.
  - Conjuntos de datos espaciales limitados al acceso público y las razones.
- Deben establecerse catálogos de datos espaciales y servicios

Interoperabilidad

Se define la interoperabilidad, como la capacidad de los sistemas de tecnologías de la información y las comunicaciones (TIC), y de los procesos empresariales a los que apoyan, de intercambiar datos y posibilitar la puesta en común de información y conocimientos. Para conseguir esta interoperabilidad debe cumplirse:

- Que existan especificaciones armonizadas que atenderán a los siguientes aspectos de los datos espaciales (Normas de aplicación):
  - Un sistema común de identificadores únicos.
  - Relaciones entre los objetos espaciales.
  - Los principales atributos y el tesauro multilingüe correspondiente.

  - La forma de intercambiar la información sobre la dimensión temporal de los datos.
  - La forma de intercambiar las actualizaciones de los datos.
- Consistencia entre ítems situados en una misma posición geográfica.
- Consistencia entre información referida al mismo objeto en escalas diferentes

Servicios de red de datos espaciales

Los estados miembros establecerán servicios de catálogo que den acceso a los metadatos y a los conjuntos y servicios de datos. Además, establecerán, operarán y darán acceso a la siguiente red de servicios:

- Servicios de localización
- Servicios de visualización
- Servicios de descarga
- Servicios de transformación de los datos espaciales
- Servicios de "acceso a servicios de datos espaciales"
- Estos servicios deberán ser fáciles de utilizar y accesibles vía Internet (Geoportal comunitario + Puntos de acceso de los Estados miembros).

Armonización y reutilización de datos

Medidas de los Estados miembros para armonización de conjuntos de datos.

- Acceso por la CE a dichos conjuntos de datos y servicios.
- Medidas de los Estados miembros que impidan falseamiento de la competencia, cuando se desarrollen actividades comerciales.
- CE regulará derechos acceso y reutilización

5.2 Organismos internacionales

En este apartado se muestran las características principales de las organizaciones que se dedican a establecer normativa dentro del mundo de la información geográfica.

5.2.1 ISO

La Organización Internacional para la Estandarización (ISO) es una organización internacional no gubernamental, compuesta por representantes de los organismos de normalización (ONs) nacionales, que produce normas internacionales industriales y comerciales. Dichas normas se conocen como normas ISO y su finalidad es coordinar las normas nacionales, en consonancia con el Acta Final de la Organización Mundial del Comercio, con el propósito de facilitar el comercio, facilitar el intercambio de información y contribuir a la transferencia de tecnologías.

En el ámbito de la información geográfica, la familia ISO 19100 es una familia de normas para la información geográfica desarrollada por el Comité Técnico 211, llamado Geomática/Información Geográfica. Entre las normas ISO pertenecientes a esta familia cabe destacar las siguientes:

- ISO 19112 ”Información Geográfica - Sistema de Referencia Basado en Identificadores Geográficos”
- ISO 19115 ”Información Geográfica - Metadatos”
- ISO 19119 ”Información Geográfica - Servicios”

5.2.2 OGC

El *Open Geospatial Consortium* (OGC) es una organización sin ánimo de lucro compuesta por más de 335 compañías, agencias gubernamentales y universidades, que participan en un proceso de consenso para desarrollar especificaciones de geoprocesamiento públicas.

El OGC pretende liderar el desarrollo, la promoción y la armonización de los estándares abiertos y arquitecturas a escala global con el fin de que las distintas aplicaciones para usuarios puedan integrar la información y los servicios geoespaciales independientemente de su localización o nivel.

En el proceso de desarrollo de estándares del OGC se distinguen dos tipos: las especificaciones abstractas y las especificaciones de implementación. El fin de las especificaciones abstractas es crear y documentar un modelo conceptual suficiente para permitir la creación de las especificaciones de implementación. Éstas son plataformas tecnológicas no ambiguas para la implementación de estándares industriales o aplicaciones de software.

Como ya se ha visto, la ISO también desarrolla una serie de estándares relacionados con la información geográfica, por lo que se pretende armonizar las especificaciones abstractas del OGC y los estándares ISO, de forma que sean o lleguen a ser iguales.

A continuación se muestra un gráfico en el que se puede ver el flujo de trabajo dentro del OGC y de éste con ISO y con los usuarios.

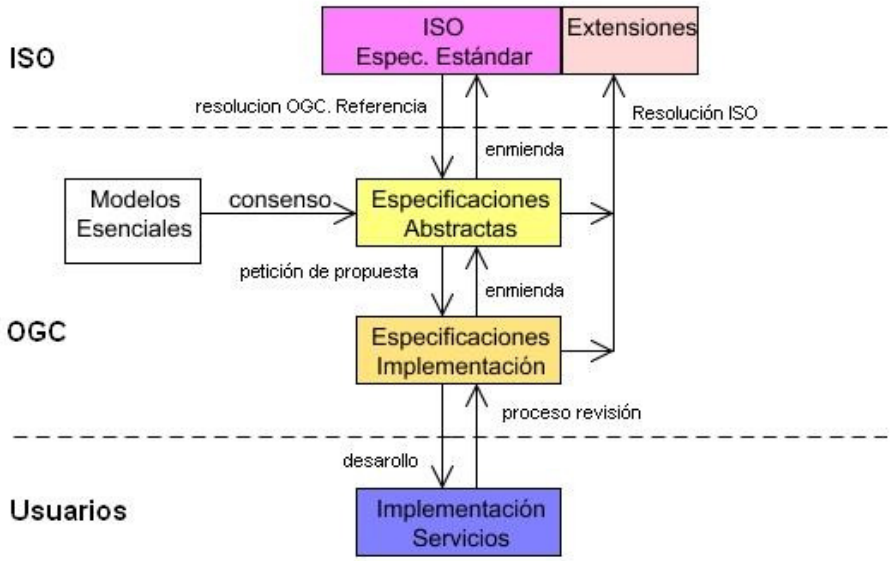


Figura 4: Flujo de trabajo entre OGC, ISO y Usuarios

5.2.3 W3C

El Consorcio *World Wide Web* (W3C) es una asociación internacional formada por organizaciones miembro del Consorcio, personal y público en general que trabajan conjuntamente para desarrollar estándares web. La misión del W3C es guiar la Web hacia su máximo potencial a través del desarrollo de protocolos y pautas que aseguren el crecimiento futuro de la Web.

El W3C fue creado en 1994 por Tim Berners-Lee, creador del *World Wide Web*, para asegurar la compatibilidad y el acuerdo entre los miembros industriales en la adopción de nuevos estándares. Para que un estándar sea aprobado por el W3C tiene que pasar los siguientes trámites:

- Borrador de trabajo (*Working Draft*)
- Última convocatoria (*Last Call*)
- Propuesta de recomendación (*Proposed Recommendation*)
- Recomendación candidata (*Candidate Recommendation*)

Finaliza con la aprobación de la "Recomendación", lo que equivale a la homologación de la propuesta, es decir, un nuevo estándar público y abierto para la Web. La mayoría de estas recomendaciones están secundadas por los fabricantes de herramientas (navegadores, editores, buscadores) y tecnologías (servicios web, directorios, registros). Esta competencia en exclusiva del W3C para crear estándares abiertos es crucial, pues de ella depende que ningún fabricante alcance nunca el monopolio de explotación de la Web. W3C no tiene un programa de certificación.

Uno de los estándares aprobados por el W3C de gran uso en el mundo de la información geográfica es la recomendación sobre el lenguaje de marcas extendido (XML).

6 Cartografía base

Para el desarrollo de este trabajo, se ha hecho uso de la cartografía proveniente de un proyecto en el cual han colaborado diferentes administraciones públicas españolas.

El modelo del proyecto Cartociudad, que aunque está más orientado a cartografía urbana, puede tener cabida el conjunto de información rústica y urbana, sobre todo si se tiene en cuenta que es un proyecto en el cual colaboran diferentes administraciones, y que el objetivo final de la cartografía catastral es la unificación de las cartografías rústica y urbana, creando una cartografía continua.

El Proyecto CartoCiudad tiene como objetivo la creación de la Base de Datos Oficial de la Administración General del Estado de red viaria, con estructura topológica de SIG, (Sistema de Información Geográfica) de ciudades y núcleos de población españoles, basada en cartografía digital oficial con viales e información textual, que permitirá la navegación asistida y diversos usos temáticos, con continuidad geográfica asegurada en todo el territorio nacional, utilizando como entramado de soporte la base de datos BCN25 del Instituto Geográfico Nacional.

En consecuencia, CartoCiudad pretende conseguir la cartografía digital «oficial» de la AGE (Administración General del Estado), con continuidad asegurada en los ámbitos urbano y rústico, utilizable como información geográfica de referencia por todos.

La producción de la Base de Datos CartoCiudad se realiza a partir de las bases de datos cartográficos oficiales de la Administración General del Estado, que incluyen, para las ciudades y entidades de población españolas, la estructura urbana, las redes viarias, y la información toponímica asociada de nombres de calles, numeración de portales, distritos y secciones censales y códigos postales. Por tanto, el proyecto integra y armoniza:

- Información extraída de la Base Cartográfica Numérica a escala 1:25.000 (BCN25) de la Dirección General del Instituto Geográfico Nacional (IGN), especialmente la referente a redes hidrográfica y de transportes y a las entidades de población, que actúa como base de referencia para dar continuidad territorial al producto resultante.
- Información cartográfica extraída de los Catastros Inmobiliarios, de la Dirección General de Catastro (DGC), especialmente en los correspondientes al ámbito urbano.
- Información sobre nombres de calles obtenida del Censo Electoral mantenido por el Instituto Nacional de Estadística (INE).
- Información sobre distritos y secciones censales del INE.
- Información sobre los distritos postales que elabora y mantiene la Sociedad Estatal Correos y Telégrafos S.A.

6.1 Ámbito del proyecto

CartoCiudad, como proyecto de ámbito nacional, comprende todo el territorio español: la España peninsular, las Islas Canarias e Islas Baleares así como las Ciudades Autónomas de Ceuta y Melilla.

Y como base de datos oficial de la red viaria, el ámbito de los datos es fundamentalmente urbano, constituyendo la mayoría de la información la relativa a núcleos de población, si bien también se representan carreteras y caminos para posibilitar la navegación continua por todo el territorio.

6.2 Escala

Debido a que los datos proceden de distintas fuentes y que pertenecen tanto al ámbito urbano como interurbano, existen distintas escalas de captura de datos:

- Zonas urbanas: Escala de captura 1: 1.000
- Zonas interurbanas: Escala de captura 1: 25.000

6.3 Sistema de Referencia

El Sistema de Referencia de los datos de Cartociudad es el Sistema de Referencia Geodésico oficial de España: *European Terrestrial Reference System* 1989 - ETRS89 (según la codificación del *European Petroleum Survey Group*: EPSG 4258). En cuanto al Sistema de Representación Cartográfica utilizado, CartoCiudad emplea las coordenadas geográficas latitud y longitud. Conforme a la finalidad del proyecto CartoCiudad, sólo se incluye información planimétrica, no disponiendo de datos altimétricos asociados.

Para este trabajo se ha empleado una reproyección de la cartografía al sistema WGS84 EPSG 4326, que también utiliza Latitud – Longitud.

6.4 Capas de información

Nombre	División Territorial
Definición	Divisiones administrativas de municipios, provincias y comunidades autónomas.
Origen de datos	Geometría: Registro Central de Cartografía del Instituto Geográfico Nacional. Denominación: Ministerio de Administraciones Públicas.
Atributos	Código INE de Provincia y Municipio Nombres oficiales de Comunidad Autónoma, Provincia y Municipio
Geometría	Polígono
Escala visualización	Comunidad Autónoma a todas las escalas Provincia > 1:5.000.000 Municipio > 1:40.000

Tabla 2: Información sobre la capa de División Territorial

Nombre	Código Postal
Definición	Todo el territorio nacional está dividido en distritos postales. El código postal es un número de 5 dígitos que identifica un área de reparto postal (o distrito postal), de los cuales los dos primeros se corresponden con el código de provincia. Hay municipios con un único distrito postal, otros que por su tamaño y población poseen varios códigos y, por último, algunos municipios comparten un único código postal.
Origen de datos	Sociedad Estatal de Correos y Telégrafos
Atributos	Código postal
Geometría	Polígono
Escala visualización	Código Postal> 1:40.000

Tabla 3: Información sobre la capa de Código Postal

Nombre	Sección censal
Definición	Subdivisión de los términos municipales empleada para aquellos trabajos encomendados al INE para los que es necesaria una división infra municipal, como, por ejemplo, para aquellos relacionados con fines electorales. De acuerdo con la Ley Electoral cada sección incluye un máximo de 2.000 electores y un mínimo de 500
Origen de datos	Instituto Nacional de Estadística (INE)
Atributos	Código INE del Municipio en que se encuentra la Sección Censal Nombre del Municipio en que se encuentra la Sección Censal Código de Sección Censal Código de Distrito Censal
Geometría	Polígono
Escala visualización	Sección Censal > 1:4.000

Tabla 4: Información sobre la capa de Sección Censal

Nombre	Topónimo
Definición	Nombre propio de una entidad geográfica de interés que está georreferenciado. De momento, la toponimia incluida en CartoCiudad comprende únicamente el ámbito urbano.
Origen de datos	Dirección General del Catastro (DGC)
Atributos	Texto
Geometría	Punto
Escala visualización	Topónimo> 1:10.000

Tabla 5: Información sobre la capa de Topónimos

Nombre	Portal-Punto kilométrico
Definición	Esta capa contiene tanto los portales como los puntos kilométricos de CartoCiudad. El número de portal o de policía permite la identificación del acceso desde una vía a cada construcción, y se representa como un punto en el límite de la parcela a la que pertenece. El punto kilométrico (PK) marca el kilometraje de una carretera, y se representa como un elemento puntual sobre el trazado de la carretera.
Origen de datos	Portal: Dirección General del Catastro (DGC) PK: Base Cartográfica Numérica a escala 1:25.000 (BCN25) del Instituto Geográfico Nacional (IGN)
Atributos	Número Letra
Geometría	Punto
Escala visualización	Portal-Punto kilométrico> 1:5.000

Tabla 6: Información sobre la capa de Portal – Punto Kilométrico

Nombre	Vial
Definición	Esta capa está constituida por el entramado viario urbano e interurbano: calles, carreteras y caminos, representados por su eje.
Origen de datos	Calles: su geometría procede de Catastro, aunque es revisada y actualizada con las ortofotografías del Plan Nacional de Ortofotografía Aérea; por otra parte, el INE aporta la denominación de dichas vías. Carreteras y caminos: su geometría procede de la BCN25 del IGN; la denominación se corresponde también con la almacenada en la BCN25, contrastada con el Mapa Oficial de Carreteras del Ministerio de Fomento.
Atributos	Tipo de vía Nombre de la vía Código INE de la vía (en caso de vial urbano) Código de Catastro de la vía (en caso de vial urbano) Longitud del tramo de vía
Geometría	Línea
Escala visualización	Calle > 1:10.000 Carretera y Camino > 1:160.000

Tabla 7: Información sobre la capa de Vial

Nombre	Fondo Urbano
Definición	Esta capa contiene manzanas, parcelas y construcciones, así como aquellas líneas que configuran el modelado urbano, como, por ejemplo, las líneas de acera. Las manzanas se definen como aquellos polígonos formados por una o varias parcelas, las cuales, a su vez, son las porciones de suelo que delimitan los bienes inmuebles. Por último, las construcciones existentes dentro de las parcelas se corresponden con edificios, instalaciones industriales u obras de urbanización y de mejora.
Origen de datos	Dirección General del Catastro (DGC)
Atributos	Referencia catastral de la parcela Cobertura (altura y tipología de los edificios)
Geometría	Polígono
Escala visualización	10.000 > Manzana > 1:200.000 2.000 > Parcela > 1:10.000 Construcción y acera > 1:2.000

Tabla 8: Información sobre la capa de Fondo Urbano

7 Arquitectura

7.1 Arquitectura física

Las arquitecturas multinivel o programación por capas son las más utilizadas actualmente para el diseño de sistemas informáticos. En estas arquitecturas a cada nivel se le designa una función simple, lo que permite el diseño de arquitecturas escalables, que pueden ampliarse con facilidad en caso de que las necesidades aumenten.

El diseño más utilizado actualmente es el diseño en tres capas:

- **Capa de presentación**. Es la que ve el usuario, también se la denomina "capa de usuario", presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso. También es conocida como interfaz gráfica y debe tener la característica de ser "amigable", entendible y fácil de usar para el usuario. Esta capa se comunica únicamente con la capa de negocio.
- **Capa de negocio**. Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio, o lógica de negocio, porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. También se consideran aquí los programas de aplicación.
- **Capa de datos**. Es donde residen los datos y es la encargada de acceder a los mismos. Está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Para un proyecto como este pero de mayor envergadura, es decir con más volumen de información espacial y servicios a publicar, lo ideal sería proponer una arquitectura de tres capas a tres niveles. Esto significa que cada capa de la arquitectura se distribuiría en una máquina distinta. De esta forma, se conseguiría que el rendimiento del producto fuera mayor.

Para el presente caso, el volumen de datos a almacenar y servicios a publicar no requieren gran potencial de máquina. Por esta razón, y para este caso, se ha centralizado en un solo servidor las capas de negocio, datos y presentación.

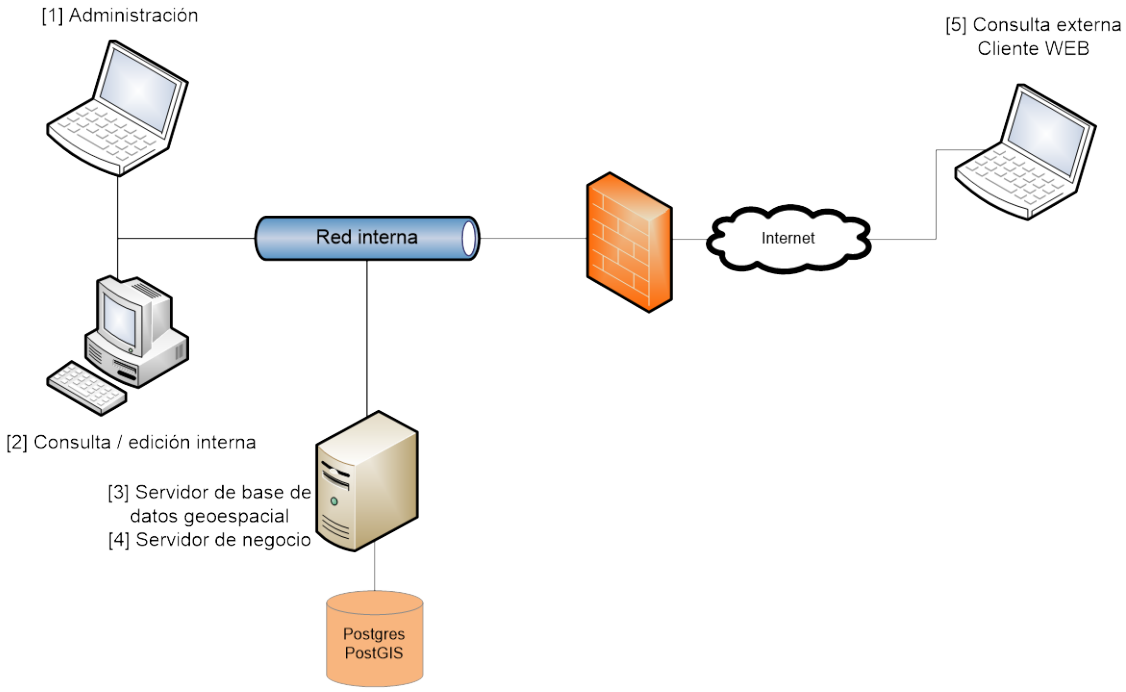


Figura 5: Esquema de la arquitectura física del proyecto

[1] Administración: Equipo que consulta la aplicación.

[2] Consulta interna: Equipo de consulta interna.

[3] Servidor de base de datos: Servidor de datos espaciales y alfanuméricos.

[4] Servidor de negocio: Servidor de servicios de cartografía y de acceso a la base de datos.

[5] Consulta web externa: Consulta de la WEB.

7.2 Arquitectura lógica

En este gráfico se muestran de forma resumida las partes a nivel funcional que pueden diferenciarse en la arquitectura.

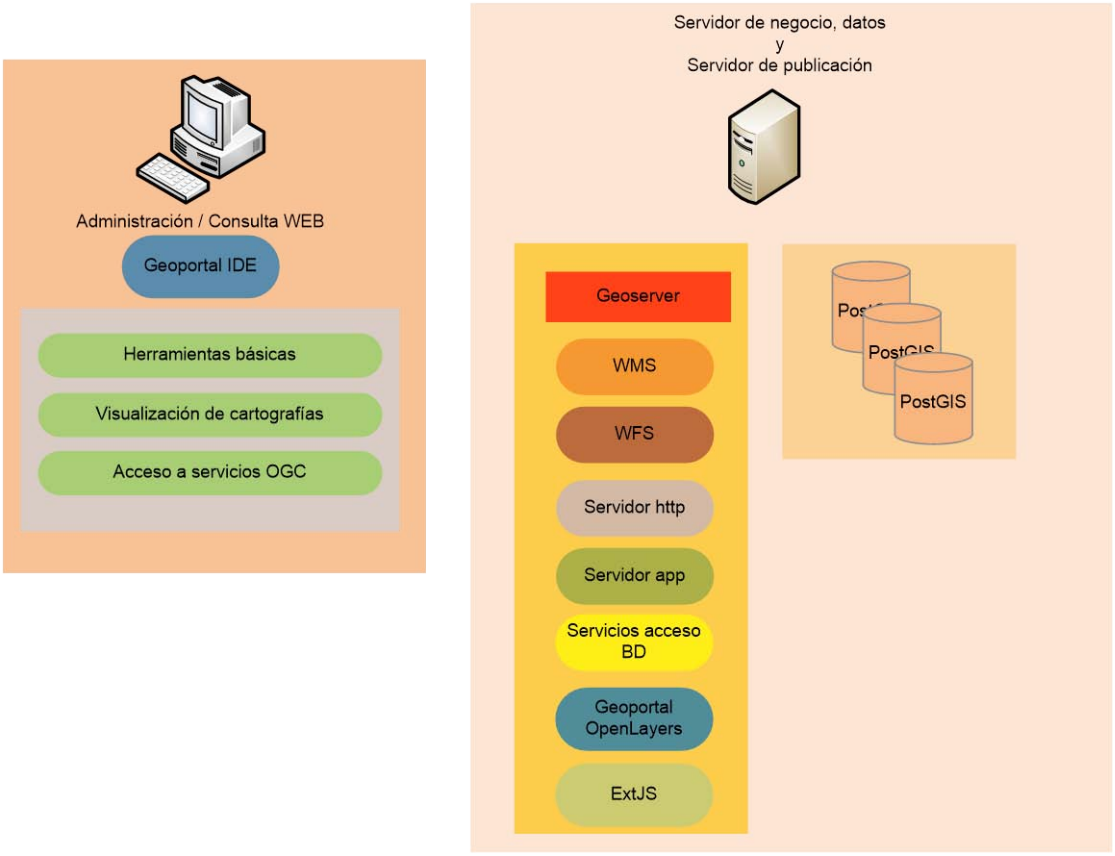


Figura 6: Esquema de la arquitectura lógica del proyecto

7.2.1 Máquina virtual de Java

Java 6

Una Máquina virtual Java (en inglés *Java Virtual Machine*, JVM) es un programa nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (el Java bytecode), el cual es generado por el compilador del lenguaje Java. Para evitar incompatibilidades entre las aplicaciones y simplificar la administración, se han unificado las versiones de las distintas aplicaciones que utilizan de forma que puedan funcionar con la misma versión de la máquina virtual. El código binario de Java no es un lenguaje de alto nivel, sino un verdadero código máquina de bajo nivel, viable incluso como lenguaje de entrada para un microprocesador físico. Como todas las piezas del rompecabezas Java, fue desarrollado originalmente por Sun Microsystems.

La JVM es una de las piezas fundamentales de la plataforma Java. Básicamente se sitúa en un nivel superior al Hardware del sistema sobre el que se pretende ejecutar la aplicación, y este actúa como un puente que entiende tanto el bytecode, como el sistema sobre el que se pretende ejecutar. Así, cuando se escribe una aplicación Java, se

hace pensando que será ejecutada en una máquina virtual Java en concreto, siendo ésta la que en última instancia convierte de código bytecode a código nativo del dispositivo final.

7.2.2 Servidor WEB

Apache 2

El servidor de mapas requiere, para su funcionamiento, un servidor web al cual realizar las peticiones de mapa para que éste pueda dar respuesta a las solicitudes del usuario. Se ha seleccionado Apache como servidor web debido a su integración con el servidor de aplicaciones seleccionado (Tomcat) y su carácter multiplataforma.

El servidor HTTP Apache es un software libre servidor HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo.

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo.

La arquitectura del servidor Apache es muy modular. El servidor consta de una sección core y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor web. Algunos de estos módulos son:

- mod_ssl - Comunicaciones Seguras vía TLS.
- mod_rewrite - reescritura de direcciones (generalmente utilizado para transformar páginas dinámicas como php en páginas estáticas html para así engañar a los navegantes o a los motores de búsqueda en cuanto a cómo fueron desarrolladas estas páginas).
- mod_dav - Soporte del protocolo WebDAV (RFC 2518).
- mod_deflate - Compresión transparente con el algoritmo deflate del contenido enviado al cliente.
- mod_auth_ldap - Permite autentificar usuarios contra un servidor LDAP.
- mod_proxy_ajp - Conector para enlazar con el servidor Jakarta Tomcat de páginas dinámicas en Java (servlets y JSP).

El servidor de base puede ser extendido con la inclusión de módulos externos entre los cuales se encuentran:

- mod_perl - Páginas dinámicas en Perl.
- mod_php - Páginas dinámicas en PHP.
- mod_python - Páginas dinámicas en Python.
- mod_rexx - Páginas dinámicas en REXX y Object REXX.

- mod_ruby - Páginas dinámicas en Ruby.
- mod_aspdotnet - Páginas dinámicas en .NET de Microsoft (Módulo retirado).
- mod_mono - Páginas dinámicas en Mono
- mod_security - Filtrado a nivel de aplicación, para la seguridad.

7.2.3 Servidor de aplicaciones
Apache Tomcat 6

Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

De esta forma Apache se encarga de publicar el contenido estático de un sitio web y redirige las peticiones dinámicas al servidor de aplicaciones Tomcat.

Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

7.2.4 Sistema de gestión de base de datos
Postgres 9.3 + PostGIS 2.1

PostGIS es un módulo para el motor de bases de datos objeto-relacional PostgreSQL, que añade soporte para tipos geográficos y lo habilita para ser usado como contenedor de información geoespacial, permitiéndole realizar operaciones de análisis geográfico. PostGIS ha sido desarrollado por Refractions Research Inc., empresa de consultoría y desarrollo especializada en aplicaciones de bases de datos y sistemas SIG responsable de proyectos como uDig. Está disponible bajo licencia GNU General Public License, sigue las especificaciones OpenGIS y cumple la norma Simple Feature Specification for SQL del Open Geospatial Consortium.

A día de hoy, PostGIS dispone de una interfaz de usuario con herramientas para la gestión de datos, soporta funciones de básicas de topología, transformación de coordenadas, validación de datos, programación de ApIs, etc. En sus próximos desarrollos tiene previsto proporcionar funcionalidades topológicas más avanzadas, soportar el almacenamiento de información ráster e incorporar herramientas para la

realización de cálculos de rutas, gestión de redes, superficies 3D y features complejas como curvas y splines.

Se trata de proyecto muy activo y muy difundido, con importantes referencias a nivel mundial. Como lugares de implantación con éxito, sus desarrolladores destacan proyectos como GlobeXplorer (agosto 2006), EU Joint Reserch Centre (Octubre 2006), Infoterra (United Kingdom, Octubre 2006), Institut Geografphique National (France, Octubre 2006), etc.

Otra prueba de su aceptación en el ámbito de los Sistemas de Información Geográficos de código abierto, es el gran número de productos informáticos que pueden usar esta base de datos. Clientes GIS como uDig, QGIS o GRASS trabajan directamente sobre PostGIS, el cliente Java JUMP puede acceder directamente a través de un plugin. En el campo de los servidores de mapas MapServer puede utilizar PostGIS como fuente de datos, al igual que GeoServer. Las librerías de GeoTools también soportan acceso a esta base de datos.

Los datos guardados en este repositorio también se pueden exportar a otros formatos GIS, como por ejemplo shapes o GML, utilizando la librería OGR C++ y herramientas basadas en líneas de comandos. En cuanto a lenguajes de programación se pueden utilizar todos los que soporta PostgreSQL como Perl, PHP, Pitón, TCL, C, C++, Java, C#, etc.

El cumplimiento de las especificaciones OGC y el continuo de desarrollo de nuevas funcionalidades GIS, sitúan a PostGIS en una posición aventajada dentro de la oferta de repositorios de datos espaciales de código abierto.

7.2.5 Servidor de mapas

Geoserver

GeoServer es un servidor de mapas que forma parte de la nueva generación de aplicaciones desarrolladas sobre la especificación J2EE. Está implementado sobre la plataforma Java, permitiendo el despliegue de la aplicación sobre cualquier servidor de aplicaciones conforme a la especificación J2EE, tanto libres (como Tomcat o Geronimo de Apache o JBoss de RedHat) como propietarios (WebLogic de BEA o WebSphere de IBM, entre otros).

Geoserver ha sido la referencia para el desarrollo de la norma OGC sobre la implementación de Web Feature Service (WFS). Además, destaca especialmente por dar soporte al protocolo Web Feature Service Transaccional (WFS-T), convirtiéndose no sólo en un servidor de cartografía, sino en un intermediario para la edición remota de información geográfica mediante estándares. El proyecto GeoServer ha hecho importantes esfuerzos, además, para posibilitar operaciones más avanzadas como el Styled Layer Descriptor (SLD), DescribeLayer, etc.

GeoServer es el producto estrella sobre la biblioteca GeoTools y está potenciado por la empresa Canadiense Refractions Research. Es un servidor multiplataforma, que puede ser instalado en sistemas operativos Windows y Linux. De entre a los formatos vectoriales que soporta se encuentran SHP, GML, PostGIS, MySQL, Oracle Spatial,

ArcSDE, Geomedia, MapInfo, Tigre y VPF. En este sentido destacar que, gracias al uso de un plugin de GeoTools para GeoServer, no necesita ser recompilado para incorporar nuevos formatos. En cuanto a formatos ráster resaltan ArcGRID, GeoTiff, World images (tiff/png/jpeg) y GT0P030, además de la implementación de mosaicos de imágenes y pirámides. GeoServer soporta, por último, proyecciones “on-the-fly” proporcionando más de 100 proyecciones por defecto.

Como ficheros de imágenes de salida Web utiliza jpeg, gif, png, SVG y KML. Además, es capaz de proporcionar a través de servicios WFS datos vectoriales en formato GML y Shapesfiles comprimidos. Mencionar, también, que incorpora un motor de validación para el chequeo de reglas topológicas y de atributos en operaciones de inserción para el mantenimiento de la integridad.

Está disponible bajo licencia GPL, siendo un producto totalmente liberalizado para su uso y difusión. A día de hoy la versión estable de GeoServer es la 2.5.1

8 Servicios

8.1 Servicio WMS

Un servicio web de mapas (Web Map Service, WMS) es un componente de software que produce dinámicamente “mapas” o “cartoimágenes” espacialmente georeferenciados a partir de información geográfica. Este estándar internacional define un “mapa” como una interpretación de la información geográfica en formato de imagen digital que puede ser vista en una pantalla de ordenador. De esta forma, no se accede a los datos en sí mismos, sino a una representación gráfica de ellos. Los mapas producidos por WMS se presentan normalmente en formatos de imagen, como PNG, GIF o JPEG, aunque también es posible hacerlo como SVG (Scalable Vector Graphics) u otros formatos de imagen.

La especificación WMS permite superponer visualmente información geográfica distribuida de forma simultánea por internet. Así, este “mapa” puede contener información de varias capas ráster y/o vectoriales, provenientes de uno o diversos servidores de cartografía, superpuestas en un determinado orden modificable y con un valor de transparencia para visualizar capas inferiores.

Figura 7: Obtención de un mapa a partir de servicios web de mapas

El WMS funciona del siguiente modo: una vez el cliente localiza el servidor WMS que aloja la información geográfica deseada, dirige una petición a dicho servidor. Éste accede a mostrar la información pedida y la devuelve al cliente de manera idónea para ser representada como una o más capas en un mapa compuesto por muchas capas.

Así, los servidores de mapas proveen al cliente de la información preparada para ser representada, y éstos, los clientes, lo hacen, con lo cual se puede representar información proveniente de fuentes distintas en una sola ventana.

En el siguiente gráfico se muestran las distintas operaciones de WMS y la respuesta que obtenemos con cada una de ellas

Figura 8: Operaciones y respuestas WMS

Entonces, un servidor de mapas básicamente puede hacer tres cosas:

- Producir un mapa.
- Responder a preguntas básicas sobre el contenido del mapa.
- Comunicar a otros programas qué mapas puede producir y cuáles de ellos pueden ser preguntados.

Para que todo esto sea posible, el OGC establece especificaciones para la interoperabilidad a escala de interfaz entre componentes para el intercambio de información geográfica, y define un vocabulario, una sintaxis y unos comandos para que tanto clientes como servidores WMS puedan comunicarse independientemente de las plataformas, formatos, etc. que se utilicen.

Para el WMS, el estándar internacional define tres operaciones: una devuelve un archivo XML con los metadatos del servicio y con los metadatos de las capas de información que contiene (GetCapabilities); otra, GetMap, devuelve un mapa en el que sus características y dimensiones están definidas de forma inequívoca, y la tercera, que es una operación opcional, GetFeatureInfo, devuelve información sobre características particulares mostradas en el mapa. Las operaciones WMS se pueden invocar usando un navegador web estándar y realizando las peticiones con el formato "*Uniform Resource Locators*", URL. El contenido de cada URL dependerá de la operación que solicitemos.

Como vemos en el siguiente dibujo, en función de las operaciones que hay disponibles tenemos un WMS básico, si implementa sólo las operaciones obligatorias, o un WMS consultable, si implementa las tres operaciones posibles.

Figura 9: Operaciones WMS

A continuación se describen de forma detallada cada una de las operaciones posibles con WMS, y se explican cada uno de los parámetros que se necesitan para realizar cada una de ellas.

Operaciones Web Map Service

Como ya se ha visto, hay tres operaciones definidas para Web Map Service: GetCapabilities, GetMap y GetFeatureInfo, esta última es opcional. Ahora vamos a ver las especificaciones de implementación y uso de estas operaciones WMS en el Hipertext Transfer Protocol (HTTP) Distributed Computing Platform (DCP).

Todas las operaciones que se describen a continuación están basadas en la especificación de implementación de OpenGIS® Web Map Server [WMS_sp], en su versión 1.1.1 de 16 de enero de 2002.Aunque existe una versión más actual de la especificación WMS, la 1.3, en la presente lección exponemos la 1.1.1 porque tanto la especificación de SLD como la de WMC están basadas en esta última especificación WMS en el momento de redactar esta lección.

GetCapabilities

Esta operación del servicio WMS es obligatoria y su propósito es obtener los metadatos del servicio, que es una descripción legible por el ordenador y los usuarios de la información del servidor, que contiene y acepta valores de parámetros de petición.

Parámetros de petición

Parámetro de la petición	Obligatorio / Opcional	Descripción
VERSION=version	Op	Versión de la petición
SERVICE=WMS	O	Tipo de servicio
REQUEST=GetCapabilities	O	Nombre de la petición
FORMAT=MIME_type	Op	Formato de salida de los metadatos del servicio
UPDATESEQUENCE=string	Op	Secuencia numérica o de cadena para el control de caché

Tabla 9: Parámetros de petición GetCapabilities WMS

Un ejemplo de una petición GetCapabilities a un servidor WMS puede ser:

```
http://localhost/cgi-bin/mapas?SERVICE=WMS&REQUEST=GetCapabilities&VERSION=1.1.1
```

Respuesta de GetCapabilities

La respuesta es un documento de capacidades en formato XML en el que podemos encontrar las siguientes secciones:

- <Service>: en esta sección encontramos información general sobre el servicio, como nombre, título, resumen o información de contacto con el proveedor del servicio.
- <Capabilities>: en esta sección se enumeran las operaciones que soporta el servicio, los formatos de salida y distintas capacidades del servidor. Se divide en las distintas subsecciones siguientes:
  - <Request>: se indican distintos parámetros para las peticiones y los formatos de salida que están disponibles para peticiones GetMap.
  - <Exception>: se enumeran los posibles formatos de salida para las peticiones erróneas.
  - <VendorSpecificCapabilities>: incluye capacidades añadidas al servicio por el proveedor.
  - <UserDefinedSymbolization>: indica la simbolización que hay que utilizar. Es aquí donde se señala si el servicio soporta SLD.
  - <Layer>: en esta subsección se muestran las capas (<Layer>) que contiene el servidor y se proporciona información sobre estas capas del tipo nombre, título, resumen, SRS, Bbox..., y el estilo asociado a dicha capa (<Style>).

GetMap

La operación GetMap devuelve un mapa o carto-imagen. Una vez se recibe la petición GetMap, un WMS debe satisfacer la petición o devolver una excepción del servicio.

Parámetros de petición

Parámetro de la petición	Obligatorio / Opcional	Descripción
VERSION=1.1.1	O	Versión de la petición
REQUEST=GetMap	O	Nombre de la petición
LAYERS=lista_capas	O	Lista separada por comas de la capa o capas demandadas. Es opcional si aparece el parámetro SLD.
STYLES=lista_estilos	O	Lista separada por comas de una interpretación de estilo por cada capa pedida. Es opcional si aparece el parámetro SLD.
SRS=namespace:identificador	O	Sistema de referencia de coordenadas
BBOX=minx,miny,maxx,maxy	O	Esquinas de la ventana de petición en unidades del CRS
WIDTH=ancho_salida	O	Ancho en píxeles de la cartoimagen
HEIGHT=alto_salida	O	Alto en píxeles de la cartoimagen
FORMAT=formato_salida	O	Formato de salida de la cartoimagen
TRANSPARENT=TRUE|FALSE	Op	Transparencia de la cartoimagen. Por defecto = FALSE
BGCOLOR=color_valor	Op	Valor de color hexadecimal en RGB, para el color de fondo. Por defecto = 255 255 255
EXCEPTIONS=formato_excepcion	Op	Formato en el que presenta las excepciones el WMS. Por defecto = XML

TIME=tiempo	Op	Valor de tiempo para la capa deseada
ELEVATION=elevacion	Op	Elevación de la capa deseada
SLD=URL_Styled_Layer_Descriptor	Op	URL del SLD
WFS=URL_WFS	Op	URL del WFS que provee de las características que se simbolizarán usando SLD

Tabla 10: Parámetros de petición GetMap WMS

Estos dos últimos parámetros sólo se usan en el caso de que el WMS soporte la especificación SLD.

Un ejemplo de petición GetMap puede ser el siguiente:

```
http://localhost/cgi-bin/mapas?
REQUEST=GetMap&
SERVICE=WMS&
VERSION=1.3.0&
LAYERS=paises&
SRS=EPSG:4326&
BBOX=-180.0,-90.0,180.0,83.623600000000001&
WIDTH=332&HEIGHT=161&
FORMAT=image/png&
STYLES=default&
TRANSPARENT=TRUE
```

Respuesta del GetMap

Figura 10: Ejemplo de respuesta de petición GetMap WMS

Una petición inválida GetMap debe devolver un fichero de error en el formato especificado en el valor EXCEPTIONS de la petición. Un ejemplo de un fichero de error puede ser el siguiente:

```
<?xml version='1.0' encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE ServiceExceptionReport SYSTEM
"http://schemas.opengeospatial.net/wms/1.1.1/exception_1_1_1.dtd">
<ServiceExceptionReport version="1.1.1">
<ServiceException>
msWMSDispatch(): WMS server error. Incomplete or unsupported WMS request
</ServiceException>
</ServiceExceptionReport>
```

GetFeatureInfo

Como ya se ha mencionado, esta operación es opcional. Es posible, pues, consultar las características de una capa cuando en éstas esté definido el atributo "queryable=1" (verdad) o lo hayan heredado.

Esta operación ha sido creada para proporcionar al cliente WMS más información sobre características que aparecen en las carto-imágenes que devuelve el servicio WMS. El cliente debe especificar las coordenadas píxel de la característica o características que desea consultar, así como la capa o capas a las que pertenece.

Parámetros de petición

Parámetro de la petición	Obligatorio / Opcional	Descripción
VERSION=1.1.1	O	Versión de la petición
REQUEST=GetFeatureInfo	O	Nombre de la petición
map request copy	O	Copia parcial de los parámetros del mapa demandado que genera el mapa del que queremos extraer información
QUERY_LAYERS=lista_capas	O	Lista separada por comas de la capa o capas demandadas
INFO_FORMAT=formato_salida	Op	Formato en el que devuelve la información de la característica (MIME_type)
FEATURE_COUNT=número	Op	Número de características de las que se va a devolver información (Por defecto = 1)
X=columna_pixel	O	Coordenadas x (columna) en píxeles de la característica del mapa
Y=fila_pixel	O	Coordenadas y (fila) en píxeles de la característica del mapa
EXCEPTIONS=*formato_excepción*	Op	Formato en el que presenta las excepciones el WMS. Por defecto = XML

Tabla 11: Parámetros de petición GetFeatureInfo WMS

Una petición GetFeatureInfo a un servidor WMS puede ser la siguiente:

```
http://ovc.catastro.meh.es/Cartografia/WMS/ServidorWMS.aspx?
REQUEST=GetFeatureInfo&
SERVICE=WMS&
QUERY_LAYERS=Cartografia&
VERSION=1.1.1&
INFO_FORMAT=application/vnd.ogc.gml&
LAYERS=Cartografia&
SRS=EPSG:23030&
BBOX=728782,4373540,729464,4373970&
WIDTH=1024&HEIGHT=645&
FORMAT=image/png&
STYLES=Default&
TRANSPARENT=TRUE&
x=462&y=215&
FEATURE COUNT=10000&
```

Respuesta del GetFeatureInfo

El servidor devuelve una respuesta en el formato especificado en el INFO_FORMAT con información sobre la característica o características situadas en el punto (i, j) si la petición es correcta, y un mensaje de excepción en caso de que no lo sea.

Un ejemplo de respuesta de una petición GetFeaturenfo puede ser:

```
<ServiceExceptionReport version="1.1.1">
<ServiceException code="InvalidFormat">
Error de programa en: Viendo otros parámetros
</ServiceException>
</ServiceExceptionReport>
```

8.2 Servicio WFS

Antes de describir qué es y cómo funciona un servidor web de características, vamos a definir qué es una característica y un tipo de característica, ya que a lo largo de todo este apartado y en adelante hacemos referencia a ellas.

Entendemos por característica un objeto que es una abstracción de un fenómeno del mundo real. Este objeto tiene una serie de propiedades asociadas: un nombre, un tipo y un valor. Un ejemplo puede ser un municipio, que tiene un nombre, un tipo (polygon) y un número de habitantes, comarca y provincia. Normalmente, estas características están almacenadas en una base de datos espacial, en un shapefile o en otros formatos.

Un tipo de característica es un tipo concreto, single, de característica de GML. Normalmente se nombra con una etiqueta “prefijo:nombre” para identificarlo de forma unívoca en relación con otros nombres de tipos de característica. A menudo podemos considerar como sinónimos un tipo de característica y una capa o un dataset. Los dataset, por su parte, pueden contener múltiples tipos de características. Por ejemplo, el dataset “Ciudad” puede contener los tipos de características “vcia:Carreteras” y “vcia:Nucleos”.

Una vez se ha descrito qué es una característica y qué es un tipo de característica, pasamos a describir el servicio de características. Web Feature Service (servicio web de características), en adelante WFS, define la interfaz para el acceso y las operaciones de manipulación de características de la información geográfica. De esta manera, en vez de tener acceso a un mapa o cartoimagen, como en WMS, tenemos acceso a la información gráfica y alfanumérica de un tema, que es servido codificado en formato GML. Esta información geográfica la podemos combinar, usar y administrar, aunque esté alojada en uno o varios servidores de características.

Con el servicio WFS, dependiendo de la petición que hagamos, podemos llevar a cabo las operaciones siguientes:

- Crear una nueva característica
- Borrar una característica,
- Actualizar una característica,
- Fijar una característica u
- Obtener o preguntar características basadas en limitaciones espaciales o no espaciales.

Son requisitos para el funcionamiento del servicio de características [WFS_sp] los siguientes:

1. Las interfaces deben estar definidas en XML.
2. GML es el formato que hay que usar para expresar características dentro de la interfaz.
3. Como mínimo WFS debe poder representar características usando GML.
4. El lenguaje de filtrado debe ser definido en XML y puede ser derivado de CQL, como se define en la especificación de implementación de la interfaz de catálogo OpenGIS® (OpenGIS® Catalogue Interface Implementation Specification).

5. El almacén de datos utilizado para almacenar las características de la información geográfica debe ser opaco para las aplicaciones cliente y sólo se deben ver los datos a través de la interfaz WFS.
6. Para referenciar las propiedades hay que utilizar un subconjunto de expresiones de XPath.

El estado de una característica geográfica se describe mediante una serie de propiedades, donde cada propiedad puede ser descrita como {nombre, tipo, valor}. El nombre y el tipo de cada propiedad de una característica están determinados por su propia definición de tipo. Las características geográficas son aquellas que pueden tener al menos una propiedad que tiene valor geométrico. Esto implica que las características también pueden tener definidas propiedades no geométricas y que se puede filtrar por cualquiera de ellas.

El proceso de petición y respuesta para el servicio de características puede ser:

1. Una aplicación cliente realiza una petición GetCapabilities a un WFS y si la petición es correcta le devuelve un documento que contiene una descripción de todas las operaciones que soporta WFS y la lista de todos los tipos de características que puede servir.
2. Además, una aplicación cliente opcionalmente puede hacer una petición a un WFS para la definición de un tipo o más de características o elementos que WFS puede servir.
3. Basado en la definición del tipo o tipos de características, la aplicación cliente genera una petición de características, como se especifica más adelante.
4. La petición se envía al servidor web.
5. El WFS es invocado para leer y servir la petición.
6. Cuando el WFS ha completado el proceso de la petición, genera un documento de estado (status report) y lo manda de vuelta al cliente. En el supuesto de que ocurra un error, el documento de estado indica el fallo.

Las operaciones que soporta WFS son las siguientes:

Figura 11: Operaciones y respuestas WFS

Y como vemos en el siguiente cuadro, hay distintos tipos de servicios WFS, en función de las operaciones que tengan disponibles.

Figura 12: Operaciones WFS

Así, vemos que hay tres clases de servidores de características, según las operaciones que permitan realizar: WFS básico, WFS XLinks, WFS transaccional.

Operaciones Web Feature Service En este apartado describimos las características de cada una de las operaciones que soporta WFS.

GetCapabilities

Con esta operación se pretende que el servicio tenga la habilidad de describir sus capacidades devolviendo un fichero con sus metadatos en respuesta a una petición GetCapabilities. Por ello ésta suele ser la primera petición que se realiza al servicio, ya que es aquí donde se describe qué operaciones son soportadas y de las que los son, dónde hay que mandarlas y qué tipos de características están disponibles en este servicio.

Parámetros de petición

Una petición GetCapabilities usando HTTP GET es la siguiente, donde solamente el parámetro REQUEST es necesario.

```
http://localhost:8080/geoserver/wfs?
REQUEST=GetCapabilities&
SERVICE=WFS&
VERSION=1.0.0&
EXCEPTIONS=XML
```

Una petición GetCapabilities usando HTTP POST es:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<GetCapabilities
service="WFS"
version="1.1.0"
/>
```

Respuesta

La respuesta a una petición GetCapabilities de un WFS describe:

- Qué operaciones soporta el servidor,
- Dónde hay que enviar las peticiones HTTP GET y POST,
- Un listado con todos los tipos de características a las que el servidor tiene acceso, como:
  - Qué operaciones (query, insert, update, delete, lock) están disponibles para cada tipo de característica,
  - El nombre y namespace del tipo de característica,
  - En qué proyección/sistema de referencia están los datos,
  - La extensión espacial del tipo de característica
  - Qué tipo de funcionalidad de preguntas soporta el servidor.

Así pues, podemos dividir el documento de capacidades devuelto en las siguientes secciones:

1. Identificador del servicio. Provee información sobre el mismo servicio WFS.
2. Proveedor del servicio. Provee metadatos sobre la organización que gestiona el WFS.
3. Operación de metadatos. Provee metadatos sobre las operaciones definidas en la especificación e implementadas para el WFS sobre el que se hace la petición.
4. Lista FeatureType. Define la lista de tipos de características y operaciones para cada una de estas características, que están disponibles para el WFS. También se encuentra en esta sección información adicional como el SRS por defecto, otros SRS posibles u otro tipo de información.
5. Lista ServesGMLObjectType. Define una lista de tipos de objetos GML, no derivados de gml:abstractFeatureType, que están disponibles para el servidor WFS que soporta la operación GetGMLObject. Estos tipos pueden estar definidos en un schema GML base, o en una aplicación schema usando su propio namespace.
6. Lista SuppportsGMLObjectType.
7. Filtro de capacidades: es una sección opcional definida en la especificación de Filter Enconding. Si esta sección existe, WFS soporta las operaciones enumeradas. Si esta sección no está definida, el usuario debe tener en cuenta que WFS sólo soportará una serie mínima de operaciones de filtrado por defecto, como se define en la especificación de implementación de Filter Encoding.

DescribeFeatureType

Esta operación genera un esquema de descripción de tipos de características servidas por el WFS. La descripción del esquema define cómo deben ser codificadas las instancias de características en la entrada y cómo serán generadas las instancias de características de salida para el WFS implementado.

Parámetros de petición

Se realiza una petición DescribeFeatureType cuando se quiere conocer información detallada sobre el esquema de un tipo de características.

Usando el método GET:

```
http://localhost:8080/geoserver/wfs?
REQUEST=DescribeFeatureType&
SERVICE=WFS&
VERSION=1.1.0&
TYPENAME=topp:cntry00
```

Usando el método POST:

```
<?xml version="1.0"?>
<wfs:DescribeFeatureType
version="1.0.0"
service="WFS"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs../wfs/1.0.0/WFS-basic.xsd">
<wfs:TypeName>topp:cntry00</wfs:TypeName>
/>
```

Respuesta

La respuesta a una petición DescribeFeatureType indica las propiedades de nombre, tipos y restricciones.

GetFeature

Esta operación permite la petición y el servicio de instancias de características. Por lo tanto, la respuesta a una petición GetFeature de un WFS es un documento GML que contiene una instancia de las características seleccionadas en la petición. Por lo tanto, es conveniente conocer la especificación de GML para poder interpretar la respuesta. Además, el cliente puede especificar los fenómenos y restringir la petición mediante variables espaciales y no espaciales.

Parámetros de petición

Un ejemplo de petición usando el método GET es:

```
http://localhost:8080/geoserver/wfs?
REQUEST=GetFeature&
```

```
SERVICE=WFS&
TYPENAME=tfc:ejes_lin&
PROPERTYNAME=the_geom,NOMBRE,TITULAR,TIPO_VIA,LEYENDA&
VERSION=1.0.0&
EXCEPTIONS=XML&
MAXFEATURES 1
```

Un ejemplo de petición usando el método POST puede ser:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<wfs:GetFeature outputFormat="GML3"
xmlns:gml="http://www.opengis.net/gml"
xmlns:wfs="http://www.opengis.net/wfs"
xmlns:ogc="http://www.opengis.net/ogc">
<wfs:Query typeName="LANL_Demo">
<wfs:PropertyName>content</wfs:PropertyName>
<ogc:Filter>
<ogc:BBOX>
<ogc:PropertyName>content</ogc:PropertyName>
<gml:Box>
<gml:coordinates>-83,25 -80,31</gml:coordinates>
</gml:Box>
</ogc:BBOX>
</ogc:Filter>
</wfs:Query>
</wfs:GetFeature>
/>
```

Respuesta

La respuesta que se obtiene a una petición GetFeature es un documento GML que contiene el tipo de característica sobre la que hemos hecho la petición. El ejemplo que mostramos a continuación es la respuesta obtenida al realizar la petición por el método GET del apartado Petición.

```
<?xml version="1.0" encoding="UTF-8"?>
<wfs:FeatureCollection xmlns:wfs="http://www.opengis.net/wfs"
xmlns:tfc="http://localhost" xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/wfs
http://localhost:8080/geoserver/schemas/wfs/1.0.0/WFS-basic.xsd http://localhost
http://localhost:8080/geoserver/wfs/DescribeFeatureType?typeName=tfc:ejes_lin">
<gml:boundedBy>
<gml:Box srsName="http://www.opengis.net/gml/srs/epsg.xml#23030">
<gml:coordinates decimal="." cs="," ts=" ">605339.80017097,4185196.02974009
797798.01056147,4523774.64445905</gml:coordinates>
</gml:Box>
</gml:boundedBy>
<gml:featureMember>
<tfc:ejes_lin fid="ejes_lin.1">
<tfc:the_geom>
<gml:MultiLineString srsName="http://www.opengis.net/gml/srs/epsg.xml#23030">
<gml:lineStringMember>
```

```
<gml:LineString>
<gml:coordinates decimal="." cs="," ts=" ">718837.66,4377134.34
718830.54,4377142.26 718812.19,4377162.11
718778.41,4377195.8 718750.48,4377222.9 718728.01,4377244.82
718689.57,4377282.43 718656.73,4377315.44
718620.8,4377351.55</gml:coordinates>
</gml:LineString>
</gml:lineStringMember>
</gml:MultiLineString>
</tfc:the_geom>
<tfc:NOMBRE>CV-368</tfc:NOMBRE>
<tfc:TITULAR>30</tfc:TITULAR>
<tfc:TIPO_VIA>J</tfc:TIPO_VIA>
<tfc:LEYENDA>J30</tfc:LEYENDA>
</tfc:ejes_lin>
</gml:featureMember>
</wfs:FeatureCollection>
```

A continuación este fichero GML se puede representar utilizando una herramienta capaz de representar información geográfica en formato GML.

GetGmlObject

La operación GetGmlObject permite recuperar características y elementos mediante su identificador de un WFS. La petición GetGmlObject es procesada por el servidor de características y devuelve al cliente un fragmento de un documento XML en el que se encuentra el resultado. Esta operación es opcional y si el servidor la soporta debe indicarse en el documento de capacidades.

Parámetros de petición

Una petición GetGmlObject está definida por el siguiente esquema XML:

```
<xsd:element name="GetGmlObject" type="wfs:GetGmlObjectType"/>
<xsd:complexType name="GetGmlObjectType">
<xsd:complexContent>
<xsd:extension base="wfs:BaseRequestType">
<xsd:sequence>
<xsd:element ref="ogc:GmlObjectId"/>
</xsd:sequence>
<xsd:attribute name="outputFormat" type="xsd:string" use="optional" default="GML3"/>
<xsd:attribute name="traverseXlinkDepth" type="xsd:string" use="required"/>
<xsd:attribute name="traverseXlinkExpiry" type="xsd:positiveInteger" use="optional"/>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

Respuesta

La respuesta a una petición GetGmlObject es el elemento indicado del documento GML del servidor devuelto como un fragmento de un documento XML. La diferencia entre esta respuesta y la de la operación GetFeature es que ésta devuelve un documento

completo que contiene una wfs:FeatureCollection, mientras que la respuesta del GetGmlObject es un fragmento de ese documento en el que se muestra sólo el elemento o el objeto indicado mediante su ID.

Transaction

Un WFS puede soportar peticiones transaccionales. Esta petición está compuesta por operaciones que modifican las características, de modo que pueden crear, modificar o borrar una característica geográfica.

Es una operación opcional y, si se implementa, se debe indicar en el documento de capacidades.

Parámetros de petición

El código para una petición de transacción está definido por el siguiente fragmento de esquema XML:

```
<xsd:element name="Transaction" type="wfs:TransactionType"/>
<xsd:complexType name="TransactionType">
<xsd:complexContent>
<xsd:extension base="ows:GetCapabilitiesType">
<xsd:sequence>
<xsd:element ref="wfs:LockId" minOccurs="0"/>
<xsd:choice minOccurs="0" maxOccurs="unbounded">
<xsd:element ref="wfs:Insert"/>
<xsd:element ref="wfs:Update"/>
<xsd:element ref="wfs:Delete"/>
<xsd:element ref="wfs:Native"/>
</xsd:choice>
</xsd:sequence>
<xsd:attribute name="releaseAction" type="wfs:AllSomeType" use="optional"/>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
<xsd:element name="LockId" type="xsd:string"/>
<xsd:element name="Insert" type="wfs:InsertElementType"/>
<xsd:complexType name="InsertElementType">
<xsd:choice>
<xsd:element ref="gml:_FeatureCollection" />
<xsd:sequence>
<xsd:element ref="gml:_Feature" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:choice>
<xsd:attribute name="idgen" type="wfs:IdentifierGenerationOptionType" use="optional"
default="GenerateNew"/>
<xsd:attribute name="handle" type="xsd:string" use="optional"/>
<xsd:attribute name="inputFormat" type="xsd:string" use="optional" default="text/xml;
subversion=gml/3.1.1"/>
<xsd:attribute name="srsName" type="xsd:anyURI" use="optional"/>
</xsd:complexType>
<xsd:simpleType name="IdentifierGenerationOptionType">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="UseExisting"/>
<xsd:enumeration value="ReplaceDuplicate"/>
```

```
<xsd:enumeration value="GenerateNew"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:element name="Update" type="wfs:UpdateElementType"/>
<xsd:complexType name="UpdateElementType">
<xsd:sequence>
<xsd:element ref="wfs:Property" maxOccurs="unbounded"/>
<xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
</xsd:sequence>
<xsd:attribute name="handle" type="xsd:string" use="optional"/>
<xsd:attribute name="typeName" type="xsd:QName" use="required"/>
<xsd:attribute name="inputFormat" type="xsd:string" use="optional" default="text/xml;
subversion=gml/3.1.1"/>
<xsd:attribute name="srsName" type="xsd:anyURI" use="optional"/>
</xsd:complexType>
<xsd:element name="Property" type="wfs:PropertyType"/>
<xsd:complexType name="PropertyType">
<xsd:sequence>
<xsd:element name="Name" type="xsd:QName"/>
<xsd:element name="Value" minOccurs="0"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="Delete" type="wfs:DeleteElementType"/>
<xsd:complexType name="DeleteElementType">
<xsd:sequence>
<xsd:element ref="ogc:Filter" minOccurs="1" maxOccurs="1"/>
</xsd:sequence>
<xsd:attribute name="handle" type="xsd:string" use="optional"/>
<xsd:attribute name="typeName" type="xsd:QName" use="required"/>
</xsd:complexType>
<xsd:element name="Native" type="wfs:NativeType"/>
<xsd:complexType name="NativeType">
<xsd:attribute name="vendorId" type="xsd:string" use="required"/>
<xsd:attribute name="safeToIgnore" type="xsd:boolean" use="required"/>
</xsd:complexType>
```

Un elemento <Transaction> puede contener cero o más elementos <Insert>, <Update> o <Delete> que describen las operaciones para crear, modificar o borrar instancias de características. El WFS debe procesar los elementos <Insert>, <Update> o <Delete> en el orden en el que aparecen en la petición de transacción.

Respuesta

La respuesta a una petición Transaction es un documento XML en el que se indica el fin de la transacción. Además, si la transacción incluye la operación <Insert>, el WFS debe devolver los identificadores de las características de las nuevas características creadas.

LockFeature

Un WFS puede procesar una petición LockFeature para una o más instancias de un tipo de característica en el momento en el que se está produciendo una operación de transacción, con el fin de evitar que dicha instancia sea editable en el proceso de

modificación. Con esto se pretende proteger ciertas características de ser modificadas mientras se realiza una operación Transaction, y asegurar así su consistencia.

Es una operación opcional, por lo que no debe ser siempre implementada, pero sí lo es, debe indicarse en el documento de capacidades.

Parámetros de petición

El siguiente fragmento de un esquema XML define la petición LockFeature:

```
<xsd:element name="LockFeature" type="wfs:LockFeatureType"/>
<xsd:complexType name="LockFeatureType">
<xsd:complexContent>
<xsd:extension base="wfs:BaseRequestType">
<xsd:sequence>
<xsd:element name="Lock" type="wfs:LockType" maxOccurs="unbounded"/>
</xsd:sequence>
<xsd:attribute name="expiry" type="xsd:positiveInteger" use="optional" default="5"/>
<xsd:attribute name="lockAction" type="wfs:AllSomeType" use="optional"/>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="AllSomeType">
<xsd:restriction base="xsd:string">
<xsd:enumeration value="ALL"/>
<xsd:enumeration value="SOME"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="LockType">
<xsd:sequence>
<xsd:element ref="ogc:Filter" minOccurs="0" maxOccurs="1"/>
</xsd:sequence>
<xsd:attribute name="handle" type="xsd:string" use="optional"/>
<xsd:attribute name="typeName" type="xsd:QName" use="required"/>
</xsd:complexType>
```

El atributo expiry indica el límite de tiempo que está bloqueada una instancia de una característica. Se especifica en minutos y el tiempo empieza a correr una vez la petición LockFeature ha sido procesada y la respuesta ha sido transmitida por completo al cliente.

Respuesta

La respuesta a una petición LockFeature viene definida por el siguiente fragmento de esquema XML:

```
<xsd:element name="LockFeatureResponse" type="wfs:LockFeatureResponseType"/>
<xsd:complexType name="LockFeatureResponseType">
<xsd:sequence>
<xsd:element ref="wfs:LockId"/>
<xsd:element name="FeaturesLocked" type="wfs:FeaturesLockedType" minOccurs="0"/>
<xsd:element name="FeaturesNotLocked" type="wfs:FeaturesNotLockedType"
minOccurs="0"/>
</xsd:sequence>
```

```
</xsd:complexType>
<xsd:complexType name="FeaturesLockedType">
<xsd:sequence maxOccurs="unbounded">
<xsd:element ref="ogc:FeatureId"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeaturesNotLockedType">
<xsd:sequence maxOccurs="unbounded">
<xsd:element ref="ogc:FeatureId"/>
</xsd:sequence>
</xsd:complexType>
```

9 Clientes de consulta

9.1 Clientes de escritorio

9.1.1 gvSIG

gvSIG 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 la primera aplicación que se desarrolló dentro del proyecto gvSIG, por lo que se conoce también como gvSIG.

gvSIG Desktop

gvSIG Desktop es un programa informático para el manejo de información geográfica con precisión cartográfica que se distribuye bajo licencia GNU GPL v2. Permite acceder a información vectorial y rasterizada así como a servidores de mapas que cumplan las especificaciones del OGC. Esta es una de las principales características de gvSIG respecto a otros Sistema de Información Geográfica, la importante implementación de servicios OGC: WMS (Web Map Service), WFS (Web Feature Service), WCS (Web Coverage Service), Servicio de Catálogo y Servicio de Nomenclátor.

Está desarrollado en lenguaje de programación Java, funcionando con los sistemas operativos Microsoft Windows, Linux y Mac OS X, y utiliza bibliotecas estándar de GIS reconocidas, como Geotools o Java Topology Suite (JTS). Asimismo, gvSIG posee un lenguaje de scripting basado en Jython y también se pueden crear extensiones en Java utilizando las clases de gvSIG.

Entre los formatos gráficos de fichero más habituales cuenta entre otros con acceso a formatos vectoriales GML, SHP, DXF, DWG, DGN, KML y formatos de imagen rasterizada como MrSID, GeoTIFF, ENVI o ECW.

Iniciado en el año 2004, es un proyecto de desarrollo informático impulsado inicialmente por la Conselleria de Infraestructuras y Transportes de la Generalidad Valenciana y la Unión Europea mediante el Fondo Europeo de Desarrollo Regional (FEDER). Actualmente está impulsado por un conjunto de entidades (empresas, administraciones, universidades) englobadas bajo la Asociación gvSIG (Iver, Prodevelop, Software Colaborativo, Creativa, Conselleria de Infraestructuras y Transportes).

Funcionalidades gvSIG Desktop

En gvSIG Desktop encontramos las herramientas propias de un completo cliente SIG de escritorio, entre otras:

- Acceso a formatos vectoriales: SHP, GML, KML, DXF, DWG, DGN.
- Acceso a formatos ráster: BMP, GIF, TIF, TIFF, JPG, JPEG, PNG, VRT, DAT de ENVI, ERDAS (LAN, GIS, IMG), PCI Geomatics (PIX, AUX), ADF de ESRI, ILWIS (MPR, MPL), MAP de PC Ráster, ASC, PGM, PPM, RST de IDRISI, RMF, NOS, KAP, HDR, RAW.

- Acceso a servicios remotos: OGC (WMS, WFS, WCS, WFS-T, WPS), ArcIMS, Ecwp.
- Acceso a bases de datos y tablas: PostGIS, MySQL, ArcSDE, Oracle, JDBC, CSV.
- Navegación: zooms, desplazamiento, gestión de encuadres, localizador.
- Consulta: información, medir distancias, medir áreas, hiperenlace.
- Selección: por punto, por rectángulo, por polígono, por capa, por atributos, invertir selección, borrar selección.
- Búsqueda: por atributo, por coordenadas.
- Geoprocesos: área de influencia, recortar, disolver, juntar, envolvente convexa, intersección, diferencia, unión, enlace espacial, translación 2D, reproyección (sólo vectorial), geoprocesos Sextante.
- Edición gráfica: añadir capa de eventos, snapping, rejilla, flatness, pila de comandos, deshacer/rehacer, copiar, simetría, rotar, escalar, desplazar, editar vértice, polígono interno, matriz, explotar, unir, partir, autocompletar polígono, insertar punto, multipunto, línea, arco, poli línea, polígono, rectángulo, cuadrado, círculo, elipse.
- Edición alfanumérica: modificar estructura tabla, editar registros, calculadora de campos.
- Servicio de catálogo y nomenclátor.
- Representación vectorial: símbolo único, cantidades (densidad de puntos, intervalos, símbolos graduados, símbolos proporcionales), categorías (expresiones, valores únicos), múltiples atributos, guardar/recuperar leyenda, editor de símbolos, niveles de simbología, bibliotecas de símbolos.
- Representación ráster: brillo, contraste, realce, transparencia por píxel, opacidad, tablas de color, gradientes.
- Etiquetado: etiquetado estático, etiquetado avanzado, etiquetado individual.
- Tablas: estadísticas, filtros, orden ascendente/descendente, enlazar, unir, mover selección, exportar, importar campos, codificación, normalización.
- Constructor de mapas: composición de página, inserción de elementos cartográficos (Vista, leyenda, escala, símbolo de norte, cajetín, imagen, texto, gráfico), herramientas de maquetación (alinear, agrupar/desagrupar, ordenar, enmarcar, tamaño y posición), grid, plantillas.
- Impresión: impresión, exportación a PDF, a Postcript, a formato de imagen.
- Redes: topología de red, gestor de paradas, costes de giro, camino mínimo, conectividad, árbol de recubrimiento mínimo, matriz orígenes-destinos, evento más cercano, área de servicio.
- Ráster y teledetección: estadísticas, filtrado, histograma, rango de escalas, realce, salvar a ráster, vectorización, regiones de interés, componentes generales, georreferenciación, geolocalización, clasificación supervisada, cálculo de bandas, perfiles de imagen, árboles de decisión, componentes principales, tasselep cap, fusión de imágenes, diagramas de dispersión, mosaicos.
- Publicación: WMS, WFS, WCS de MapServer, WFS de Geoserver.
- 3D y animación: Vista 3D plana y esférica, capas 3D, simbología 3D, extrusión, edición de objetos 3D, encuadres 3D, animación 2D y 3D, visualización estéreo (anaglifo, horizontal split).

- Topología: construcción topológica, edición topológica, generalizar, suavizar, invertir sentido de líneas, convertir capa de líneas/polígonos a puntos, convertir capa de polígonos a líneas, triangulación de Delaunay/Poligonación de Thiessen, build, clean, correcciones topológicas en modo Batch.
- Otros: gestión de Sistemas de Referencia Coordenados, exportar/importar WMC, scripting, gestión de traducciones.

9.1.2 Udig

uDig es una construcción de la empresa Refractions Research La licencia de uDig es GNU LGPL, a esta misma empresa se le debe el desarrollo de PostGIS y aportes significativos a Geoserver.

Características

Es un desarrollo rompiendo muchos convencionalismos de las herramientas tradicionales, con un rostro parecido a qGIS. Entre sus características se puede mencionar:

- Construido en Java, bajo el entorno Eclipse al igual que gvSIG.
- La libertad de configurar la interface es envidiable, pudiéndose arrastrar ventanas casi a cualquier lugar, ejecutarlas en segundo plano, arrastre externo e interno, minimizarlas a botones y modificando libremente los bordes de frames.
- La velocidad de ejecución es muy buena a pesar de ser sobre Java. Corre sobre Linux y Mac, obviamente con mejor rendimiento.
- En cuanto a lectura de formatos vectoriales, es limitado con archivos discretos (no lee dgn, kml, dxf, o dwg) pero sí los afamados (gml, xml). El único tradicional que lee es el shape file.
- Con imágenes ráster también tiene sus limitantes, pero se pueden agregar servicios wms y otros servicios en línea.
- En cuanto a bases de datos sí es robusto, ArcSDE, DB2, MySQL, Oracle Spatial, PostgreSQL/PostGIS y WFS, así que por medio de algunos de estos puede integrar los datos vectoriales a los que no accede de forma convencional.
- La rejilla, barra de escala y leyenda se integran como si fueran capas. Esto es interesante pues no son funcionalidades de la interface de despliegue sino de los datos. Aunque su configuración parece algo complicada a primera impresión.
- Tiene características que lo hacen práctico, como:
  - Copy/paste de features (como Manifold GIS).
  - Pegar como xml en notepad.
  - Simbología temática muy práctica, con alertas para evitar problemas con daltónicos, monitores CRT, proyectores, monitores LCD, impresión a color y fotocopiadora.
- Es interesante que la herramienta normalmente viene con un ejemplo estructurado, que incluye una comunidad de Canadá y una base mundial de ciudades, países, zonas horarias e imágenes de satélite. Esta estrategia es

muy buena para entender su capacidad al momento de verlo por primera vez.

- La búsqueda de actualizaciones en línea es otra característica práctica que otros proyectos deberían considerar. En esto, un tanto similar a gvSIG, existe una barrera en la primera impresión, y es que la riqueza que está en las extensiones no tiene suficiente marketing o le falta un hilo conductor que promueva su utilidad (y en ese caso oficialidad). Al menos, con esta actualización en línea, tras unos minutos de descarga puedo ver muchas capacidades que recibe en las extensiones Grass, JGrass, SEXTANTE, Horton Machine y de Axios en aplicaciones hidrológicas, modelos 3D, interacción GPS, ráster y vector.

Desventajas

uDig hace cosas interesantes, al igual que qGIS se complementa con JGrass, pero como solución GIS no es la mejor herramienta open source, en cuanto a funcionalidades de construcción vectorial y manejo de topologías le superan qGIS (con las extensiones que trae) y gvSIG (sin extensiones). Si bien es madura, y tiene lo que un usuario común podría requerir, su potencial es para el usuario con capacidades de desarrollo en Java; su enfoque Internet GIS tiene sentido para conectarse a datos y búsqueda de actualizaciones pero en cuanto a publicación tiene poco que ofrecer.

Lee pocos formatos CAD/GIS, no ha logrado integrar la comunidad al nivel que gvSIG, y, en esto la demanda de usuarios y alianzas estratégicas son un motor importante para agilizar el desarrollo, aspecto que no parece estar logrando en el ímpetu de gvSIG.

El hecho de romper esquemas hace que muchos usuarios se descontrolen al inicio en el manejo de proyectos, catálogos y perspectivas. En cambio es destacable la sencillez que tiene una vez que se conoce su forma de operar, aunque en la filosofía de escalabilidad Java hay que cuidar el equilibrio para que la versión básica tenga más extensiones.

9.2 Clientes WEB

9.2.1 Tecnología a emplear

9.2.1.1 HTML

HTML, siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

9.2.1.1.1 Historia de HTML

9.2.1.1.1.1 Primeras especificaciones

La primera descripción de HTML disponible públicamente fue un documento llamado HTML Tags (Etiquetas HTML), publicado por primera vez en Internet por Tim Berners-Lee en 1991. Describe 22 elementos comprendiendo el diseño inicial y relativamente simple de HTML. Trece de estos elementos todavía existen en HTML 4.

Berners-Lee consideraba a HTML una ampliación de SGML, pero no fue formalmente reconocida como tal hasta la publicación de mediados de 1993, por la IETF, de una primera proposición para una especificación de HTML: el boceto Hypertext Markup Language de Berners-Lee y Dan Connolly, el cual incluía una Definición de Tipo de Documento SGML para definir la gramática. El boceto expiró luego de seis meses, pero fue notable por su reconocimiento de la etiqueta propia del navegador Mosaic usada para insertar imágenes sin cambio de línea, reflejando la filosofía del IETF de basar estándares en prototipos con éxito. Similarmente, el boceto competidor de Dave Raggett HTML+ (Hypertext Markup Format) (Formato de marcaje de hipertexto), de 1993 tardío, sugería, estandarizar características ya implementadas tales como tablas.

9.2.1.1.2 Marcado HTML

HTML consta de varios componentes vitales, incluyendo elementos y sus atributos, tipos de data, y la declaración de tipo de documento.

9.2.1.1.2.1 Elementos

Los elementos son la estructura básica de HTML. Los elementos tienen dos propiedades básicas: atributos y contenido. Cada atributo y contenido tiene ciertas restricciones para que se considere válido al documento HTML. Un elemento generalmente tiene una etiqueta de inicio (p.ej. <nombre-de-elemento>) y una etiqueta de cierre (p.ej. </nombre-de-elemento>). Los atributos del elemento están contenidos en la etiqueta de inicio y el contenido está ubicado entre las dos etiquetas (p.ej. <nombre-de-elemento atributo="valor">Contenido</nombre-de-elemento>). Algunos elementos, tales como <br>, no tienen contenido ni llevan una etiqueta de cierre. Debajo se listan varios tipos de elementos de marcado usados en HTML.

9.2.1.1.2.2 Estructura general de código en el lenguaje de etiquetas HTML.

El marcado estructural describe el propósito del texto. Por ejemplo, <h2>Golf</h2> establece a "Golf" como un encabezamiento de segundo nivel, el cual se mostraría en un navegador de una manera similar al título "Marcado HTML" al principio de esta sección. El marcado estructural no define cómo se verá el elemento, pero la mayoría de los navegadores web han estandarizado el formato de los elementos. Un formato específico puede ser aplicado al texto por medio de hojas de estilo en cascada.

El marcado de la presentación describe la apariencia del texto, sin importar su función. Por ejemplo, negrita indica que los navegadores web visuales deben mostrar el texto en negrita, pero no indica qué deben hacer los navegadores web que muestran el contenido de otra manera (por ejemplo, los que leen el texto en voz alta). En el caso de negrita e itálica, existen elementos que se ven de la misma manera pero tienen una naturaleza más semántica: énfasis fuerte y énfasis. Es fácil ver cómo un lector de pantalla debería interpretar estos dos elementos. Sin embargo, son equivalentes a sus correspondientes elementos de la presentación: un lector de pantalla no debería decir más fuerte el nombre de un libro,

aunque éste esté en itálicas en una pantalla. La mayoría del marcado de presentación ha sido desechado con HTML 4.0, en favor de Hojas de estilo en cascada.

El marcado hipertextual se utiliza para enlazar partes del documento con otros documentos o con otras partes del mismo documento. Para crear un enlace es necesario utilizar la etiqueta de ancla <a> junto con el atributo href, que establecerá la dirección URL a la que apunta el enlace. Por ejemplo, un enlace a la Wikipedia sería de la forma Wikipedia. También se pueden crear enlaces sobre otros objetos, tales como imágenes .

9.2.1.1.2.3 Atributos

La mayoría de los atributos de un elemento son pares nombre-valor, separados por un signo de igual "=" y escritos en la etiqueta de comienzo de un elemento, después del nombre de éste. El valor puede estar rodeado por comillas dobles o simples, aunque ciertos tipos de valores pueden estar sin comillas en HTML (pero no en XHTML). De todas maneras, dejar los valores sin comillas es considerado poco seguro. En contraste con los pares nombre-elemento, hay algunos atributos que afectan al elemento simplemente por su presencia (tal como el atributo” ismap” para el elemento “img”).

Figura 13: Atributos HTML

9.2.1.1.3 Códigos HTML básicos

- <html>: define el inicio del documento HTML, le indica al navegador que lo que viene a continuación debe ser interpretado como código HTML. Esto es así de facto, ya que en teoría lo que define el tipo de documento es el DOCTYPE, significando la palabra justo tras DOCTYPE el tag de raíz, por ejemplo:

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

- <script>: incrusta un script en una web, o se llama a uno mediante src="url del script" Se recomienda incluir el tipo MIME en el atributo type, en el caso de JavaScript text/javascript.
- <head>: define la cabecera del documento HTML, esta cabecera suele contener información sobre el documento que no se muestra directamente al usuario. Como por ejemplo el título de la ventana del navegador. Dentro de la cabecera <head> podemos encontrar:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
<html>
    <head>
        <title>Example</title>
        <link href="screen.css" rel="sty
    </head>
    <body>
        <h1>
            <a href="/">Header</a>
        </h1>
        <ul id="nav">
            <li>
                <a href="one/">One</a>
            </li>
            <li>
                <a href="two/">Two</a>
            </li>
```

Figura 14: Ejemplo de cabecera HTML

- <title>: define el título de la página. Por lo general, el título aparece en la barra de título encima de la ventana
- <link>: para vincular el sitio a hojas de estilo o iconos Por ejemplo:<link rel="stylesheet" href="/style.css" type="text/css">
- <style>: para colocar el estilo interno de la página; ya sea usando CSS, u otros lenguajes similares. No es necesario colocarlo si se va a vincular a un archivo externo usando la etiqueta <link>
- <meta>: para metadatos como la autoría o la licencia, incluso para indicar parámetros http (mediante http-equiv="") cuando no se pueden modificar por no estar disponible la configuración o por dificultades con server-side scripting.

- <body>: define el contenido principal o cuerpo del documento. Esta es la parte del documento html que se muestra en el navegador; dentro de esta etiqueta pueden definirse propiedades comunes a toda la página, como color de fondo y márgenes. Dentro del cuerpo <body> podemos encontrar numerosas etiquetas. A continuación se indican algunas a modo de ejemplo:

  - <h1> a <h6>: encabezados o títulos del documento con diferente relevancia.
  - <table>: define una tabla
    - <tr>: fila de una tabla
    - <td>: columna de una tabla
  - <a>: Hipervínculo o enlace, dentro o fuera del sitio web. Debe definirse el parámetro de pasada por medio del atributo href. Por ejemplo: Wikipedia se representa como Wikipedia).
  - : división de la página. Se recomienda, junto con css, en vez de <table> cuando se desea alinear contenido.
  - <img>: imagen. Requiere del atributo src, que indica la ruta en la que se encuentra la imagen. Por ejemplo: . Es conveniente, por accesibilidad, poner un atributo alt="texto alternativo".
  - : Etiquetas para listas.
  - <b>: texto en negrita (Etiqueta desaprobada. Se recomienda usar la etiqueta)
  - <i>: texto en cursiva (Etiqueta desaprobada. Se recomienda usar la etiqueta)

- <s>: texto tachado (Etiqueta desaprobada. Se recomienda usar la etiqueta <del>)
- <u>: texto subrayado

La mayoría de etiquetas deben cerrarse como se abren, pero con una barra ("/") tal como se muestra en los siguientes ejemplos:

- <table><tr><td>Contenido de una celda</td></tr></table>
- <script>Código de un [[script]] integrado en la página</script>.

9.2.1.1.4 Nociones básicas de HTML

El lenguaje HTML puede ser creado y editado con cualquier editor de textos básico, como puede ser Gedit en Linux, el Bloc de Notas de Windows, o cualquier otro editor que admita texto sin formato como GNU Emacs, Microsoft Wordpad, TextPad, Vim, Notepad++, entre otros.

Existen además, otros editores para la realización de sitios Web con características WYSIWYG (What You See Is What You Get, o en español: "lo que ves es lo que obtienes"). Estos editores permiten ver el resultado de lo que se está editando en tiempo real, a medida que se va desarrollando el documento. Ahora bien, esto no significa una manera distinta de realizar sitios web, sino que una forma un tanto más simple ya que estos programas, además de tener la opción de trabajar con la vista preliminar, tiene su propia sección HTML la cual va generando todo el código a medida que se va trabajando. Algunos ejemplos de editores WYSIWIG son Macromedia Dreamweaver, o Microsoft FrontPage.

Combinar estos dos métodos resulta muy interesante, ya que de alguna manera se ayudan entre sí. Por ejemplo; si se edita todo en HTML y de pronto se olvida algún código o etiqueta, simplemente me dirijo al editor visual o WYSIWYG y se continúa ahí la edición, o viceversa, ya que hay casos en que sale más rápido y fácil escribir directamente el código de alguna característica que queramos adherirle al sitio, que buscar la opción en el programa mismo.

Existe otro tipo de editores HTML llamados WYSIWYM (Lo que ves es lo que quieres decir) que dan más importancia al contenido y al significado que a la apariencia visual. Entre los objetivos que tienen estos editores es la separación del contenido y la presentación, fundamental en el diseño Web.

HTML utiliza etiquetas o marcas, que consisten en breves instrucciones de comienzo y final, mediante las cuales, se determina la forma en la que debe aparecer en su navegador el texto, así como también las imágenes y los demás elementos, en la pantalla del ordenador.

Toda etiqueta se identifica porque está encerrada entre los signos menor que y mayor que (<>), y algunas tienen atributos que pueden tomar algún valor. En general las etiquetas se aplicarán de dos formas especiales:

- Se abren y se cierran, como por ejemplo: negrita que se vería en su navegador web como negrita.

- No pueden abrirse y cerrarse, como <hr> que se vería en su navegador web como una línea horizontal.
- Otras que pueden abrirse y cerrarse, como por ejemplo <p>.
- Las etiquetas básicas o mínimas son:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
 <html lang="es">
   <head>
     <title>Ejemplo</title>
   </head>
   <body>
     <p>ejemplo</p>
   </body>
 </html>
```

Seleccionando la opción Ver código fuente en el navegador, se puede ver realmente la información que está recibiendo éste y cómo la está interpretando. Por ejemplo: en Internet Explorer o en Firefox, simplemente hay que desplegar el menú Ver y luego elegir Código fuente. De esta forma, se abrirá el editor de texto configurado como predeterminado en el sistema con el código fuente de la página que se esté viendo en ese momento en el explorador. Otra forma más rápida consiste en hacer clic con el botón derecho del ratón en cualquier punto del área donde el navegador muestra la página web y elegir Ver código fuente.

Para el navegador Firefox existe el plugin FireBug, un depurador que permite entre otras cosas visualizar el código HTML de la página que estamos visualizando de forma dinámica, y que incluso resalta el trozo de código por el que está pasando el ratón en cada momento, por lo que es una herramienta muy útil para aprender diversos conceptos de este lenguaje.

9.2.1.1.5 Historia del estándar

En 1989 existían dos técnicas que permitían vincular documentos electrónicos, por un lado los hipervínculos (links) y por otro lado un poderoso lenguaje de etiquetas denominado SGML. Por entonces, Tim Berners-Lee da a conocer a la prensa que estaba trabajando en un sistema que permitirá acceder a ficheros en línea, funcionando sobre redes de computadoras o máquinas electrónicas basadas en el protocolo TCP/IP.

A principios de 1990, Tim Berners-Lee define por fin el HTML como un subconjunto del conocido SGML y crea algo más valioso aún, el World Wide Web. En 1991, Tim Berners-Lee crea el primer navegador web, que funcionaría en modo texto y sobre un sistema operativo UNIX.

Los trabajos para crear un sucesor del HTML, denominado HTML +, comenzaron a finales de 1993. HTML+ se diseñó originalmente para ser un súper conjunto del HTML que permitiera evolucionar gradualmente desde el formato HTML anterior. A la primera especificación formal de HTML+ se le dio, por lo tanto, el número de versión 2 para distinguirla de las propuestas no oficiales previas. Los trabajos sobre HTML+ continuaron, pero nunca se convirtió en un estándar, a pesar de ser la base formalmente más parecida al aspecto compositivo de las especificaciones actuales.

El borrador del estándar HTML 3.0 fue propuesto por el recién formado W3C en marzo de 1995. Con él se introdujeron muchas nuevas capacidades, tales como facilidades para crear tablas, hacer que el texto fluyese alrededor de las figuras y mostrar elementos matemáticos complejos. Aunque se diseñó para ser compatible con HTML 2.0, era demasiado complejo para ser implementado con la tecnología de la época y, cuando el borrador del estándar expiró en septiembre de 1995, se abandonó debido a la carencia de apoyos de los fabricantes de navegadores web. El HTML 3.1 nunca llegó a ser propuesto oficialmente, y el estándar siguiente fue el HTML 3.2, que abandonaba la mayoría de las nuevas características del HTML 3.0 y, a cambio, adoptaba muchos elementos desarrollados inicialmente por los navegadores web Netscape y Mosaic. La posibilidad de trabajar con fórmulas matemáticas que se había propuesto en el HTML 3.0 pasó a quedar integrada en un estándar distinto llamado MathML.

El HTML 4.0 también adoptó muchos elementos específicos desarrollados inicialmente para un navegador web concreto, pero al mismo tiempo comenzó a limpiar el HTML señalando algunos de ellos como «desaprobados» o deprecated en inglés.

9.2.1.1.6 Accesibilidad Web

El diseño en HTML aparte de cumplir con las especificaciones propias del lenguaje debe respetar unos criterios de accesibilidad web, siguiendo unas pautas, o las normativas y leyes vigentes en los países donde se regule dicho concepto. Se encuentra disponible y desarrollado por el W3C a través de las Pautas de Accesibilidad al Contenido Web 1.0 WCAG (actualizadas recientemente con la especificación 2.0), aunque muchos países tienen especificaciones propias, como es el caso de España con la Norma UNE 139803.

9.2.1.1.7 Entidades HTML

Los caracteres especiales como signo de puntuación, letras con tilde o diéresis, o símbolos del lenguaje; se deben convertir en entidad HTML para mostrarse en un navegador. La siguiente es una lista de caracteres españoles y su correspondiente entidad HTML:

Carácter	Entidad HTML	Carácter	Entidad HTML
á	&aacute;	Á	&Aacute;
é	&eacute;	É	&Eacute;
í	&iacute;	Í	&Iacute;
ó	&oacute;	Ó	&Oacute;
ú	&uacute;	Ú	&Uacute;
ü	&uuml;	Ü	&Uuml;
ñ	&ntilde;	Ñ	&Ntilde;
¡	&iexcl;	¿	&iquest;

Figura 15: Ejemplo de algunos de los caracteres especiales HTML

9.2.1.2 JavaScript

JavaScript es un lenguaje de scripting basado en objetos, no tipeado y liviano, utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas web dinámicas. JavaScript es un dialecto de ECMAScript y se caracteriza por ser un lenguaje basado en prototipos, con entrada dinámica y con funciones de primera clase. JavaScript ha tenido influencia de múltiples lenguajes y se diseñó con una sintaxis similar al lenguaje de programación Java, aunque más fácil de utilizar para personas que no programan.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, la que desarrolló los primeros navegadores web comerciales. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

Tradicionalmente, se venía utilizando en páginas web HTML, para realizar operaciones y en el marco de la aplicación cliente, sin acceso a funciones del servidor. JavaScript se ejecuta en el agente de usuario, al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Inicialmente los autores lo llamaron Mocha y más tarde LiveScript pero fue rebautizado como JavaScript en un anuncio conjunto entre Sun Microsystems y Netscape, el 4 de diciembre de 1995.

En 1997 los autores propusieron JavaScript para que fuera adoptado como estándar de la European Computer Manufacturers 'Association ECMA, que a pesar de su nombre no es europeo sino internacional, con sede en Ginebra. En junio de 1997 fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también como un estándar ISO.

JScript es la implementación de ECMAScript de Microsoft, muy similar al JavaScript de Netscape, pero con ciertas diferencias en el modelo de objetos del navegador que hacen ambas versiones sean incompatibles con frecuencia.

Para evitar estas incompatibilidades, el World Wide Web Consortium diseñó el estándar Document Object Model (DOM, ó Modelo de Objetos del Documento en castellano), que incorporan Konqueror, las versiones 6 de Internet Explorer y Netscape Navigator, Opera la versión 7, y Mozilla Application Suite, Mozilla desde su primera versión.

9.2.1.2.1 Historia y denominación

JavaScript fue desarrollado originalmente por Brendan Eich de Netscape con el nombre de Mocha, el cuál fue renombrado posteriormente a LiveScript, para finalmente quedar como JavaScript. El cambio de nombre coincidió aproximadamente con el momento en que Netscape agregó soporte para la tecnología Java en su navegador web Netscape Navigator en la versión 2.0B3 en diciembre de 1995. La denominación produjo confusión, dando la impresión de que el lenguaje es una prolongación de Java, y se ha caracterizado por muchos como una estrategia de

mercadotecnia de Netscape para obtener prestigio e innovar en lo que eran los nuevos lenguajes de programación web.

Microsoft dio como nombre a su dialecto de JavaScript, JScript, para evitar problemas relacionados con la marca. JScript fue adoptado en la versión 3.0 de Internet Explorer, liberado en agosto de 1996, e incluyo compatibilidad con el Efecto 2000 Con las funciones de fecha, una diferencia de los que se basaban en ese momento. Los dialectos pueden parecer tan similares que los términos "JavaScript" y "JScript" a menudo se utilizan indistintamente, pero la especificación de JScript es incompatible con la de ECMA en muchos aspectos.

Netscape envió JavaScript a ECMA para su estandarización, resultando la versión normalizada llamada ECMAScript.

9.2.1.3 AJAX

Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

9.2.1.3.1 Tecnologías incluidas en Ajax

Ajax es una combinación de cuatro tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
- Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto XMLHttpRequest para intercambiar datos de forma asíncrona con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- XML es el formato usado generalmente para la transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre formateado, texto plano, JSON y hasta EBML.

Como el DHTML, LAMP o SPA, Ajax no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.

9.2.1.3.2 Antecedentes de Ajax

A pesar de que el término «Ajax» fuese creado en 2005, la historia de las tecnologías que permiten Ajax se remonta a una década antes con la iniciativa de Microsoft en el desarrollo de Scripting Remoto. Sin embargo, las técnicas para la carga asíncrona de contenidos en una página existente sin requerir recarga completa remontan al tiempo del elemento iframe (introducido en Internet Explorer 3 en 1996) y el tipo de elemento layer (introducido en Netscape 4 en 1997, abandonado durante las primeras etapas de desarrollo de Mozilla). Ambos tipos de elemento tenían el atributo src que podía tomar cualquier dirección URL externa, y cargando una página que contenga javascript que manipule la página paterna, pueden lograrse efectos parecidos al Ajax.

El Microsoft's Remote Scripting (o MSRS, introducido en 1998) resultó un sustituto más elegante para estas técnicas, con envío de datos a través de un applet Java el cual se puede comunicar con el cliente usando JavaScript. Esta técnica funcionó en ambos navegadores, Internet Explorer versión 4 y Netscape Navigator versión 4. Microsoft la utilizó en el Outlook Web Access provisto con la versión 2000 de Microsoft Exchange Server.

La comunidad de desarrolladores web, primero colaborando por medio del grupo de noticias microsoft.public.scripting.remote y después usando blogs, desarrollaron una gama de técnicas de scripting remoto para conseguir los mismos resultados en diferentes navegadores. Los primeros ejemplos incluyen la biblioteca JSRS en el año 2000, la introducción a la técnica imagen/cookie en el mismo año y la técnica JavaScript bajo demanda (JavaScript on Demand) en 2002. En ese año, se realizó una modificación por parte de la comunidad de usuarios al Microsoft's Remote Scripting para reemplazar el applet Java por XMLHttpRequest.

Frameworks de Scripting Remoto como el ARSCIF aparecieron en 2003 poco antes de que Microsoft introdujera Callbacks en ASP. NET.

Desde que XMLHttpRequest está implementado en la mayoría de los navegadores, raramente se usan técnicas alternativas. Sin embargo, todavía se utilizan donde se requiere una mayor compatibilidad, una reducida implementación, o acceso cruzado entre sitios web. Una alternativa, el Terminal SVG (basado en SVG), emplea una conexión persistente para el intercambio continuo entre el navegador y el servidor.

9.2.1.3.3 Problemas e Inconvenientes

Las páginas con AJAX son más difíciles de desarrollar que las páginas estáticas.

- Las páginas creadas dinámicamente mediante peticiones sucesivas AJAX, no son registradas de forma automática en el historial del navegador, así que haciendo clic en el botón de "volver" del navegador, el usuario no será devuelto a un estado anterior de la página, en cambio puede volver a la última página que visitó. Soluciones incluyen el uso de IFrames invisible para desencadenar cambios en el historial del navegador y el cambio de la porción de anclaje de la dirección (después de un #).

- Los motores de búsquedas no entienden JavaScript. La información en la página dinámica no se almacena en los registros del buscador.
- Hay problemas usando Ajax entre nombres de dominios. Eso es una función de seguridad.
- El sitio con Ajax usa más recursos en el servidor.
- Es posible que páginas con Ajax no puedan funcionar en teléfonos móviles, PDA u otros aparatos. Ajax no es compatible con todas las aplicaciones para ciegos u otras discapacidades.

9.2.1.3.4 Navegadores que permiten Ajax

Ha de tenerse en cuenta que ésta es una lista general, y el soporte de las aplicaciones Ajax dependerá de las características que el navegador permita.

- Navegadores basados en Gecko como Mozilla, Mozilla Firefox, SeaMonkey, Camino, K-Meleon, IceWeasel, Flock, Epiphany, Galeon y Netscape versión 7.1 y superiores
- Google Chrome
- Microsoft Internet Explorer para Windows versión 5.0 y superiores, y los navegadores basados en él
- Navegadores con el API KHTML versión 3.2 y superiores implementado, incluyendo Konqueror versión 3.2 y superiores, Apple Safari versión 1.2 y superiores, y el Web Browser for S60 de Nokia tercera generación y posteriores
- Opera versión 8.0 y superiores, incluyendo Opera Mobile Browser versión 8.0 y superiores.

9.2.1.4 ExtJS

ExtJS es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM.

Originalmente construida como una extensión de la biblioteca YUI, en la actualidad puede usarse como extensión para las bibliotecas jQuery y Prototype. Desde la versión 1.1 puede ejecutarse como una aplicación independiente.

Entre las funcionalidades de ExtJS, destacan un conjunto de componentes (widgets) para incluir dentro de una aplicación web, como:

- Cuadros y áreas de texto.
- Campos para fechas.
- Campos numéricos.
- Combos.
- Radiobuttons y checkboxs.
- Editor HTML.
- Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
- Árbol de datos.
- Pestañas.
- Barra de herramientas.
- Menús al estilo de Windows.
- Paneles divisibles en secciones.
- Sliders.

Varios de estos componentes estás capacitados para comunicarse con el servidor usando AJAX. También contiene numerosas funcionalidades que permiten añadir interactividad a las páginas HTML, como Cuadros de diálogo y quicktips para mostrar mensajes de validación e información sobre campos individuales.

9.2.1.5 OpenLayers

OpenLayers es una librería javascript que nos permite visualizar mapas en cualquier navegador web actual. Además no tiene ninguna dependencia con el servidor web en el que se aloja: al estar todo escrito en javascript sólo se ejecuta en el navegador.

OpenLayers se parece a las librerías de Google Maps o MSN Virtual Earth pero su principal punto fuerte frente a ellos es que es software abierto y utiliza métodos estándares para acceder a los datos cartográficos entre ellos Web Map Service (WMS) i Web Feature Service (WFS). También permite combinar información de distintos servidores, y emplear información en GML para importar datos, localizar elementos, etc.

En los últimos tiempos se han hecho muy populares los que hacen uso de los más populares servicios de mapas de la actualidad -como por ejemplo: Google Maps, Microsoft VE, Yahoo! Maps o Ask.com- para usarlos como herramientas para la representación de información muy variada. Sin embargo, a los ya mencionados, y cuyos derechos pertenecen a las empresas o instituciones que les da nombre, se suma

un nuevo servicio denominado OpenLayers cuyo principal particularidad es ser de código abierto.

En este proyecto se ha empleado la versión 2.13.1 de OpenLayers.

9.2.1.5.1 Contenido del paquete OpenLayers

El paquete de instalación de la librería OpenLayers, contiene el código fuente en JavaScript de la propia librería, y versiones de la librería comprimidos para producción, desarrollo y versión móvil.

La versión de producción y desarrollo de la librería se encuentra en la raíz de este directorio, con las diferentes versiones:

- OpenLayers.js: Versión de producción de la librería que contiene todas las clases comprimidas en un único fichero.
- OpenLayers.debug.js: Versión de desarrollo de la librería que unifica todas la clases en un único fichero.
- OpenLayers.light.js: Incluye un subconjunto de OpenLayers de uso básico de visualización de mapas. Esta versión permite el acceso a OpenStreetMap, Bing, Google, WMS y capas vectoriales, los controles básicos del mapa, y herramientas vectoriales de interacción.
- OpenLayers.light.debug.js: Versión de desarrollo del conjunto básico de clases de OpenLayers.
- OpenLayers.mobile.js: Incluye un subconjunto de los OpenLayers de uso básico para web móviles. Esta versión permite el acceso a OpenStreetMap, Bing, WMS, WFS y capas vectoriales, controles táctiles optimizados, geolocalización; vector de edición y herramientas de interacción.

En cuanto al contenido de las carpetas:

- Build: Contiene herramientas de compresión de la librería en sus diferentes versiones.
- Doc: Contiene documentación de la librería.
- Examples: Contiene ejemplos de utilización de la librería.
- Img: Contiene imágenes empleadas en los ejemplos.
- Theme: Contiene hojas de estilo para aplicar a los ejemplos y desarrollos.
- Tools: Contiene las utilidades de compresión de código JavaScript desarrolladas con el lenguaje Python.
- Lib: Contiene el código fuente de la librería.

9.2.1.5.2 Introducción al API

Para poder utilizar las clases proporcionadas por OpenLayers habrá que incorporar una referencia al script de la librería en la cabecera de nuestro documento HTML:

```
<script src="http://www.openlayers.org/api/OpenLayers.js"></script>
```

En esta introducción al API de OpenLayers desarrollaremos una página HTML que muestre un mapa del mundo obtenido desde el servicio de mapas del Instituto Geográfico Nacional (España).

Antes vamos a introducir algunas de las clases que proporciona la API de OpenLayers.

La clase Map

Todo gira alrededor de la clase 'Map', en realidad 'OpenLayers.Map'. La clase 'Map' encapsula el mapa que se mostrará en pantalla. Para ello habrá que crear una instancia de la clase. A este objeto creado, que llamaremos 'mapa', se le añadirán capas de información, que contendrán la referencia a los datos, y controles que permitirán manipular como se muestra la información.

Al constructor de la clase 'Map' hay que pasarle como primer argumento una referencia al elemento de la página web donde queremos que se muestre el mapa. Lo que se le pasa es el valor del atributo 'id' del elemento . En el código javascript aparecerá:

```
var mapa = new OpenLayers.Map("mapa");
```

Y en el <body> del código HTML aparecerá el elemento :

```
<div id="mapa" style="width: 600px; height: 300px;"></div>
```

Ahora tenemos que añadir al mapa al menos una capa con información. La información cartográfica, los datos, se introducen a través de la clase OpenLayers.Layer. En realidad a través de las clases especializadas derivadas de la clase base OpenLayers.Layer. Hay muchos tipos de capas especializadas para distintos tipos de información. Por ejemplo hay una capa 'Vector' (OpenLayers.Layer.Vector) que permite añadir features vectoriales. Hay capas especializadas para añadir marcadores: OpenLayers.Layer.Marker; Hay capas especializadas para acceder a los servicios de mapas comerciales más conocidos y también hay una capa OpenLayers.Layer.OSM que permite acceder a los mapas de OpenStreetMap (OSM).

La clase Map tiene un atributo 'layers' que guarda las referencias a las capas que se añaden al mapa. Las capas se añaden al mapa a través de sus métodos addLayer() y addLayers().

En el ejemplo que estamos construyendo utilizaremos una capa OpenLayer.Layer.WMS que permite acceder a los Web Map Services. Habrá que pasarle al constructor de la capa el URL del servicio y los parámetros de la petición 'GetMap' que queramos especificar, normalmente al menos el parámetro 'layers' y el parámetro 'projection' que

se corresponde con el parámetro 'srs' de la petición 'GetMap'. En nuestro caso accederemos al Mapa Base del servicio de mapas del IDEE (Infraestructura de Datos Espaciales de España). El URL del servicio es:

```
http://www.idee.es/wms/IDEE-Base/IDEE-Base?
```

El mapa base proporciona una capa denominada "Todas" que da acceso a todas las capas en una. Esta es la capa que utilizaremos en nuestra petición. El primer parámetro pasado al constructor de la capa es un String con un nombre descriptivo para la capa creada. Con todo ello, la sentencia que crea la capa quedará:

```
var capa = new OpenLayers.Layer.WMS( "OpenLayers WMS",
            "http://www.idee.es/wms/IDEE-Base/IDEE-Base",
            {layers: 'Todas', projection: 'EPSG:4230'} );
```

Habrá que añadir la capa creada al mapa mediante:

```
mapa.addLayer(capa);
```

Finalmente utilizaremos el método 'zoomToMaxExtent()' de la clase Map para mostrar el mapa en pantalla:

```
mapa.setZoomToMaxExtent();
```

Veamos ahora el código html completo de la página que estamos desarrollando:

```
<html>
   <head>
      <script src="http://www.openlayers.org/api/OpenLayers.js"></script>
      <script type="text/javascript">
         var mapa, capa;
         function init(){
            mapa = new OpenLayers.Map('mapa');
            capa = new OpenLayers.Layer.WMS( "Mapa Base",
             "http://www.idee.es/wms/IDEE-Base/IDEE-Base",
             {layers: 'Todas', projection: 'EPSG:4230'} );
            mapa.addLayer(capa);
            mapa.zoomToMaxExtent();
          }
      </script>
   </head>
   <body onload="init()">
      <h2>Hola Mundo (OpenLayers)</h2>
      <div id="mapa"  style="width: 600px; height: 300px"></div>
   </body>
</html>
```

Vemos que la llamada a la función init(), que es la que crea el mapa, se hace a partir del evento 'onload' del elemento <body>.

Podemos observar que el mapa por defecto añade controles para zooming y panning en la esquina superior izquierda. Al elemento le podemos dar estilo CSS para posicionarlo, añadir recuadro, etc.

9.2.1.5.3 Tipos básicos de datos

OpenLayers está construida sobre Javascript y por tanto dispone de los mismos tipos de datos.

En Javascript hay cinco tipos de datos primitivos: Undefined, Null, Boolean, Number y String.

Además se dispone de una colección de objetos nativos: Object, Boolean, Error, SintaxError, Function, Number, EvalError, TypeError, Array, Date, RangeError, URIError, String, RegExp, y ReferenceError.

OpenLayers ofrece unas clases con métodos útiles para operar con los tipos de datos básicos:

OpenLayers.String

Esta clase proporciona los siguientes métodos:

- startsWidth(string, substr): Comprueba si la cadena 'string' comienza con la cadena 'substr'. Devuelve un Boolean
- contains(str, substr): Comprueba si la cadena 'substr' está contenida dentro de la cadena 'str'. Devuelve 'TRUE' en caso afirmativo y 'FALSE' en caso negativo.
- trim(str): Devuelve una cadena en la que se han eliminado todos los espacios que pudiera haber al principio o al final de la cadena 'str'
- camelize(str): Camel-Case. Convierte una cadena a base de guiones en el convenio Camelize. Por ejemplo la cadena 'lat-punto' la convierte en latPunto y la cadena -lat-punto en 'LatPunto'. Devuelve una cadena con las modificaciones, sin alterar el original.
- isNumeric(str): Devuelve un Boolean si la cadena corresponde a un número. Reconoce el formato exponencial para números reales. (p. ej isNumeric("2.5e3") devuelve 'TRUE').
- numericIf(str): Si la cadena 'str' es un valor numérico devuelve un Number con ese valor. En otro caso devuelve un String con la misma cadena recibida.

OpenLayers.Number

La clase OpenLayers.Number tiene dos propiedades que se utilizan para dar formato a los números: 'decimalSeparator' y 'thousandsSeparator'.

Además contiene las siguientes funciones para manipular datos tipo Number:

- limSigDigs(float, sig): Redondea el valor del Float 'float' al número de decimales indicado por el Integer 'sig'. Devuelve un Float con el número redondeado.
- format(num, dec, tsep, decsep): Devuelve un String con el Float 'num' redondeado al número de decimales indicados por el Integer 'dec'. Como

separadores de millares y decimales se utilizarán los String 'tsep' y 'decsep' respectivamente.

OpenLayers.Pixel

Representa una posición de pantalla. Tiene dos propiedades: 'x' e 'y'.

El constructor admite como parámetros los valores de las coordenadas x e y:

```
OpenLayers.Pixel( x: Number, y: Number);
```

La clase OpenLayers.Pixel tiene los siguientes métodos de utilidad:

- toString(): Devuelve una cadena de la forma 'x=200.4,y=242.2', supuestas estas las coordenadas del objeto
- clone(): Devuelve un 'clon', una instancia de OpenLayers.Pixel con idénticas x e y que el original.
- equals(px: OpenLayers.Pixel) : Devuelve un Boolean indicando si el Pixel pasado como argumento es equivalente al objeto propietario del método.
- add(dx: Number, dy: Number): Devuelve un OpenLayers.Pixel cuyas componentes son la suma de las originales más los incrementos 'dx' y 'dy'.
- offset(px: OpenLayers.Pixel): Devuelve un OpenLayers.Pixel cuyas coordenadas son la suma del original mas las coordenadas del punto offset

OpenLayers.LonLat

Representa una posición geográfica identificada por sus propiedades longitud, 'lon', y latitud 'lat'. Las dos únicas propiedades específicas de la clase LonLat son 'lon' y 'lat' que son del tipo Number y que se corresponderán con las coordenadas de un punto expresadas en las unidades de la proyección del mapa. Así si el mapa está trabajando en una proyección con coordenadas geográficas, el par representara una longitud y una latitud expresadas en grados sexagesimales. En el caso de que el mapa esté en otro tipo de proyección, la pareja 'lon', 'lat' representará un punto en las coordenadas y unidades de la proyección del mapa. Si nuestro mapa utiliza la proyección Spherical Mercator, la pareja 'lon', 'lat' serán las coordenadas de un punto en dicha proyección y expresadas en metros.

Para crear un objeto LonLat se le pasan al constructor la pareja de datos 'lon' 'lat':

```
var pto = new OpenLayers.LonLat( -3.54, 42.37);
```

Una vez creado un objeto de la clase OpenLayers.LonLat podremos acceder a sus propiedades individuales de forma sencilla:

```
var longitud = pto.lon;
var latitud = pto.lat;
```

La clase LonLat expone los siguientes métodos:

- toString(): Devuelve un String formateado con las coordenadas del punto.
- toShortString(): Devuelve un String formateado con las coordenadas del punto en formato compacto.
- clone(): Devuelve un objeto LonLat con los mismos valores de 'lon' y 'lat'.
- add(inclon, inclat): Devuelve un objeto LonLat cuyas coordenadas son el resultado de sumar ordenadamente a las coordenadas del objeto original los valores pasados en los parámetros 'inclon' y 'inclat'.
- equal(otherLonLat): Este método devuelve un Boolean que indica si el objeto 'otherLonLat' pasado como parámetro tiene los mismos valores de las propiedades 'lon' y 'lat' que el objeto original.
- transform(sourceProy, destProy): En este método los dos parámetros que se pasan como argumentos son dos objetos OpenLayers.Proyection. El método modifica las coordenadas del LonLat original mediante la aplicación del cambio de coordenadas entre la proyección 'sourceProy' y la proyección 'destProy'. El método devuelve una referencia al objeto original, con las coordenadas transformadas. (Este método modifica las coordenadas del objeto LonLat original).
- wrapDateLine(maxExtent):
- fromString (str): Esta función crea un objeto LonLat a partir de un String que tenga las coordenadas separadas por una coma. Por ejemplo : 'var ll = OpenLayers.LonLat.fromString("-3,43")'

OpenLayers.Size

Representa un tamaño en dos dimensiones. Tiene dos propiedades 'w', anchura y 'h', altura.

El constructor admite dos parámetros Number que se corresponden con los valores de la anchura 'w' y la altura 'h' del objeto 'Size' que se desea construir:

```
OpenLayers.Size( w: Number, h: Number)
```

La clase OpenLayers.Size expone los siguientes métodos públicos:

- toString() : Devuelve una Cadena (String) del tipo : 'w=55,h=66'
- clone() : Devuelve un objeto OpenLayers.Size idéntico al propietario del método
- equals(px: OpenLayers.Pixel): Devuelve un Boolean que nos indica si los objetos son equivalentes (misma anchura, misma altura).

OpenLayers.Bounds

Representa una región rectangular, identificada por sus propiedades 'left', 'bottom', 'right' y 'top'.

Dispone de los siguientes métodos:

- toString(): Devuelve el rectángulo expresado como una cadena.
- toArray(reverseAxisOrder): Devuelve un vector con los elementos que componen el rectángulo. Es posible indicar si se tienen que invertir el orden de los elementos.
- toBBOX(decimal,reverseAxisOrder): Devuelve una cadena con el rectángulo expresado con el número de dígitos deseado y si se quiere invertir el orden de los parámetros.
- toGeometry(): Crea un nuevo polígono basado en la geometría del rectángulo.
- getWidth(): Devuelve el ancho del rectángulo.
- getHeight(): Devuelve el alto del rectángulo.
- getSize(): Devuelve el tamaño del rectángulo.
- getCenterPixel(): Devuelve el pixel central del rectángulo.
- getCenterLonLat(): Devuelve el centro del mapa del rectángulo.
- scale(ratio,origin): Escala el rectángulo en función de un ratio y un origen. Por defecto el origen es el centro del rectángulo.
- add(x,y): Modifica el rectángulo de forma que contenga los valores de x e y pasados.
- extend(object): Extender los límites para incluir el punto, lonlat o límites especificados.
- containsLonLat(lonlat,inclusive): Comprueba si un punto expresado en latitud-longitud está contenido en un rectángulo. Por defecto el borde está contenido en el rectángulo.
- containsPixel(px,inclusive): Comprueba si un pixel está contenido en un rectángulo. Por defecto el borde está contenido en el rectángulo.
- contains(xy, inclusive): Comprueba si un punto expresado en xy está contenido en un rectángulo. Por defecto el borde está contenido en el rectángulo.
- intersectsBounds(bounds, inclusive): Comprueba si un rectángulo intersecta con el rectángulo actual. Por defecto el borde está contenido en el rectángulo.
- containsBounds(bounds, partial,inclusive): Comprueba si un rectángulo está contenido con el rectángulo actual, de forma completa o parcial. Por defecto el borde está contenido en el rectángulo.
- transform(srid_in,srid_out): Transforma la extensión de un sistema de referencia de entrada a uno de salida.

OpenLayers.Icon

Encapsula un icono. Tiene propiedades 'url', 'size', 'px' y 'offset'. También ofrece una propiedad 'calculateOffset' que permite calcular el offset en función del tamaño.

OpenLayers.Projection

Clase de transformación entre sistemas de coordenadas. Depende de la librería de proyecciones proj4js.

Dispone de los siguientes métodos:

- getCode(): Devuelve una cadena con el código del sistema de referencia.
- getUnits(): Devuelve una cadena con las unidades del sistema de referencia. Devuelve null si la librería proj4js no está disponible.
- addTransform(from, to, method)
- transform(point, source, dest): Transformar un punto de coordenadas de una proyección a otra.

9.2.1.5.4 La clase OpenLayers.Util

La clase OpenLayers.Util es una colección de métodos y propiedades utilitarios que permiten realizar determinadas tareas frecuentes de un modo sencillo. La documentación de la clase la puedes encontrar en:

http://dev.openlayers.org/docs/files/OpenLayers/Util-js.html

Desarrollamos a continuación la explicación de algunos de los métodos de la clase Util:

GetElement()

```
OpenLayers.Util.getElement(
      idElemento : String [, idElemento2 : String, ... ] )
      : [ domElement o Array[ domElements] ]
```

Admite como parámetro uno o más id's correspondientes a elementos del DOM que queremos seleccionar. Nos devuelve un elemento o un Array con los elementos solicitados.

El caso habitual con un solo parámetro de entrada es equivalente a 'document.getElementById(idElemento)' o al clásico '$('idElemento')'. Un ejemplo de utilización sería:

```
var element = OpenLayers.Util.getElement("panelIzquierdo");
```

En este caso la variable 'element' recibirá una referencia al elemento del documento html cuyo id es 'panelIzquierdo'.

isElement()

```
OpenLayers.Util.isElement( elemento : Object ) : Boolean
```

Recibe como parámetro el elemento a analizar y devuelve 'TRUE' si se trata de una instancia de la clase Element y 'FALSE' en caso contrario.

extend()

```
OpenLayers.Util.extend( dest: Object, source: Object ) : Object
```

Este método copia las propiedades del objeto 'source' en el objeto destino 'dest', sin modificar propiedades que no se especifiquen en el objeto 'source'. Es muy útil en situaciones en las que creamos un objeto con las propiedades por defecto, y luego con el método 'extend()' afinamos el valor de las propiedades que consideremos adecuado.

Un ejemplo de utilización lo tenemos en la creación de un estilo de visualización para una Feature punto. Podemos crear un estilo con los valores por defecto de la siguiente manera:

```
// Crear un objeto estilo con valores por defecto
var point_style = OpenLayers.Util.extend({},
    OpenLayers.Feature.Vector.style['default']);
```

Ahora solo necesitamos modificar los valores de las propiedades que queramos personalizar:

```
point_style.strokeColor = "#000011";
point_style.fillColor = "#ffa500";
```

removeItem()

```
OpenLayers.Util.removeItem( miArray: Array, item: Object ) : Array
```

Esta función recorre los elementos del Array 'miArray' y elimina el elemento 'item' si existe.

La función devuelve una referencia al Array modificado.

Para vaciar un Array hay que utilizar 'miArray.length = 0 '.

9.2.1.5.5 La clase OpenLayers.map

OpenLayers.Map es la clase fundamental de la librería OpenLayers. Todo programa de OpenLayers tiene como objeto crear un mapa que se visualizará en pantalla. Los objetos de la clase OpenLayers.Map son una colección de capas, OpenLayers.Layer, que contienen la información que se quiere mostrar, y controles, OpenLayers.Control, que permiten interactuar con el usuario. El objeto Map también es el responsable de gestionar la visualización del mapa en cuanto a Zoom y Panning se refiere.

El constructor de la clase Map tiene la siguiente forma:

```
OpenLayers.Map( div : [DOMElement/ String], options: Object)
```

El primer parámetro es una referencia al elemento 'div' del documento html destinado a contener el mapa. En lugar de una referencia se puede pasar una cadena de texto con el 'id' del elemento.

El segundo parámetro es un Array de opciones en la forma 'key:value'. Las opciones son valores de las propiedades del objeto 'Map' que queramos fijar con un valor determinado en el constructor.

Un ejemplo de mapa sin opciones adicionales podría ser:

```
var map = new OpenLayers.Map("divMapa");
```

Podemos añadir las opciones adicionales directamente en el constructor:

```
var map = new OpenLayers.Map("divMapa", {
        projection: 'EPSG:4326',
        units: 'm'
    }
  );
```

Pero también podemos preparar primero el objeto de opciones y añadirlo luego al constructor por referencia:

```
var opciones = {projection: 'EPSG:4326', units: 'm'};
var map = new OpenLayers.Map("divMapa", opciones);
```

La clase OpenLayers.Map expone una buena colección de propiedades y métodos.

Propiedades del objeto Map

Básicamente un objeto Map es una colección de capas (Layer) y controles (Control). Veamos en primer lugar una serie de propiedades que son colecciones de objetos pertenecientes al mapa:

- events: Objeto que maneja los eventos del mapa.
- allOverlays: Establece si todas las capas del mapa se tratarán como superpuestas “overlays”. Implica que todas las capas estarán marcadas como isBaseLayer como false al añadirse al mapa.
- div: Elemento que contiene el mapa.
- Layers: La propiedad 'layers' es un Array de objetos OpenLayers.Layer, que contiene la colección de capas del mapa. Para gestionar la colección de capas se dispone de los métodos: addLayer(), addLayers(), removeLayer(), getNumLayers(), getLayerIndex(), setLayerIndex(), raiseLayer(), setBaseLayer(), getLayer(), getLayersBy(), getLayersByClass(), getLayersByName(), setLayerZIndex(), resetLayersZIndex().
- baseLayer: Capa base actualmente seleccionada.
- tileSize: Tamaño del mosaico predeterminado del mapa.
- projection: Código de la proyección del mapa. Por defecto se asigna la proyección EPSG:4326
- units: Unidades del mapa.
- resolutions: Una lista de las resoluciones del mapa (unidades de mapa por píxel) en orden descendente.

- maxResolution: Resolución máxima del mapa. Por defecto el máximo es 360 ° / 256 píxeles, lo que equivale a ampliar el nivel 0 en gmaps.
- minResolution: Resolución mínima del mapa.
- maxScale: Denominador de la escala máxima del mapa.
- minScale: Denominador de la escala mínima del mapa.
- maxExtent: La extensión máxima para el mapa. El valor predeterminado es el mundo entero en grados (-180, -90, 180, 90).
- minExtent: La extensión mínima para el mapa, se define mediante un rectángulo.
- restrictedExtent: Limite de navegación del mapa en la medida que sea posible.
- numZoomLevels: Número de niveles de zoom para el mapa.
- Theme: Ruta de acceso relativa a un archivo CSS desde el que cargar estilos de tema. Especifique nulo en las opciones del mapa (por ejemplo, {tema: null}) si desea obtener declaraciones de estilo en cascada - al poner enlaces a hojas de estilo o de las declaraciones de estilo directamente en la página.
- displayProjection {OpenLayers.Projection} Requiere soporte proj4js. Proyección utilizada por varios controles para mostrar los datos de usuario.
- panMethod {Function} La función de aceleración que se utilizará para la interpolación.
- controls: Colección de controles asociados al mapa. Para manejar la colección se utilizan los métodos: addControl(), addControls(), addControlToMap(), getControl(), getControlsBy(), getControlsByClass() y removeControl().

9.2.1.6 GeoExt

GeoExt es una librería para la creación de aplicaciones web destinadas al manejo de mapas, ya que combina toda la potencia de OpenLayers y la toda la interface gráfica proporcionada por ExtJS, combinando los controles geoespaciales de OpenLayers con los componentes de interfaz de usuario de Ext JS.

GeoExt permite construir aplicaciones GIS de estilo similar a las de escritorio, pero en un navegador. En tan solo 12 líneas de código es posible crear una ventana con un mapa en un GeoExt.MapPanel.

Figura 16: GeoExt

9.2.1.6.1 Componentes de la librería GeoExt

Los componentes de GeoExt y sus clases, amplían la funcionalidad de ExtJS y sus clases con nuevas clases extendidas con soporte para mapas. En la API de GeoExt documentada, se incluyen propiedades, métodos y eventos sobre los objetos de la librería.

GeoExt contiene una colección de componentes especializados para la gestión de mapas y sus datos. Algunos de sus componentes son:

- Widgets:
  - GeoExt.Action: Crea una instancia de GeoExt.Action. Se crea una GeoExt.Action para insertar un control OpenLayers en una barra de herramientas como un botón o en un menú como un elemento de menú. Una instancia GeoExt.Action se puede utilizar como un Ext.Action regular o como una herramienta.
  - GeoExt.FeatureRenderer: Crear un componente de caja para la prestación de una capa con datos vectoriales.
  - GeoExt.LayerLegend: Clase base para los componentes de GeoExt.LegendPanel.
  - GeoExt.LayerOpacitySlider: Crear un control deslizante para controlar la opacidad de una capa.

- GeoExt.LegendImage: Muestra una imagen en una leyenda de una capa en un BoxComponent.
- GeoExt.LegendPanel: Se trata de un panel que muestra las leyendas de todas las capas. Dependiendo del tipo de capa, se elegirá un renderizador para cada leyenda. El LegendPanel incluirá leyendas para todas las capas incluidas en layerStore asociado al árbol de capas.
- GeoExt.MapPanel: Crear un contenedor de panel para un mapa. El mapa que aparece es un mapa de OpenLayers con todas sus propiedades.
- GeoExt.Popup: Popups es una ventana especializada que fija el anclaje a un lugar determinado en un MapPanel pulsado por el usuario. Cuando una ventana emergente que está anclada a un lugar, significa que la ventana emergente visible señala un punto y si se desplaza el mapa, la ventana se desplazará también.
- GeoExt.PrintMapPanel: Panel de mapa que controla la previsualización del mapa para proceder a su impresión. Será necesario un servicio para completar la funcionalidad de impresión.
- GeoExt.UrlLegend: Muestra una imagen de una leyenda en un BoxComponent.
- GeoExt.VectorLegend: Muestra la leyenda de una capa vectorial.
- GeoExt.WMSLegend: Muestra una imagen en una leyenda de una capa WMS. La imagen de la leyenda se obtendrá con una solicitud GetLegendGraphic para recuperar la imagen.
- GeoExt.ZoomSlider: Crea un control deslizante para controlar el nivel de zoom del mapa.
- GeoExt.LayerOpacitySliderTip: Crea un control para regular los valores de opacidad de GeoExt.LayerOpacitySlider.
- GeoExt.ZoomSliderTip: Crea un control para regular los valores del control GeoExt.ZoomSlider.
- GeoExt.form: El módulo GeoExt.form contiene clases que amplían la funcionalidad de mapa relacionado con las clases Ext.form de ExtJS.
  - GeoExt.form.BasicForm
  - GeoExt.form.FormPanel
  - GeoExt.form.SearchAction
- GeoExt.grid: El módulo GeoExt.grid contiene clases que amplían la funcionalidad de mapa relacionado con las clases Ext.grid de ExtJS.
  - GeoExt.grid.FeatureSelectionModel
  - GeoExt.grid.SymbolizerColumn
- GeoExt.tree: El módulo GeoExt.tree contiene clases que amplían la funcionalidad de mapa relacionado con las clases Ext.tree de ExtJS. Es utilizado para la gestión del árbol de capas.
  - GeoExt.tree.BaseLayerContainer
  - GeoExt.tree.LayerContainer
  - GeoExt.tree.LayerLoader
  - GeoExt.tree.LayerNode
  - GeoExt.tree.LayerParamLoader
  - GeoExt.tree.LayerParamNode
  - GeoExt.tree.OverlayLayerContainer

- GeoExt.tree.TreeNodeUIEventMixin
- GeoExt.tree.WMSCapabilitiesLoader

9.2.1.6.2 Ejemplo de generación de un mapa básico con GeoExt

Figura 17: Ejemplo de mapa básico con GeoExt

Para la generación de este ejemplo es necesaria la carga de las librerías desde una página de arranque HTML y un script de JavaScript que construye los objetos.

Index.html

```
<html>
   <head>
      <title>GeoExt MapPanel in an Ext Window</title>
      <script type="text/javascript" src="http://extjs.cachefly.net/ext-3.4.0/adapter/ext/ext-base.js"></script>
      <script type="text/javascript" src="http://extjs.cachefly.net/ext-3.4.0/ext-all.js"></script>
      <link rel="stylesheet" type="text/css" href="http://extjs.cachefly.net/ext-3.4.0/resources/css/ext-all.css" />
      <link rel="stylesheet" type="text/css" href="http://extjs.cachefly.net/ext-3.4.0/examples/shared/examples.css" />
      <script src="http://www.openlayers.org/api/2.11/OpenLayers.js"></script>
      <script type="text/javascript" src="../script/GeoExt.js"></script>

      <script type="text/javascript" src="mappanel-window.js"></script>

   </head>
   <body>
```

```
        <h1>GeoExt.MapPanel in an Ext.Window</h1>
        <p>This example shows the how to place a MapPanel in another Ext container
        without creating a map first.  See <a href="mappanel-div.html">mappanel-
div.html</a>
        for an example that creates the map before creating the map panel.<p>
        <p>The js is not minified so it is readable. See <a href="mappanel-
window.js">mappanel-window.js</a>.</p>
    </body>
</html>
```

Mappanel-window.js

```
/** api: example[mappanel-window]
 *  Map Panel (in a Window)
 *  -----------------------
 *  Render a map panel in a Window.
 */

var mapPanel;
Ext.onReady(function() {
    new Ext.Window({
        title: "GeoExt MapPanel Window",
        height: 400,
        width: 600,
        layout: "fit",
        items: [{
            xtype: "gx_mappanel",
            id: "mappanel",
            layers: [new OpenLayers.Layer.WMS(
                "Global Imagery",
                "http://maps.opengeo.org/geowebcache/service/wms",
                {layers: "bluemarble"}
            )],
            extent: "-5,35,15,55"
        }]
    }).show();
    mapPanel = Ext.getCmp("mappanel");
});
```

10 Desarrollo de un Geoportal WEB IDE

En esta sección se describirá el proceso completo de conversión de los datos y los proceso de creación de los diferentes servicios y las herramientas de consulta para que sirviendo de base, puedan crearse nuevas herramientas de consulta y volcado de los datos. La instalación de las herramientas básicas del servidor se describirá en el anexo.

Inicialmente se seguirá un proceso de creación que partirá de los datos, la creación de los repositorios necesarios, conversión y volcado de datos. Posteriormente se crearán los servicios de cartografía y de consulta y posteriormente se verán las herramientas para la consulta de estos datos a través de los servicios.

10.1 Creación del repositorio de datos espacial

Para la creación del repositorio de datos espacial y alfanumérico se emplearán bases de datos de PostgreSQL+PostGIS.

La administración de un sistema de gestión de bases de datos como PostgreSQL puede realizarse de dos formas, mediante la línea de comandos, o mediante el uso de una herramienta de administración que visualmente facilite las tareas de administración. Existen diferentes herramientas para la administración de PostgresSQL, citaremos dos de ellas que pueden obtenerse mediante las últimas distribuciones de PostgresSQL:

- pgAdmin III: Herramienta de escritorio para la administración de PostgreSQL.
- phpPgAdmin: Herramienta en entorno web que permite la administración remota sin necesidad de instalación de ningún software en el equipo cliente.

Describiremos la administración con pgAdmin III por ser más completa y dar respuesta a todas las necesidades de administración de PostgreSQL.

Una vez instalado PostgresSQL en un entorno Windows, pgAdmin III estará disponible desde la entrada del menú inicio.

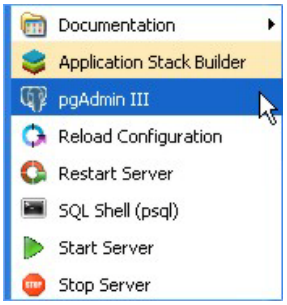


Figura 18: Ejecución de pgAdmin III

Una vez ejecutado pgAdmin III encontraremos una interface similar a la siguiente:

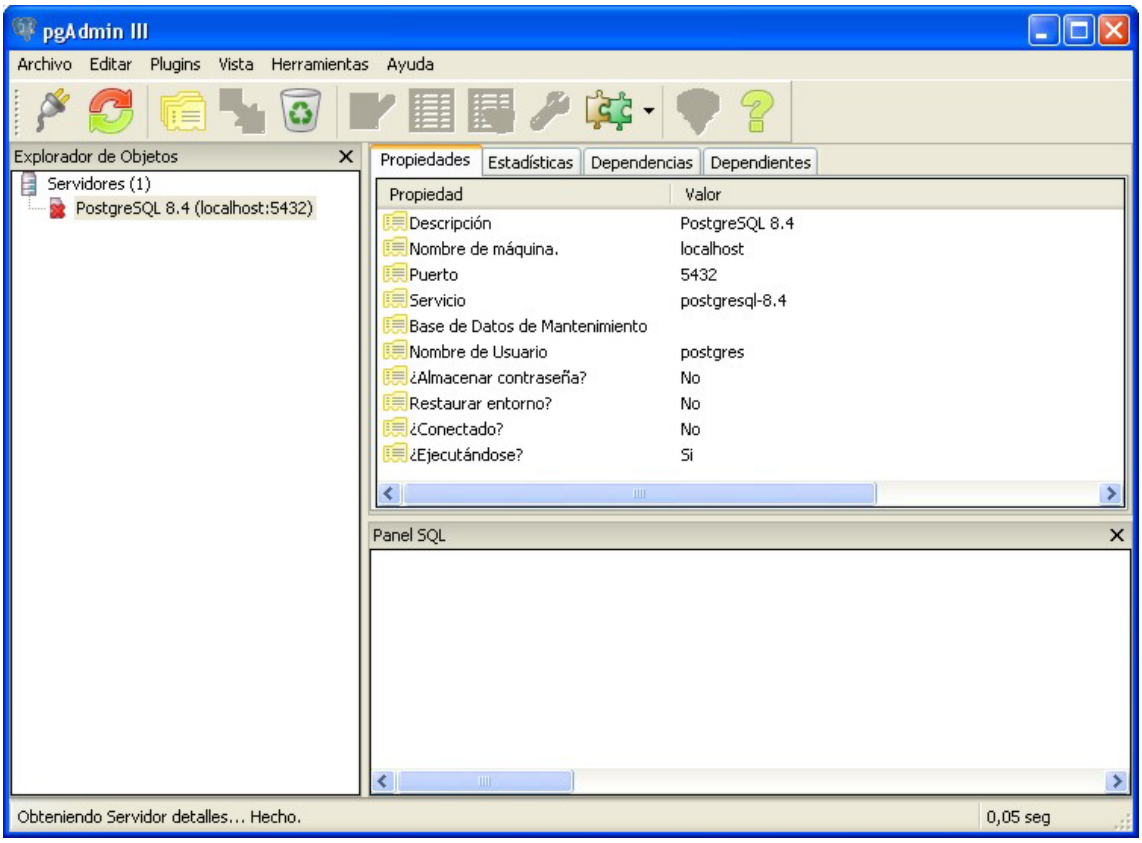


Figura 19: Conexión a la base de datos PostgreSQL

En la sección de explorador de objetos encontraremos los diferentes servidores registrados para administrar. Por defecto aparece el servidor instalado en la máquina local como localhost en el puerto por defecto 5432.

Explorando el servidor localhost encontraremos:

- Bases de datos: Conjunto de bases de datos que existen en la instancia de PostgreSQL de la máquina. Las bases de datos están estructuradas por esquemas de información para la organización de la información.
- Tablespaces: Los tablespaces son los entornos de trabajo y espacios reservados en disco para el almacenamiento de datos. La instalación de PostgreSQL define dos tablespaces por defecto pg_default y pg_global.
- Roles de grupos: Definición de los grupos de usuarios del servidor de bases de datos.
- Roles de login: Definición de los usuarios del servidor de la base de datos.

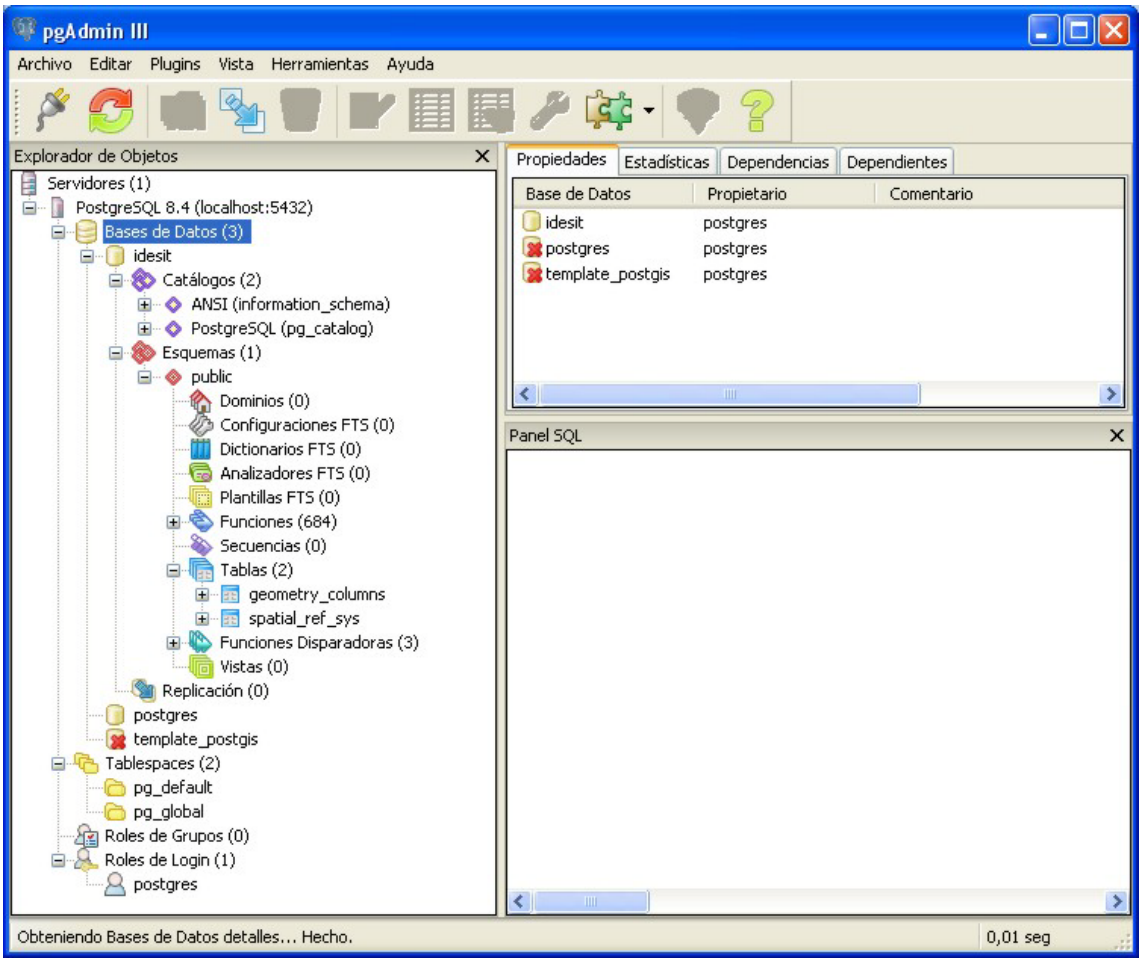


Figura 20: Exploración de la base de datos PostgreSQL

Para cada uno de los elementos del explorador de objetos, existe un menú contextual accesible desde el botón secundario del ratón. En este caso para la creación de la base de datos pulsaremos con el botón secundario del ratón sobre bases de datos.

Figura 21: Creación de una base de datos PostgreSQL con extensión PostGIS

Para la creación de la base de datos, será necesario especificar una serie de datos como:

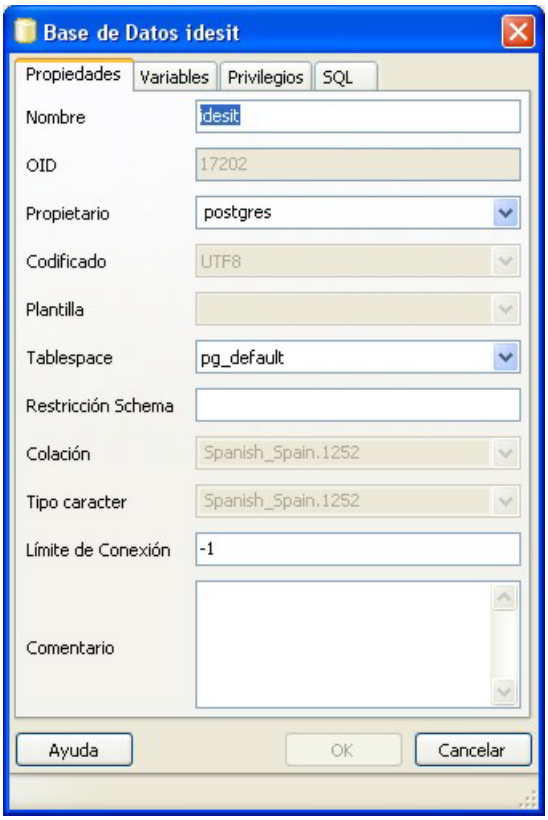


Figura 22: Parámetros de creación de una base de datos PostgreSQL con extensión PostGIS

- Nombre de la base de datos.
- Propietario de la base de datos.
- Codificación de la base de datos. Perfil de la página de códigos a emplear.
- Plantilla con la cual se creará la base de datos. Inicialmente si la base de datos a emplear albergará datos espaciales, será necesaria la creación con la plantilla de PostGIS.
- Tablespace.
- Colación. Idioma para el cual se genera la base de datos.
- Tipo carácter. Idioma para el cual se genera la base de datos.

Una vez generada la base de datos podremos explorar los elementos que contiene:

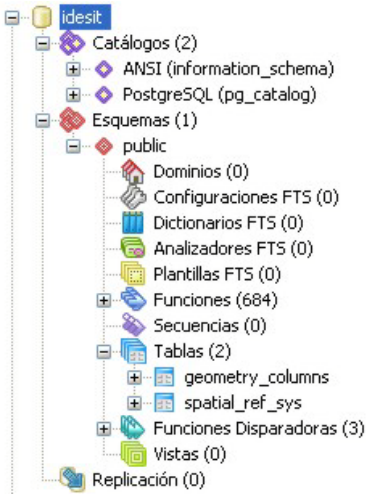


Figura 23: Exploración de la base de datos creada

En cuanto a los objetos de la base de datos, los más importantes son:

- Esquemas: Diferentes secciones de la base de datos para la organización de los datos. Por defecto en la creación de cualquier base de datos se genera el esquema public, en este esquema se almacenarán las entidades o tablas de información alfanumérica y espacial, las funciones, secuencias, vistas, funciones disparadoras, etc.
- Tablas: Las tablas almacenarán la información. Por defecto cuando una base de datos es creada con la platilla de PostGIS, deben existir dos tablas para el registro de entidades gráficas:
  - Geometry_columns: En esta tabla se referencian las entidades que contienen información geográfica.
  - Spatial_ref_sys: En esta tabla se almacenan los diferentes sistemas de referencia espacial que podremos emplear.
- Funciones: Son las funciones propias que PostGIS que nos permitirán interactuar con los datos geográficos.
- Secuencias: Las secuencias registran los identificadores generados de forma automática para los elementos de diferentes entidades o tablas.

10.2 Conversión de datos geográficos

Para el volcado de datos geográficos sobre bases de datos de PostgreSQL+PostGIS, disponemos de varias opciones, pero la más recomendada es emplear el cargador de figuras shp2pgsql que se incluye con la instalación de PostgresSQL, otra de las opciones es emplea gvSIG para exportar datos a PostGIS.

Se parte de la base que la información geográfica se encuentra en un formato conocido como ShapeFile y en un sistema de referencia conocido.

Esta utilidad se emplea desde la línea de comandos y está accesible desde la entrada de menú de la instalación de PostgreSQL.

Existen alternativas a la hora de cargar información geográfica, la opción recomendada es la transformación de la información geográfica en sentencias SQL de inserción que podremos recuperar en cualquier momento.

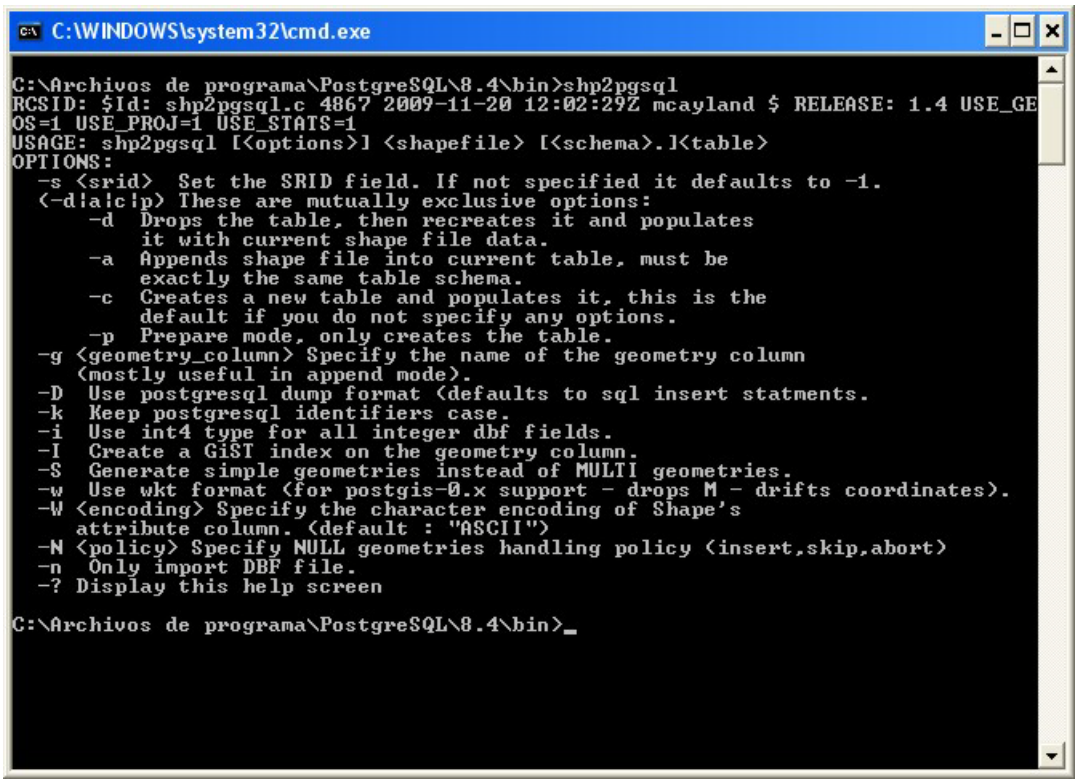


Figura 24: Conversión de datos .SHP a .SQL

La sintaxis del cargador de figuras es la siguiente:

Shp2pgsql –s sistema_de_refencia –I shapefile esquema.tabla>shape.sql

- -s sistema de referencia: Especifica el sistema de referencia en el cual se encuentran los datos de origen en el fichero ShapeFile.
- -I: Crea el índice espacial en la capa una vez cargada.
- Shapefile: Ruta hasta el fichero ShapeFile a convertir.
- Esquema.tabla: Esquema y tabla en la cual se convertirán los datos. Por defecto el esquema es el esquema public.
- >shape.sql: Formato de salida de la conversión. Esta salida en consulta SQL se puede ejecutar e insertar directamente los datos sobre PostGIS.

En la conversión de datos se empleará cartografía de CartoCiudad con su modelo de datos. Las sentencias de conversión de las diferentes capas es la siguiente:

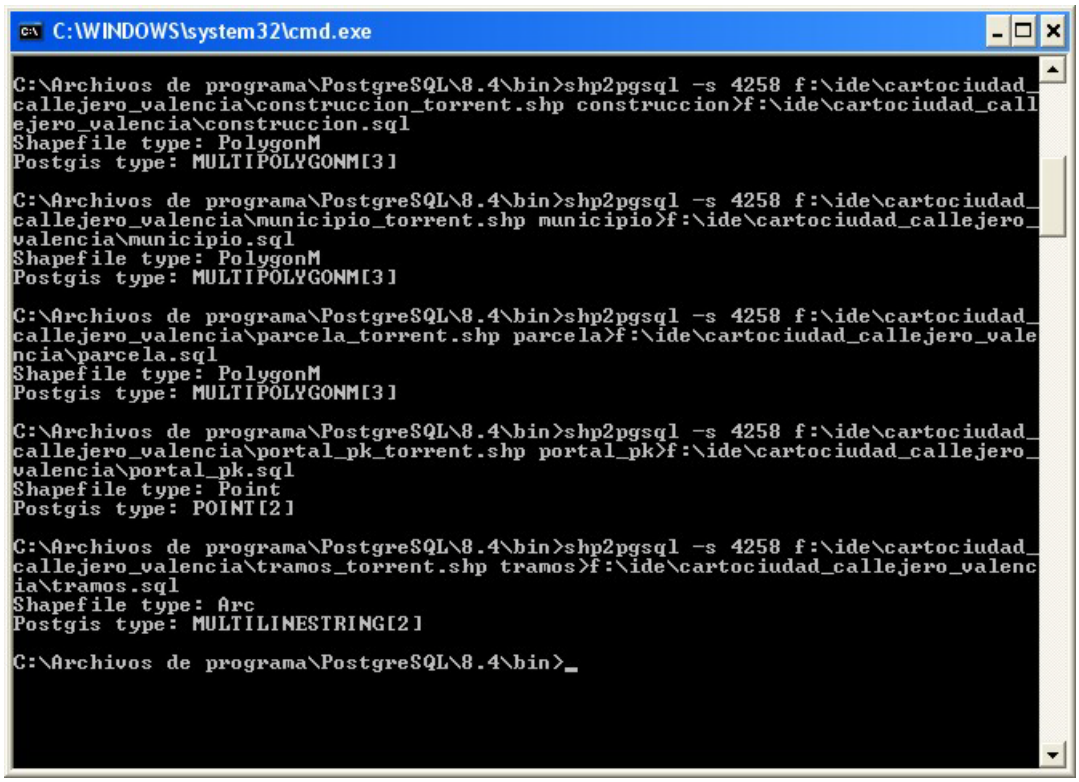


Figura 25: Obtención de ficheros .SQL desde ficheros .SHP

10.3 Volcado de datos geográficos

Una vez se han convertidos los datos geográficos a sentencias SQL de inserción de PostGIS, será necesario ejecutar las SQL sobre PostGIS, para ello emplearemos la herramienta de administración de PostGIS.

Figura 26: Volcado de datos geográficos

Al arrancar una consulta SQL aparecerá una ventana en la cual escribir las sentencias SQL a ejecutar sobre la base de datos activa. En este caso abriremos las sentencias SQL existentes creadas con el cargador de figuras shp2pgsql.

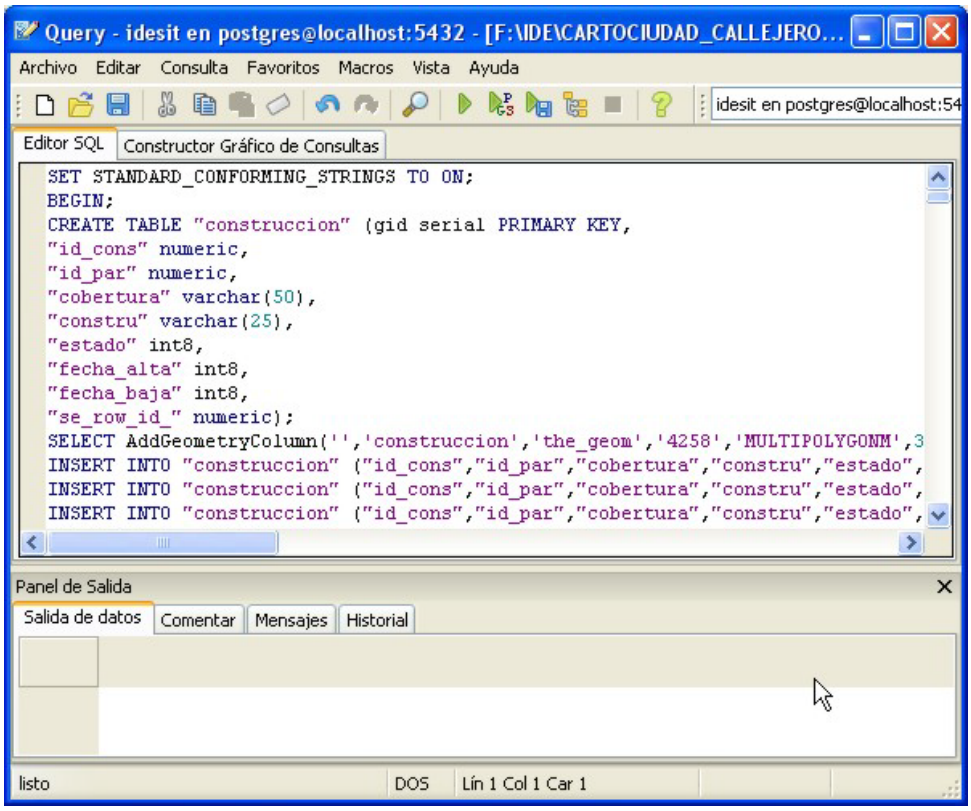


Figura 27: Muestra de una consulta SQL

Ejecutaremos la consulta. Del resultado de esta consulta no se obtendrán resultados visibles, pero si la creación de deferentes objetos en la base de datos. Como por ejemplo la creación de una nueva tabla, una nueva secuencia, y la inserción de una nueva referencia en la tabla geometry_columns.

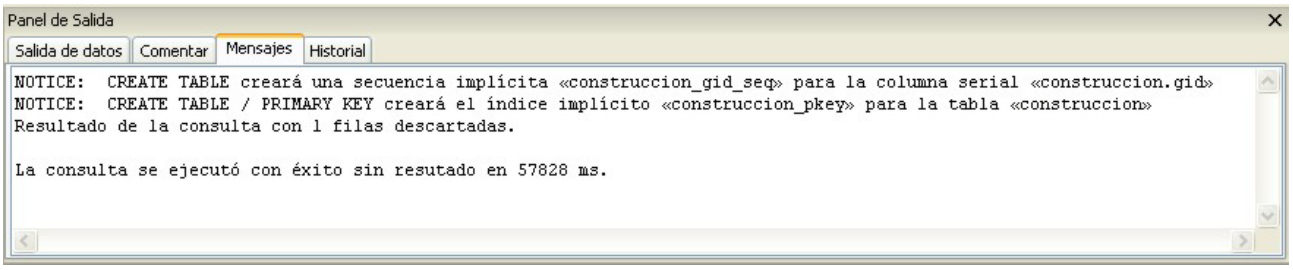


Figura 28: Resultado de la ejecución de la SQL

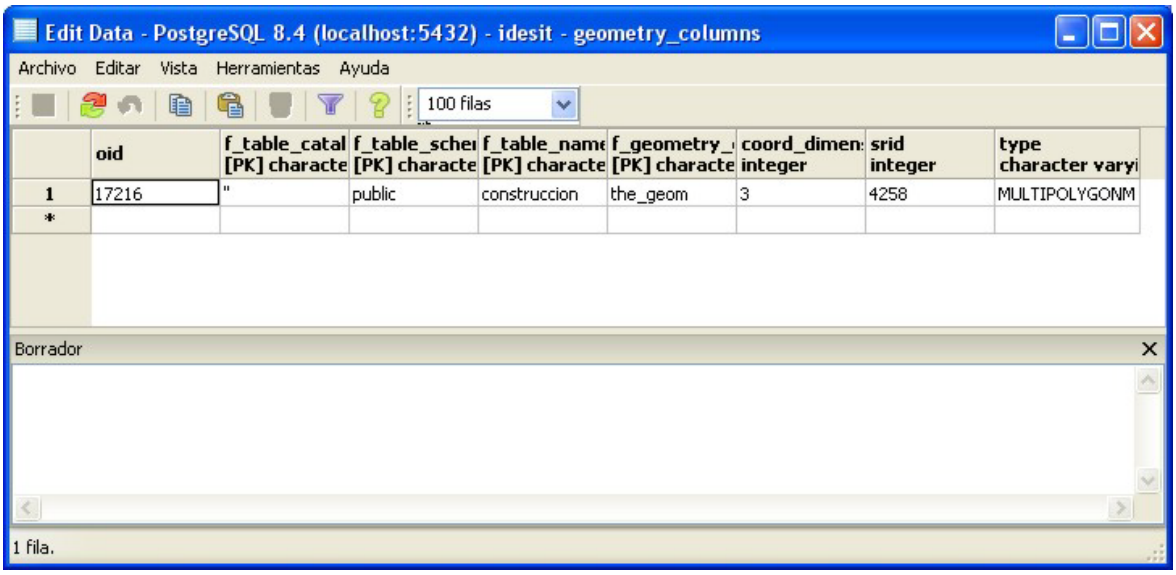


Figura 29: Registro de capa espacial

Repitiendo el proceso para una de las capas que sea necesario insertar en la base de datos obtendremos una entidad o tabla para cada una de las capas.

Se podrá explorar el resultado de cada uno de los objetos creados desde el explorador de objetos de pgAdmin III y en particular observar el resultado de la inserción de datos sobre las nuevas tablas. Para ello con el botón secundario de cada una de las entidades se abre el menú contextual, y en la opción **Ver datos >Ver las 100 filas superiores**

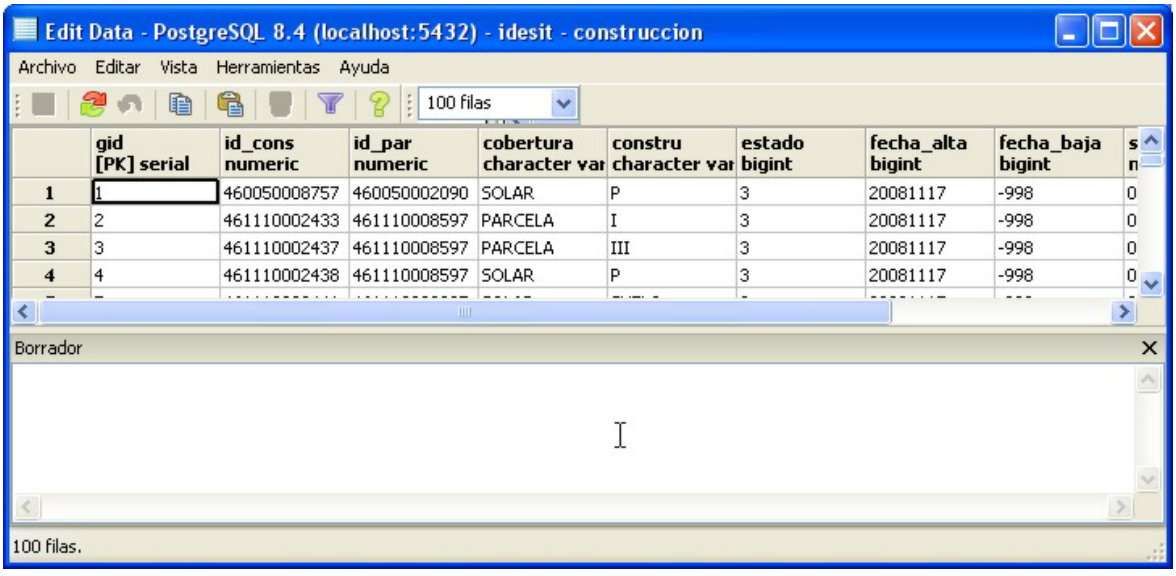


Figura 30: Datos de una tabla espacial

10.4 Optimización de la base de datos

Una vez realizada la inserción de datos sobre la base de datos, será recomendable la optimización de la base de datos. Esta optimización se recomienda hacerla de diferentes formas, mediante la creación de índices, tanto alfanuméricos como espaciales, y hacer una optimización y re indexado completo de la base de datos.

10.4.1 Creación de índices

La creación de índices sobre las diferentes entidades o tablas de la base de datos acelera el rendimiento de las tablas y el conjunto de la base de datos. Existen diferentes tipos de índices alfanuméricos y un índice espacial para la información geográfica almacenada en el campo de geometría, por defecto the_geom, de las entidades.

La creación de índices es similar en todos sus tipos. En el explorador de objetos, se localiza una entidad en la cual generar un índice, los objetos índices por defecto aparecerán vacíos, y con el menú contextual pulsaremos sobre **Nuevo índice**.

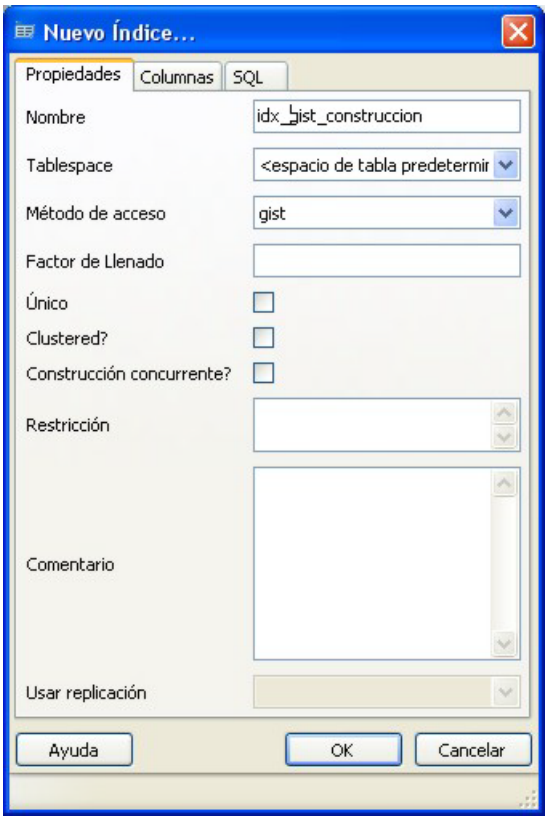


Figura 31: Creación de índices espaciales

En la definición del índice será necesario definir:

- Nombre del índice.
- Tablespace donde se creará el índice.
- Método de acceso.

- Columna sobre la cual se aplicará el índice.

10.4.1.1 Índices alfanuméricos

Se emplean para el acceso a columnas que contienen datos alfanuméricos, principalmente interesa para realizar clasificaciones de capas mediante valores únicos o intervalos. Emplean métodos de acceso a datos del tipo btree.

10.4.1.2 Índices espaciales

Se emplean para el acceso a columnas que contienen datos alfanuméricos, principalmente interesa para realizar clasificaciones de capas mediante valores únicos o intervalos. Se emplean métodos de acceso a datos del tipo gist.

10.4.2 Mantenimiento de la base de datos

En las opciones de mantenimiento de la base de datos, se encuentran las utilidades de optimización y re indexado de la base de datos.

Figura 32: Mantenimiento de la base de datos PostgreSQL

Entre estas utilidades se encuentran las opciones de **vacuum** y **reindex** que permiten la optimización de la base de datos.

A partir de este momento el repositorio está preparado para servir datos para las diferentes aplicaciones y servicios. Un método de comprobación es la creación de un acceso mediante aplicaciones de escritorio que soporten el acceso a bases de datos espaciales de PostGIS como son uDIG y gvSIG.

10.5 Creación de servicios básicos de cartografía

La creación de servicios básicos de cartografía servirá para la ir construyendo la IDE e ir comprobando el funcionamiento de cada una de las capas que la componen. Estos servicios se generarán de diferentes formas y con diferentes servidores de mapas, como son Geoserver y Mapserver.

10.5.1 Geoserver

Geoserver es un servidor de mapas desarrollado en java que cumple los estándares del OGC como ya se ha comentado. Se parte de la base que Geoserver se encuentra instalado en el servidor; consultar la sección de instalación de Geoserver.

10.5.1.1 Inicio de Geoserver

Tras la instalación Geoserver se encuentra accesible para ser empleado en la dirección http://localhost:8080/geoserver

La apariencia inicial al entrar en la aplicación de Geoserver es la siguiente:

Figura 33: Interfaz de gestión de Geoserver

Por defecto Geoserver tras la instalación genera un usuario para administrar el servidor de mapas. Este usuario se valida en la aplicación con las siguientes credenciales, usuario: admin, clave: geoserver.

Tras esta validación se tiene acceso a las diferentes herramientas de administración, en la tabla de contenidos de la sección izquierda de la aplicación. Aquí encontraremos herramientas clasificadas en los tipos: servidor, servicios, datos, seguridad y previsualización de capas.

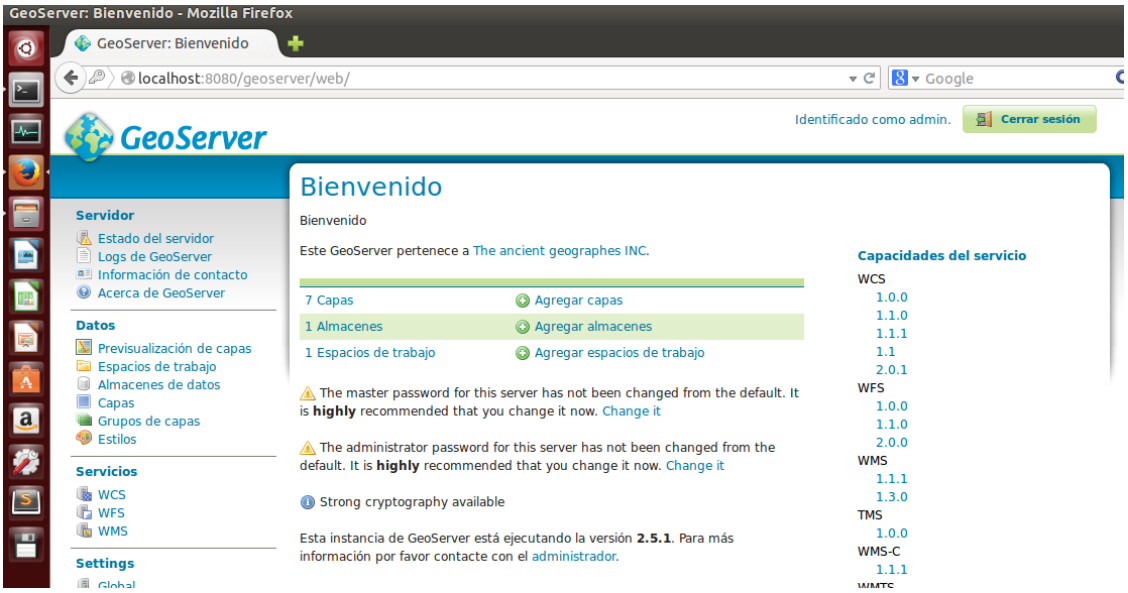


Figura 34: Administración Geoserver

10.5.1.2 Estado del servidor

En esta sección se muestra información sobre el estado del servidor y su funcionamiento, uso de memoria, máquina virtual empleada, etc. Desde esta sección es posible liberar memoria y bloqueos de la aplicación.

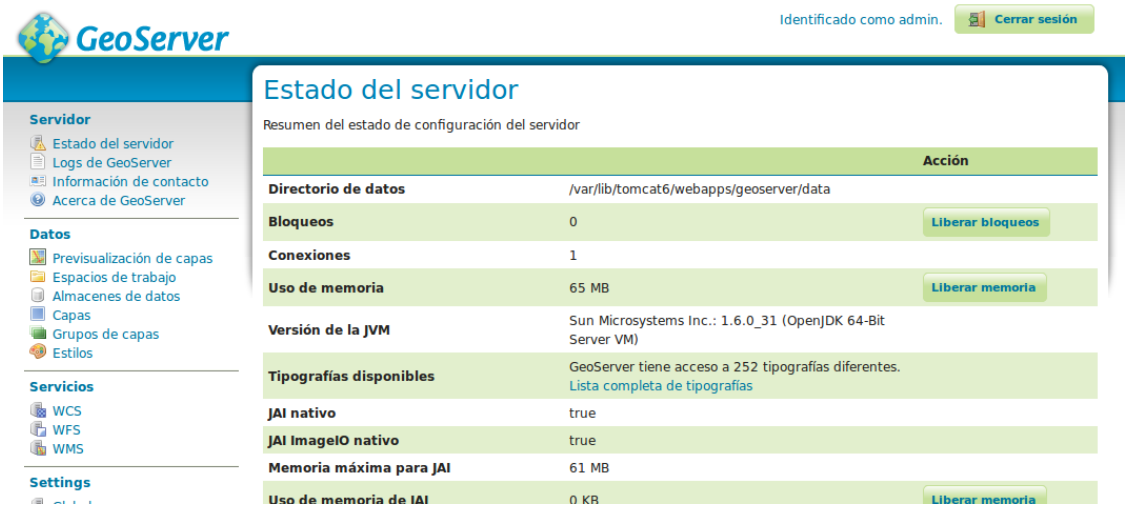


Figura 35: Consulta del estado del servidor

10.5.1.3 Información del contacto

En esta sección se incluye la descripción básica del servidor y la persona de contacto.

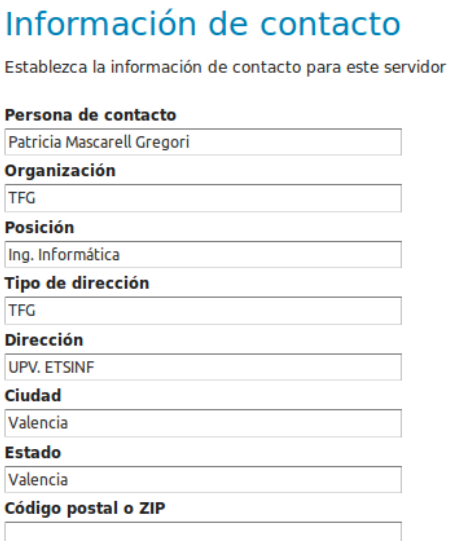


Figura 36: Información de contacto

10.5.1.4 Configuración global

Esta es una configuración global que se aplica a todo el servidor, donde se definen el número de decimales con el cual operar y el perfil de página de códigos a emplear en los datos.

También se define la salida del fichero de registro que podremos emplear para hacer debug ante los errores encontrados.

Figura 37: Configuración global

10.5.1.5 Configuración de JAI

JAI Java Advanced Imaging es una librería java de acceso y manipulación de imágenes empleadas por Geoserver. Desde este panel podrá configurarse su uso.

Figura 38: Configuración JAI

10.5.1.6 Servicios de Geoserver

Geoserver proporciona una serie de servicios que serán enumerados a continuación. Entre ellos se encuentran:

- GWC Geowebcaché: Proporciona utilidades de tileado y caché de imágenes para el servicio WMS, soporta multitud de formatos entre los que se encuentran WMS-C, WMTS, KML, Google Maps and Virtual Earth.
- WCS. Servicio de coberturas.
- WFS. Servicio de elementos vectoriales.
- WMS. Servicio de mapas.

Geoserver dispone de una configuración básica a nivel de servicio sobre la cual se publicarán los diferentes datos. Cada uno de estos servicios dispone de una serie de metadatos, que serán los enviados en el servicio de descubrimiento de respuesta ante cualquier petición GetCapabilities. Adicionalmente cada servicio dispone de parámetros de configuración particulares. Los metadatos de los diferentes servicios son similares:

10.5.1.6.1 Metadatos para WCS

Figura 39: Configuración WCS

10.5.1.6.2 Metadatos para WFS

Figura 40: Configuración WFS

10.5.1.6.3 Metadatos para WMS

Figura 41: Configuración WMS

10.5.1.7 Datos en Geoserver

Geoserver es capaz de conectarse a un conjunto de datos definido que puede ser extendido mediante el uso de drivers o librerías java de acceso a datos. Los datos en Geoserver se emplean mediante la creación de objetos de espacios de nombre y almacenes para la organización de los mismos.

10.5.1.7.1 Espacios de trabajo

En espacio de trabajo o espacio de nombres es una agrupación de datos de forma que puedan identificarse los datos que guardan relación, ya que Geoserver sólo es capaz de generar una instancia de servicio WMS, WFS, WCS. Así para hacer referencia a un determinado dato se hace anteponiendo el espacio de trabajo al mismo, por ejemplo **mi_espacio_trabajo:mi_dato**.

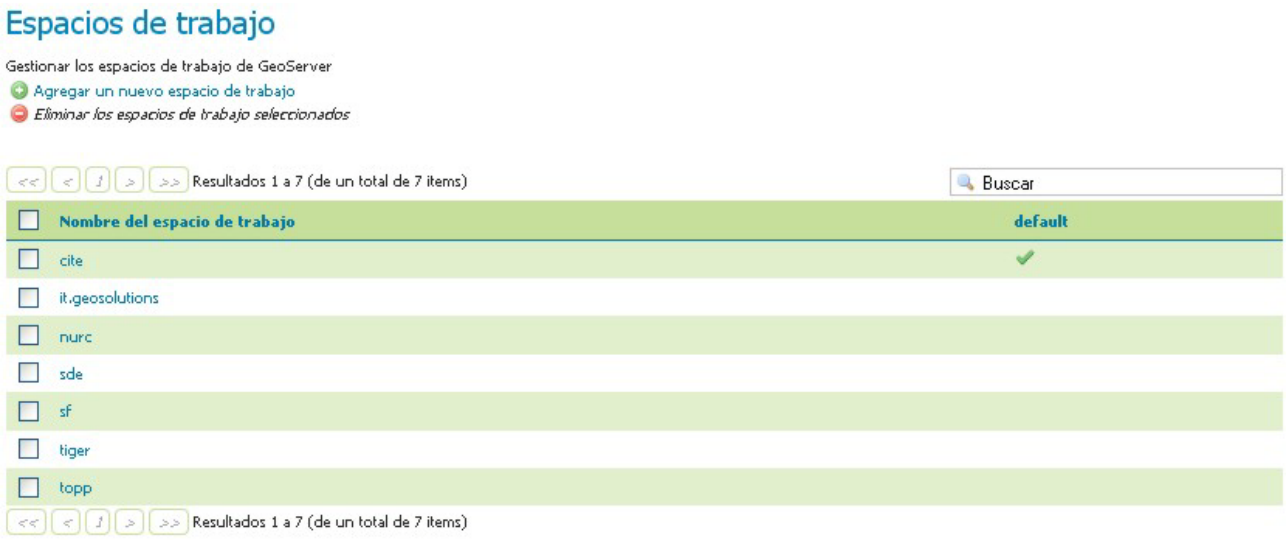


Figura 42: Creación de espacios de trabajo

Por defecto en Geoserver siempre existe un espacio de trabajo por defecto.

10.5.1.7.2 Creación de un espacio de trabajo

Para crear un nuevo espacio de trabajo, sólo es necesario pulsar en **Agregar un nuevo espacio de trabajo**, esta opción nos permitirá la creación.

Figura 43: Espacio de trabajo de Geoserver

10.5.1.7.3 Almacenes de datos

Los almacenes de datos son conjuntos de datos que provienen de la misma fuente. En este caso se puede decir que un almacén de datos se trata de una conexión a una fuente de datos.

10.5.1.7.4 Nuevo almacén de datos

Es posible la creación de nuevos almacenes de datos de diferentes tipos. Al pulsar sobre la opción de nuevo almacén de datos se deberá definir el tipo del mismo.

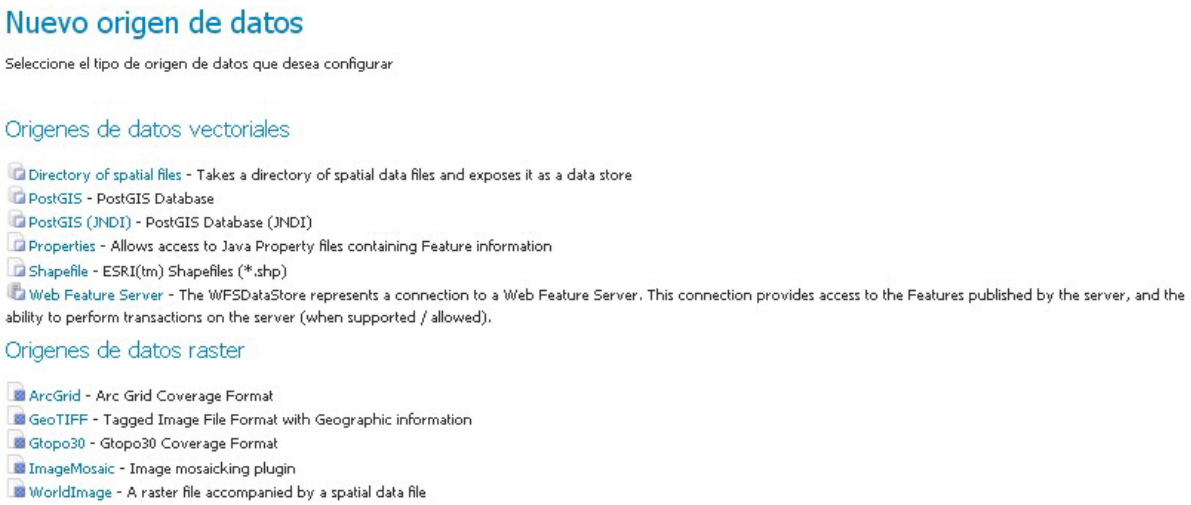


Figura 44: Definición de un origen de datos

Una vez definido el tipo de almacén, en nuestro caso el PostGIS, será necesario completar los datos de la conexión.

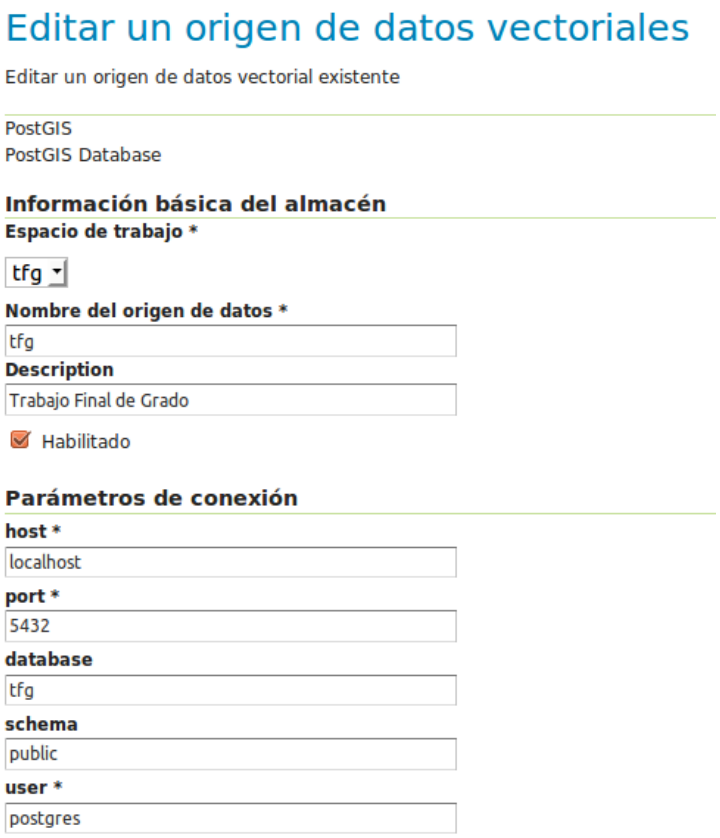


Figura 45: Configuración de un origen de datos

10.5.1.7.5 Publicar una nueva capa

Cuando ya se ha creado la nueva conexión al almacén de datos, el almacén será explorado y presentará las entidades que contiene, de forma que será posible seleccionar la entidad o la tabla que queremos publicar como una determinada capa. En esta exploración de datos aparecerán también las vistas de la base de datos de PostGIS con geometrías registradas en la tabla geometry_columns.

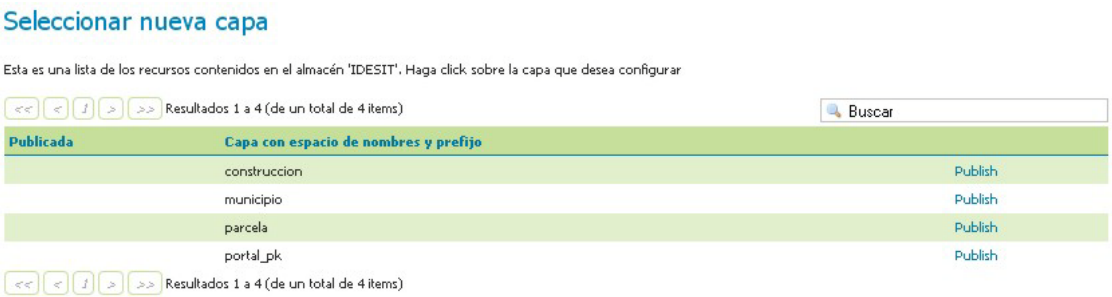


Figura 46: Publicación de una capa

Para la publicación de la capa será necesario definir una serie de parámetro de ésta, como por ejemplo el nombre, descripción, palabras clave, sistema de referencia, etc.

Con Geoserver será sencillo incluso calcular la extensión de la misma, ya que realiza comprobaciones directas sobre el origen de datos explorando los atributos de la misma.

Figura 47: Configuración de una capa

10.5.1.7.6 Visualización una nueva capa

Geoserver proporciona una forma sencilla de comprobación de la publicación de capas, en el panel de herramientas, dispone de una previsualización de capas en diferentes formatos.

Previsualización de capas

Despliega todas las capas configuradas en GeoServer y proporciona una vista previa en varios formatos.

<< < 1 > >> Resultados 1 a 7 (de un total de 7 ítems) Buscar

Tipo	Nombre	Título	Formatos habituales	Todos los formatos
	tfg:tramos	tramos	OpenLayers KML GML	Seleccionar una
	tfg:manzana	manzana	OpenLayers KML GML	Seleccionar una
	tfg:parcela	parcela	OpenLayers KML GML	Seleccionar una
	tfg:toponimos	toponimos	OpenLayers KML GML	Seleccionar una
	tfg:construccion	construccion	OpenLayers KML GML	Seleccionar una
	tfg:municipio	municipio	OpenLayers KML GML	Seleccionar una
	tfg:portal_pk	portal_pk	OpenLayers KML GML	Seleccionar una

<< < 1 > >> Resultados 1 a 7 (de un total de 7 ítems)

Figura 48: Listado de capas a previsualizar

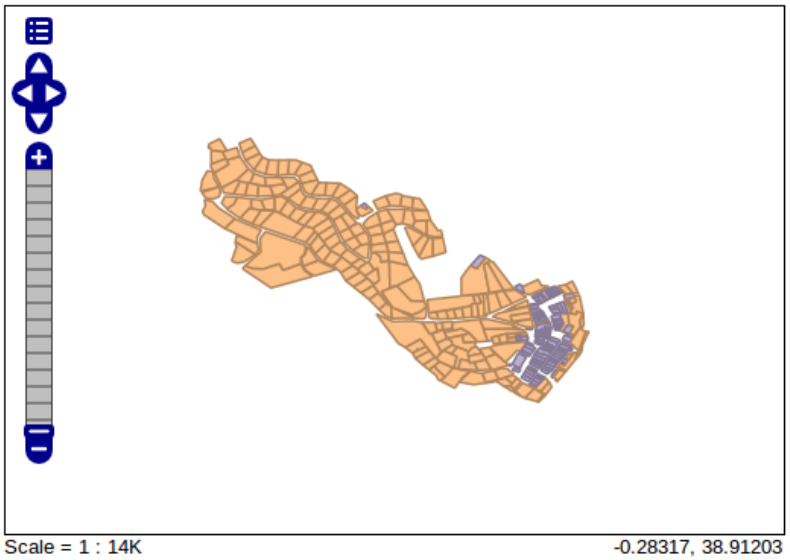


Figura 49: Visualización de una capa

10.5.1.7.7 Grupos de capas

Los grupos de capas permiten la publicación de varias capas agrupadas bajo el nombre del grupo, de esta forma sólo es necesario invocar al grupo para consultar todas las capas que contiene. Es posible generar nuevos grupos de capas y gestionar el orden de visualización de las capas dentro del grupo.

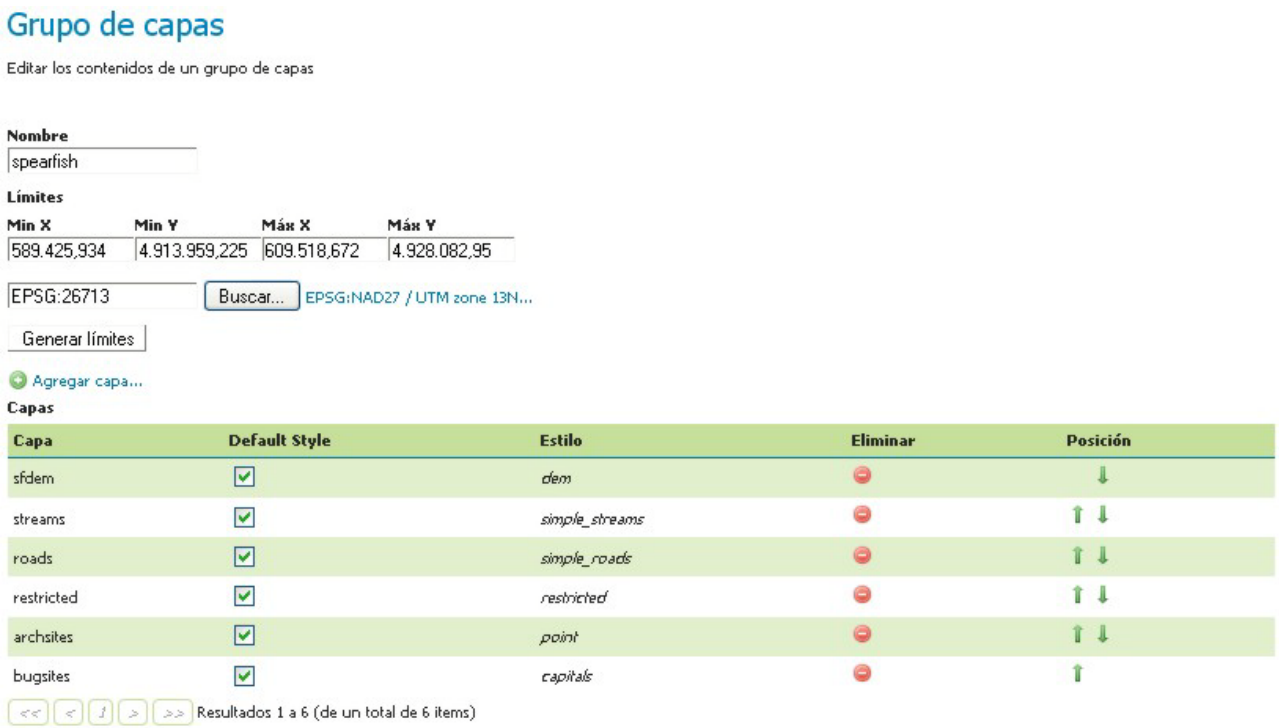


Figura 50: Definición de un grupo de capas

10.5.1.7.8 Estilo de visualización de una nueva capa

Figura 51: Selección del estilo de visualización de una capa

En Geoserver es posible incluir estilos de visualización de capas para el servicio WMS. Geoserver emplea el estándar SLD Style Layer Descriptor. SLD es una configuración de la forma en la que se tiene que visualizar la capa, está almacena en XML, pero Geoserver no dispone de editor de SLD en la última versión, para ello se deberá apoyarse en otras herramientas que si implementan soluciones de edición de SLD como gvSIG o uDIG.

Para la modificación de la simbología de una capa es necesario acceder a la pestaña de publicación en las propiedades de la capa.

Por defecto Geoserver asigna una simbología por defecto, y es posible añadir una serie de símbolos adicionales de la colección disponible.

10.5.1.7.9 Creación de nuevo estilo de visualización

Figura 52: Conexión base de datos

Para la creación de un nuevo estilo de visualización será necesaria la creación de la definición de un nuevo fichero de SLD. Para la generación de un nuevo estilo SLD emplearemos gvSIG, así se accederá a la base de datos espacial y se cargará la capa directamente ya que gvSIG es capaz de acceder a datos PostGIS. Se añade una nueva capa a la vista de tipo **GeoDB** e incluyendo los parámetros de conexión. De esta forma la capa se cargará en la tabla de contenidos.

Posteriormente se aplica la simbología deseada haciendo uso del editor de simbología propio de gvSIG. Una vez aplicada la simbología deseada sólo bastará con guardar la leyenda generada en formato SLD.

10.5.1.8 Estilos. Carga de nuevos estilos de visualización

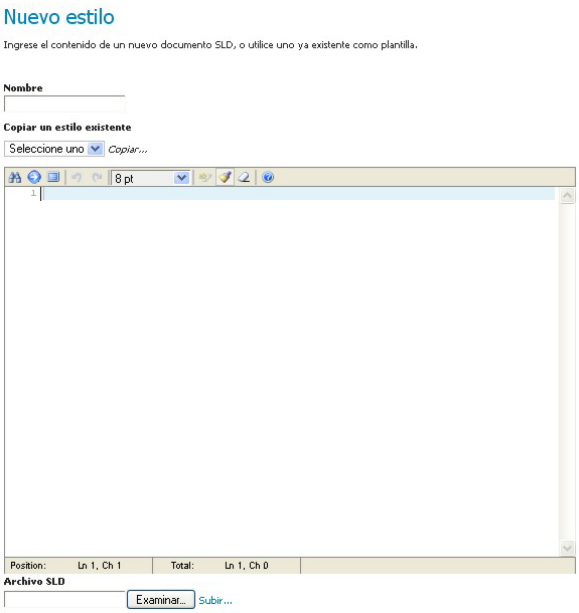


Figura 53: Editor de estilos SLD

Si ya disponemos de la simbología de una capa definida en un formato de fichero SLD, sólo será necesario cargarla y hacer que Geoserver la aplique para servir la capa con la nueva simbología. Para ello será necesario acceder en el panel de herramientas en la sección de estilos, y crear un nuevo estilo.

Se deberá dar un nuevo nombre al estilo y definirlo directamente sobre el editor de texto, o es posible cargar un fichero SLD existente. Se cargará el fichero de resultado de guardar la leyenda de gvSIG, y el símbolo estará disponible para ser empleado.

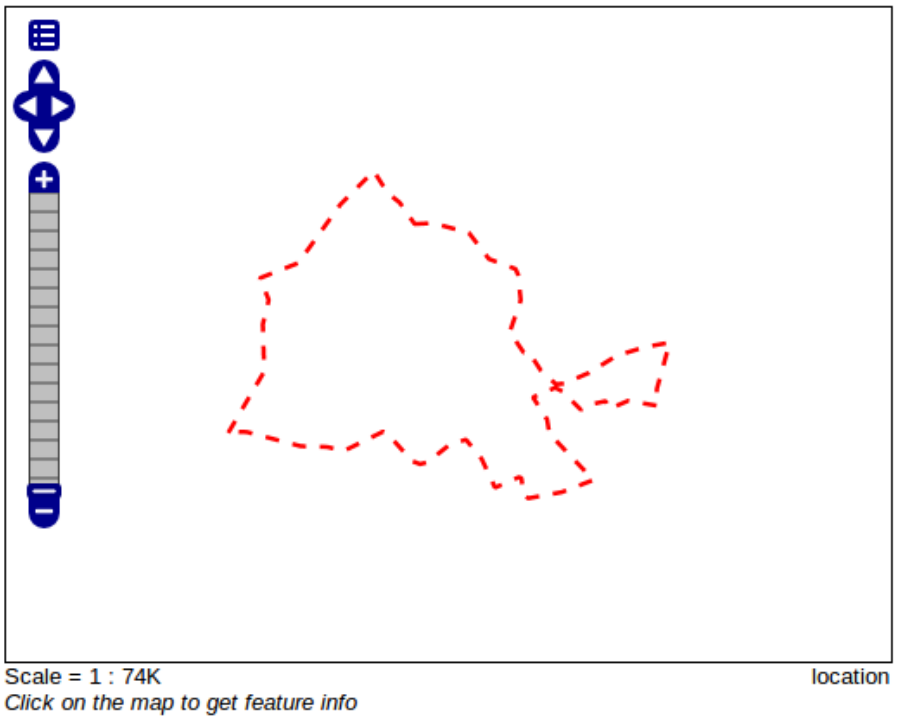


Figura 54: Previsualización de capa con estilo

10.6 Creación cliente de consulta básico web

En este punto se describirá la creación de un cliente web de consulta web básico. Este cliente de consulta será capaz de acceder a servicios de cartografía WMS. Está desarrollado empleando diferentes tecnologías como Ajax, HTML y JavaScript. Inicialmente se emplea JavaScript en diferentes librerías para construir el cliente, como ExtJS, OpenLayers y GeoExt.

10.6.1 Funcionalidad

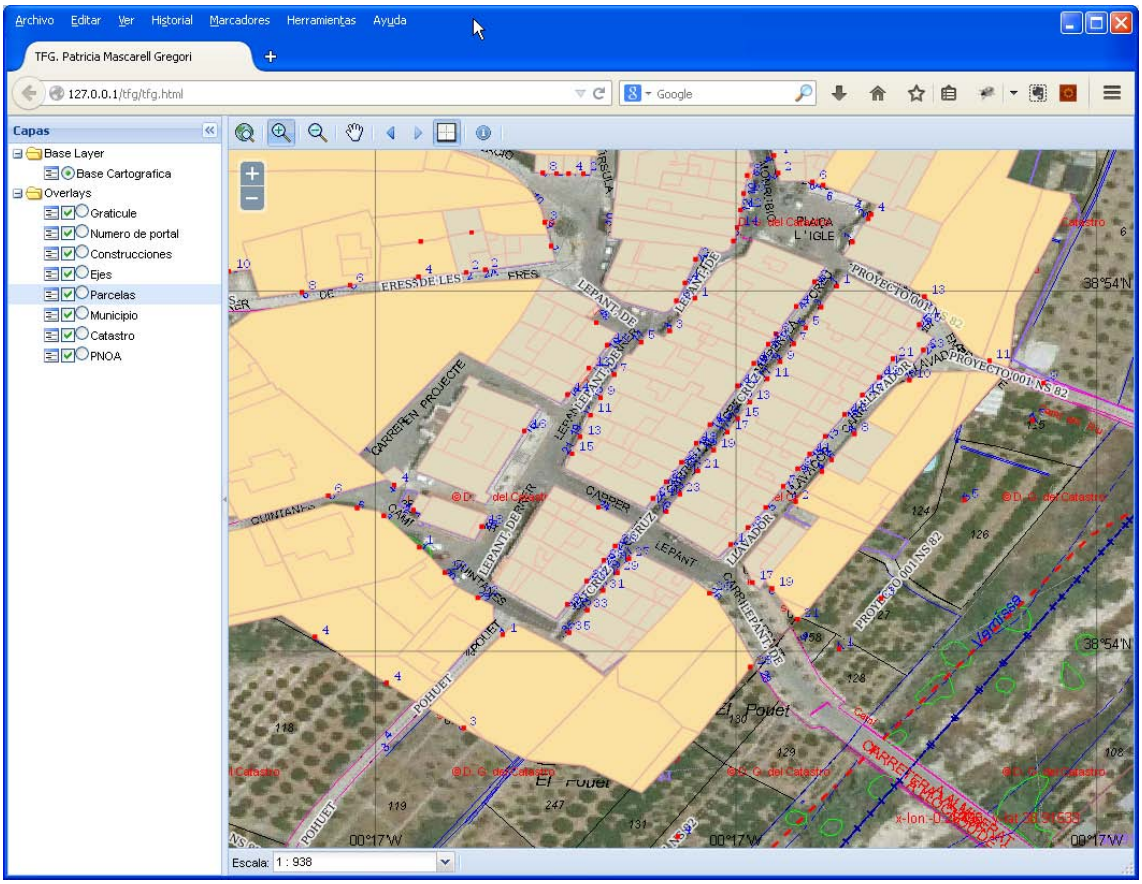


Figura 55: Aspecto general del geoportal WEB IDE

El visor básico contiene un conjunto de herramientas básico que permiten realizar la navegación por la cartografía e interrogar a la misma mediante las operaciones de GetFeatureInfo. Se trata de un cliente muy extensible que puede completarse con nuevas herramientas de consulta y búsqueda.

10.6.1.1 Áreas del cliente web

- Tabla de contenidos del mapa: Muestra las diferentes capas incluidas en la visualización del visor. Estas capas están definidas en el servicio WMS sobre el cual se inicia el visor. Sobre esta tabla de contenidos o TOC se puede consultar la capa activa sobre la cual solicitar información, consultar la leyenda de capa una de las capas incluidas.
- Posición: Panel de información de las coordenadas del puntero del ratón.
- Barra de herramientas: Barra que contiene diferentes herramientas que interactúan con el mapa.
- Mapa: Área de representación de la cartografía.

Todas las áreas del cliente web son redimensionables, permitiendo ajustar las dimensiones a las necesidades del usuario.

10.6.1.2 Herramientas disponibles

La barra de herramientas almacena las herramientas interactivas con el área del mapa. Esta barra de herramientas contiene herramientas de navegación y de información. Las herramientas de navegación son las herramientas típicas de navegación por la cartografía, que permiten ampliar, reducir y desplazarse por la misma.

	Zoom todo	Herramienta que realiza un zoom a la totalidad de la cartografía
	Zoom acercar	Herramienta que permite ampliar la imagen y aumentar su detalle. Esta herramienta funciona mediante un clic o mediante la creación de una ventana
	Zoom alejar	Herramienta que permite disminuir la imagen
	Encuadre	Herramienta que permite desplazar la cartografía sin alterar la escala
	Extensión previa	Permite volver a extensiones de zoom anteriores
	Extensión siguiente	Permite volver a extensiones de zoom anteriores
	Rejilla de coordenadas	Permite mostrar y ocultar una rejilla de coordenadas dinámica que se actualiza con la escala de visualización.
	Información	Muestra la información del elemento seleccionado. La información se muestra sobre plantillas de datos donde se definen los datos a mostrar para cada una de las capas.

Tabla 12: Herramientas de la IDE

10.6.1.3 Herramienta de información

La herramienta de información devuelve resultados de atributos de las capas en el punto pulsado sobre la cartografía. La herramienta de información muestra información de aquellas capas que se encuentren visibles en el mapa.

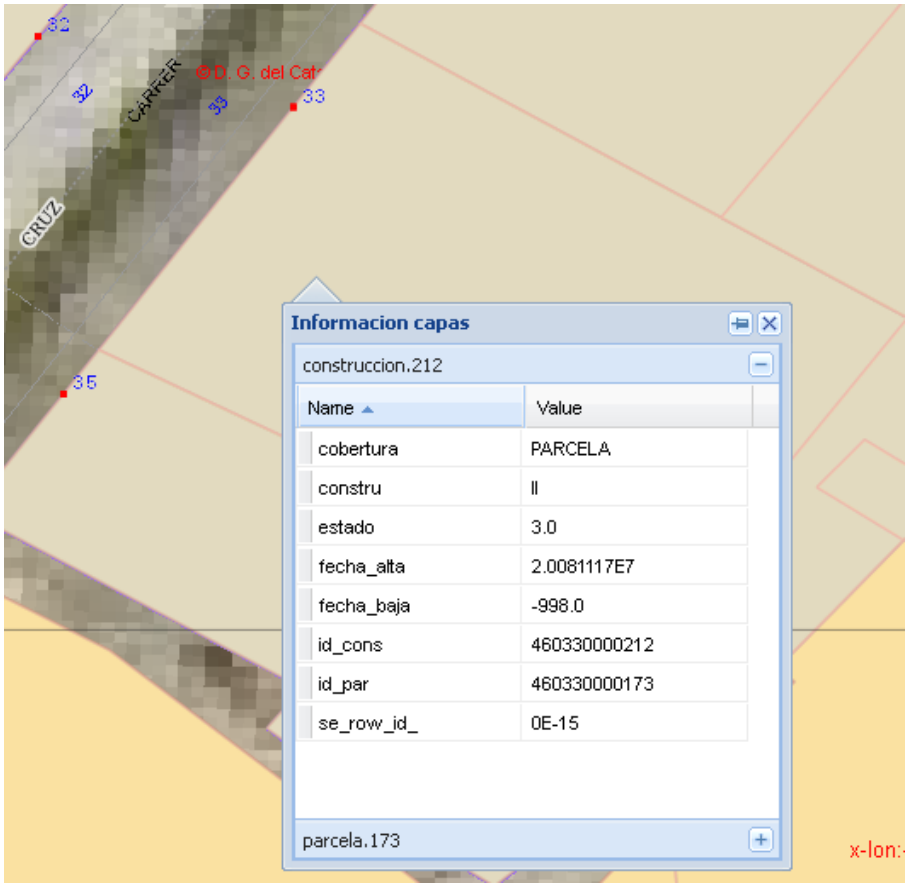


Figura 56: Resultado de la herramienta de información

Los datos de las capas se organizan en paneles extensibles en acordeón mostrando los campos de atributos de cada unas de las capas y sus valores.

10.6.2 Proceso de desarrollo del cliente WEB

Para el desarrollo del cliente web se emplea un cargador de librerías en formato HTML que inicializa las librerías y crea los objetos que componen la aplicación.

Este HTML inicial cargará las librerías de ExtJS, OpenLayers y GeoExt. Además se cargarán las hojas de estilo necesarias para la aplicación.

Tfg.html

```
<html xmlns="http://www.w3.org/1999/xhtml">
   <head>
      <title>TFG. Patricia Mascarell Gregori</title>
      <!--Libreria ExtJS-->
        <script type="text/javascript" src="ext-3.4.1/adapter/ext/ext-base.js"></script>
                <script type="text/javascript" src="ext-3.4.1/ext-all.js"></script>
                <link rel="stylesheet" type="text/css" href="ext-3.4.1/resources/css/ext-all.css" />
                <link rel="stylesheet" type="text/css" href="ext-3.4.1/examples/shared/examples.css" />

                <!--Libreria Openlayers-->
                <script src="OpenLayers-2.13.1/OpenLayers.js"></script>

                <!--Libreria Proj4-->
                <script type="text/javascript" src="proj4js-1.0.0/lib/proj4js-compressed.js"></script>

                <!--Libreria GeoExt-->
                <script type="text/javascript" src="GeoExt/script/GeoExt.js"></script>
                <link rel="stylesheet" type="text/css" href="GeoExt/resources/css/popup.css">

                <!--Librerias desarrolladas-->
                <script type="text/javascript" src="tfg.js"></script>
      <link rel="stylesheet" type="text/css" href="css/icons.css" />

        </head>
   <body>
                <div id="desc"></div>

   </body>
</html>
```

Una vez cargadas las librerías se procede a la creación de la aplicación con diferentes objetos que disponen de propiedades y funciones que permiten interactuar con ellos.

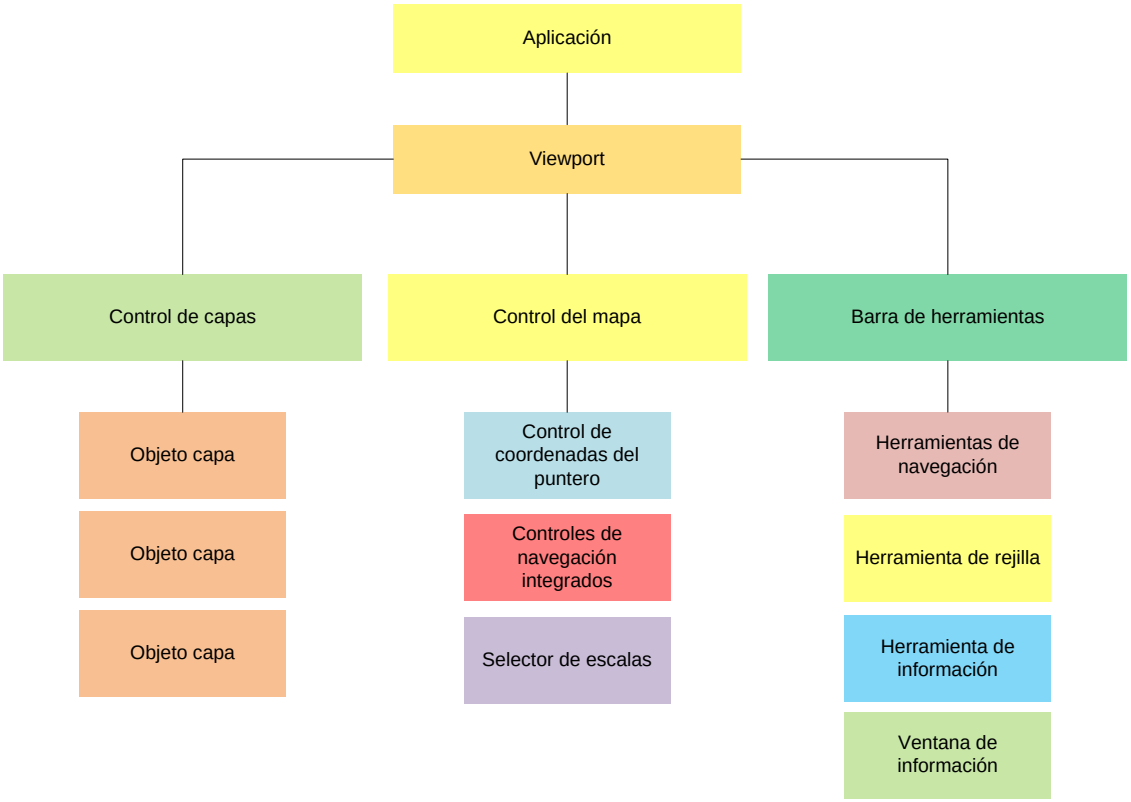


Figura 57: Esquema de componentes del Geoportal

Entre los objetos más destacables que se emplean:

- Mappanel: Objeto de GeoExt que contiene un mapa OpenLayers encapsulado en un panel de ExtJS, que permite la visualización de capas WMS y WFS. Dispone de múltiples propiedades y funciones para interactuar con el mapa.
    - La creación del mapa de OpenLayers necesita una serie de opciones que marcarán el comportamiento del mapa generado:

```
//Definición de las opciones para la creación del mapa.
mapOptions = {
                    units: 'm',
                    maxExtent: new OpenLayers.Bounds(-10, 35, 15, 45),
                    numZoomLevels: 24,
                    projection: new OpenLayers.Projection("EPSG:4326"),
                    displayProjection: new OpenLayers.Projection("EPSG:4326")
};
//Creación del mapa que será encapsulado en el mapPanel
 mapOpenLayers = new OpenLayers.Map(mapOptions);
```

    - El objeto MapPanel de GeoExt también dispone de una serie de parámetros en su creación que definen su comportamiento y como se presenta inicialmente, como por ejemplo las coordenadas donde se centrará al iniciarse, el mapa de OpenLayers que encapsula, los diferentes componentes de ExtJS que contendrá, ya que se trata de una

extensión de la clase Pannel de ExtJS, y las diferentes capas que cargará el mapa inicialmente en formato WMS o WFS.

```
mapPanel = new GeoExt.MapPanel({
    border: true,
    region: "center",
    map: mapOpenLayers,
    center: [-0.28788, 38.92495],
    zoom: 10,
    tbar: toolbarItems,
    bbar: ["Escala:",zoomSelector,'-'],
    layers: [
        new OpenLayers.Layer.WMS (
                "Base Cartografica",
                "http://localhost:8080/geoserver/tfg/wms",
                {layers: 'municipio', transparent: true, format: "image/png"},
                {isBaseLayer: true, visibility:false}
        ),
```

- La creación de capas de mapa de OpenLayers utiliza una serie de parámetros que son un estándar en los protocolos de acceso a mapas WMS y WFS.

```
new OpenLayers.Layer.WMS (
        "Nombre de la capa",
        "http://<url del servicio de mapas>:8080/geoserver/tfg/wms",
        {layers: 'nombre_capa_servidor',
         transparent: true, //Transparencia de la capa
         format: "image/png"},//Formato de la imagen de mapa pedida
        {isBaseLayer: false,  //Define si la capa es base cartográfica
        visibility:false} //Define el comportamiento inicial de la capa
),
```

- El objeto MapPanel de GeoExt también dispone de una serie de parámetros en su creación que definen su comportamiento y como se presenta inicialmente, como por ejemplo las coordenadas donde se centrará al iniciarse, y si dispondrá de otros objetos integrados como una barra de herramientas superior e inferior.

```
mapPanel = new GeoExt.MapPanel({
    border: true,
    region: "center",
    map: mapOpenLayers,
    center: [-0.28788, 38.92495],
    zoom: 10,
    tbar: toolbarItems,
    bbar: ["Escala:",zoomSelector,'-'],
    layers: [
```

- La barra de herramientas superior del objeto MapPanel contendrá herramientas que interaccionan con el mapa de OpenLayers para por ejemplo ampliar o reducir el mapa. Esta barra de herramientas es una clase propia de la librería GeoExt que se extiende de la clase Acticion de ExtJS y organizadas en un

Toolbar. En el siguiente fragmento se muestra la generación de dos herramientas, zoom extensión y zoom ampliar.

```
//Creación de la barra de herramientas
  toolbarItems = [];
  action = null;
  actions = {};

  action = new GeoExt.Action({
    control: new OpenLayers.Control.ZoomToMaxExtent(),
    map: mapOpenLayers,
    //text: "max extent",
    tooltip: 'Zoom to max extent',
              iconCls: 'zoomfull',
  });
  actions["max_extent"] = action;
  toolbarItems.push(action);
  toolbarItems.push("-");

  action = new GeoExt.Action({
    control: new OpenLayers.Control.ZoomBox(),
    map: mapOpenLayers,
    //text: "zoomin",
    tooltip: "zoom in",
    iconCls: 'zoomin',
    toggleGroup: 'map'
  });
  actions["zoomin"] = action;
  toolbarItems.push(action);
  toolbarItems.push("-");
```

- La herramienta de información es algo más peculiar puesto que es necesario configurar más parámetros para poder recibir la respuesta adecuada. Inicialmente se construye la herramienta y posteriormente se inserta la ToolBar del MapPanel.

```
action = new GeoExt.Action({
    control: new OpenLayers.Control.WMSGetFeatureInfo({
      url: 'http://192.168.32.4:8080/geoserver/tfg/wms',
      autoActivate:false,
                    infoFormat:"application/vnd.ogc.gml",
                    //maxFeatures:3, //Máximo de capas que se consultarán
      queryVisible: true, //Muestra información sólo de las capas visibles
      eventListeners: {
        "getfeatureinfo": function(e){
                                  var items = [];
                                  Ext.each(e.features, function (feature) {
                                        items.push({
                                              xtype:"propertygrid",
                                              title:feature.fid,
                                              source:feature.attributes
                                              });
                                        });
                                  new GeoExt.Popup({
                                        title:"Informacion capas",
                                        width: 300,
                                        height: 350,
                                        layout:"accordion",
                                        map:mapOpenLayers,
                                        location:e.xy,
                                        items:items
                                        }).show();
                            }
        }
      }),
      map: mapOpenLayers,
      //text: "info",
      tooltip: "info",
      iconCls: 'info',
      toggleGroup: 'map'
    });
    actions["info"] = action;
    toolbarItems.push(action);
    toolbarItems.push("-");
```

- LayerTree: Objeto de GeoExt que hereda del tree de ExtJS; proporciona el árbol de capas para construcción de la tabla de contenidos, en las cuales se extienden más propiedades para las capas de cartografía.

    - Creación de una entrada de capa para el árbol de capas y del objeto GeoExt.tree de GeoExt. Se indican el tipo de capas que formarán parte del árbol, capas base y capas superpuestas.

```
// Creación de un nodo de capa para el árbol
  var LayerNodeUI = Ext.extend(GeoExt.tree.LayerNodeUI, new
GeoExt.tree.TreeNodeUIEventMixin());

  // Creación del árbol de capas
  var treeConfig = [{
    nodeType: "gx_baselayercontainer"
  }, {
    nodeType: "gx_overlaylayercontainer",
    expanded: true,
    loader: {
      baseAttrs: {
        radioGroup: "foo",
        uiProvider: "layernodeui"
      }
    }
  }
      ];
  treeConfig = new OpenLayers.Format.JSON().write(treeConfig, true);
```

    - Creación del panel de árbol de capas y su comportamiento con plugins, eventos, etc.

```
// Creación del panel del árbol de capas
  tree = new Ext.tree.TreePanel({
    border: true,
    region: "west",
    title: "Capas",
    width: 200,
    split: true,
    collapsible: true,
    collapseMode: "mini",
    autoScroll: true,
    plugins: [
      new GeoExt.plugins.TreeNodeRadioButton({
        listeners: {
          "radiochange": function(node) {
            alert(node.text + " es la nueva capa activa.");
          }
        }
      })
    ],
    loader: new Ext.tree.TreeLoader({
      applyLoader: false,
      uiProviders: {
        "layernodeui": LayerNodeUI
      }
```

```
    }),
    root: {
      nodeType: "async",
      children: Ext.decode(treeConfig)
    },
    listeners: {
      "radiochange": function(node){
        alert(node.layer.name + " es la nueva capa activa.");
      }
    },
    rootVisible: false,
    lines: false
  });
```

- Viewport: Objeto de ExtJS que proporciona la distribución de paneles sobre la ventana de la aplicación. El resto de objetos se acoplan a las regiones y paneles proporcionados por este objeto.

```
app = new Ext.Viewport({
    layout: "fit",
    hideBorders: true,
    items:[
                {
      layout: "border",
      deferredRender: false,
      items: [mapPanel,tree]
    }]
  });
```

Tfg.js

```
/**
 * TFG: Patricia Mascarell Gregori
 *
 * ETSINF Escuela Técnica Superior de Informática
 * UPV.Universidad Politécnica de Valencia
 */

/**
 *  Geoportal WEB-IDE.
 */

var mapPanel, tree;

OpenLayers.ProxyHost = "../cgi-bin/proxy.cgi?url=";

Ext.onReady(function() {
    //Definición de las opciones para la creación del mapa.
        mapOptions = {
                        units: 'm',
                        maxExtent: new OpenLayers.Bounds(-10, 35, 15, 45),
                        numZoomLevels: 24,
                        projection: new OpenLayers.Projection("EPSG:4326"),
                        displayProjection: new OpenLayers.Projection("EPSG:4326")
        };
        //Creación del mapa que será encapsulado en el mapPanel
    mapOpenLayers = new OpenLayers.Map(mapOptions);

    //Creacion del combo de escala
    scaleStore = new GeoExt.data.ScaleStore({map: mapOpenLayers});
    zoomSelector = new Ext.form.ComboBox({
        id:'comboScale',
        hideLabel: false,
        fieldLabel: 'Escala',
        store: scaleStore,
        emptyText: "Zoom Level",
        tpl:            '<tpl            for="."><div            class="x-combo-list-item">1            :
{[parseInt(values.scale)]}</div></tpl>',
        editable: false,
        triggerAction: 'all',
        mode: 'local'
    });

        //Registrando el evento de selección del combo de escalas
    zoomSelector.on('select',
        function(combo, record, index) {
            mapOpenLayers.zoomTo(record.data.level);
        },
        this
    );

    mapOpenLayers.events.register('zoomend', this, function() {
        var scale = scaleStore.queryBy(function(record){
            return mapOpenLayers.getZoom() == record.data.level;
```

```
    });

    if (scale.length > 0) {
        scale = scale.items[0];
        zoomSelector.setValue("1 : " + parseInt(scale.data.scale));
    } else {
        if (!zoomSelector.rendered) return;
        zoomSelector.clearValue();
    }
});

//Creación de la barra de herramientas
toolbarItems = [];
action = null;
actions = {};

action = new GeoExt.Action({
    control: new OpenLayers.Control.ZoomToMaxExtent(),
    map: mapOpenLayers,
    //text: "max extent",
    tooltip: 'Zoom to max extent',
                iconCls: 'zoomfull',
});
actions["max_extent"] = action;
toolbarItems.push(action);
toolbarItems.push("-");

action = new GeoExt.Action({
    control: new OpenLayers.Control.ZoomBox(),
    map: mapOpenLayers,
    //text: "zoomin",
    tooltip: "zoom in",
    iconCls: 'zoomin',
    toggleGroup: 'map'
});
actions["zoomin"] = action;
toolbarItems.push(action);
toolbarItems.push("-");

action = new GeoExt.Action({
    control: new OpenLayers.Control.ZoomBox({out: true}),
    map: mapOpenLayers,
    //text: "zoomout",
    tooltip: "zoom out",
    iconCls: 'zoomout',
    toggleGroup: 'map'
});
actions["zoomin"] = action;
toolbarItems.push(action);
toolbarItems.push("-");

action = new GeoExt.Action({
    control: new OpenLayers.Control.DragPan({isDefault: true}),
    map: mapOpenLayers,
    //text: "pan",
```

```
    tooltip: "pan",
    iconCls: 'pan',
    toggleGroup: 'map'
});
actions["pan"] = action;
toolbarItems.push(action);
toolbarItems.push("-");

ctrl = new OpenLayers.Control.NavigationHistory();
mapOpenLayers.addControl(ctrl);

action = new GeoExt.Action({
    //text: "previous",
    control: ctrl.previous,
    disabled: true,
    tooltip: "previous in history",
                iconCls: 'back'
});
actions["previous"] = action;
toolbarItems.push(action);

action = new GeoExt.Action({
    //text: "next",
    control: ctrl.next,
    disabled: true,
    tooltip: "next in history",
                iconCls: 'next'
});
actions["next"] = action;
toolbarItems.push(action);

action = new GeoExt.Action({
    control: new OpenLayers.Control.Graticule({numPoints: 2, labelled: true}),
    map: mapOpenLayers,
    //text: "graticule",
    tooltip: "graticule",
    iconCls: 'graticule',
    toggleGroup: 'graticule'
});
actions["graticule"] = action;
toolbarItems.push(action);
toolbarItems.push("-");

       action = new GeoExt.Action({
    control: new OpenLayers.Control.WMSGetFeatureInfo({
        url: 'http://192.168.32.4:8080/geoserver/tfg/wms',
        autoActivate:false,
                        infoFormat:"application/vnd.ogc.gml",
                        //maxFeatures:3, //Máximo de capas que se consultarán
        queryVisible: true, //Muestra información sólo de las capas visibles
        eventListeners: {
            "getfeatureinfo": function(e){
                                        var items = [];
                                        Ext.each(e.features, function (feature) {
                                                items.push({
```

```
                                                xtype:"propertygrid",
                                                title:feature.fid,
                                                source:feature.attributes
                                                });
                                        });
                                new GeoExt.Popup({
                                        title:"Informacion capas",
                                        width: 300,
                                        height: 350,
                                        layout:"accordion",
                                        map:mapOpenLayers,
                                        location:e.xy,
                                        items:items
                                        }).show();
                                }
                        }
                }),
                map: mapOpenLayers,
                //text: "info",
                tooltip: "info",
                iconCls: 'info',
                toggleGroup: 'map'
        });
        actions["info"] = action;
        toolbarItems.push(action);
        toolbarItems.push("-");

        //Creacion del control de coordenadas del puntero del ratón
        action = new GeoExt.Action({
                        control: new OpenLayers.Control.MousePosition({prefix: 'x-lon:',separator: ',
y-lat:'}),
                map: mapOpenLayers,
                text: "coordenadas",
                tooltip: "coordenadas",
                iconCls: 'coordenadas',
                toggleGroup: 'coordenadas'
        });
        actions["coordenadas"] = action;

                //Esta herramienta no se añadirá a la barra de herramientas
                //toolbarItems.push(action);
        //toolbarItems.push("-");

        //Creacion del panel del mapa
        mapPanel = new GeoExt.MapPanel({
                border: true,
                region: "center",
                map: mapOpenLayers,
                center: [-0.28788, 38.92495],
                zoom: 10,
                tbar: toolbarItems,
                bbar: ["Escala:",zoomSelector,'-'],
                layers: [

                                new OpenLayers.Layer.WMS (
```

```
                        "Base Cartografica",
                        "http://192.168.32.4:8080/geoserver/tfg/wms",
                        //"http://localhost:8080/geoserver/tfg/wms",
                        {layers: 'municipio', transparent: true, format: "image/png"},
                        {isBaseLayer: true, visibility:false}
                ),

                //Preparando capa PNOA
                new OpenLayers.Layer.WMS (
                        "PNOA",
                        "http://www.idee.es/wms/PNOA/PNOA?",
                        {layers: 'pnoa', transparent: true, format: "image/png"},
                        {isBaseLayer: false, visibility:false}
                ),

                new OpenLayers.Layer.WMS(
                        "Catastro",
        "http://ovc.catastro.meh.es/Cartografia/WMS/ServidorWMS.aspx?",
                        {layers: 'Catastro', transparent: true, format: "image/png"},
                        {isBaseLayer: false, visibility:false}
        ),

                new OpenLayers.Layer.WMS (
                        "Municipio",
                        "http://192.168.32.4:8080/geoserver/tfg/wms",
                        //"http://localhost:8080/geoserver/tfg/wms",
                        {layers: 'municipio', transparent: true, format: "image/png"},
                        {isBaseLayer: false, visibility:false}
                ),

                new OpenLayers.Layer.WMS (
                        "Parcelas",
                        "http://192.168.32.4:8080/geoserver/tfg/wms",
                        //"http://localhost:8080/geoserver/tfg/wms",
                        {layers: 'parcela', transparent: true, format: "image/png"},
                        {isBaseLayer: false, visibility:false}
                ),

                new OpenLayers.Layer.WMS (
                        "Ejes",
                        "http://192.168.32.4:8080/geoserver/tfg/wms",
                        //"http://localhost:8080/geoserver/tfg/wms",
                        {layers: 'tramos', transparent: true, format: "image/png"},
                        {isBaseLayer: false, visibility:false}
                ),

                new OpenLayers.Layer.WMS (
                        "Construcciones",
                        "http://192.168.32.4:8080/geoserver/tfg/wms",
                        //"http://localhost:8080/geoserver/tfg/wms",
                        {layers: 'construccion', transparent: true, format:
"image/png"},
                        {isBaseLayer: false, visibility:false}
                ),
```

```
                        new OpenLayers.Layer.WMS (
                                "Numero de portal",
                                "http://192.168.32.4:8080/geoserver/tfg/wms",
                                //"http://localhost:8080/geoserver/tfg/wms",
                                {layers: 'portal_pk', transparent: true, format: "image/png"},
                                {isBaseLayer: false, visibility:false}
                        )
    ]
});

// Creación de un nodo de capa para el árbol
var LayerNodeUI = Ext.extend(GeoExt.tree.LayerNodeUI, new
GeoExt.tree.TreeNodeUIEventMixin());

// Creación del árbol de capas
var treeConfig = [{
    nodeType: "gx_baselayercontainer"
}, {
    nodeType: "gx_overlaylayercontainer",
    expanded: true,
    loader: {
        baseAttrs: {
            radioGroup: "foo",
            uiProvider: "layernodeui"
        }
    }
}
      ];
treeConfig = new OpenLayers.Format.JSON().write(treeConfig, true);

// Creación del panel del árbol de capas
tree = new Ext.tree.TreePanel({
    border: true,
    region: "west",
    title: "Capas",
    width: 200,
    split: true,
    collapsible: true,
    collapseMode: "mini",
    autoScroll: true,
    plugins: [
        new GeoExt.plugins.TreeNodeRadioButton({
            listeners: {
                "radiochange": function(node) {
                    alert(node.text + " es la nueva capa activa.");
                }
            }
        })
    ],
    loader: new Ext.tree.TreeLoader({
        applyLoader: false,
        uiProviders: {
            "layernodeui": LayerNodeUI
        }
    }),
```

```
        root: {
            nodeType: "async",
            children: Ext.decode(treeConfig)
        },
        listeners: {
            "radiochange": function(node){
                alert(node.layer.name + " es la nueva capa activa.");
            }
        },
        rootVisible: false,
        lines: false
    });

    app = new Ext.Viewport({
        layout: "fit",
        hideBorders: true,
        items:[
                    {
            layout: "border",
            deferredRender: false,
            items: [mapPanel,tree]
        }]
    });
});
```

10.6.3 Estructura de carpeta del cliente WEB.

El cliente web desarrollado utiliza varias librerías de mapas y de desarrollo de componentes. Estas librerías están estructuradas en la carpeta de publicación del cliente web de la siguiente forma:

- Librería ExtJS. Almacenada en la carpeta extjs-3.4.1
- Librería OpenLayers. Almacenada en la carpeta OpenLayers-2.13.1
- Librería Proj4. Almacenada en la carpeta Proj4js-1.0.0
- Librería GeoExt. Almacenada en la carpeta GeoExt.
- Imágenes usadas en la aplicación. Carpeta imágenes
- Hojas de estilo. Carpeta css
- Tfg.html. Fichero HTML lanzador de la aplicación.
- Tfg.js. Aplicación JavaScript

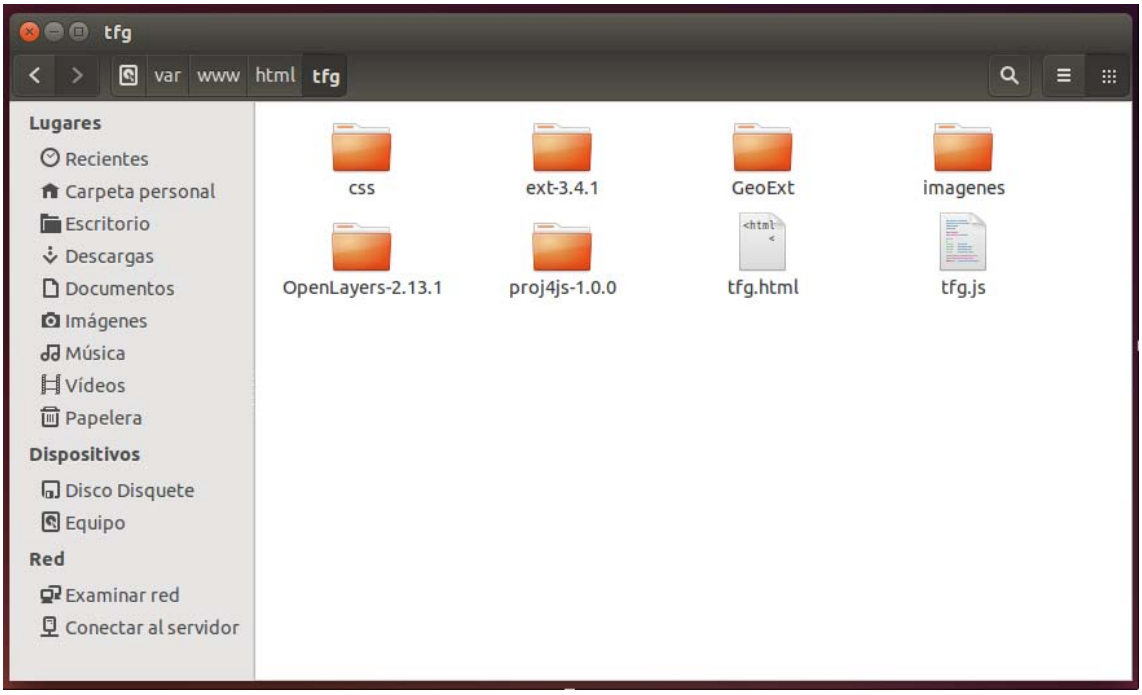


Figura 58: Estructura de carpetas del Geoportal WEB

11 Planificación y presupuesto

11.1 Planificación

Para la elaboración de la planificación del desarrollo del Geoportal WEB se han tenido en cuenta las diferentes fases que intervienen en la creación de un producto de software, aunque algunas de ellas son personalizadas.

Una vez generadas las fases generales del proyecto se especifican las tareas que son necesarias realizar en cada una de ellas, estimando una duración, el perfil usado y el número de recursos necesarios. Para la estimación del coste se ha empleado únicamente una unidad de cada uno de los recursos.

11.1.1 Fases que componen el proyecto

De forma continua se ha planificado una tarea que tiene una duración igual a la duración total del proyecto, y que es desempeñada por un perfil de Director de Proyecto, que será el encargado de la supervisión de todas las tareas a realizar a lo largo del proyecto.

El resto de fases se componen de las tareas necesarias para cumplir los objetivos de la fase y que serán desarrolladas a continuación:

Fase	Duración
Dirección y supervisión del proyecto	40 horas
Toma de requisitos y análisis	24 horas
Bases de datos y tratamiento de datos	72 horas
Diseño de aplicaciones	40 horas
Desarrollo	160 horas
Publicación de servicios de cartografía	48 horas
Pruebas y validación	24 horas
Implantación	32 horas

Tabla 13: Fases comprendidas en el proyecto

Con el cálculo de estas fases se llega a la conclusión final que el proyecto necesita 440 horas para su realización, de diferentes tipos de recursos.

11.1.2 Calendario del proyecto

El calendario del proyecto marca la fecha inicial de comienzo del proyecto y la fecha estimada de finalización. Se estima una fecha de inicio 01/04/2014 y una fecha de finalización de 09/06/2014, con lo que transcurren 50 días desde el inicio y la finalización.

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**TFG. Geoportal WEB**	**50 días**	**mar 01/04/14**	**lun 09/06/14**
2	**Toma de requisitos y análisis**	**3 días**	**mar 01/04/14**	**jue 03/04/14**
3	Recopilación y análisis de requisitos	3 días	mar 01/04/14	jue 03/04/14
4	**Bases de datos y tratamiento de datos**	**9 días**	**vie 04/04/14**	**mié 16/04/14**
5	Análisis de datos	3 días	vie 04/04/14	mar 08/04/14
6	Diseño del modelo de datos	3 días	mié 09/04/14	vie 11/04/14
7	Tratamiento de datos	3 días	lun 14/04/14	mié 16/04/14
8	**Diseño de aplicaciones**	**5 días**	**jue 17/04/14**	**mié 23/04/14**
9	Diseño Geoportal WEB	5 días	jue 17/04/14	mié 23/04/14
10	**Desarrollo**	**20 días**	**jue 24/04/14**	**mié 21/05/14**
11	**Geoportal WEB**	**20 días**	**jue 24/04/14**	**mié 21/05/14**
12	Interfaz gráfica y disposición de paneles	3 días	jue 24/04/14	lun 28/04/14
13	Control de mapa	2 días	mar 29/04/14	mié 30/04/14
14	Controles de escala	2 días	jue 01/05/14	vie 02/05/14
15	Barra de herramientas	2 días	lun 05/05/14	mar 06/05/14
16	Tabla de contenidos y propiedades de capas.	2 días	mié 07/05/14	jue 08/05/14
17	Barra de estado	2 días	vie 09/05/14	lun 12/05/14
18	Herramientas de navegación en el mapa	2 días	mar 13/05/14	mié 14/05/14
19	Herramienta de información	5 días	jue 15/05/14	mié 21/05/14
20	**Publicación de servicios de cartografía**	**6 días**	**jue 22/05/14**	**jue 29/05/14**
21	Tratamiento de datos	3 días	jue 22/05/14	lun 26/05/14
22	Generación de servicios	3 días	mar 27/05/14	jue 29/05/14
23	**Pruebas y validación**	**3 días**	**vie 30/05/14**	**mar 03/06/14**
24	Pruebas Geoportal WEB	3 días	vie 30/05/14	mar 03/06/14
25	**Implantación**	**4 días**	**mié 04/06/14**	**lun 09/06/14**
26	Instalación de herramientas de servidor	1 día	mié 04/06/14	mié 04/06/14
27	Despliegue de servicios de cartografía	2 días	jue 05/06/14	vie 06/06/14
28	Despliegue del Geoprotal WEB	1 día	lun 09/06/14	lun 09/06/14

Figura 59: Diagrama de Gantt

11.2 Presupuesto

El presupuesto describe el coste a realizar en el desarrollo e implantación del Geoportal WEB objeto de este trabajo final de grado.

En este presupuesto se recogen únicamente las dedicaciones de los perfiles y recursos que se estima son necesarios para su desarrollo, no se tiene en cuenta ningún otro tipo de gasto como pueda ser la adquisición de equipos, tarifas de telecomunicaciones, etc.

Así para desglosar el presupuesto se parte de las horas de dedicación por perfiles y el precio medio estimado para cada uno de ellos.

Recurso	€/hora
Director proyecto	55.00 €/hora
Analista programador	50.00 €/hora
Programador	30.00 €/hora
Técnico de pruebas	25.00 €/hora

Tabla 14: Presupuesto

11.2.1 Dedicación

En la dedicación se van a desglosar cada una de las tareas realizadas y el perfil del recurso que las realiza, con el fin de obtener un presupuesto final.

Perfil	Tarea	Horas
Director proyecto	Dirección y supervisión del proyecto	**40 horas**
Analista programador		**184 horas**
	Tratamiento de datos	24 horas
	Generación de servicios	24 horas
	Recopilación y análisis de requisitos	24 horas
	Diseño Geoportal WEB	40 horas
	Análisis de datos	24 horas
	Diseño del modelo de datos	24 horas
	Tratamiento de datos	24 horas
Programador		**160 horas**
	Interfaz gráfica y disposición de paneles	24 horas
	Control de mapa	16 horas
	Controles de escala	16 horas
	Barra de herramientas	16 horas
	Tabla de contenidos y propiedades de capas. Gestión de capas	16 horas
	Barra de estado	16 horas
	Herramientas de navegación en el mapa	16 horas
	Herramienta de información	40 horas
Técnico de pruebas		**56 horas**
	Pruebas Geoportal WEB	24 horas
	Instalación de herramientas de servidor	8 horas
	Despliegue de servicios de cartografía	16 horas
	Despliegue del Geoportal WEB	8 horas

Tabla 15: Dedicación

11.2.2 Cálculo del presupuesto

Para el cálculo del presupuesto como se ha comentado anteriormente, se empleará sólo la dedicación en horas de los perfiles que desempeñan tareas. Así tras el cálculo de las tablas que contienen las dedicaciones en horas por perfil y la tabla de precios unitarios por hora de perfil se obtiene un coste total estimado del proyecto de: 17600€.

12 Conclusiones

En cuanto a los objetivos marcados al inicio de este trabajo final de grado, se puede afirmar que se han cumplido todos y cada uno de ellos:

- Se ha definido claramente el concepto de IDE (Infraestructura de Datos Espacial) para que el lector conozca los componentes, niveles y usuarios que participan en ella. Y que a su vez, se entienda el concepto de IDE como la mejor forma de trabajar con información cartográfica y espacial de forma distribuida.
- Se han descrito adecuadamente las políticas y acuerdos internacionales que han impulsado a la investigación, el desarrollo y la estandarización en el desarrollo y uso de IDEs. Pues, el gran éxito de las Infraestructuras de Datos Espaciales reside en el lenguaje propio de comunicación que tienen gracias a los estándares y la interoperabilidad.
- Se han citado las fuentes de las qué se han tomado las capas de información cartográfica utilizadas en el presente trabajo.
- Se ha analizado, estudiado y definido las arquitecturas física y lógica necesarias para sustentar el trabajo realizado. Este análisis es muy importante, y determinará el éxito y buen funcionamiento de la herramienta Geoportal WEB.
- Se han descrito los servicios de mapas utilizados, las tecnologías y las herramientas necesarias para la creación de un Geoportal WEB IDE que sea capaz de mostrar cartografía y consultar datos sobre ella.
- Por último, se ha realizado un estudio a partir de las fases de desarrollo y los recursos disponibles y con ello se ha obtenido la planificación y el presupuesto necesarios para llevar a cabo este proyecto.

El hecho de que hoy en día existan tantas herramientas software y frameworks, que permitan trabajar de acuerdo a los estándares y normativas definidas por INSPIRE, ISO, OGC, W3C, etc., hace que resulte relativamente fácil encontrar información suficiente para la investigación y desarrollo de un Geoportal WEB IDE.

Por iniciativa propia, y dado los recursos destinados a la elaboración del presente trabajo, he tomado la decisión de desarrollar el Geoportal WEB IDE para este estudio utilizado como base un sistema operativo libre de licencias, tal como es Ubuntu Desktop, y herramientas de software libre que se han descrito en el presente documento. La principal ventaja de utilizar tecnología sin licencias es el ahorro que supone no tener que pagar el costo de las mismas. Esto se traduce en la posibilidad de invertir ese presupuesto en desarrollar productos a medida, es decir, productos más adaptados a las necesidades del cliente. Y por otra parte, destinar parte de este presupuesto a la revisión y pruebas del producto para alcanzar mayor calidad.

Por último, queda mencionar que este proyecto puede ser la base para la implantación de herramientas de consulta geográfica en cualquier municipio o empresa. También cabría la posibilidad de extender más herramientas y ampliar su desarrollo, adaptándolo a las necesidades de cualquier empresa o administración pública que precise de este tipo de herramientas para la gestión de sus datos espaciales. O incluso hacer uso de datos espaciales públicos para apoyar la información de los datos propios de la empresa o institución y así, realizar estudios estadísticos, estratégicos, etc.

13 Glosario

En este glosario se recogen los términos y abreviaturas más empleadas en el contenido de este documento.

SIG / GIS	Sistema de Información Geográfica
IDE	Infraestructura de Datos Espaciales
IDEE	Infraestructura de Datos Espaciales de España
INSPIRE	INfraestructure for SPatial InfoRmation in Europe
OGC	Open Geospatial Consortium
J2EE	Estándar Java2 Enterprise Edition
HTTP	Hypertext Transfer Protocol
NEM	Núcleo Español de Metadatos
WMS	Web Map Service
WFS	Web Feature Service
WCS	Web Coverage Service
WCAS	Web Catalog Service
WFS-G	Web Gazetteer Service
WCTS	Web Coordinate Transformation Service

Tabla 16: Glosario

14 Anexos

Como anexo se detallará el proceso de puesta a punto del servidor realizando la instalación y configuración del software necesario para que la herramienta WEB IDE funcione correctamente.

El sistema operativo empleado es Ubuntu Desktop 14.04 LTS. Ubuntu es un potente sistema operativo libre de licencia y está mantenido por Canonical, que utiliza el núcleo de LINUX y está basado en Debian.

Las versiones estables de Ubuntu se liberan cada 6 meses y Canonical proporciona soporte técnico y actualizaciones de seguridad para la mayoría de las versiones de Ubuntu durante 18 meses, excepto para las versiones LTS (del inglés Long Term Support). Para las versiones LTS de escritorio y servidor Canonical ofrece 5 años, a partir de la fecha del lanzamiento. Actualmente, la versión más reciente de Ubuntu Server LTS es la 14.04 y es la versión de sistema operativo que se ha utilizado.

Aunque Ubuntu Desktop tiene interfaz gráfica, la instalación de todos los componentes se ha realizado usando el modo consola. Puede que en un principio el modo consola sea menos amigable, pero es una forma de realizar la instalación más rápida y eficaz.

En realidad, la mejor opción sería utilizar un sistema operativo Ubuntu Server pues, al no tener interfaz gráfica no se consumirían los recursos innecesariamente en proporcionar rendimiento a los gráficos de la interfaz, y todo el rendimiento de la máquina estaría destinado al cálculo y uso de las herramientas instaladas. Dado que este trabajo de final de grado es una demostración y se desea mostrar en la misma máquina las consultas que puede realizar el usuario administrador de la web IDE, finalmente la solución empleada ha sido Ubuntu Desktop.

14.1 Java 6

La instalación de Java se puede realizar como usuario root con la siguiente instrucción:

>apt-get install openjdk-6-jdk

Y se debe confirmar marcando 'y' para que el proceso de instalación continúe.

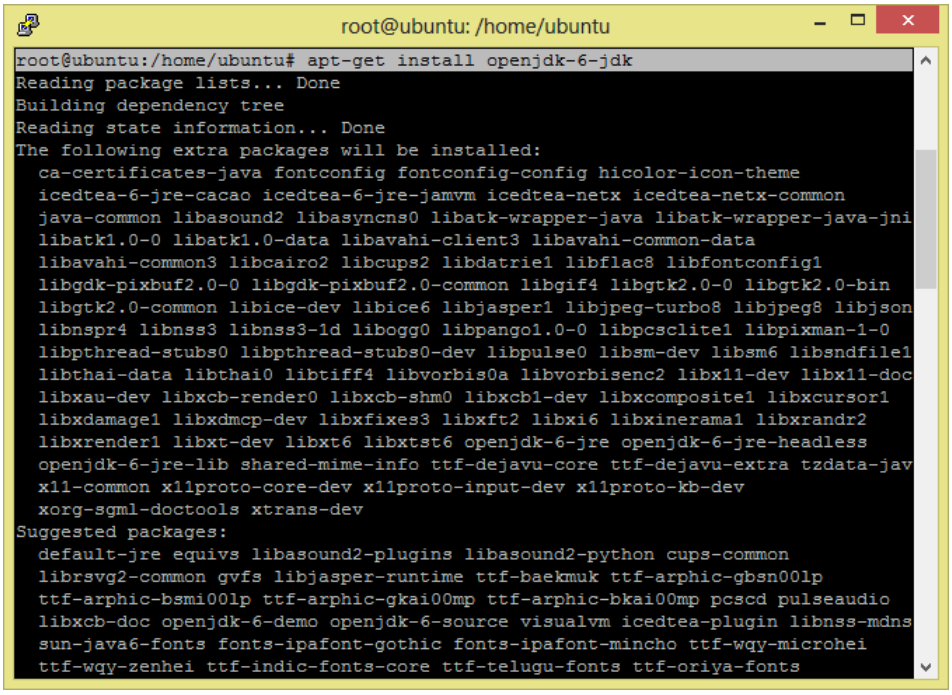


Figura 60: Instalación de Java

14.2 Apache 2

La instalación del servidor http Apache, se debe realizar estando logueado como usuario administrador root. Recordamos que para estar como usuario root se deberá escribir en consola la siguiente orden:

>sudo su

>password_root

Dónde 'password_root' es la contraseña que se ha determinado a la cuenta de usuario administrador del sistema en el momento de realizar la instalación del sistema operativo.

Una vez recordado este punto, y estando el usuario como root, se podrá seguir con la instalación del software necesario.

La instalación empezará cuando el usuario ejecute la orden de instalación y confirme la orden marcando 'y':

>apt-get install apache2

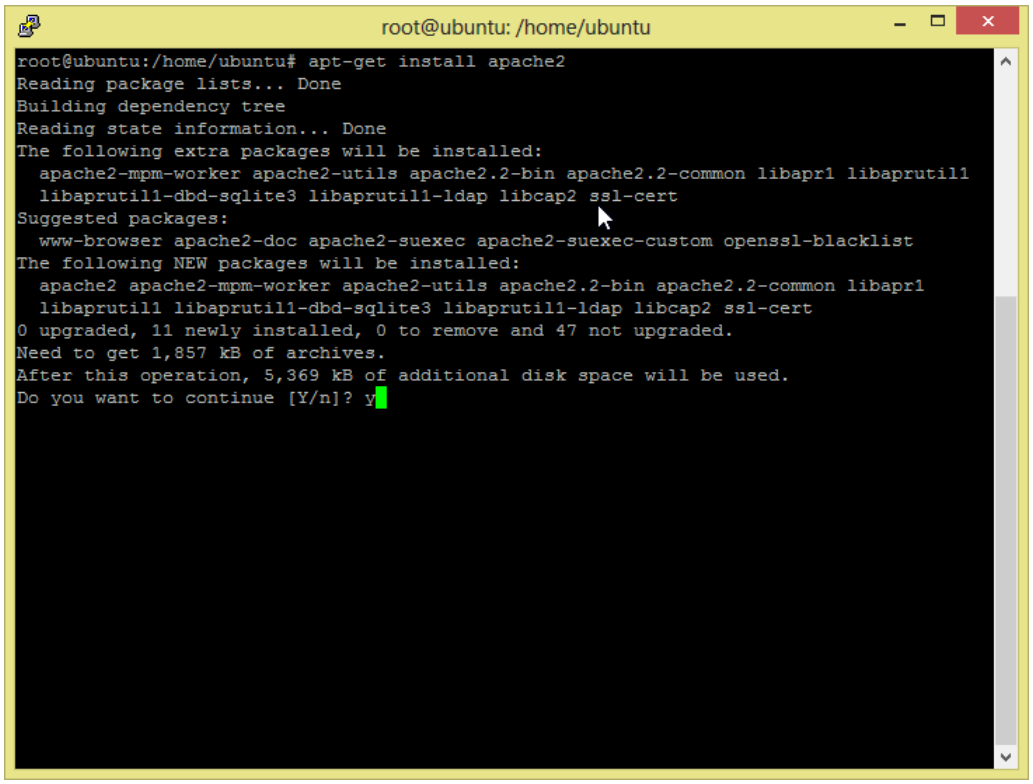


Figura 61: Instalación de Apache 2

Una vez instalado Apache, se podrá encontrar los ficheros de configuración en el siguiente directorio:

>cd /etc/apache2

El fichero de configuración de Apache será en Ubuntu el fichero apache2.conf, que está ubicado dentro de este directorio. El fichero httpd.conf, que en otros sistemas suele ser el fichero de configuración, estará vacío. Entonces, para que Apache use como ServerName predeterminado localhost, se deberá editar el fichero httpd.conf con nano

>nano /etc/apache2/httpd.conf

Y añadir la siguiente línea

ServerName localhost

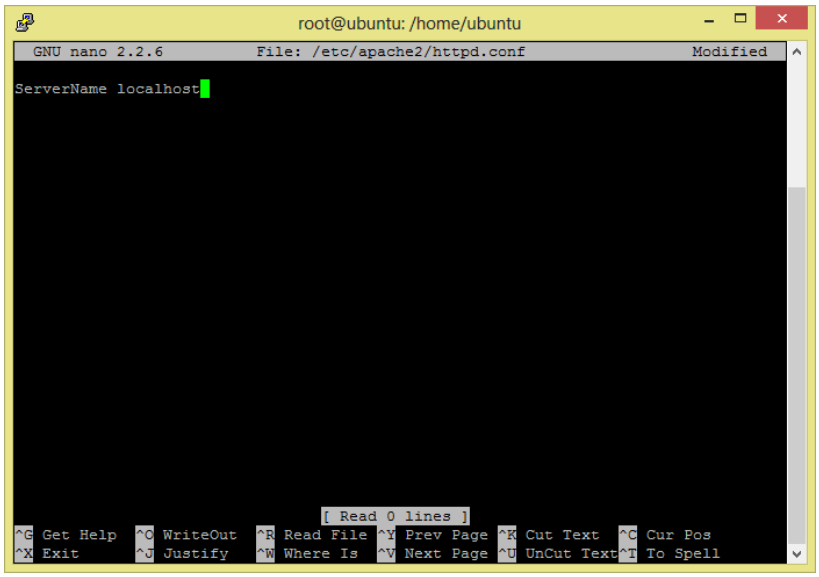


Figura 62: Configuración del ServerName

Una vez finalizados los cambios se guardará el documento (ctrl+o) y se cerrará (ctrl+x).

14.3 Python

Se requerirá realizar la instalación de python en esta máquina. Para proceder a su instalación se deberá ejecutar la siguiente orden:

>apt-get install python-dev

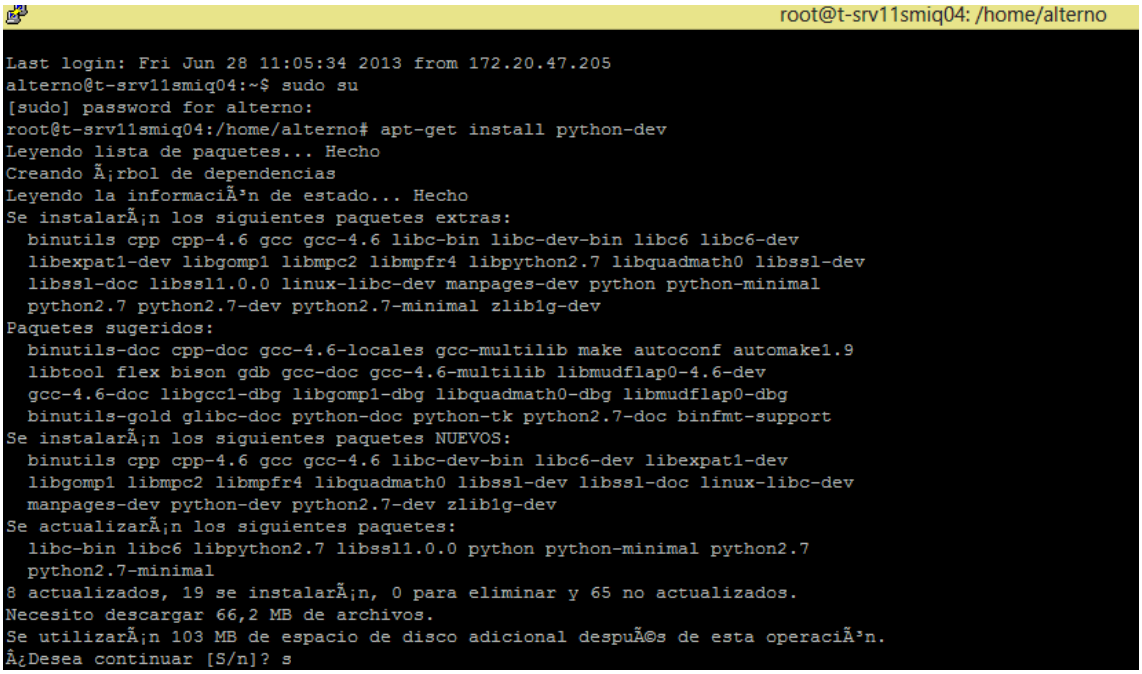


Figura 63: Instalación de Python

14.4 Proxy.cgi

En esta máquina también se requerirá la instalación del fichero proxy.cgi

Este fichero se puede obtener de la web

>wget http://trac.osgeo.org/openlayers/browser/trunk/openlayers/examples/proxy.cgi?format=txt

Y se deberá primero copiar a la carpeta de usuario en /home y desde allí se deberá mover a su ubicación final

>mv /home/alterno/proxy.cgi /usr/lib/cgi-bin/.

Para configurar este proxy.cgi se deberá añadir las url que se desean permitir dentro de la lista de "allowedHosts". Simplemente habrá que editar el fichero ejecutando nano y añadir las urls pertinentes.

14.5 Apache Tomcat 6

La instalación del servidor de aplicaciones Apache Tomcat se debe realizar cómo usuario root, empleando la siguiente instrucción:

>apt-get install tomcat6 tomcat6-admin tomcat6-common tomcat6-user

En pantalla el sistema solicitará una confirmación por parte del usuario root para continuar el proceso de instalación.

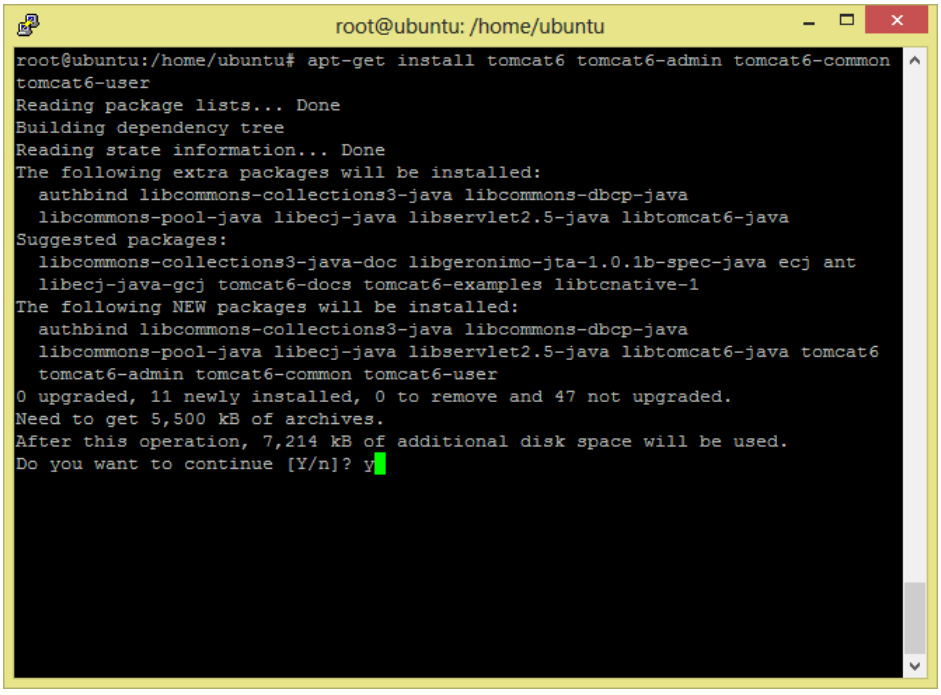


Figura 64: Instalación de Apache Tomcat

Lo siguiente que se deberá realizar al respecto es configurar las claves y roles del usuario administrador de Apache Tomcat.

Para configurar las claves de administración y roles de administrador Tomcat, habrá que editar el fichero tomcat-users.xml, siempre como usuario root.

>nano /etc/tomcat6/tomcat-users.xml

Y se deberá añadir el siguiente código:

<role rolename=”manager” />

<role rolename=”admin” />

<user username=”admin” password=”admin” roles=”admin, manager” />

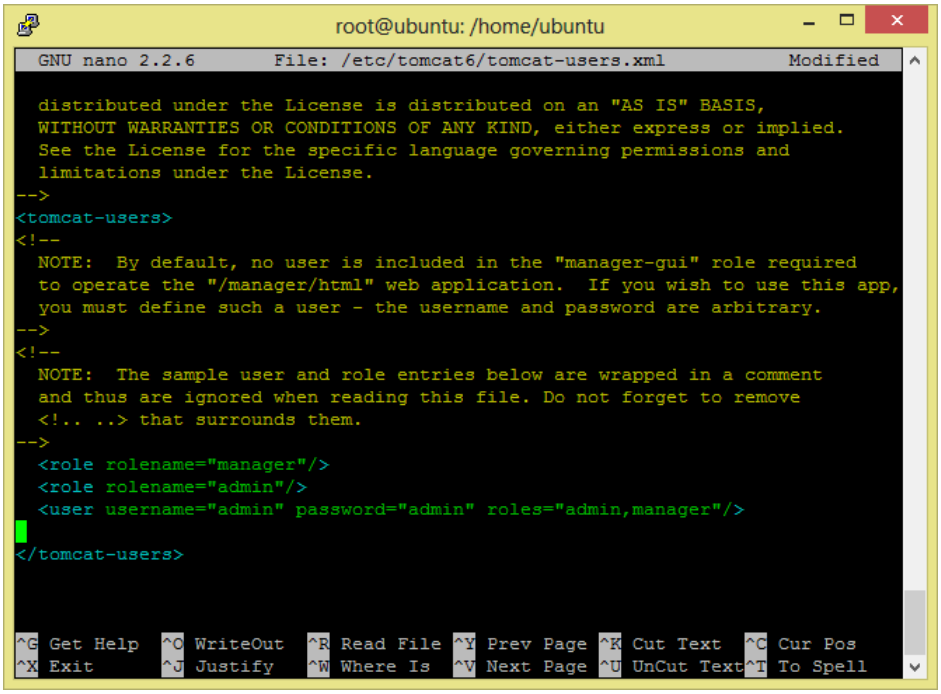


Figura 65: Configuración de los roles de acceso y usuarios Tomcat

En el ejemplo, username y password tienen asignado el valor “admin”. Este valor es el que normalmente se asigna para los ejemplos pero se recomienda que el administrador determine un username y password adecuados y seguros. Después de asignar los valores deseados, se guardarán los cambios en el fichero (ctrl+o) y se cerrará la edición del documento (ctrl+x).

Por último, y para que los cambios sean efectivos, se deberá reiniciar el servicio de Apache Tomcat.

>/etc/init.d/tomcat6 restart

14.6 Geoserver

Para realizar la instalación de Geoserver, lo primero que se deberá hacer es ir a la página web oficial de Geoserver y descargar el archivo WAR correspondiente a la versión que se desea utilizar. En la web, se dispondrá de la versión estable más reciente, así como de todas las versiones estables anteriores.

La versión que se va a utilizar en este caso es la 2.5.1 pero puede que en otro momento se requiera descargar otra versión, por ello se aconseja visitar primero la web desde un entorno gráfico y consultar cuál es la URL a la versión del fichero de instalación WAR que interesa.

Para la versión 2.5.1 de Geoserver, la URL para descargar el WAR es la siguiente:

http://downloads.sourceforge.net/geoserver/geoserver-2.5.1-war.zip

Hay que recordar que para otra versión de Geoserver que se desee utilizar esta URL no será válida. Se tendrá que consultar previamente desde la web de descargas de Geoserver.

http://geoserver.org/display/GEOS/Download

Situarse sobre la versión WAR elegida y copiar esa URL.

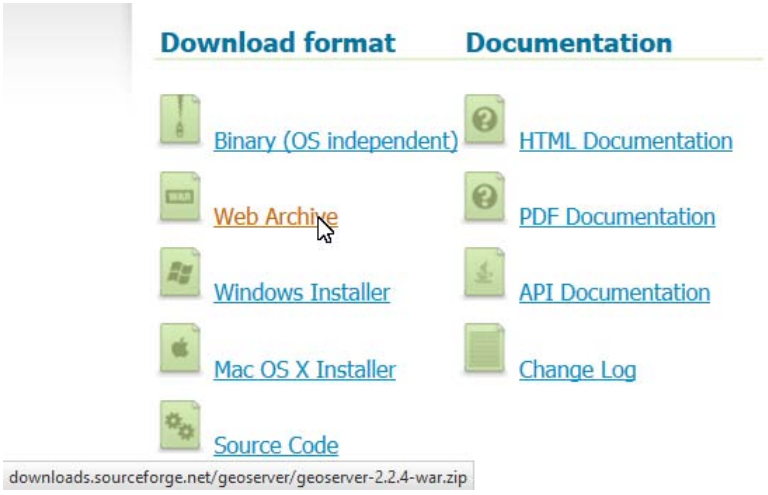


Figura 66: Selección de archivo WAR para instalar Geoserver

Una vez se tiene esta URL ya podremos descargar el archivo WAR para hacer su instalación.

Para poder realizar uno de los pasos que se va a requerir más adelante, se necesitará instalar la utilidad unzip, que permitirá descomprimir archivos comprimidos.

>apt-get install unzip

Después de haber instalado la utilidad unzip, se proseguirá a la instalación de Geoserver.

En primer lugar, se deberá descargar el archivo WAR en la carpeta de usuario situada en home, puesto que por permisos puede que no podamos descargar en otra carpeta.

>cd /home/ubuntu

>wget http://downloads.sourceforge.net/geoserver/geoserver-2.5.1-war.zip

Y una vez esté descargado el archivo, se deberá descomprimir con la siguiente orden

>unzip geoserver-2.5.1-war.zip

Esta orden descomprimirá el fichero zip donde estará el archivo WAR que se utilizará para la instalación de Geoserver.

Lo siguiente que se debe hacer es mover el archivo WAR al directorio de Apache Tomcat en dónde, tras ser reiniciado el servicio, desplegará Geoserver y lo instalará.

>mv /home/ubuntu/geoserver.war /var/lib/tomcat6/webapps/.

Una vez movido geoserver.war al directorio correspondiente, se deberá reiniciar el servicio de Apache Tomcat.

>/etc/init.d/tomcat6 restart

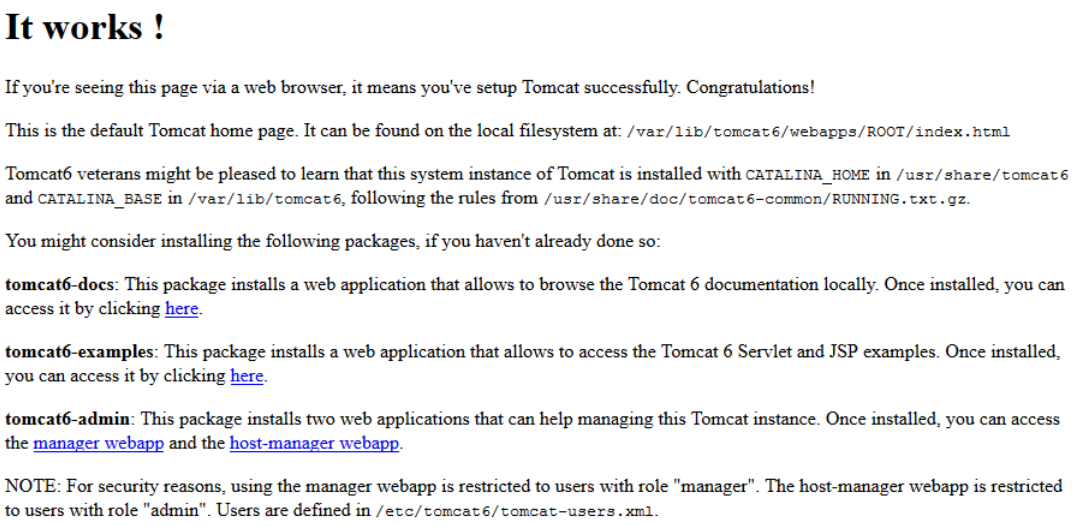
Otra forma de realizar la instalación es desde un navegador con entorno gráfico. La forma de hacerlo sería desde la interfaz del Manager de Apache Tomcat.

Se deberá acceder desde cualquier máquina con acceso a Internet e interfaz gráfica. Se abrirá un navegador de internet y se accederá por IP:8080 o nombreDeDominio:8080.

Por ejemplo, si la máquina dónde está el Apache Tomcat dónde queremos acceder tiene la IP 1.2.3.4 la url a la que se deberá acceder será la siguiente:

http://1.2.3.4:8080

Una vez se accede a esta URL aparecerá en pantalla la siguiente interfaz de Tomcat.



It works !

If you're seeing this page via a web browser, it means you've setup Tomcat successfully. Congratulations!

This is the default Tomcat home page. It can be found on the local filesystem at: `/var/lib/tomcat6/webapps/ROOT/index.html`

Tomcat6 veterans might be pleased to learn that this system instance of Tomcat is installed with `CATALINA_HOME` in `/usr/share/tomcat6` and `CATALINA_BASE` in `/var/lib/tomcat6`, following the rules from `/usr/share/doc/tomcat6-common/RUNNING.txt.gz`.

You might consider installing the following packages, if you haven't already done so:

tomcat6-docs: This package installs a web application that allows to browse the Tomcat 6 documentation locally. Once installed, you can access it by clicking here.

tomcat6-examples: This package installs a web application that allows to access the Tomcat 6 Servlet and JSP examples. Once installed, you can access it by clicking here.

tomcat6-admin: This package installs two web applications that can help managing this Tomcat instance. Once installed, you can access the manager webapp and the host-manager webapp.

NOTE: For security reasons, using the manager webapp is restricted to users with role "manager". The host-manager webapp is restricted to users with role "admin". Users are defined in `/etc/tomcat6/tomcat-users.xml`.

Figura 67: Interfaz Tomcat

Al intentar acceder al manager webapps, se solicitarán las claves de usuario administrador de Tomcat (las claves que se registraron en la instalación).

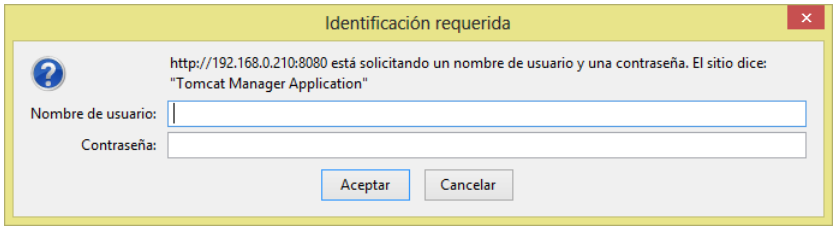


Figura 68: Identificación de usuario Tomcat

Una vez el usuario administrador esté validado, aparecerá otra ventana interfaz de administración de todas las aplicaciones desplegadas en Apache Tomcat. La interfaz será como la que se puede apreciar en la imagen.

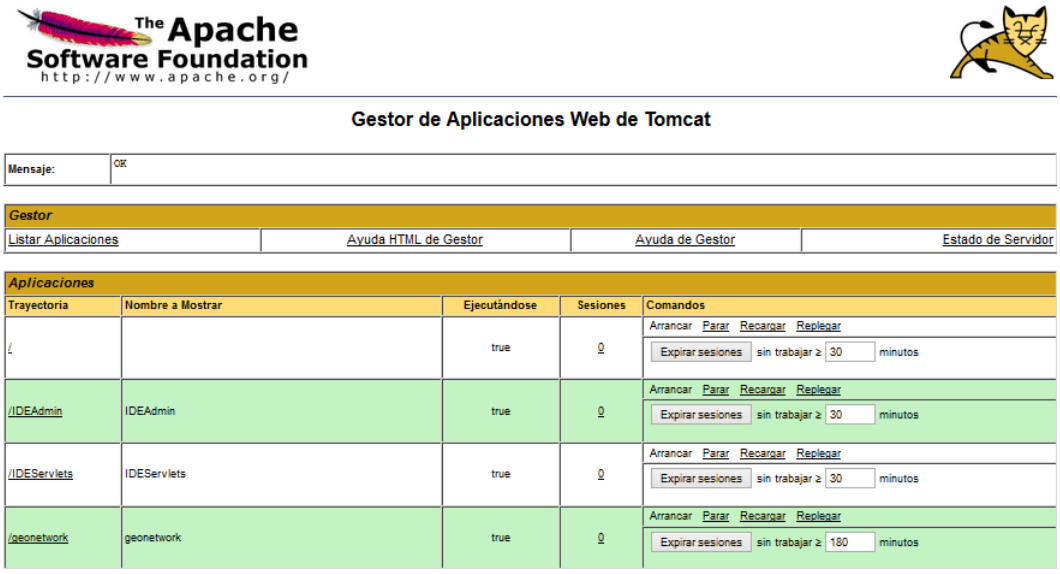


Figura 69: Interfaz de gestión de aplicaciones web de Tomcat

Previamente, el usuario habrá descargado en la máquina local el fichero de instalación WAR. Este fichero suele estar comprimido en formato ZIP, por lo que deberá ser descomprimido para poder obtener el instalable geoserver.war.

Desde la interfaz Manager de Apache Tomcat existe una utilidad para el despliegue de aplicaciones WAR. De esta forma Tomcat se encarga de realizar la descompresión del fichero y la instalación sobre el servidor.

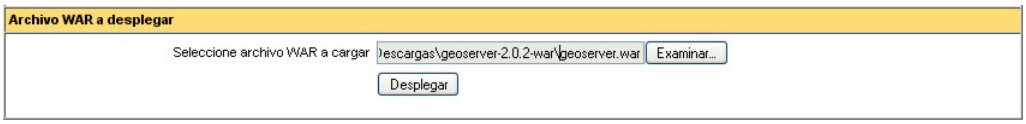


Figura 70: Instalación de aplicaciones desde la interfaz de Tomcat

Las dos formas de realizar la instalación de Geoserver son igual de válidas, por lo que queda en manos del usuario administrador la elección de qué método de instalación le resulta más cómodo.

Una vez instalado Geoserver, la forma de acceder a su administración será siempre desde un navegador, en un entorno gráfico a través de la URL (si por ejemplo la IP es 1.2.3.4)

http://1.2.3.4:8080/geoserver

14.7 Postgres 9.3 + PostGIS 2.1

Para realizar la instalación de PostgreSQL el usuario deberá estar logueado como usuario root. La versión que se va a instalar será PostgreSQL 9.3 con extensión PostGIS 2.1, y la instalación se realiza de la siguiente forma:

- >apt-get install postgresql-9.3-postgis

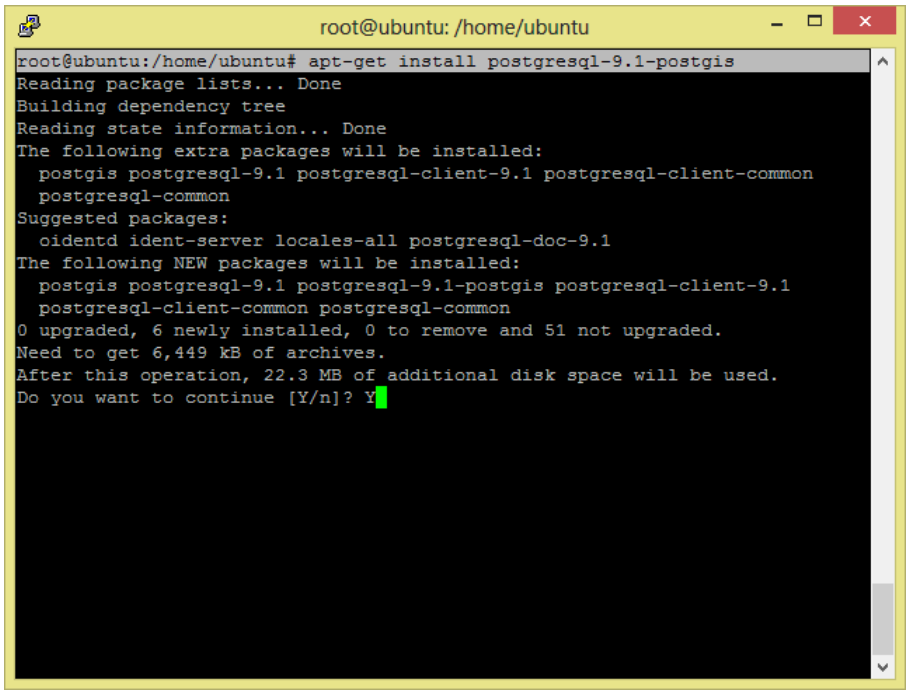


Figura 71: Instalación de PostgreSQL + PostGIS

Después de realizar la instalación de PostgreSQL y PostGIS, se deberán hacer ciertas configuraciones.

- Definir contraseña del usuario 'postgres'

>sudo su postgres

>psql

>ALTER USER postgres WITH PASSWORD 'postgres';

Dónde 'postgres' es el password que se le ha asignado al usuario postgres.

>\q

Para salir del modo psql.

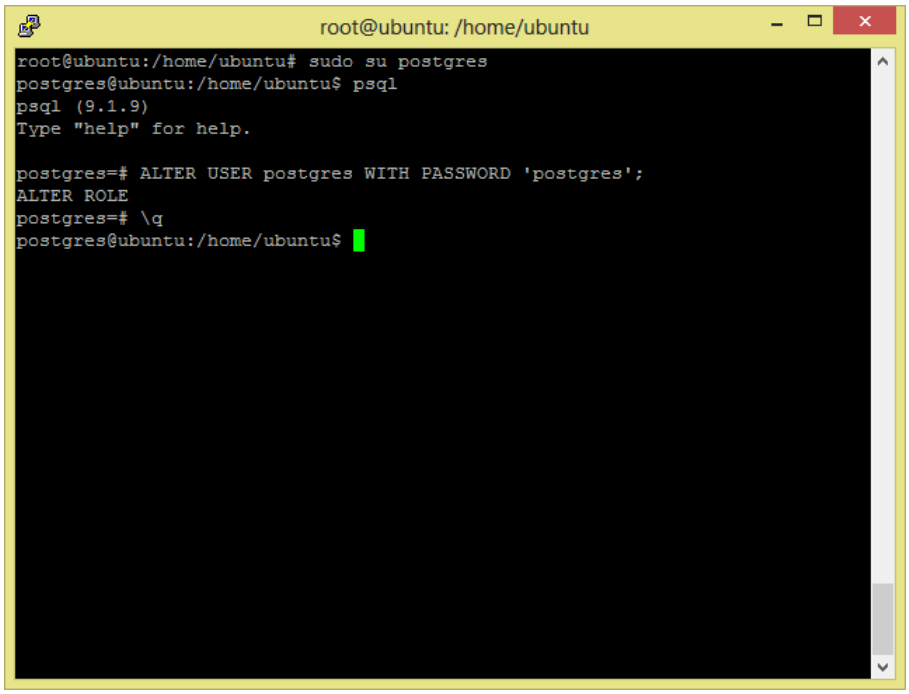


Figura 72: Definición de password para usuario PostgreSQL

- Creación de una plantilla PostGIS

>sudo su postgres

>createdb pgistemplate

>createlang plpgsql pgistemplate

>psql -d pgistemplate -f /usr/share/postgresql/9.1/contrib/postgis-1.5/postgis.sql

>psql -d pgistemplate -f /usr/share/postgresql/9.1/contrib/postgis-1.5/spatial_ref_sys.sql

>psql -d pgistemplate -c "SELECT postgis_full_version();"

>exit

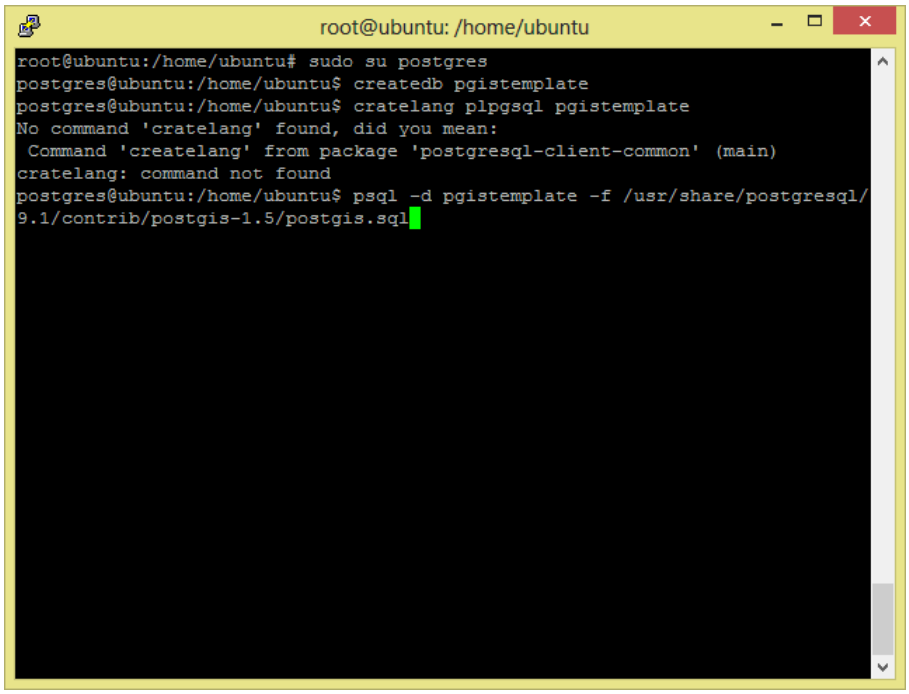


Figura 73: Creación de una plantilla PostGIS. Paso 1

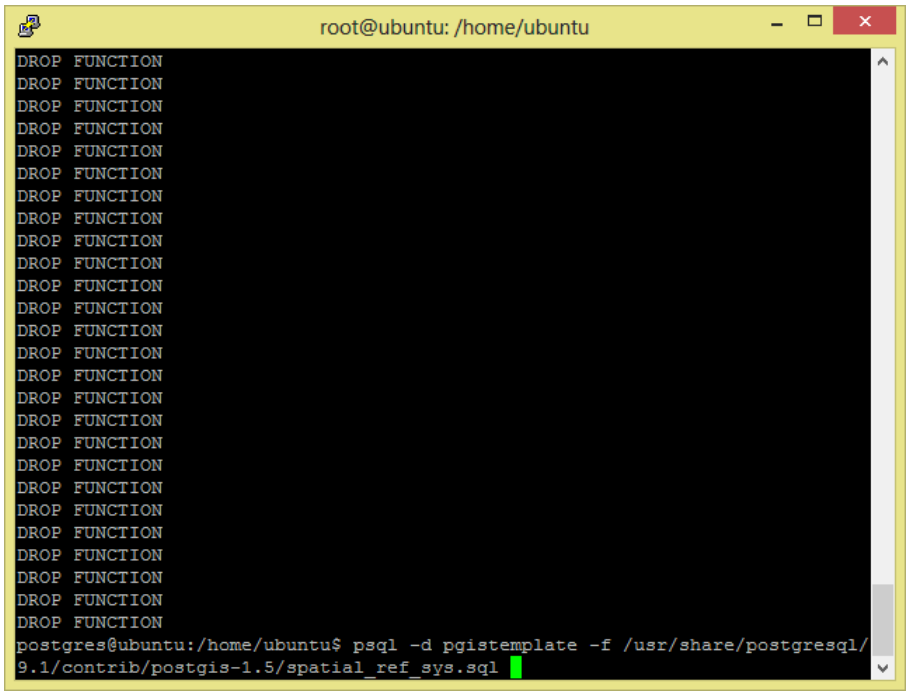


Figura 74: Creación de una plantilla PostGIS. Paso 2

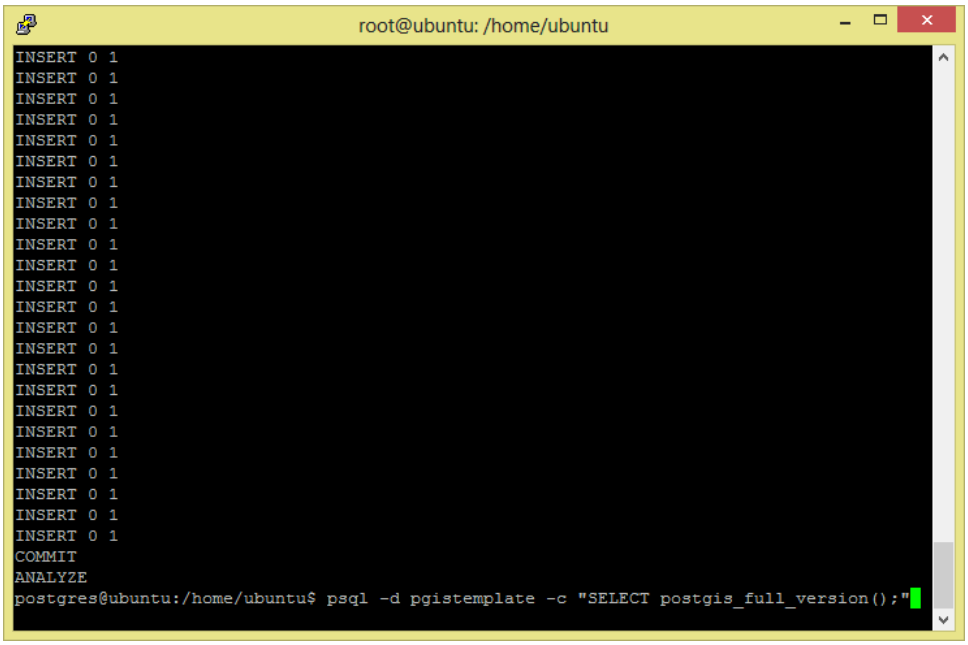


Figura 75: Creación de una plantilla PostGIS. Paso 3

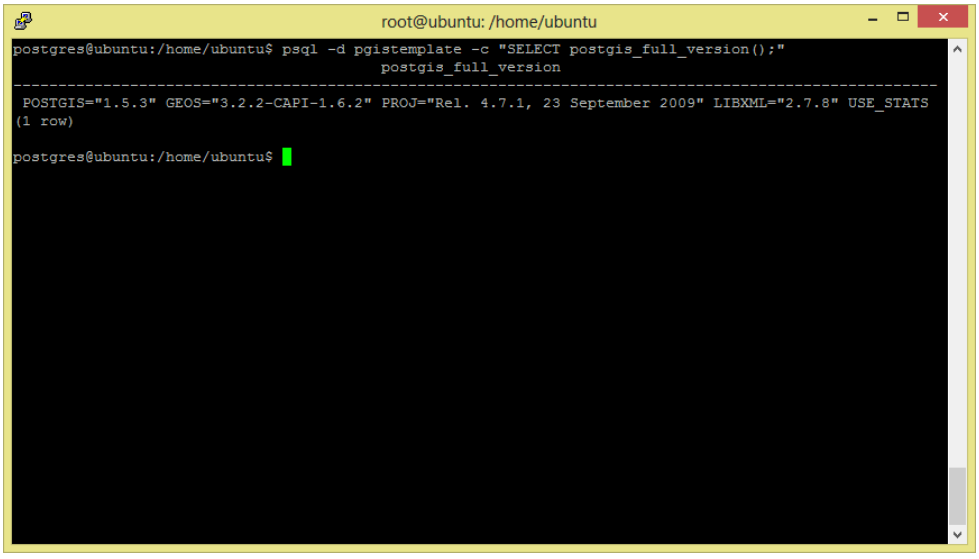


Figura 76: Creación de una plantilla PostGIS. Paso 4

- Configuración de los permisos de conexión a la base de datos

>nano /etc/postgresql/9.1/main/postgresql.conf

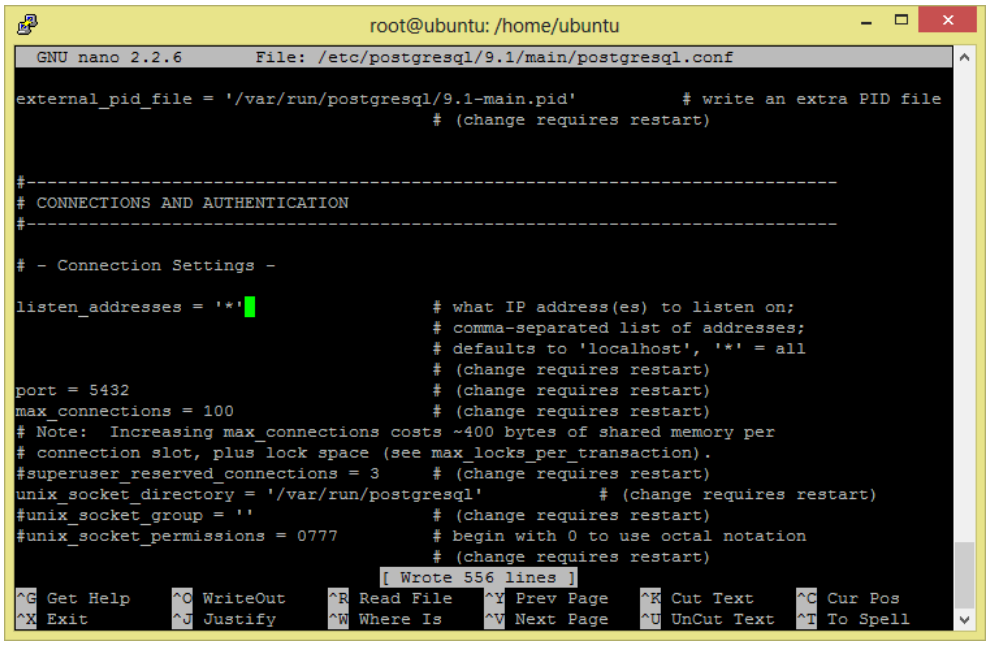


Figura 77: Configuración del fichero postgresql.conf

En este fichero se deberá comprobar el valor de los parámetros 'listen_adresses' y 'password_encryption'. Estas variables deberán tener los valores '*' y on respectivamente. Y por último, si estas líneas estás comentadas (están comentadas si tienen el carácter # delante de la línea), habrá que descomentarlas borrando el carácter # que está delante de ellas.

Las líneas deberán quedar así:

listen_adresses = '*'

...

password_encryption = on

Nota 1: Las líneas no están consecutivas en el fichero.

Nota 2: Para guardar los cambios en el documento, hay que marcar las teclas 'ctrl'+O para guardar y 'ctrl'+X para cerrar el documento.

- Configuración de los permisos de conexión a la base de datos

>nano /etc/postgresql/9.1/main/pg_hba.conf

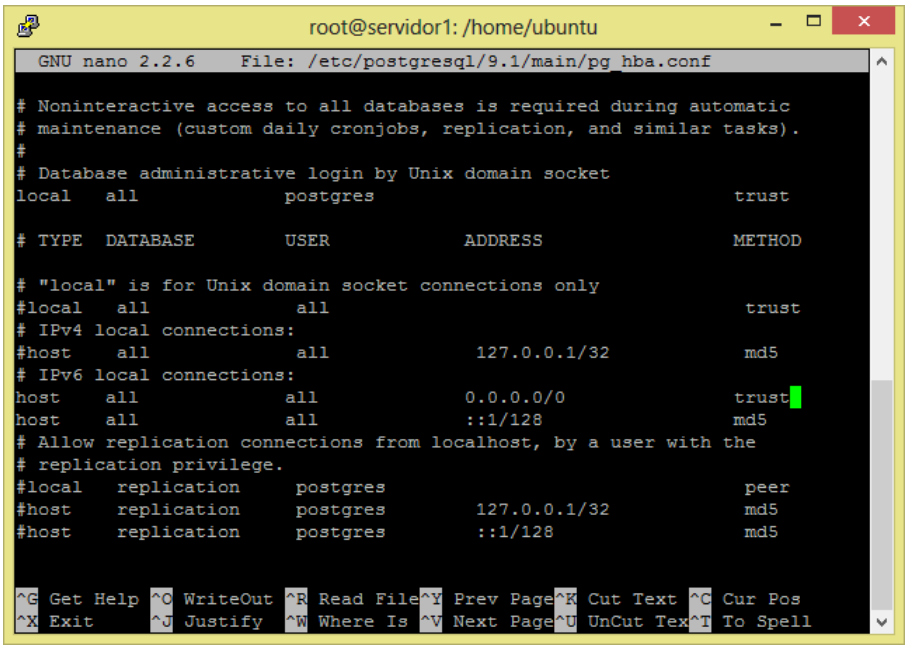


Figura 78: Configuración del fichero pg_hba.conf

En este fichero se realizará la configuración de forma que permita la conexión a la base de datos desde cualquier IP, siempre y cuando el usuario esté validado. Para ello los parámetros de configuración deberán ser los siguientes, y el resto que hay de ejemplo, comentarlos.

local all postgres trust

host all all 0.0.0.0/0 trust

Y por último, faltará reiniciar el servicio para que los cambios realizados sean efectivos

>/etc/init.d/postgresql restart

15 Bibliografía

En este apartado se recogen las fuentes de información empleadas.

1. European Commission: Draft Implementing Rules for Metadata (Version 3). http://inspire.jrc.it/reports/ImplementingRules/INSPIRE_Metadata_ImplementingRule_v3_20071026.pdf (2007)
2. International Organization for Standardization (ISO): Geographic information - Metadata. ISO 19115:2003 (2003)
3. International Organization for Standardization (ISO): Information and documentation - The Dublin Core metadata element set. ISO 15836:2003 (2003)
4. European Commission: Relation between ISO 19115 and ISO 19119 and the elements of the INSPIRE draft metadata implementing rules (informative)
5. International Organization for Standardization (ISO): Geographic information-Services. ISO 19119:2005 (2005)
6. International Organization for Standardization (ISO): Geographic information-Metadata-XML schema implementation. ISO/TS 19139:2007 (2007)

Páginas web de consulta:

7. Ubuntu: http://www.ubuntu.com/
8. Apache: https://httpd.apache.org/
9. OpenJava: http://openjdk.java.net/
10. Apache Tomcat: http://tomcat.apache.org/
11. Postgres: http://www.postgresql.org/
12. Geoserver: http://geoserver.org/
13. ExtJS: http://www.sencha.com/products/extjs/
14. OpenLayers: http://openlayers.org/
15. GeoExt: http://geoext.org/
16. INSPIRE. http://inspire.ec.europa.eu/
17. Open Geospatial Consortium. http://www.opengeospatial.org/