Proyecto Fin de Carrera

Linked Map Service: Aplicación de estándares abiertos para la explotación transparente de datos geográficos y Linked Data

Autor:

Elena Oliván Salvador

Director:

Francisco Javier López Pellicer

Departamento de Informática e Ingeniería de Sistemas

Escuela de Ingeniería y Arquitectura

Junio 2016

Resumen

En la actualidad podemos encontrar grandes volúmenes de información geográfica y numerosas aplicaciones para trabajar con ellos. Con este proyecto se pretende dar una vía de desarrollo en las aplicaciones de explotación de información geográfica, a través de la incorporación de los datos enlenzados, más conocidos como *Linked Data* en estas aplicaciones. Los Datos Enlazados o *Linked Data* es una metodología que permite vincular, compartir y exponer información estructurada utilizando RDF como modelo de datos. Si conectamos la información geoespacial con otros recursos relacionados obtenemos como resultado un gran volumen de información interconectado que permite el desarrollo de nuevas aplicaciones.

Para ello en este proyecto se ha desarrollado un servicio capaz de comunicarse con servicios de mapas externos que generan mapas referenciados espacialmente de forma dinámica a partir de información geográfica, y a su vez permite el acceso en modo lectura y escritura a almacenes de datos RDF, lo que permite enlazar estos recursos. Para ver los resultados y la información vinculada se ofrece un cliente, en el que se puede trabajar con diferentes capas que se obtienen tanto de servicios de mapas externos, como propios, y almacenes de datos RDF. Todo el proyecto se encuentra accesible como código abierto en un repositorio online:

`https://github.com/IAAA-Lab/LinkedMapService`

Agradecimientos

A mis padres por estar siempre ahí en estos años, apoyándome y animándome a seguir adelante, sin ellos no hubiera llegado hasta aquí. Al resto de familiares y amigos por el interés y palabras de ánimo que siempre me habéis ofrecido. A mis amigas por aguantarme y seguir ahí en cada paso del camino. A mis amigos de la universidad, por los malos y buenos momentos que hemos vivido a lo largo de este viaje, al final los que quedan son los buenos. A mis compañeros de laboratorio por ayudar a que los días de trabajo fueran más llevaderos. A Javier, mi director de proyecto, por sus consejos, disponibilidad y paciencia. Y especialmente a mi madre por confiar siempre en mí.

Contenido

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

En este documento se presenta la memoria del proyecto final de carrera "*Linked Map Service: Aplicación de estándares abiertos para la explotación transparente de datos geográficos y Linked Data*", en el que se desarrolla un sistema que permite trabajar con datos geográficos oficiales y vincularlos a los datos representados en dichas imágenes que han sido publicados siguiendo el modelo RDF y las buenas prácticas de publicación de datos en la web conocidas como Datos Enlazados o *Linked Data*. De esta forma se ha conseguido una herramienta que permite la interacción de estos dos ámbitos, el geoespacial y el de *Linked Data*.

Este proyecto nace con el propósito de simplificar, optimizar y publicar como código abierto un sistema desarrollado dentro de un proyecto de investigación denominado Linked Map [1]. Este proyecto lleva a cabo la conversión de datos geográficos de carácter oficial gubernamental, para que puedan ser vinculados con fuentes de datos según el modelo RDF [2] enlazados entre sí, también conocidos como *Linked Data*.

1.1. Motivación

Con este proyecto se quiere demostrar el gran potencial que existe vinculando diferentes fuentes de información geográfica bajo el paradigma del *Linked Data* y generar nuevas posibilidades de investigación y desarrollo de aplicaciones, dado que con este proyecto se permite su uso en lectura y escritura, abriendo nuevas vías de desarrollo. Los grandes productores de información geográfica (NASA, ESA, agencias nacionales de cartografía, catastro, etc.) tienen un ritmo lento de adopción de sistemas basados en el uso de *Linked Data*, incluso con proyectos exitosos como los desarrollados en Ordnance Survey [3], la entidad responsable de la publicación de mapas oficiales en Gran Bretaña. Esto se debe al esfuerzo técnico y económico necesario para adaptar las herramientas y

estándares actuales a este nuevo modelo de información. Es por ello que estos grandes productores de información necesitan ver importantes ventajas en la utilización de aplicaciones con este tipo de tecnología.

En general, el *Linked Data* se percibe como una tecnología novedosa, pero no interoperable con los estándares OGC [4] (Open Geospatial Consortium), que son los que se siguen habitualmente para el tratamiento y trabajo con información geográfica. Además existe la percepción de que el principal enfoque que se le da al *Linked Data* es de "solo lectura", este es uno de los motivos de su lenta adopción. Este proyecto revela que las dos críticas anteriores deben ser reconsideradas mostrando cómo se puede establecer una interoperabilidad transparente entre el mundo de OGC y *Linked Data*, y cómo se puede trabajar con *Linked Data* en modo lectura/escritura con información geográfica.

1.2. Trabajo previo

En esta sección se proporciona información sobre trabajos previos de servicios geoespaciales OGC, *Linked Data* y servicios geoespaciales que incluyen web semántica, es decir, extensión semántica de servicios geoespaciales estándar.

1.2.1. Servicios geoespaciales de OGC

Open Geospatial Consortium (OGC) es un consorcio internacional conformado por 521 empresas (a fecha 14/05/2016), agencias gubernamentales y universidades que participan en un proceso de consenso para desarrollar estándares de interfaz de acceso público. Los estándares OGC apoyan la interoperabilidad entre sistemas de información geográfica y la Web. Aportando así a los desarrolladores una tecnología que les permite trabajar con compleja información geográfica y sus servicios de forma útil y accesible en todo tipo de aplicaciones.

Desde sus orígenes el OGC ha adaptado sus especificaciones a especificaciones basadas en web, desarrollando su propia arquitectura, siendo esta similar a la arquitectura orientada a servicios (*Service Oriented Architecture*. SOA) [5]. SOA es una arquitectura conceptual para diseñar y desarrollar sistemas que permiten la integración de los datos y la lógica de negocio en un mismo sistema. SOA es una arquitectura de aplicación, todas las funciones están definidas como servicios independientes y sus interfaces pueden ser

instanciadas y llamadas en secuencias bien definidas para formar los procesos de negocio. En SOA la clave está en la interfaz que es la encargada de definir los parámetros requeridos y la naturaleza del resultado. Con ello detalla la naturaleza del servicio y no la tecnología utilizada. Esta función permite realizar dos de los puntos críticos: los servicios son realmente independientes y pueden ser manejados. La arquitectura SOA puede utilizarse para desarrollar sistemas en diversos ámbitos, ya que no es una solución específica de problemas GIS (Sistema de Información Geográfica), pero si está ampliamente utilizada en este ámbito.

The OpenGIS Web Service Common Implementation Standard [6] especifica detalles de la implementación SOA de OGC que son comunes a los estándares de interfaz de aplicación de los servicios web OGC. Estos aspectos comunes son principalmente algunos de los parámetros y las estructuras de datos utilizadas en las solicitudes de operación y respuestas. Cada uno de estos estándares detalla aspectos adicionales de la interfaz, incluyendo la especificación de todos los parámetros adicionales y estructuras de datos necesarios en todas las solicitudes de operación y respuestas. En cualquier caso, OGC reconoce la necesidad de dar cabida a múltiples estilos, por ejemplo los que corresponden a servicios web WSLD/SOAP y RESTful.

OGC proporciona diferentes servicios para trabajar con datos geográficos:

- **Web Feature Service (WFS)** [7]: servicio que ofrece una interfaz que permite interactuar con los mapas, editando o analizando la imagen siguiendo criterios geográficos.
- **Web Coverage Server (WCS)** [8]: servicio que ofrece una interfaz que permite realizar peticiones web independientes de la plataforma de cobertura geográfica. La cobertura geográfica son imágenes/objetos editables de un área en concreto.
- **Sensor Observation Service (SOS)** [9]: servicio que permite consultar datos generados por sensores en tiempo real, y series de datos.
- **Catalogue Service (CSW)** [10]: servicio define una interfaz común para la búsqueda y consulta de servicios, metadatos y recursos de tipo geográfico.
- **Web Map Service (WMS)** [11]: servicio que genera mapas referenciados espacialmente, de forma dinámica a través de información geográfica.

Este último, el WMS, es el más utilizado en la actualidad, es el que proporciona la información geográfica representada en una imagen que se puede ver en cualquier

dispositivo que soporte el formato imagen como PNG o JPEG y opcionalmente como gráficos vectoriales como SVG.

El estándar WMS, ofrece operaciones para recuperar mapas, descripción de los mapas y opcionalmente información asociada a los recursos representados en el mapa. Estas operaciones son *GetMap*, *GetCapabilities* y *GetFeatureInfo*, respectivamente. Una operación de un WMS está definida por una combinación de métodos HTTP, una serie de parámetros para especificar la consulta, y un cuerpo con el contenido del mensaje y el contenido de la respuesta según el tipo de petición.

A continuación se muestra un ejemplo de las peticiones *GetCapabilities*, *GetMap*, *GetFeatureInfo* con un WMS que sirve imágenes de mapas a partir de los datos de OpenStreetMap (OSM)[1], el cual es un mapa del mundo creado por los usuarios, de uso libre y licencia abierta.

Ejemplo de petición *GetCapabilities*:

```
http://b.maps.omniscale.net/v2/osmoscde-8a05ef58/style.default/map?
service=WMS&request=GetCapabilities&version=1.1.1
```

El parámetro resaltado en negrita en la petición indica el tipo de petición que se está realizando, en este caso *GetCapabilities*. Como resultado devuelve un XML con la información que describe el servicio. A continuación una parte del resultado que devuelve:

```
<?xml version="1.0"?>
<!DOCTYPE WMT_MS_Capabilities SYSTEM "http://schemas.opengis.net/wm
s/1.1.1/WMS_MS_Capabilities.dtd">
<WMT_MS_Capabilities version="1.1.1">
  <Service>
  <Name>OGC:WMS</Name>
  <Title>Omniscale OpenStreetMap WMS</Title>
  <Abstract>
    Omniscale OpenStreetMap WMS © Omniscale,
    Map Data: OpenStreetMap - (License:ODbL)
  </Abstract>
  (... otras definiciones ...)
  </Service>
  (... otras definiciones ...)
  <Layer queryable="1">
```

[1] https://www.openstreetmap.org/

```
  <Name>world</Name>
  <Title>Base map</Title>
  </Layer>

  <Layer queryable="1">
  <Name>landusages</Name>
  <Title>Land use</Title>
  </Layer>

  <Layer queryable="1">
  <Name>admin</Name>
  <Title>Boundaries</Title>
  </Layer>
  (... otras definiciones ...)
</WMT_MS_Capabilities>
```

El XML que devuelve la petición *GetCapabilities*, muestra la descripción de varias de las capas que ofrece el WMS de OpenStreetMap, las descripciones obtenidas corresponden a las capas denominadas: "world" que se corresponde a la capa base del mundo, "landusages" que es el nombre que recibe la capa de usos del suelo y "admin", correspondiente a una capa con información de la administración.

Ejemplo de petición *GetMap*:

`http://b.maps.omniscale.net/v2/osmoscde-8a05ef58/style.default/map?service=WMS&`**`request=GetCapabilities`**`&version=1.1.1& `**`layers=osm`**`&styles=&`**`format=image%2Fpng`**`&transparent=false&hq=false&continuousWorld=true&height=256&width=256&`**`srs=EPSG%3A3857`**`&`**`bbox=-469629.1017841229, 4852834.051769271, -391357.5848201024,4931105.568733294`**

Los parámetros resaltados en negrita y subrayados, son los que sirven para especificar el resultado que estamos solicitando a través de esta petición.

Parámetros	**Descripción**
Service	Especifica el tipo de servicio al que se le envía la petición, en este caso un WMS, otras posibilidades serían los servicios descritos anteriormente en esta misma sección (WFS, WCS, SOS, CSW)
Request	Especifica el tipo de petición que se está realizando, en este caso *GetMap*, otras posibilidades al tratarse de un WMS serían *GetCapabilities* o *GetFeatureInfo*
Layers	Especifica las capas sobre las que se está realizando la petición
Format	Especifica el tipo de formato de la respuesta que se quiere obtener

Parámetros	Descripción
SRS	Especifica el sistema de referencia con el que trabaja el servicio
Bbox	Especifica el bounding box[2] sobre el que se está realizando la petición

Tabla 1: Descripción parámetros petición

El resultado obtenido es una imagen en formato PNG:

Figura 1.1: Resultado ejemplo petición *GetMap*

Ejemplo de petición *GetFeatureInfo*:

```
http://demo.opengeo.org/geoserver/wms?SERVICE=WMS&VERSION=1.3.0&
REQUEST=GetFeatureInfo&FORMAT=image%2Fpng&TRANSPARENT=true&
QUERY_LAYERS=ne%3Ane&LAYERS=ne%3Ane&INFO_FORMAT=text %2Fplain&I=230&
J=243&WIDTH= 256&HEIGHT=256&CRS=EPSG%3A3857& STYLES=&BBOX=
-1252344.271424327%2C5009377.085697312%2C6.984919309616089e-10%2C
6261721.35712164
```

En los parámetros resaltados en negrita y subrayados se puede ver el tipo de servicio al que se le hace la petición, el tipo de petición y el formato en el que se solicita la respuesta (descripción más detalla del resto de parámetros en la tabla 5). El resultado obtenido tras realizar la petición se muestra a continuación y consiste en un fichero HTML.

[2] Bounding box: es el área definida por dos longitudes y dos latitudes. bbox = min Longitud , min Latitud , max Longitud , max Latitud

```
the_geom = [GEOMETRY (MultiPolygon) with 2980 points]
scalerank = 0
featurecla = Admin-0 country
labelrank = 2.0
sovereignt = Spain
sov_a3 = ESP
adm0_dif = 0.0
level = 2.0
type = Sovereign country
admin = Spain
adm0_a3 = ESP
geou_dif = 0.0
geounit = Spain
gu_a3 = ESP
su_dif = 0.0
subunit = Spain
su_a3 = ESP
brk_diff = 0.0
name = Spain
name_long = Spain
brk_a3 = ESP / brk_name = Spain / brk_group =
abbrev = Sp.
postal = E
formal_en = Kingdom of Spain / formal_fr =
note_adm0 = / note_brk =
name_sort = Spain // name_alt =
mapcolor7 = 4.0/mapcolor8 = 5.0/mapcolor9 = 5.0/mapcolor13 =5.0
pop_est = 4.0525002E7/ gdp_md_est = 1403000.0/ pop_year = -99.0
lastcensus = 2001.0
gdp_year = -99.0
economy = 2. Developed region: nonG7
income_grp = 1. High income: OECD
wikipedia = -99.0
fips_10 =
iso_a2 = ES / iso_a3 = ESP / iso_n3 = 724
un_a3 = 724
wb_a2 = ES / wb_a3 = ESP
```

```
woe_id = -99.0
adm0_a3_is = ESP / adm0_a3_us = ESP
adm0_a3_un = -99.0 / adm0_a3_wb = -99.0
continent = Europe
region_un = Europe
subregion = Southern Europe
region_wb = Europe & Central Asia
name_len = 5.0 / long_len = 5.0 / abbrev_len = 3.0
tiny = -99.0
homepart = 1.0
```

Tabla 2: Resultados *GetFeatureInfo*

En la respuesta se obtiene la información que contiene el servicio de mapas sobre la *feature*[3] u objeto espacial que se ha realizado la petición. En este caso la información describe una *feature* de tipo país relacionado con la administración.

1.2.2. Linked Data

Linked Data es el uso de la Web para vincular una información con otra que previamente no estaba enlazada. El *Linked Data* es una metodología que permite exponer, compartir y vincular información y conocimientos de web semántica mediante el uso de URIs y RDF. Las URIs como su propio nombre indica son los identificadores de recursos uniformes, y son una serie de cadenas de caracteres que nos permiten identificar los recursos de forma única. RDF es el modelo de almacenamiento de los datos en la web semántica. Este formato soporta la evolución del esquema de almacenamiento de los datos en el tiempo sin la necesidad de que los consumidores de datos tengan que ser modificados, también facilita la fusión de los datos, aunque sigan diferentes esquemas. La relación entre los diferentes recursos almacenados en formato RDF se realiza mediante URIs, a su vez, cada recurso tiene una URI asociada, por tanto

[3] *Feature* es la forma de denominar al objeto que contiene información asociada y se representa en el mapa.

estas tres URIs forman lo que se conoce como tripleta. Con el uso de este modelo, las aplicaciones pueden mezclar, fusionar, consultar y compartir los datos.

Para trabajar con la información en RDF existe un lenguaje estandarizado que es SPARQL (SPARQL Protocol and RDF Query Language) [12]. SPARQL es una tecnología fundamental en el desarrollo de la Web Semántica. Originalmente solo incorpora funciones para la recuperación de sentencias RDF. Sin embargo, actualmente algunas propuestas también incluyen operaciones de creación, modificación y borrado de datos. SPARQL soporta la consulta de patrones obligatorios y opcionales de grafo, junto con sus conjunciones y disyunciones, también soporta la ampliación o restricciones del ámbito de las consultas indicando los grafos sobre los que se opera. Los resultados de las consultas SPARQL pueden ser conjuntos de resultados o grafos RDF.

Ejemplo de una consulta en SPARQL

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT?name
WHERE f ?x foaf:name ?name.g
ORDER BY ?name
LIMIT 3
OFFSET 1
```

Dentro de SPARQL se encuentra un vocabulario más específico denominado GeoSPARQL. GeoSPARQL es un estándar para la representación y consulta de *Linked Data* geoespaciales para la Web Semántica del OGC. Tiene por objeto proporcionar una base de intercambio estandarizado para datos geoespaciales RDF que puede apoyar tanto el razonamiento espacial cualitativo y cuantitativo y consultar con el lenguaje de consulta de base de datos SPARQL. Un ejemplo de una consulta en GeoSPARQL

```
PREFIX geo: <http://www.opengis.net/def/geosparql/>
PREFIX geof: <http://www.opengis.net/def/geosparql/function/>
PREFIX sf: <http://www.opengis.net/def/sf/>

SELECT ?f

WHERE {

  ?f geo:hasGeometry ?g .

  ?g geo:asWKT ?gWKT .

  FILTER (geof:sfWithin(?gWKT,"POLYGON ((-77.2 38.8, -77 38.8, -77
  39, -77.2    39.9,-77.2 38.8))"^^sf:wktLiteral))

}
```

1.2.3. Extensión semántica de servicios geoespaciales estándar

Uno de los principales objetivos de este proyecto es poder trabajar con servicios geoespaciales, incluyendo el concepto de web semántica, es por ello que se utiliza el término semántica geoespacial. La semántica geoespacial se encarga de la compresión de los atributos de un objeto geoespacial, en concreto la semántica geoespacial trata la información geoespacial como un componente de la web semántica.

Lo que se pretende conseguir con la semántica geoespacial web, es trabajar con la información geoespacial de forma significativa para las personas y las máquinas. Sin embargo, existen todavía retos abiertos en el descubrimiento, recuperación, traducción, transformación, interpretación e integración de la semántica geoespacial.

El término habilitación semántica en el contexto de los servicios geoespaciales significa encapsular o añadir un servicio de web semántica (operaciones de búsqueda y recuperación, etc.), en servicios basados en especificaciones OGC. Este enfoque proporciona una gran ventaja, ya que los desarrolladores pueden integrar este tipo de servicios de web semántica, en sus servicios con estándares OGC. Pero esto supone la adaptación del servicio de web semántica a los estándares OGC. Como alternativa para la habilitación semántica en un servicio OGC Web Lopez-Pellicer [13], propone incluir catálogos basados en CSW con un servicio que expone sus contenidos como *Linked Data*.

1.3. Objetivos del proyecto

Linked Map es una aplicación que permite trabajar con datos geográficos y *Linked Data*. Es importante diferenciar el Linked Map como aplicación y tecnología, ya que la tecnología que utiliza el proyecto que consiste en vincular información geográfica con información en formato RDF, también se conoce con el mismo nombre que este tipo de aplicaciones.

Este proyecto tiene como objetivo ofrecer un sistema simple, optimizado y de código abierto, que permite trabajar con información geográfica y *Linked Data*. Otro de los objetivos que se quiere satisfacer con este proyecto es el obtener una aplicación que permite relacionar información geográfica con información en formato RDF,

vinculando de esta forma el ámbito de la información geográfica con el paradigma del *Linked Data*. Por ello se ha querido obtener un cliente web que le permite al usuario visualizar, consultar y editar información, siendo para este transparente el funcionamiento y procesamiento de las peticiones y la naturaleza de la información, es decir, el cliente realiza las peticiones sin tener que especificar o conocer si la petición es geográfica o de *Linked Data*.

Otro de los objetivos es proporcionar una solución modular que permite la expansión del sistema o el reemplazo de servicios u almacenes de datos de forma sencilla, para facilitar las posibles vías de investigación y desarrollo posteriores. También es uno de los objetivos obtener un sistema de fácil configuración y uso para el usuario para ello se ha generado un manual de usuario, además de tener toda la documentación y fuentes del sistema en un repositorio online.

El Linked Map Service es el servicio núcleo del sistema y es el que contiene la lógica de este. El objetivo de implementar este servicio es: el permitir trabajar con ambas fuentes de información, geoespacial y RDF, siendo este el encargado de gestionar las peticiones entre el cliente y el servidor.

En este proyecto se utiliza la tecnología de Linked Map para trabajar con los datos del "*Nomenclátor geográfico básico de España*" [14] proporcionados por el Centro Nacional de Información Geográfica.

1.4. Tecnologías utilizadas

Como he mencionado anteriormente un objetivo principal de este proyecto es generar una solución de código abierto, para ello todas las tecnologías utilizadas en su desarrollo e implementación también lo son.

A continuación nombro las diferentes tecnologías y herramientas utilizadas en el desarrollo del proyecto.

- **Base de datos geoespacial**: se ha utilizdo el sistema gestor de la base de datos geoespacial PostgreSQL 9.2 con su extensión PostGIS 2.1.8.
- **Extracción y carga de datos**: se ha utiliza Java 1.8 y la herramienta GeoKettle.
- **Servidor de mapas** conforme con OGC WMS 1.3.0 mediante el uso de Mapserver MS4W.

- **Servidor SPARQL**: se ha utilizado Apache Fuseki 2.3.1, que es un servicio que trabaja con SPARQL.
- **Implementación de LMS**: se ha utilizado Java 1.8 con el framwork Spring Boot, JavaScript, y Apache Jena. Todas las dependencias del servicio se gestionan mediante Apache Maven.
- **Implementación demostrador** utiliza Bootstrap y OpenLayers que es una librería JavaScript.

Como se puede observar todos los lenguajes, servicios, herramientas utilizadas en el desarrollo e implementación del proyecto son de código abierto.

En el apéndice C se describen con más detalle las tecnologías aquí nombradas.

Capítulo 2. Análisis del sistema

En el presente capítulo se describen las funcionalidades que poseen el sistema y los diferentes bloques que se pueden diferenciar en él. En el sistema se diferencian dos bloques principales. Un primer bloque corresponde al Linked Map Service (LMS), y otro bloque que es el demostrador, que permite probar el funcionamiento del bloque del LMS.

2.1. Funcionalidades del sistema

A continuación se enumeran las principales funcionalidades que tiene el sistema separadas por los bloques antes mencionados:

Funcionalidades del Linked Map Service (LMS):

1. Permite gestionar peticiones dirigidas a un servicio de mapas, o a un almacén de datos RDF de forma transparente al usuario final.
    a. Si se pide una imagen conforme a la operación *GetMap* del estándar OGC WMS se obtiene como respuesta la imagen correspondiente a la petición del WMS.
    b. Si se piden datos se obtiene un RDF procedente de un almacén RDF con los datos usados para generar la imagen.
2. Funciones para realizar operaciones de lectura/escritura sobre almacenes de datos RDF.
3. Funciones de actualización de una base de datos externa al servicio.

4. Ofrece una API con funciones de consulta para datos RDF cuyo contenido es información geográfica, basada en Apache Jena y con los vocabularios de RDF Spatial[4] y GeoSPARQL[5].

Funcionalidades del demostrador:

1. Representación visual de los recursos pertenecientes a un área geográfica específica mediante iconos en el mapa.
2. Mostrar breve información sobre el recurso cuando el usuario pasa el ratón por encima.
3. Mostrar información detallada del recurso cuando el usuario selecciona un recurso haciendo clic en el mapa y a continuación hace uso del botón "más información". La información aparecerá al lado del mapa.
4. Editar la información asociada a un recurso, exceptuando su identificador.
5. Actualizar la información de los almacenes de datos del sistema.
6. Permitir ver diferentes tipos de recurso (editado, original, etc.)
7. Herramientas típicas utilizadas en un mapa: desplazar, zoom, etc.

2.2. Módulos del sistema

En esta sección se expone una visión general de la estructura en módulos del sistema. Como se explica anteriormente el sistema está compuesto por dos bloques principales: Linked Map Service y el demostrador o cliente web. Además de estos dos bloques principales el sistema también cuenta con un bloque que podemos denominar de persistencia, en el que se engloban los diferentes almacenes de datos con los que trabaja el sistema.

[4] https://www.w3.org/2005/Incubator/geo/XGR-geo-20071023/

[5] http://www.opengeospatial.org/standards/geosparql

2.2.1. Módulo Linked Map Service

Este módulo denominado Linked Map Service (LMS) es a través del cual se accede a los almacenes de datos. El LMS implementa una habilitación semántica dentro de un servicio geoespacial, basada en la idea que se nombre en la sección 1.3.3. Por tanto el LMS es un proxy inverso sobre un WMS externo, que a su vez puede servir información RDF sobre los recursos almacenados en el sistema. El LMS también implementa sus propias URIs directas y una interfaz REST que permite acceder y actualizar los datos del almacén RDF. Esta interfaz es distinta a la interfaz del WMS pero ambas están entrelazadas. Ya que el LMS es el encargado de trabajar con ambas interfaces y su integración en el sistema. En la figura 2.1 Se puede ver la interacción del LMS con la interfaz de web semántica y la del WMS.

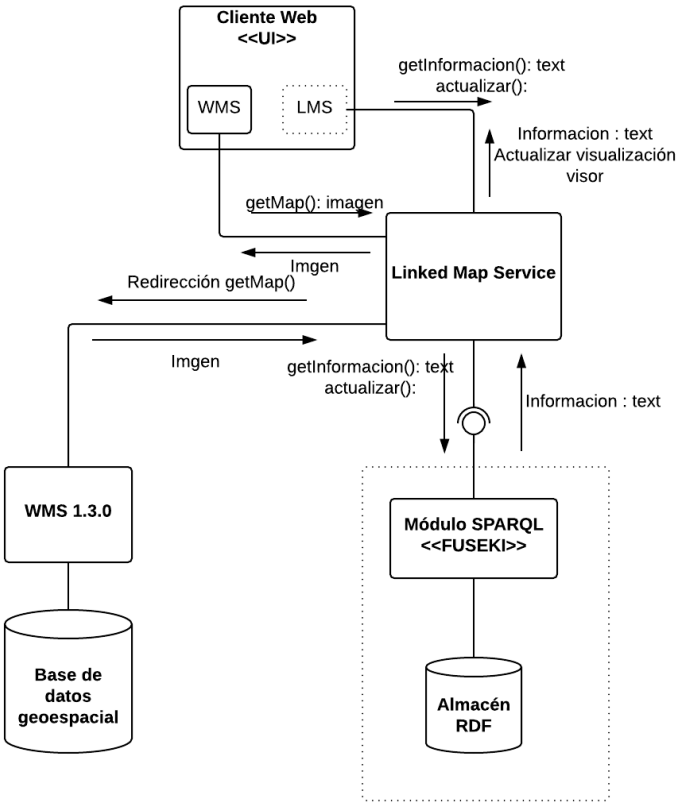

Figura 2.1: Diagrama interacción LMS con interfaz de web semántica y servicio de mapas

2.2.2. Demostrador

El demostrador sirve para probar el sistema, así como para que el usuario pueda comparar la información con la que está trabajando. El demostrador ofrece al usuario

diferentes capas de información, que es proporcionada por el LMS, es por ellos que puede comparar la información original del sistema obtenida de las fuentes oficiales, con la información aportada por otros usuarios o modificada por el mismo. En el siguiente diagrama (figura 2.2) se puede ver la interacción del cliente con el sistema.

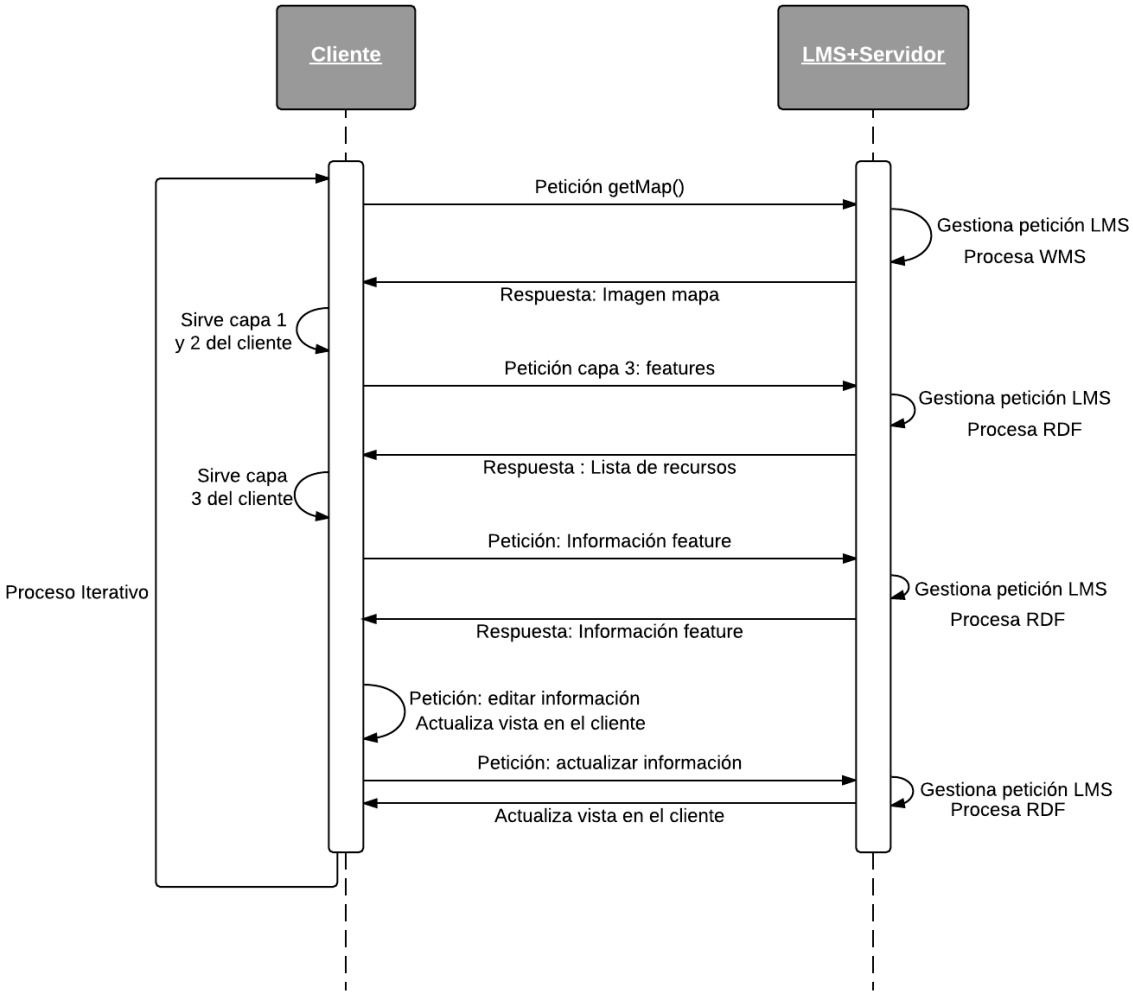

Figura 2.2: Diagrama de secuencia interacción cliente con el sistema

2.2.3. Módulo de persistencia

El módulo de persistencia de los datos se encarga como su nombre indica del almacenamiento de la información en el sistema. Dentro de este módulo de persistencia podemos diferenciar dos componentes, la base de datos geoespacial y el almacén RDF. La base de datos geoespacial es el componente donde se almacena la información obtenida de los conjuntos de datos obtenidos de fuentes oficiales del gobierno. Antes de almacenar la información en la base de datos geoespacial, ha pasado un pre-procesado,

en la que se ha realizada la adecuación de la información para que cumpla las especificaciones de la base de datos utilizada.

El almacén RDF obtiene la información que almacena de la base de datos geoespacial, teniendo los mismos recursos almacenados. Para ello se realiza una conversión de los datos almacenados en la base de datos geoespacial a formato RDF. Como resultado de esta conversión el almacén RDF contiene las tripletas de información RDF, lo que permite ser interrogado mediante el lenguaje de interrogación de *Linked Data* SPARQL.

Capítulo 3. Diseño del sistema

En este capítulo se describe la arquitectura del sistema, tomando como referencia las especificaciones dadas en el capítulo 2 de análisis del sistema.

3.1. Arquitectura del sistema

El proyecto está formado por dos grandes módulos diferenciados: el servicio denominado LMS y un prototipo de demostración que a su vez está compuesto por un servidor de datos, una instancia de LMS y un cliente web. En la figura 3.1 se pueden ver todas las partes del sistema y como se comunican entre ellas.

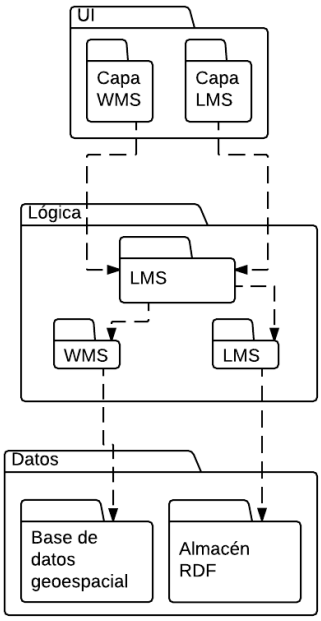

Figura 3.1: Diagrama del sistema

El LMS ofrece al usuario, una interfaz de forma que este, puede entender la representación visual de la información que ofrece el mapa, enlazándola a su vez con otra información geográfica relacionada con la misma. El LMS se implementa de forma simultánea y sobre las mismas URIs que los interfaces de servicio definidos en el estándar OGC WMS 1.3.0 para dar acceso a mapas y las buenas prácticas *Linked Data* para acceder a datos relacionados a ellos. Este servicio, a su vez, obtiene las imágenes de un WMS remoto que implementa el estándar internacional OGC WMS 1.3.0 que permite visualizar la información en un mapa, y los datos enlazados de un servidor SPARQL remoto de consulta y edición de información RDF que permite al sistema tratar consultas de web semántica. SPARQL es un protocolo para acceder a datos RDF.

Dentro del prototipo de demostración, el módulo de persistencia de datos contiene la base de datos almacenando la información geográfica con la que trabaja el sistema.

El cliente web, será el medio utilizado por el usuario, para visualizar la información en forma de mapa, realizar consultas sobre puntos de interés y editar información. Y por último el LMS, será el encargado de comunicar el cliente web con el servidor, será con este servicio con el que se procesarán las peticiones y se servirán las respuestas al cliente.

3.2. Componentes del sistema

En este apartado se explica más en detalle el funcionamiento de los diferentes módulos que componen el sistema. En primer lugar se describen los módulos correspondientes al LMS (LMS, API del LMS, servicio SPARQL y WMS), después los módulos correspondientes al demostrador (cliente web) y por el último los componentes del módulo de persistencia del sistema.

3.2.1. LMS

El LMS es el núcleo del sistema. Es en este servicio donde se encuentra la lógica del sistema. Los controladores están implementados en Java utilizando el framework Spring Boot. Al utilizar el framework Spring Boot este permite desplegar la instancia al servicio LMS en un servidor externo, o en un Tomcat embebido. El LMS está implementado para servir diferentes tipos de peticiones, que están agrupadas en dos grupos diferenciados.

De forma que las URIs validas que acepta el LMS son de la siguiente forma:

- `/service/{query}`
- `/resource/{id_resource}[?{query}]`

El espacio URI incluye selectores parámetros de consulta. La instancia del LMS procesa las URIs para luego reenviarlas al módulo correspondiente para que sirva las respuestas, es decir, en caso que la petición sea para el WMS, es en el LMS donde se procesa la URI para reenviar la petición al WMS con el formato adecuado para que este pueda procesarla, y de igual manera con las peticiones que tendrán que ser enviadas al módulo de SPARQL.

3.2.2. API del LMS

La API del LMS está compuesta por las siguientes operaciones: lectura de recursos y edición/actualización de recursos. También haría las funciones de proxy inverso en la parte referente al WMS, ya que se encarga del redireccionamiento de las peticiones.

Las librerías utilizadas para el desarrollo de esta API son:

- **Spring Boot** es parte de Spring IO Platform, un framework que facilita el desarrollo de proyectos y su posterior testeo. Spring Boot es un componente importante dentro del LMS porque, se encarga de la configuración de los componentes, principalmente a través de la inversión de control, utiliza el modelo vista-controlador para la implementación de servicio web, utiliza el Tomcat embebido, y la herramienta de pruebas durante la implementación.
- **Apache Jena** es un framework de código abierto que permite trabajar en el ámbito de web semántica desde Java. Este framework ofrece operaciones para trabajar con información en formato RDF mediante el uso de grafos RDF, dichos grafos se representan a través de modelos, que pueden generarse de diversas fuentes o incluso fuentes combinadas (ficheros, bases de datos, etc.) y soportan las consultas, mediante el lenguaje SPARQL.

3.2.3. Servicio SPARQL

El LMS es el servicio desde el cual se generan las peticiones para el acceso a los recursos del almacén RDF. Las operaciones permitidas son consultar, y editar recursos,

todo ellos siguiendo las prácticas de *Linked Data*. Para conectar el LMS con el almacén de datos RDF es necesario un punto de SPARQL, que sea el encargado de acceder a los datos. El punto de SPARQL que se ha elegido para este proyecto es Apache Fuseki(figura 3.2).

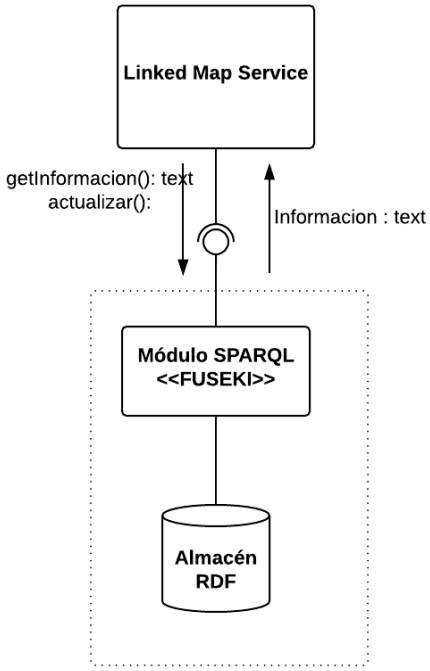

Figura 3.2: Diagrama interacción LMS con el módulo de web semántica

Apache Fuseki permite trabajar con datos en formato RDF de una forma sencilla, dado que es similar a un protocolo REST. Las operaciones que permite son las siguientes: SPARQL Query, y SPARQL Update.

Existen dos posibles tipos de peticiones que tiene que servir el servicio de SPARQL. El primer tipo es aquella petición que solicita la información de un recurso almacenado en el almacén RDF, en cuyo caso el LMS manda la petición al servicio SPARQL y este obtiene la información sobre el recurso solicitado. Y el segundo tipo que puede ser una petición que solicite información sobre una región del mapa. En este caso, la petición vendrá con el formato de una petición del WMS, por tanto se procesa la petición en el LMS y se transforma en una petición con la sintaxis de SPARQL que solicita la información de los recursos almacenados en RDF que se encuentran en el bounding box que se pasaba como parámetro en la petición original. Además de estas peticiones de

consultar información, también se pueden realizar peticiones de actualización, para ello utiliza Jena.

Existe diferencia entre la petición de edición y actualización, ya que el usuario puede editar la información asociada a un recurso, ya sea el nombre o la posición de este en el mapa, y visualizarlo en el cliente, pero con esta petición de edición los cambios no se actualizaran en la información del sistema. Si el usuario desea que las modificaciones realizas sean de forma permanente y modificar los datos fuente del sistema, es cuando tendrá que hacer la petición de actualización y es en ese momento cuando se actualizara la información en el almacén RDF, y esto desencadenara la actualización de la información en la base de datos geoespacial, de esta forma se consigue tener la información sincronizada tanto en la base de datos PostgreSQL como en el almacén RDF.

Hay disponible en la distribución de Apache Fuseki de un cliente de línea de comandos que permite interactuar con el servicio de SPARQL. Las peticiones que soporta el servicio de SPARQL son de la siguiente forma:

Petición de consulta

```
s-query --service http://localhost:3030/ds/query 'SELECT * {?s ?p ?o}'
```

Petición de actualización

```
s-update --service http://localhost:3030/ds/update --file=update.ru
```

3.2.4. Servicio de mapas

El servicio de mapas (WMS) recibe la petición del LMS y le envía la respuesta al mismo y este es el encargado de devolverle la imagen al cliente. El WMS está implementado con Mapserver y es el encargado de servir las peticiones de tipo geoespacial.

En el uso que se le da en este sistema principalmente es servir las peticiones que contienen imágenes, aunque este WMS soporta todas las peticiones que se especifican en el estándar OGC WMS 1.3.0, las obligatorias *GetCapabilities* y *GetMap*, así como la opcional *GetFeatureInfo*, esta última devuelve la información sobre el punto de interés seleccionado en el formato que se le especifique.

3.2.5. Cliente web

El cliente web esta implementado en JavaScript utilizando la librería OpenLayers y Bootstrap. Este cliente web permite visualizar la información, tanto la servida por el WMS como la información RDF. De esta forma si el usuario está haciendo modificaciones en los datos RDF podrá visualizarlos y darle una visión directa de lo que está modificando para evaluar posteriormente si quiere hacer los cambios permanentes o no.

La interacción del LMS con el cliente es la de recibir y servir las peticiones, al devolver las respuestas al LMS se encarga de mostrar la información tanto en formato texto, como la información representada en el mapa, y es el encargado de gestionar las peticiones de edición, que al realizarse también se ve afectada la información que se muestra en el cliente, ya que deberá de ser actualizada.

En el siguiente diagrama (figura 3.3) se puede observar la interacción del cliente con los diferentes módulos del sistema. En el apéndice A se encuentran los diagramas de secuencia explicados con mayor detalle.

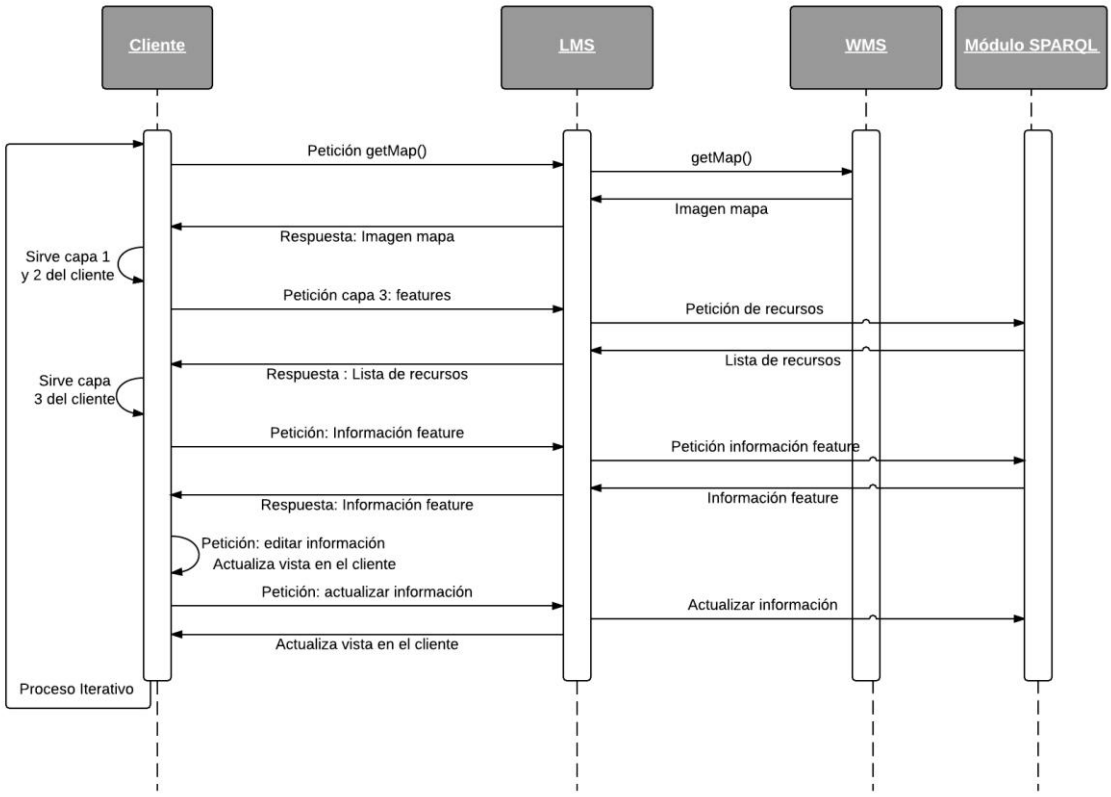

Figura 3.3: Diagrama de secuencia completo del sistema

Como he mencionado anteriormente el cliente está basado en la librería JavaScript OpenLayers. OpenLayers es una biblioteca JavaScript de código abierto desarrollado por la Open Source Geospatial Foundation (OSGeo) que permite la visualización de información geoespacial en aplicaciones web. Se ha elegido OpenLayers para desarrollar el cliente del LMS porque permite integrar un visor de mapas dentro del cliente web, es fácil la configuración para que sirva la información de un WMS, permite trabajar con capas de diferentes tipos (ráster, vectorial), y está en constante desarrollo ya que numerosos programadores trabajan con ella y continuamente se están añadiendo funcionalidades, y optimizando las ya existentes.

El visor que incluye el cliente web está formado por tres capas: una capa base que es la que sirve el mapa del mundo que la obtiene de un WMS externo, en concreto la capa que sirve la aplicación es OSM, que la obtiene del WMS de OSM, una segunda capa denominada que obtiene del WMS descrito anteriormente, implementado para este proyecto, que sirve la información cargada en la base de datos del sistema, y una tercera capa denominada LMS de tipo vectorial, que se obtiene de la información RDF del sistema, sobre la cual el cliente realiza las peticiones y en la que se muestran las modificaciones que este realiza.

En la figura 3.4 se puede ver la estructura conceptual de capas del cliente, gestionadas por el cliente web.

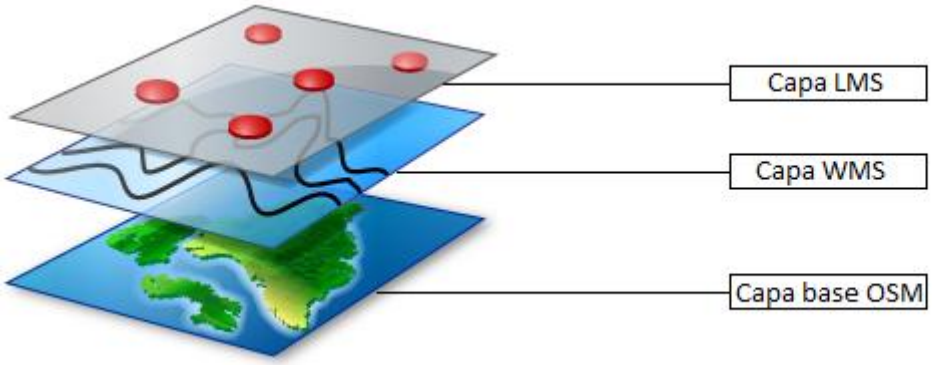

Figura 3.4: Estructura de capas del cliente

3.2.6. Base de datos

La base de datos es el recurso donde se almacenan los datos geográficos con los que trabaja el sistema. Dado que la naturaleza de los datos es geoespacial, la tecnología

utilizada para la creación y funcionamiento de la base de datos es PostgreSQL con el módulo de PostGIS que añade el soporte necesario para trabajar con objetos espaciales. La base de datos solo interactúa con el servicio de mapas, ya que es de esta de donde el WMS obtiene los datos que sirve al cliente.

La información almacenada puede ser modificada exclusivamente cuando el sistema recibe una petición de actualización que será direccionada al módulo de SPARQL, pero el sistema mantiene la información sincronizada, por tanto si se produce una modificación en el almacén de datos RDF, se producirá la misma actualización en la base de datos, con lo que esto implica, es decir, los datos que se le sirven al WMS cambiaran, por tanto la información que este sirve al cliente también se actualizara en el momento de la petición.

Se encuentra una descripción más en detalle de la implementación de la base de datos en el apéndice B.

3.2.7. Almacén RDF

El almacén RDF se obtiene a partir de los datos almacenados en la base de datos geoespacial. Para ello se genera un almacén de tipo TDB que contiene toda la información almacenada en la base de datos en formato RDF y posteriormente se carga en el servicio de SPARQL, para su uso en la parte de web semántica del sistema.

TDB es un componente de Apache Jena. Soporta todas las operaciones contenidas en la API de Jena. TDB se utiliza como un almacén de RDF de alto rendimiento. TDB está protegido contra la corrupción, las terminaciones de procesos inesperados y los fallos del sistema. Para que TDB soporte concurrencia se debe acceder mediante el uso de un servicio de SPARQL.

Los recursos almacenados en la instancia de TDB son los mismos que se encuentran en la base de datos geoespacial del sistema, ya que es a partir de estos de los que se genera la información que contiene este fichero.

Se encuentra una descripción más en detalle sobre la implementación de este en el apéndice B.

Capítulo 4. Demostrador

En este capítulo se presenta el demostrador del sistema, y la visualización de las diferentes funcionalidades que el sistema posee.

4.1. Pantalla principal

Cuando se arranca el sistema, nos aparece la pantalla principal del cliente web. En esta pantalla podemos ver un mapa que nos ofrece tres capas: una capa con la representación del mapa del mundo, una capa con la información cargada en el sistema, que obtiene del WMS desarrollado para el sistema que obtiene la información de la base de datos y una capa que obtiene de la información del almacén RDF.

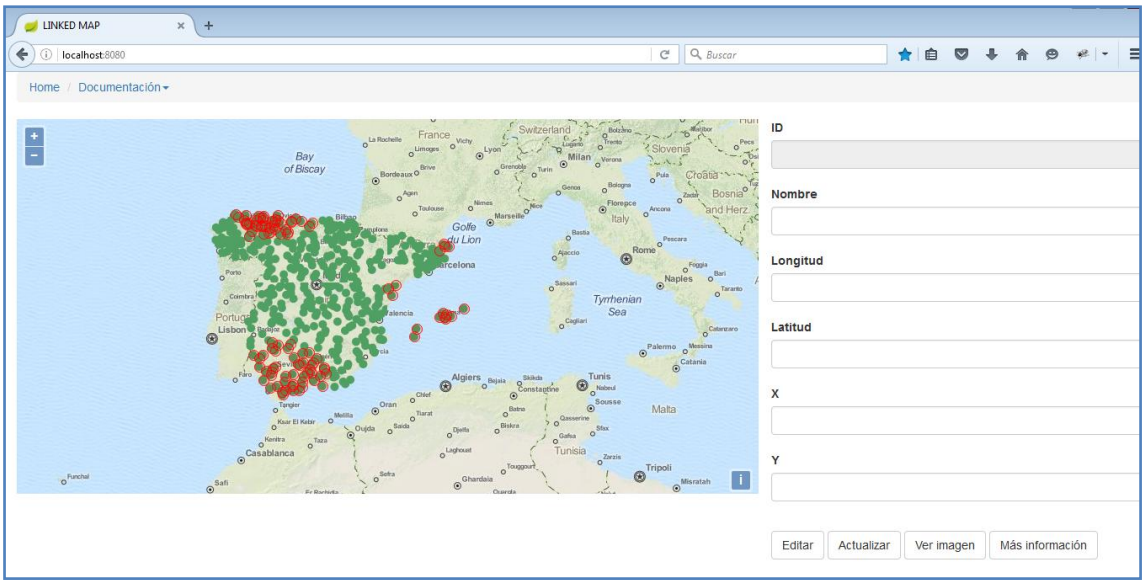

Figura 4.1: Pantalla de inicio de la aplicación

A nivel del servidor cargar la pantalla inicial de la aplicación genera una serie de peticiones hacia el LMS. Las peticiones generadas son de la siguiente forma:

```
http://localhost:8080/service/getMap?service=WMS&request=GetMap&
version=1.3.0&layers=ngbe&styles=&format=image%2Fpng&transparent=false
&height=256&width=256&srs=EPSG%3A3857&bbox=-469629.1017841229,
4852834.051769271,-391357.584820102,4931105.568733294
```

Como se puede observar es una petición *GetMap* con una serie de parámetros, es el LMS el encargado de procesarla y redireccionarla hacia el servicio de mapas o hacia el módulo de RDF. Mediante la cabecera HTTP "Accept" se indica al LMS como procesar la petición. Si la petición solicita una imagen ("*Accept: image/png*" por ejemplo) se redirecciona al WMS y como respuesta el LMS devuelve una imagen en formato PNG que se visualiza en el mapa, en caso de que la petición solicite datos RDF ("*Accept: text/turtle*" por ejemplo), la petición es gestionada por el LMS al módulo de web semántica y como respuesta se obtiene la información de los recursos asociados al área indicada en la petición (parámetro *bbox*) en formato RDF.

4.2. Opciones del cliente

A continuación se muestran diferentes respuestas que obtiene el cliente según la petición realizada.

4.2.1. Petición información

Al realizar una petición de más información sobre un punto el usuario puede visualizar en el desplegable donde aparece la información completa que se tiene sobre ese punto.

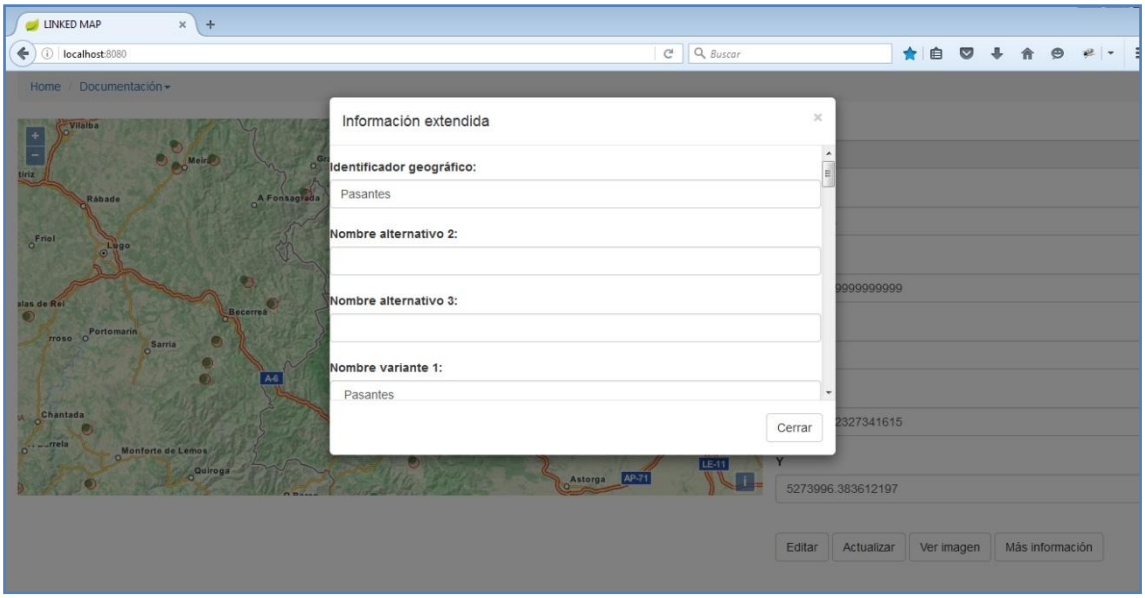

Figura 4.2: Petición más información

En este caso el cliente envía una petición solicitando información sobre un recurso al LMS, la petición generada es de la siguiente forma:

```
http://localhost:8080/informacion/resource/{id_recurso}
```

Dado que la petición es de información, esta la gestiona el módulo de RDF, por tanto se redirecciona a dicho módulo, que es el encargado de realizar la petición contra el almacén RDF, como respuesta se obtiene la información que contiene el almacén RDF sobre el recurso solicitado en formato RDF. Para facilitar la comprensión al usuario en el cliente se procesa la respuesta y se muestra en el formulario que se puede ver en la figura 4.2.

4.2.2. Petición edición

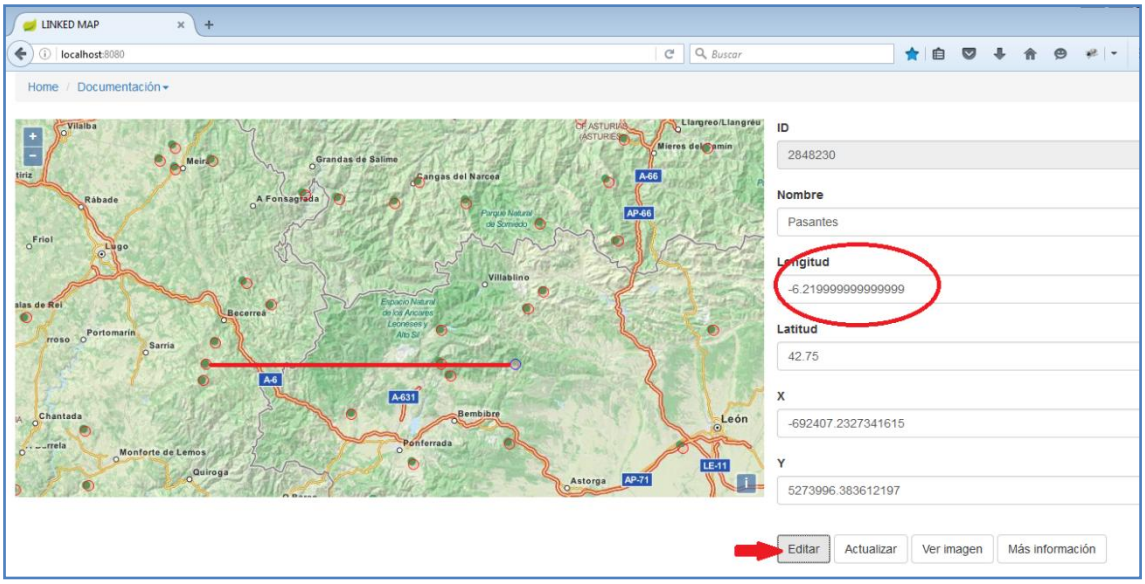

Figura 4.3: Petición editar

Al realizar una petición de edición en el cliente se puede visualizar el punto original y el editado, permitiendo así al usuario poder comprar ambos. La petición de edición no actualiza la información del sistema de forma permanente, por tanto no genera ninguna petición dirigida al LMS.

4.2.3. Petición actualización

Al realizar una petición de actualización se guardan los cambios en todos los puntos que hayan sido previamente modificados.

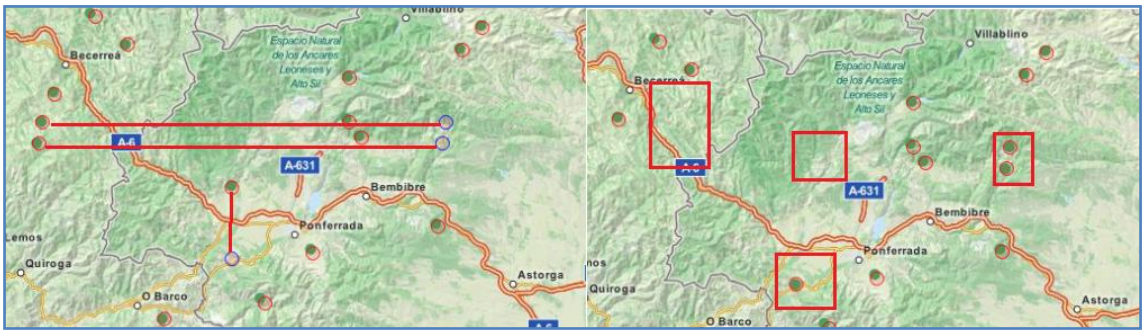

Figura 4.4: Petición actualización

Cuando en el cliente se genera una petición de actualización va dirigida al LMS, la petición es de la siguiente forma:

```
http://localhost:8080/service/updateRDF/resource/{id_recurso}
```

El LMS es el encargado de procesar la petición y actualizar el contenido del almacén RDF con la nueva información asociada al recurso y de actualizar la base de datos geoespacial del sistema, para que toda la información sea coherente y este actualizada. A su vez tras finalizar las operaciones de actualización de los almacenes de datos del sistema, se generan nuevamente peticiones del tipo *GetMap* como las explicadas anteriormente para el mapa que se está visualizando en el cliente.

Las peticiones se encuentran de manera más detallada en el apéndice E, correspondiente al manual de usuario.

Capítulo 5. Gestión del proyecto

En este capítulo se da una visión de los tiempos invertidos en el análisis y desarrollo del proyecto, para tener una idea general del trabajo empleado y en que fases del proyecto ha sido dedicado. Además de la metodología de trabajo aplicada y las herramientas de gestión.

5.1. Planificación y horas de trabajo

Para el desarrollo de este proyecto han sido necesarias 702 horas de trabajo, distribuidas en las tareas de aprendizaje, análisis y diseño, implementación, pruebas, documentación y reuniones, en la tabla 3 se muestra la planificación temporal de dichas tareas. Se puede encontrar descrito de forma más detallada en el apéndice D.

Tarea	May 15	Jun 15	Jul 15	Ago 15	Sep 15	Oct 15	Nov 15	Dic 15	Ene 16	Feb 16	Mar 16	Abr 16	May 16	Jun 16	Horas
Aprendizaje	■	■	■		■	■	■	■							135
Análisis y diseño					■	■	■								84
Implementación								■	■	■	■	■	■		242
Pruebas										■	■	■	■		76
Documentación											■	■	■		145
Reuniones	■	■			■	■	■	■	■	■	■	■	■	■	20

Tabla 3: Planificación del proyecto

5.2. Metodología de trabajo

La metodología de trabajo aplicada durante el desarrollo de este proyecto ha partido de una fase de análisis, posteriormente el estudio de las tecnologías más apropiadas para su

implementación, seguido de una fase detallada de diseño, y por último una fase de implementación y pruebas. A lo largo del desarrollo del proyecto se ha llevado un control de versiones y copias de seguridad.

En paralelo a todas las fases antes mencionadas se han llevado reuniones de forma frecuente con el director del proyecto, para concretar requerimientos del sistema, detalles de diseño y tecnologías a utilizar.

5.2.1. Control de versiones

Para mantener un control de versiones sobre las modificaciones que se han ido realizando en los diferentes módulos del sistema, se ha contado con un repositorio en Github, el cual permite tener un control de cambios de todos los fuentes del sistema, así como la posibilidad de volver a una versión anterior del sistema en caso de que algún a fichero resulte dañado.

5.2.2. Copias de seguridad

Durante el desarrollo del proyecto se han seguido diferentes técnicas para generar copias de seguridad. Para las copias de seguridad del código fuente y documentación del sistema se utilizaban las siguientes:

- Diariamente se actualizaba el repositorio en Github que contiene todos los ficheros que conforman el proyecto, con posibilidad de descarga. Además de la disponibilidad online, existían copias locales actualizadas en diferentes equipos.
- A su vez diariamente se realizaba una copia del repositorio de Github para almacenarla en Dropbox.
- Como copia de seguridad extra y completa de todo el sistema se tienen dos máquinas virtuales con todo el sistema desplegado y documentación asociada.

5.3. Herramientas de gestión

Para la gestión del proyecto se han utilizado las siguientes tecnologías:

Tecnología	Uso
Github	Plataforma control de versiones, copia de seguridad, y corrección de incidencias encontradas
Dropbox	Copia de seguridad
Procesadores de textos	Creación de documentación
Hojas de cálculo	Creación de documentación, organización de listas de tareas
Visio	Creación de diagramas para la documentación

Tabla 4: Herramientas de gestión

Se puede encontrar una explicación más detallada sobre estas tecnologías en el apéndice C.

Capítulo 6. Conclusiones

En este capítulo se presentan los resultados obtenidos en el desarrollo del proyecto, posibles vías de desarrollo futuras y para finalizar una conclusión a nivel personal.

6.1. Resultados

El proyecto se ha desarrollado cumpliendo los requisitos y funcionalidades especificadas con anterioridad en este documento. Con la implementación de este proyecto se ha conseguido ofrecer una herramienta que permite a los usuarios trabajar con información geográfica enlazada con *Linked Data*, que era una de los objetivos principales de este proyecto. Con este se pretendía demostrar que la explotación de datos geoespaciales mediante *Linked Data*, es una solución viable y con grandes posibilidades de desarrollo. Observando la arquitectura y las tecnologías que se han utilizado a lo largo del desarrollo e implementación de este proyecto se puede observar que el resultado obtenido es un sistema de fácil configuración, y que no requiere una gran cantidad de recursos para instalarlo y utilizarlo.

En resumen se ha conseguido:

- Un sistema compuesto por módulos independientes que permiten la integración de varios almacenes de datos (base de datos geoespacial y almacén RDF).
- Un WMS específico que trabaja con los datos del sistema
- Un servidor de SPARQL que trabaja con los datos del sistema.
- Un servicio LMS ligero y portable, que permite el trabajo simultaneo con información geográfica proporcionada por WMS externos e información en formato RDF.

- Un cliente web sencillo que permite al usuario trabajar con información geográfica y *Linked Data* de manera sencilla, ya que es transparente para el usuario.

Todo ello se encuentra alojado en un repositorio en Github, se puede acceder a él mediante el siguiente enlace:

`https://github.com/IAAA-Lab/LinkedMapService`

6.2. Trabajo futuro

Debido a que la adopción del *Linked Data* por los grandes productores de información geográfica es un proceso lento, con este proyecto se ha querido demostrar la viabilidad de este tipo de sistemas. Teniendo esto en cuenta y el diseño modular del proyecto, puede ser utilizado como base para futuras investigaciones en este campo, añadiendo nuevas funcionalidades a las ya existentes, pudiendo orientar esta investigación hacia los grandes productores de información o hacia cualquier tipo de usuario mediante alguna aplicación que permita compartir información entre ellos sobre algún tema en concreto.

Del lado de los grandes productores de información sería necesario expandir alguno de los módulos del proyecto, como por ejemplo el de almacenes RDF, en este caso el sistema trabaja con un único almacén RDF, pero existe la posibilidad de trabajar con varios, modificando solamente dicho módulos, es posible que también fuera necesario añadir algún tipo de funcionalidad al LMS para cubrir alguna necesidad de estos productores de información.

En el caso de querer generar una aplicación para usuarios, en la cual compartan información entre ellos, que no sea de carácter oficial, por ejemplo rutas de senderismo, o algún otro tipo de rutas, sería necesario añadir alguna funcionalidad al sistema, pero con las herramientas ofrecidas en el LMS se tendría una buena base de la que partir.

Este tipo de proyecto, puede ser explotado por diferentes tipos de profesionales desde usuarios finales, evaluadores GIS, evaluadores de web semántica, investigadores, etc. es ahí donde reside el potencial del proyecto, ya que engloba diversos campos.

6.3. Conclusión personal

La realización de este proyecto me ha aportado el conocimiento de nuevas tecnologías como son el caso de OpenLayers, Spring Boot, conocimientos sobre *Linked Data*, entre otras, dado que a lo largo de la carrera no había tenido la oportunidad de trabajar con ellas. Para ello he tenido que hacer uso de documentación de referencia, numerosos tutoriales y consultas a mi director del proyecto, dado la complejidad de alguna de ellas. Todos estos nuevos conocimientos sobre tecnologías me van a resultar muy útiles para mi futuro trabajo en una empresa, ya que son tecnologías que tienen un uso extendido.

Además de conocimientos técnicos, también me ha permitido adquirir una serie de buenos hábitos para el desarrollo de este proyecto y proyectos futuros, respecto a la organización del trabajo, especificación de los requisitos mediante reuniones con mi director, y ver que esos requisitos se satisfacían, abordar la visión de un proyecto desde cero, diseñando su arquitectura y tomando decisiones de diseño.

En resumen el desarrollo de este proyecto me ha ayudado y dado unas bases con las que empezar a afrontar mi futuro laboral en el mundo de la empresa. Porque con el desarrollo de este proyecto me he tenido que enfrentar a problemas que con anterioridad a lo largo de la carrera no había tenido ocasión.

El hecho de tener que desarrollar un proyecto de una envergadura media, que tiene que ofrecer una solución real a un problema del mundo real, y puede ser utilizado por otros usuarios para la explotación de información geográfica, o tal vez utilizarlo en futuras investigaciones es algo gratificante, porque ves que tu trabajo sirve para algo real.

Apéndices

Apéndice A. Análisis y diseño del sistema

En este apéndice se explica más en detalle el análisis y diseño del sistema que se ha realizado a lo largo de la elaboración de este proyecto, para poder comprender completamente el sistema desarrollado.

En esta sección se presentan documentos de requisitos del sistema, diagramas de casos de uso, diagramas de secuencia y diagrama de clases.

A.1. Requisitos del sistema

En este apartado se definen y analizan los requisitos que tiene que satisfacer el sistema. Con este listado se puede comprobar que el sistema desarrollado se ajusta a los requisitos especificados.

En la siguiente tabla (Tabla 9) se enumeran los requisitos funcionales y no funcionales que tiene que satisfacer el sistema.

ID Requisito	Requisito
RF-1	Visualizar un mapa procedente de un WMS en la aplicación
RF-2	Hacer zoom en el mapa
RF-3	Desplazar el mapa
RF-4	Visualizar 2 capas (o más) en el mapa
RF-5	Pasar el cursor por un punto y obtener una breve descripción
RF-6	Obtener información detallada de un punto seleccionado del mapa
RF-7	Editar información sobre un punto
RF-8	Visualizar punto original y punto editado

ID Requisito	Requisito
RF-9	La aplicación utilice diferentes iconos para diferentes tipos de POI dibujados en el mapa
RF-10	Actualizar la información en los almacenes de datos
RF-11	La interfaz deberá de ser sencilla e intuitiva
RNF-1	El sistema debe ser sencillo y ligero
RNF-2	El usuario tiene que tener documentación con ejemplos de las posibles peticiones que soporta el sistema

Tabla 5: Requisitos del sistema

A.2. Casos de uso

A continuación se analizan los posibles casos de uso que podrán darse en la aplicación, mediante algunos diagramas de caso de uso.

En la figura A.1 se muestran las diferentes acciones que se pueden realizar con la aplicación.

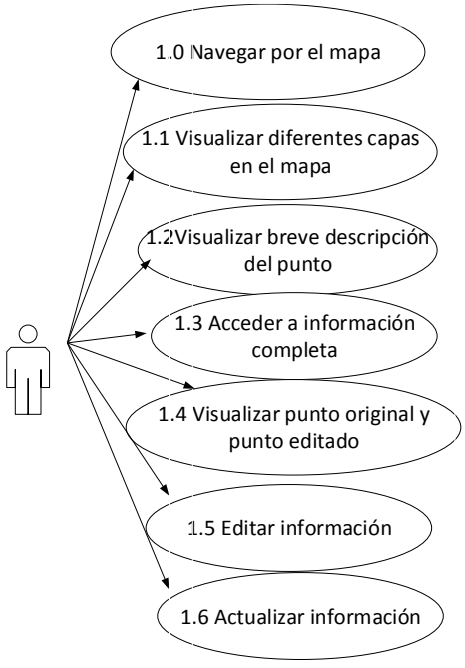

Figura A.1: Diagrama de casos de uso de la aplicación

Caso de uso: 1.0 Navegar por el mapa	
Propósito:	Poder desplazar el mapa, es decir, arrastrarlo y que se pueda ver otra zona de mapa que no esté dentro del campo de visualización inicial.
Actor principal:	Usuario de la aplicación
Descripción:	En esta opción el usuario que esté utilizando la aplicación podrá desplazar el mapa ayudándose del ratón, para mover el mapa que esta visualizando actualmente, permitiendo de esta forma que se visualicen otras secciones del mapa que anteriormente no estaban visibles
Flujo básico:	Clic en el mapa del cliente y arrastrar
Flujo alternativo:	El WMS está caído y no sirve el mapa. El usuario no podrá visualizar el mapa, y recibirá una imagen en blanco, o un icono de objeto no encontrado

Caso de uso: 1.1 Visualizar diferentes capa en el mapa	
Propósito:	Permitir visualizar diferentes capas sobre el visor de mapa que ofrece el cliente de la aplicación
Actor principal:	Usuario de la aplicación
Descripción:	En esta opción el usuario puede visualizar en el mapa diferentes capa servidas por diferentes servicios
Flujo básico:	Arrancar la aplicación./Realizar cada operación de edición./Realizar una operación de actualización
Flujo alternativo:	El WMS está caído y no sirve el mapa. El usuario no podrá visualizar el mapa, y recibirá una imagen en blanco, o un icono de objeto no encontrado

Caso de uso: 1.2 Visualizar breve descripción de un punto	
Propósito:	Ver breve descripción del punto al pasar por encima con el cursor del ratón
Actor principal:	Usuario de la aplicación
Descripción:	En esta opción el usuario obtiene a través de una ventana emergente una breve descripción del punto por el que está pasando el cursor del ratón
Flujo básico:	Pasar el ratón por los diferentes puntos del mapa
Flujo alternativo:	El WMS está caído y no sirve el mapa. El usuario no podrá visualizar el mapa, y recibirá una imagen en blanco, o un icono de objeto no encontrado. Existen problemas al leer la información y se sirve la descripción en blanco

Caso de uso: 1.3 Obtener información completa	
Propósito:	Acceder a la información completa de un punto
Actor principal:	Usuario de la aplicación
Descripción:	En esta opción el usuario seleccionara un punto y hace clic sobre "más información" y obtiene en la pantalla toda la información asociada al punto
Flujo básico:	Hacer clic sobre "más información"
Flujo alternativo:	El WMS está caído y no sirve el mapa. El usuario no podrá visualizar el mapa, y recibirá una imagen en blanco, o un icono de objeto no encontrado. Existen problemas al leer la información y se sirve la descripción en blanco

Caso de uso: 1.4 Visualizar puntos editados	
Propósito:	Visualizar punto original y punto editado
Actor principal:	Usuario de la aplicación
Descripción:	En esta opción el usuario puede ver el punto original y el editado y compararlos
Flujo básico:	Editar un punto y visualizar el mapa
Flujo alternativo:	El WMS está caído y no sirve el mapa. El usuario no podrá visualizar el mapa, y recibirá una imagen en blanco, o un icono de objeto no encontrado.

Caso de uso: 1.5 Editar información	
Propósito:	Editar la información asociada a un punto
Actor principal:	Usuario de la aplicación
Descripción:	En esta opción el usuario puede modificar la información asociada a un punto, sin alterar los datos fuentes originales, solo previsualizando los cambios en el mapa
Flujo básico:	Seleccionar un punto, modificar la información y dar al botón "Editar"
Flujo alternativo:	El WMS está caído y no sirve el mapa. El usuario no podrá visualizar el mapa, y recibirá una imagen en blanco, o un icono de objeto no encontrado. Error en la petición de edición, se notifica al usuario y se cancela la operación

Caso de uso: 1.6 Actualizar información	
Propósito:	Actualizar la información asociada a un punto en los almacenes de datos
Actor principal:	Usuario de la aplicación
Descripción:	En esta opción el usuario puede actualizar la información asociada a un punto, y modificarla en los almacenes de datos del sistema
Flujo básico:	Hacer clic en el botón actualizar
Flujo alternativo:	Error en la conexión con los almacenes de datos, se le notifica al usuario y se cancela la operación

Para cada uno de estos casos de usos se va a mostrar un ejemplo. En los siguientes diagramas se pueden observar posibles trazas de ejecución de los casos mencionados anteriormente.

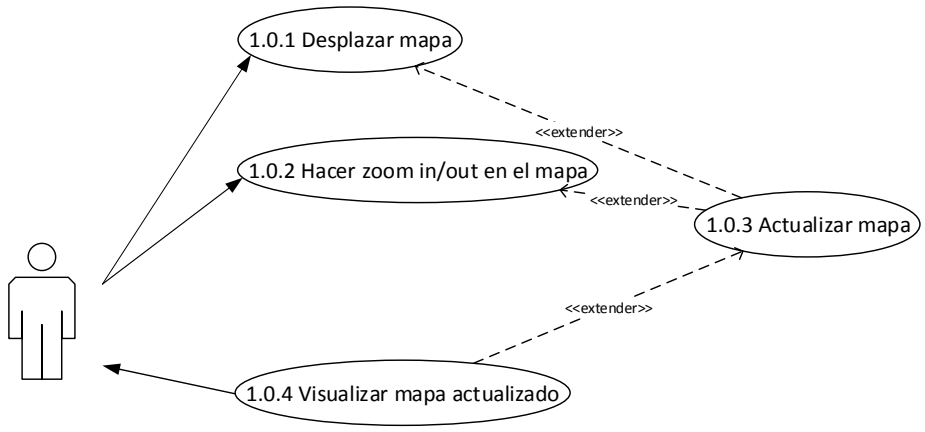

Figura A.2: Casos de uso 1.0

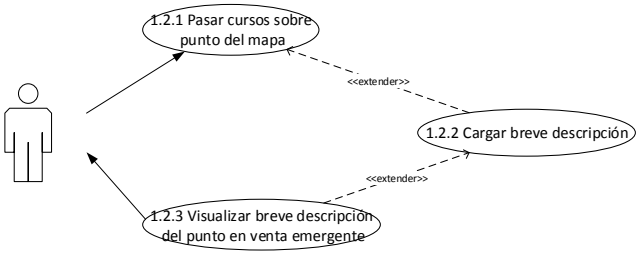

Figura A.3: Casos de uso 1.1

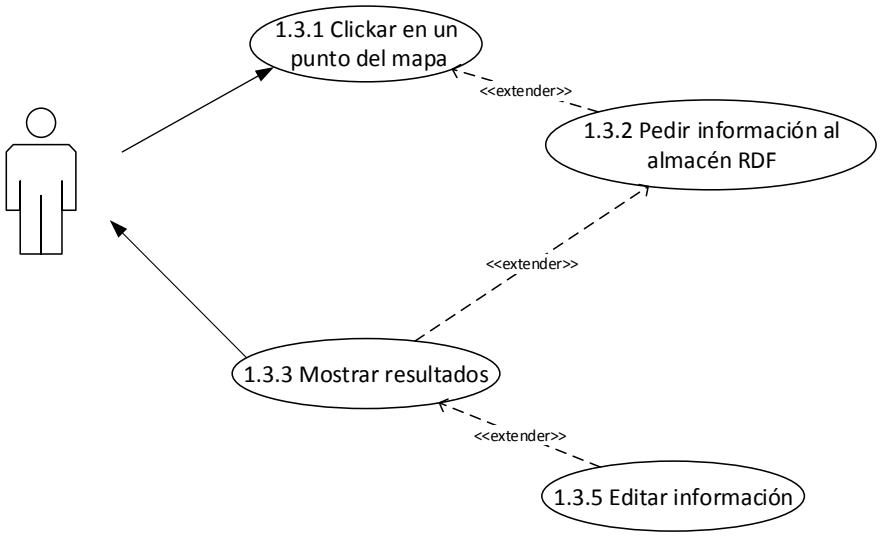

Figura A.4: Casos de uso 1.2, 1.3 y 1.4

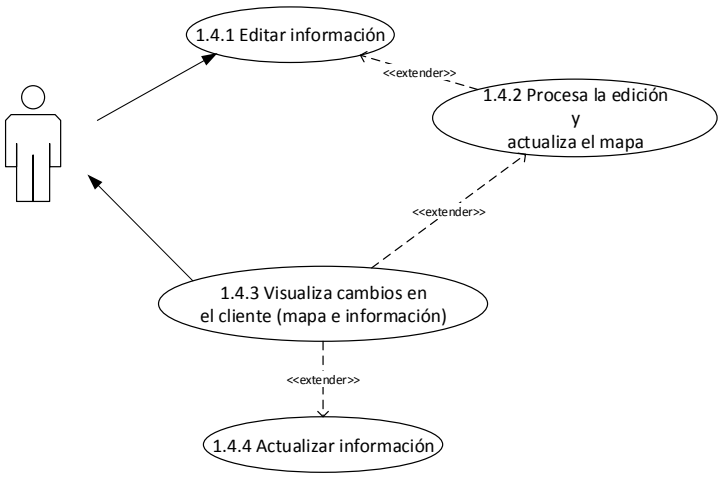

Figura A.5: Casos de uso 1.5

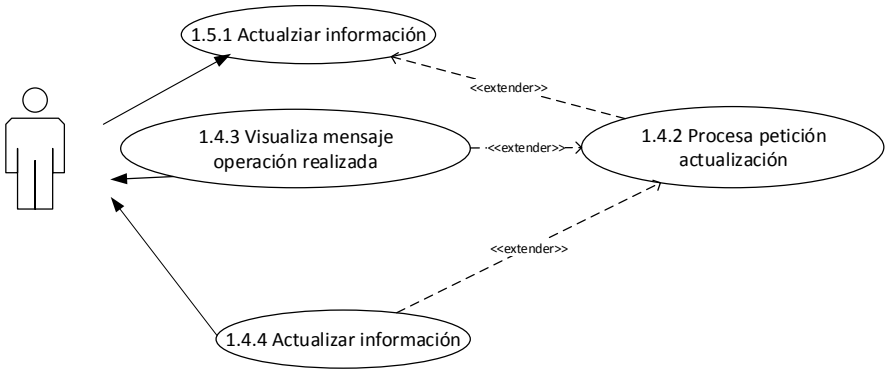

Figura A.6: Casos de uso 1.6

A.3. Diagramas de secuencia

En esta sección se muestran diferentes diagramas de secuencia del sistema, y se explican brevemente.

Diagrama de arranque

En la figura A.7 se puede ver el diagrama de secuencia de arranque de la aplicación. En él se muestra que ocurre cuando el usuario accede a la aplicación, como se realiza la carga de la página de inicio del sistema, y como el cliente se conecta a través del LMS, y este lo redirecciona al WMS para obtener los datos del mapa que sirve el cliente en el visor que contiene la página principal.

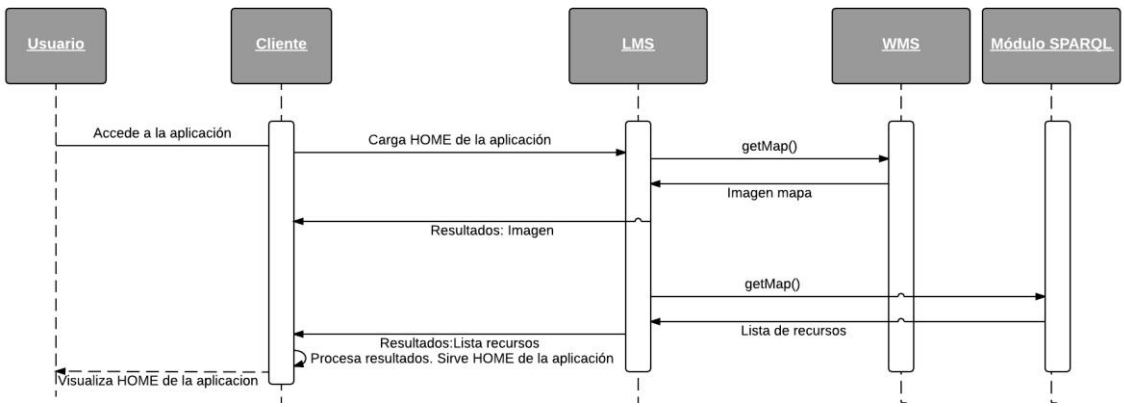

Figura A.7: Diagrama de secuencia arranque del sistema

Diagrama petición *GetMap*

En la figura A.8 se puede ver el diagrama de secuencia de una petición en la que el usuario solicita una imagen del mapa (o parte de este). Para ello vemos que el usuario tiene que realizar una selección de la sección del mapa de la cual quiere la imagen en el

cliente, posteriormente el cliente envía la petición el LMS y este es el que detecta que la petición es de tipo WMS, y se la redirecciona al WMS del sistema como una petición de *GetMap()*. La petición *GetMap()* llega al WMS, quien la procesa y devuelve la imagen solicitada al LMS y este es el encargado de servírsela al cliente, y que el usuario pueda visualizar el resultado.

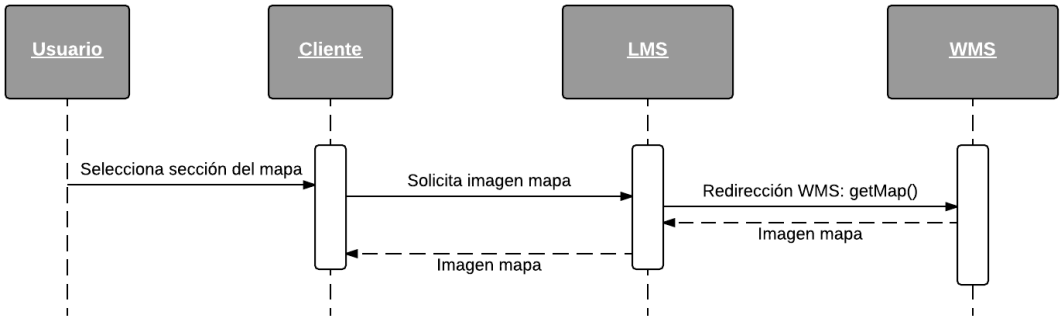

Figura A.8: Diagrama de secuencia petición imagen

Diagrama edición

En la figura A.9 se puede ver el diagrama de secuencia de una petición de edición de información. En esta petición el usuario puede querer editar información "texto" o incluso las coordenadas (latitud/longitud) del punto, alterando así su representación en el mapa.

El usuario edita la información deseada en el cliente, al ser una petición de edición y no de actualización de la información, no se envía petición al LMS, ya que es el cliente el encargado de procesarla, como es una petición de edición los cambios podemos decir que son temporales, ya que no alteran los datos fuentes originales del sistema, pero sí que se altera su visualización en el cliente para que el cliente pueda ver la modificación que ha realizado. Por tanto al procesar la petición en el cliente este hace las modificaciones correspondientes, y actualiza la vista del mismo, permitiendo al usuario ver los cambios realizados, y comparar entre los puntos e información original y los editados.

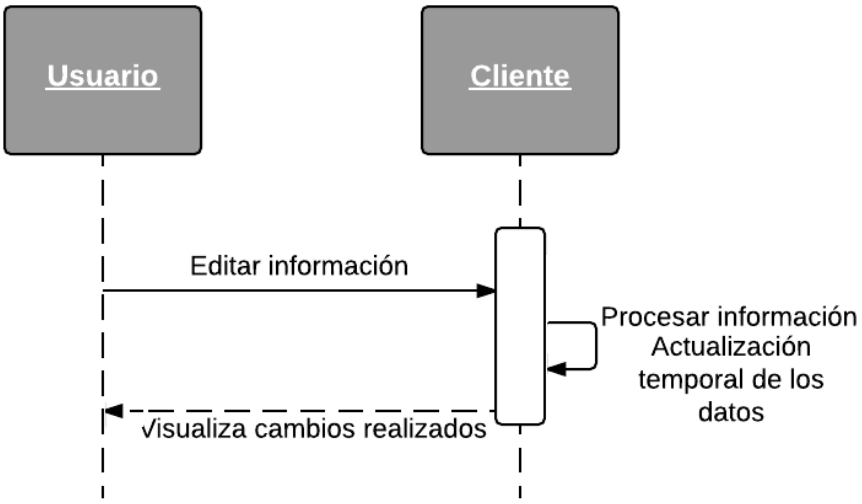

Figura A.9: Diagrama de secuencia petición edición

Diagrama actualización

En la figura A.10 se puede observar el diagrama de secuencia de una petición de actualización, en este caso el cliente quiere actualizar la información, después de una o varias peticiones de edición. Por tanto el cliente realiza una petición de actualizar desde el cliente, pasándola al LMS, que la procesa para mandarla al módulo de SPARQL del sistema, quien la trata y realiza los cambios en los almacenes de datos del sistema, para modificar los datos fuentes del sistema, si en el proceso no se produce ningún error envía al LMS un respuesta como que el proceso ha finalizado correctamente, y el LMS sirve al cliente la respuesta de que la actualización ha sido realizada correctamente, y en caso de ser necesario la actualización de la información visualizada en el cliente (mapa/información).

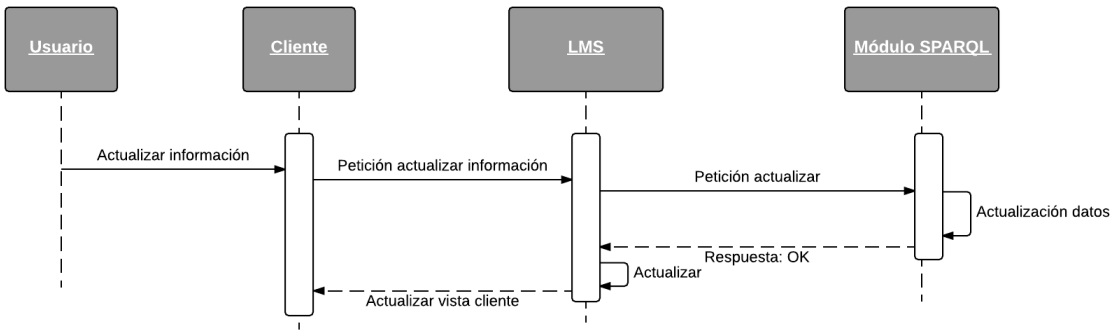

Figura A.10: Diagrama de secuencia petición actualizar

A.4. Clases y objetos

En esta sección se presenta el diagrama de clases y objetos del sistema desarrollado, ampliando así la información sobre la implementación lógica del sistema. Dado que el sistema tiene dos partes claramente diferenciadas, el módulo orientado al servicio de mapas y el módulo RDF, a continuación se muestran los diagramas de clases correspondientes a dichos módulos.

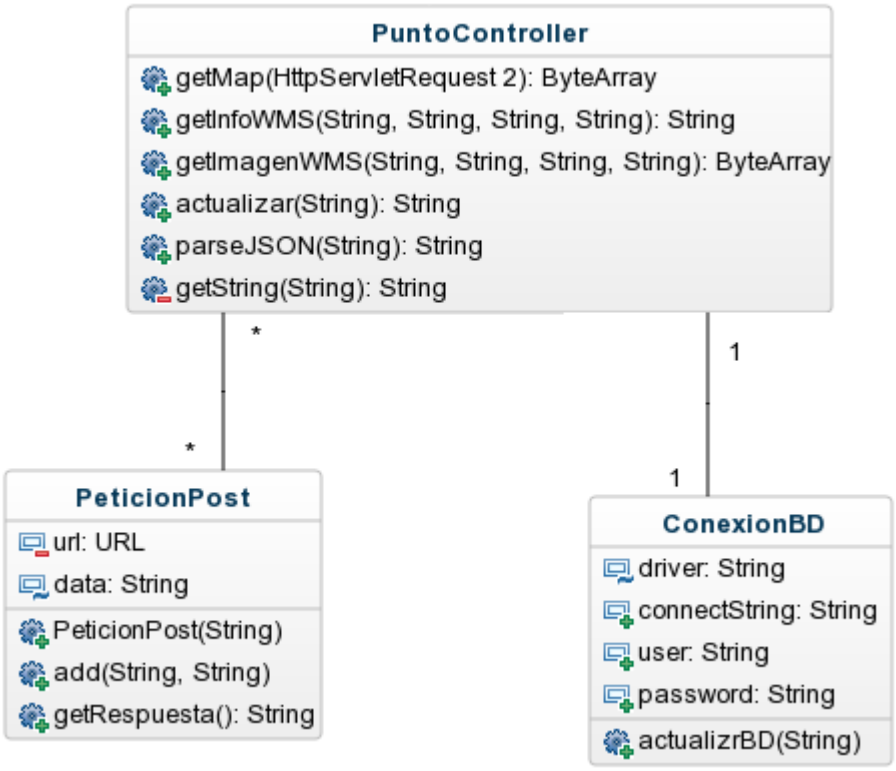

Figura A.11: Diagrama de clases módulo servicio de mapas

En la figura A.11 se puede ver las clases que componen el módulo del sistema relacionado con el servicio de mapas. La clase más compleja es “PuntoController” que como su nombre indica es el controlador de todas las peticiones por parte del cliente dirigidas hacia el servicio de mapas. Esta clase contiene los métodos de redirección y actualización de esta parte de la aplicación. La clase “PeticionPost” es la que permite crear las URLs adecuadas para el servicio de mapas, y por último la clase “conxionBD” es la que contiene el método para conectar la aplicación con la base de datos y desde ella permitir actualizar los valores.

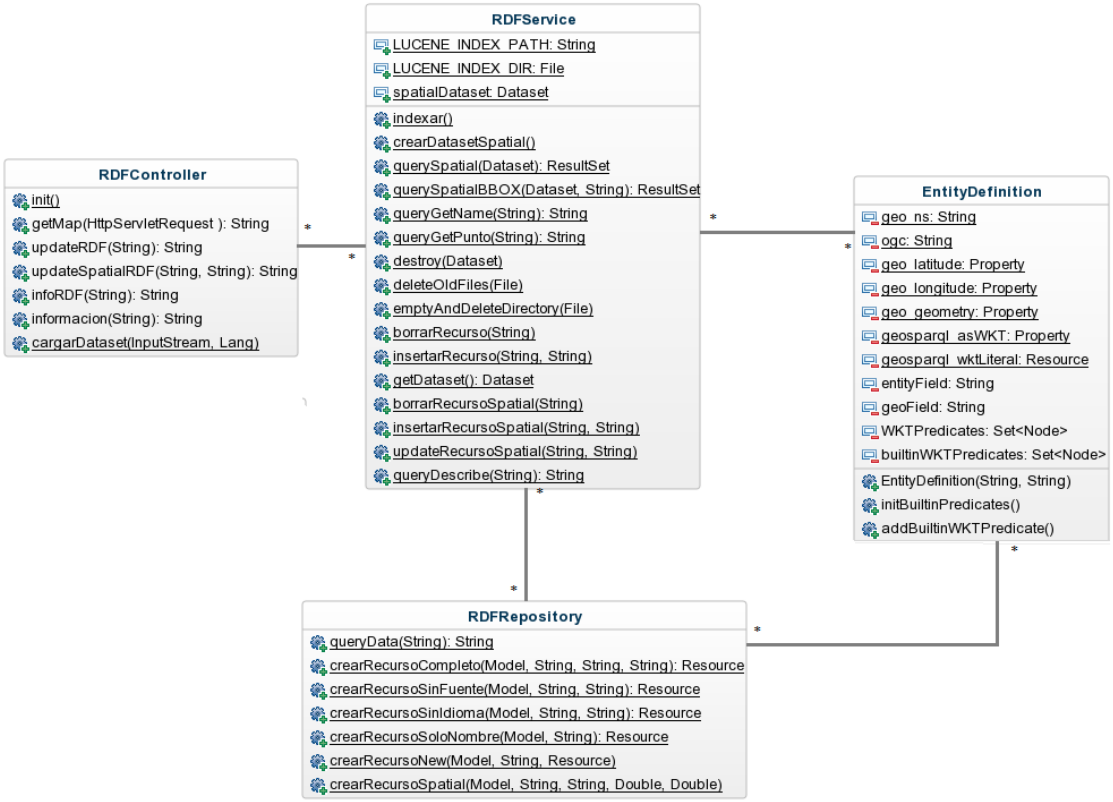

Figura A.12: Diagrama de clases módulo RDF

En la figura A.12 se puede ver las clases que componen el módulo del sistema relacionado con RDF. La clase más compleja es “RDFController” que como su nombre indica es el controlador de todas las peticiones por parte del cliente dirigidas al punto de SPARQL. Esta clase contiene los métodos de redirección y actualización de esta parte de la aplicación, además de las clases “RDFService” y “RDFRepository” que son las que contiene las funciones de crear peticiones y recursos para el servicio de SPARQL. La clase “EntityDefinition” es una clase necesaria para poder crear recursos en formato RDF que sigan las especificaciones y los vocabularios que se describen más adelante (ver sección B.2.2).

Dado que el cliente de la aplicación es un cliente web implementado con las librerías Openlayers 3 y Bootstrap, los fuentes para la implementación del mismo son ficheros Javascript. La estructura que tiene el cliente es la siguiente:

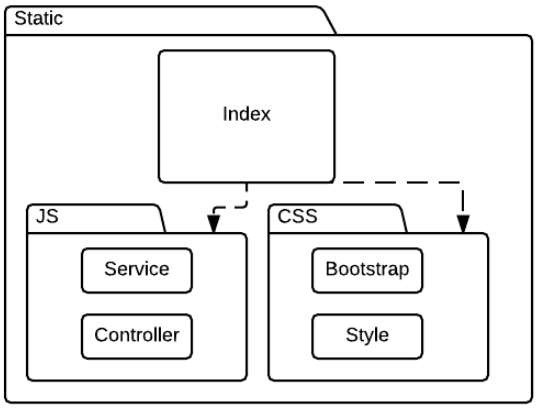

Figura A.13: Diagrama del cliente

Index representa el fichero index.html del cliente en el cual se cargan los ficheros de la carpeta CSS correspondientes a Bootstrap y la hoja de estilos, y en la carpeta JS los ficheros service.js y controller.js son los que contienen las funciones implementadas con la librería Openlayers. El fichero "service.js" es el que gestiona la creación y servicio del mapa del cliente, mientras que en el fichero "controller.js" están las funciones para gestionar las peticiones de edición y actualización.

Apéndice B. Almacenes de datos

En este apéndice se describen los almacenes de datos del sistema.

B.1. Base de datos

En esta sección se expone todo el proceso para la creación de la base de datos, así como su implementación y estructura de almacenamiento.

La información fuente con la que trabajara esta base de datos es el "*Nomenclátor geográfico básico de España*", que es una base de datos que contiene la información sobre topónimos oficiales correspondientes a las comunidades autónomas, las provincias, las islas, los municipios y las entidades locales de población y los topónimos correspondientes a la orografía, hidrografía, vías de comunicación, comarcas naturales y otras formaciones georeferenciadas que hayan sido aprobadas por la Administración pública competente y por el Consejo Superior Geográfico, y todo ello siguiendo la normativa europea INSPIRE.

B.1.1. Creación de la base de datos

Ha sido necesario crear una base de datos con el sistema gestor de base de datos PostgresSQL con el módulo de PostGIS que añade el soporte para trabajar con objetos espaciales.

La base de datos está compuesta por una tabla denominada *ngbev2013* que contiene toda la información proporcionada por el fichero fuente *Nomenclator 2013*, a la cual se le han añadido una serie de atributos, que son necesarios para algunas de las funcionalidades del sistema. Estos atributos añadidos tienen la geometría con la que posteriormente trabajara el servicio de mapas del sistema y el CRS.

B.1.2. Estructura de la base de datos

A continuación se muestra los atributos que componen la tabla que compone la base de datos.

Atributo	Descripción	Tipo de dato
ID	Identificador numérico único para cada registro.	Entero
NOMBRE_EXTENDIDO	Forma de uso referencial y principal de una entidad geográfica.	Cadena de caracteres
IDENTIFICADOR_GEOGRÁFICO	Identificador único de la instancia de localización. Topónimo referido a una entidad geográfica que está al mismo nivel de uso que los indicados en nombre alternativo 2 y nombre alternativo 3.	Cadena de caracteres
NOMBRE_ALTERNATIVO (2 y 3)	Topónimo referido a una entidad geográfica que está al mismo nivel de uso que el identificador geográfico.	Cadena de caracteres
NOMBRE_VARIANTE (1,2 y 3)	Topónimo de uso menor o restringido, referido a una entidad que tiene un identificador geográfico o un nombre alternativo.	Cadena de caracteres
FUENTE	Para el nombre extendido, identificador geográfico, nombre alternativo y nombre variante existen distintos campos en los que especifican la fuente origen del topónimo.	Cadena de caracteres
IDIOMA	En esta primera versión se han cumplimentado los idiomas correspondientes al nombre extendido, identificador geográfico, nombre alternativo y nombre variante, en base al resultado de la comparación con los del Nomenclátor Geográfico Conciso de España v.1.1. Para ello se ha seguido la Norma ISO 639-2.	Cadena de caracteres
ESTATUS	Información cualitativa que permite discernir el grado de normalización del topónimo. Se ha considerado topónimo oficial cuando ha sido declarado y publicado como tal por un órgano competente mediante el acto administrativo correspondiente.	Cadena de caracteres
LATITUD_ETRS89_REGCAN95	Coordenadas geográficas dadas en el sistema de referencia especificado en el campo Sistema_Referencia_ETRS89_REGCAN95[6].	Real

[6] El ETRS89 (Sistema de Referencia Terrestre Europeo 1989), es un sistema de referencia geodésico ligado a la parte estable de la placa continental europea.

Atributo	Descripción	Tipo de dato
LONGITUD_ETRS89_REGCAN95	Coordenadas geográficas dadas en el sistema de referencia especificado en el campo Sistema_Referencia_ETRS89_REGCAN95.	Real
HUSO_ETRS89_REGCAN95	Huso de la proyección UTM en el que se encuentra el topónimo.	Entero
XUTM_ETRS89_REGCAN95	Coordenadas UTM en el sistema de referencia especificado en el campo Sistema_Referencia_ETRS89_REGCAN95.	Real
YUTM_ETRS89_REGCAN95	Coordenadas UTM en el sistema de referencia especificado en el campo Sistema_Referencia_ETRS89_REGCAN95.	Real
SISTEMA_REFERENCIA_ETRS89_REGCAN95	Sistema de referencia geodésico en el que se encuentran especificadas todas las coordenadas cuyo campo se denomina como ETRS89_REGCAN95.	Cadena de caracteres
HOJA MTN_25	Número de hoja correspondiente al Mapa Topográfico Nacional a escala 1:25.000 (MTN25) en el que se encuentra el topónimo.	Cadena de caracteres
CÓDIGO_INE	Se ha asignado el código INE[7] a cada registro toponímico, considerándose once cifras (correspondientes al nivel de "Núcleo de Población") para los registros vinculados a entidades contempladas en el Instituto Nacional de Estadística y cinco cifras (correspondientes al nivel de "Municipio") para el resto de topónimos.	Cadena de caracteres

---

En España en 1995, la compensación de la red geodésica de Canarias, dentro del marco de la Red Geodésica Nacional por Técnicas Espaciales (REGENTE), supuso la materialización del sistema denominado REGCAN95, completamente compatible con el sistema ETRS89.

La adopción de estos sistemas de referencia permitió una completa integración de la cartografía oficial española con los sistemas de navegación y la cartografía de otros países europeos.

[7] El código INE es un identificador único creado por el Instituto Nacional de Estadística para cada entidad de población española.

Atributo	Descripción	Tipo de dato
CÓDIGO_NGBE[8]	Clasificación de tipo de entidad establecida en el marco de desarrollo del Nomenclátor Geográfico Básico de España (NGBE).	Cadena de caracteres
CRS	Sistema de referencias y coordenadas en las que está almacenado el punto	Cadena de caracteres
PUNTO	Geometría que representa la localización del punto representada en el sistema de coordenadas que indicar el CRS	Geometría

Tabla 6: Tabla *ngbev2013* de la base de datos

Codificación detallada:

Código NGBE	Descripción
1.0	Estado
1.1.1	Comunidad autónoma
1.1.2	Ciudad autónoma
1.2	Provincia
1.3	Municipio
1.4	EATIM (Entidad de ámbito territorial inferior al municipio)
1.5	Isla administrativa
1.6	Comarca administrativa
1.7	Condominios, mancomunidades y enclaves
2.1.0	Capital de Estado
2.1.1	Capital de comunidad autónoma y ciudad autónoma
2.1.2	Capital de provincia
2.1.3	Capital de municipio
2.1.4	Capital de EATIM (Entidad de ámbito territorial inferior al municipio)
2.1.5	Entidad colectiva
2.1.6	Otras entidades menores de población
2.1.7	Distrito municipal

[8] Código NGBE: Clasificación de tipo de entidad establecida en el marco de desarrollo del Nomenclátor Geográfico Básico de
España (NGBE).

2.1.8	Barrio
2.1.9	Entidad singular INE (Instituto Nacional de Estadística)
2.2.1	Instalación, construcción abierta
2.2.2	Edificación
2.3.1	Vértice Geodésico
2.3.2	Hitos de demarcación territorial
2.3.3	Hitos en vías de comunicación
3.1.1	Aeropuerto
3.1.2	Aeródromo
3.1.3	Pista de aviación, helipuerto
3.2.1	Puerto
3.2.2	Instalación portuaria
3.2.3	Muelle
3.3.1	Carretera
3.3.2	Camino , vía pecuaria
3.3.3	Vía urbana
3.3.4	Ferrocarril
4.1.1	Alineación montañosa
4.1.2	Montaña
4.1.3	Paso de montaña
4.1.4	Llanura
4.1.5	Depresión
4.1.6	Vertiente
4.2.1	Comarca geográfica
4.2.2	Paraje
4.2.3	Elemento puntual del paisaje
4.3.1	Parque Nacional, Parque Natural
4.3.2	Espacio protegido restante
5.1	Curso natural de agua
5.2	Masa natural de agua
5.3	Curso artificial de agua
5.4	Embalse
5.5	Hidrónimo puntual
5.6	Glaciar

6.1.1	Mar
6.1.2	Entrante costero, estrecho marítimo
6.2.1	Saliente costero
6.2.2	Playa
6.2.3	Isla
6.2.4	Otro relieve costero
6.3	Relieve submarino

Tabla 7: Codificación NGBE

Fuente: Nomenclátor Geográfico Básico de España (NGBE v.2013)

B.1.3. Carga de datos

Los datos se dan en formato .mdb, es decir, en un fichero de ACCESS, por tanto he creado una aplicación Java que permite la migración de los datos a la base de datos de la aplicación.

B.1.4. Adaptación de los datos

Dado que los datos almacenados en la base de datos están representados con los sistemas de referencia geodésicos oficiales de España (ETRS89 para Península, Islas Baleares, Ceuta y Melilla y REGCAN95 para Islas Canarias), y la aplicación también trabaja con mapas proporcionados por OpenStreetMap he tenido que realizar un cambio de sistema de referencias, para que ambos mapas trabajen con el mismo sistema de coordenadas. Para ellos he usado el programa Geokettle con el cual he realizado un modelo de trasformación (figura B.1) que permite cambiar el sistema de referencias de los objetos y a su vez crear un nuevo campo punto, que será el que posteriormente representaremos en el mapa de la aplicación.

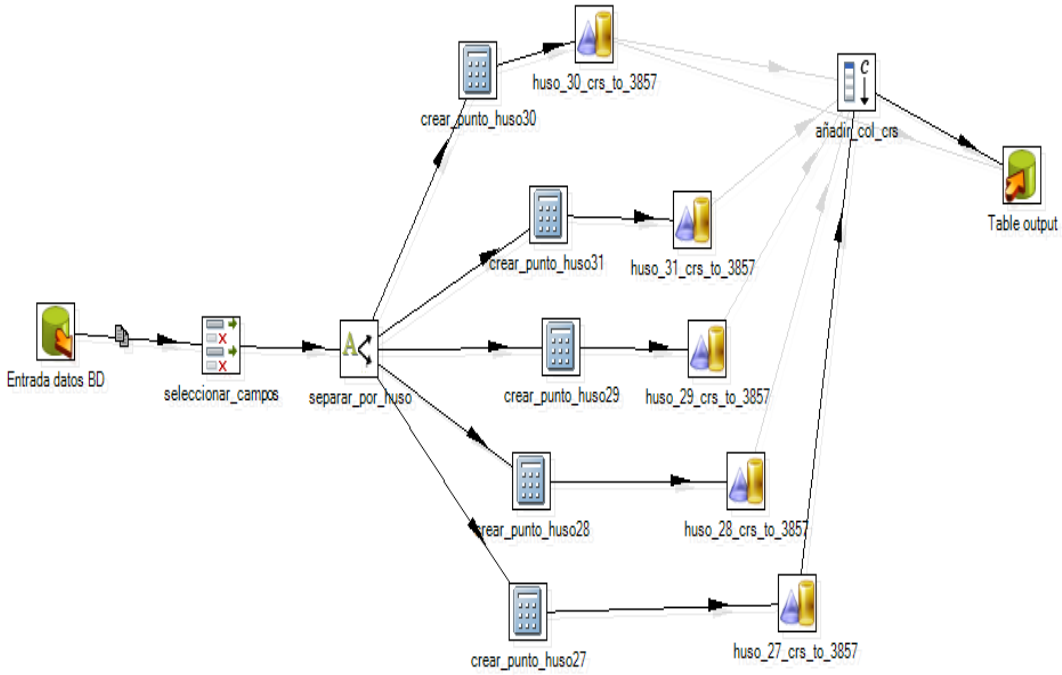

Figura B.1: Transformación ETL de los datos

B.1.5. Script de creación de la base de datos

Crear base de datos

```
CREATE DATABASE "NGBE"
WITH OWNER = postgres
       ENCODING = 'UTF8'
       TABLESPACE = pg_default
       LC_COLLATE = 'en_US.UTF-8'
       LC_CTYPE = 'en_US.UTF-8'
CONNECTION LIMIT = -1;

ALTER DATABASE "NGBE"
SET search_path = "$user", public, topology;
```

Crear tabla *ngbev2013*

```
CREATE TABLE public.ngbev2013
(
id integer NOT NULL,
nombre_extendido character varying,
identificador_geografico character varying,
nombre_alternativo_2 character varying,
nombre_alternativo_3 character varying,
nombre_variante_1 character varying,
nombre_variante_2 character varying,
nombre_variante_3 character varying,
fuente_nombre_extendido character varying,
fuente_identificador_geografico character varying,
fuente_alternativo_2 character varying,
fuente_alternativo_3 character varying,
fuente_variante_1 character varying,
fuente_variante_2 character varying,
fuente_variante_3 character varying,
idioma_nombre_extendido character varying,
idioma_identificador_geografico character varying,
idioma_alternativo_2 character varying,
idioma_alternativo_3 character varying,
idioma_variante_1 character varying,
idioma_variante_2 character varying,
idioma_variante_3 character varying,
estatus_nombre_extendido character varying,
longitud_etrs89_regcan95 double precision,
latitud_etrs89_regcan95 double precision,
huso_etrs89_regcan95 integer,
xutm_etrs89_regcan95 double precision,
yutm_etrs89_regcan95 double precision,
sistema_referencia_etrs89_regcan95 character varying,
hojamtn_25 character varying,
codigo_ine character varying,
codigo_ngbe character varying,
```

```
crs character varying,
punto geometry,
CONSTRAINT ngbev2013_pkey PRIMARY KEY(id)
)
WITH (
  OIDS=FALSE
);
ALTER TABLE public.ngbev2013
 OWNER TO postgres;
ALTER TABLE public.ngbev2013 ADD COLUMN geom geometry(Point,3042);


UPDATE ngbev2013 SET geom =
ST_SetSRID(ST_MakePoint(longitud_etrs89_regcan95,latitud_etrs89_regcan
95),3042);
```

B.2. Almacén RDF

En esta sección se expone el método seguido para la creación del almacén de datos RDF.

B.2.1. Creación del almacén RDF

El almacén RDF contiene la información de los recursos almacenados en la base de datos geoespacial. Por este motivo se ha creado una aplicación Java que permite obtener los recursos de la base de datos y generar a partir de estos un fichero en formato RDF, que posteriormente permite la creación de un TDB[9], que es el almacén de datos RDF del sistema.

TDB es un componente de Jena para el almacenamiento y consulta de datos RDF. Soporta todas las operaciones contenidas en la API de Jena. TDB se utiliza como un almacén de RDF de alto rendimiento. TDB está protegido contra la corrupción, las terminaciones de procesos inesperados y los fallos del sistema. Para que TDB soporte concurrencia se debe acceder mediante el uso de un servicio de SPARQL.

[9] https://jena.apache.org/documentation/tdb/

Para el desarrollo de esta aplicación Java se ha hecho uso del framework Jena, que proporciona librerías que facilitan la adaptación de la información de la base de datos al formato RDF.

B.2.2. Estructura del almacén RDF

La creación de la estructura que almacena los datos en formato RDF, se ha hecho siguiendo los estándares oficiales, y haciendo uso de los vocabularios ya existentes, que se adaptan a la naturaleza de los datos con los que trabaja la aplicación.

El modelo resultante en el que se estructuran los datos del sistema, hace uso de tres vocabularios:

- **GeoSPARQL** [15] es un estándar para la representación y consulta de *Linked Data* geoespaciales para la Web Semántica del Open Geospatial Consortium (OGC). Tiene por objeto proporcionar una base de intercambio estandarizado para datos geoespaciales RDF que puede apoyar tanto el razonamiento espacial cualitativo y cuantitativo y consultar con el lenguaje de consulta de base de datos SPARQL.
- **SKOS-XL** [16] proporciona un modelo para representar la estructura básica y el contenido de esquemas conceptuales como listas encabezamientos de materia, taxonomías, esquemas de clasificación, tesauros y cualquier tipo de vocabulario controlado.
- **Dublin Core** [17] es un sistema de 15 definiciones semánticas descriptivas que pretenden transmitir un significado semántico a las mismas, proporcionando un vocabulario de características "base", capaces de proporcionar la información descriptiva básica sobre cualquier recurso, sin que importe el formato de origen, el área de especialización o el origen cultural.

La estructura genérica del modelo es la siguiente:

```
@prefix geo:    <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix wkt:    <http://www.opengis.net/ont/geosparql#> .
@prefix skosxl: <http://www.w3.org/2008/05/skos-xl#> .

uri recurso
      geo:geometry
            [
            wkt:asWKT "POINT(lon,lat)"
            ]
```

```
skosxl:labelrelation
        [
        skosxl:preferLabel
                [
                skosxl:literalForm "nombre"
                dc:source   "fuente"
                dc:language "idioma"
                ]
        skosxl:altLabel
                [
                skosxl:literalForm "nombre"
                dc:source   "fuente"
                dc:language "idioma"
                ]
        skosxl:hiddelabel
                [
                skosxl:literalForm "nombre"
                dc:source   "fuente"
                dc:language "idioma"
                ]
        ]
```

Tabla 8: Modelo recurso RDF

En este modelo se puede observar al principio la definición de los prefijos con los que se va a declarar cada vocabulario, en el caso de Dublin Core no es necesario, ya que reconoce implícitamente su prefijo como DC.

Después de la definición de los prefijos se encuentra, la URI del recurso, y a continuación la declaración de las propiedades que puede tener asociado un recurso.

- *geo:geometry* especifica el punto de la localización del recurso en el mapa.
- *skosxl:preferLabel* especifica el nombre, fuente e idioma oficiales del recurso.
- *skosxl:altLabel* especifica el nombre, fuente e idioma alternativos del recurso.
- *skosxl:hiddelabel* especifica el nombre, fuente e idioma variantes del recurso.

En el caso de las propiedades *geo:geometry* y *skosxl:preferLabel* deben tener valor obligatoriamente, ya que son las propiedades mínimas que se pueden tener sobre un recurso.

Las propiedades *skosxl:altLabel* y *skosxl:hiddelabel* son multivaluadas, es decir, pueden repetirse, dado que un recurso puede tener varios nombres alternativos/variantes, y de esta forma se permite almacenar todos los valores, así como

en caso de que el recurso no tenga valores asociados para estas propiedades, el recurso almacenado no tendrá esas propiedades asociadas.

A continuación se muestra un ejemplo de un recurso almacenado en la estructura descrita anteriormente.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
    xmlns:wkt="http://www.opengis.net/ont/geosparql#"
    xmlns:skosxl="http://www.w3.org/2008/05/skos-xl#">
  <rdf:Description
rdf:about="http://localhost/collections/1/features/1898601">
  <skosxl:labelRelation rdf:parseType="Resource">
    <skosxl:altLabel rdf:parseType="Resource">
      <dc:source>NGBE</dc:source>
      <skosxl:literalForm>
       "Barranco de la Torquilla"
      </skosxl:literalForm>
    </skosxl:altLabel>
    <skosxl:hiddenLabel rdf:parseType="Resource">
      <dc:source>NGBE</dc:source>
      <skosxl:literalForm>
       "Embalse de Joaquín Costa"
      </skosxl:literalForm>
    </skosxl:hiddenLabel>
    <skosxl:altLabel rdf:parseType="Resource">
      <dc:source>NGBE</dc:source>
      <skosxl:literalForm>
       "Baranco de Riasol"
      </skosxl:literalForm>
    </skosxl:altLabel>
    <skosxl:prefLabel rdf:parseType="Resource">
      <dc:source>NGBE</dc:source>
      <skosxl:literalForm>
       "Barranco de la Torquilla"
      </skosxl:literalForm>
    </skosxl:prefLabel>
      </skosxl:labelRelation>
    <geo:geometry rdf:parseType="Resource">
      <wkt:asWKT>POINT(0.54,42.16)</wkt:asWKT>
    </geo:geometry>
  </rdf:Description>
</rdf:RDF>
```

Tabla 9 Ejemplo recurso RDF

En este ejemplo se puede ver que el recurso tiene como URI: http://localhost/collections/1/features/1898601.

Tiene definidas las propiedades *geo:geometry* y *skosxl:prefLabel* y en este caso concreto este recurso tiene dos nombres alternativos (*skosxl:altLabel*) y dos nombres variantes (*skosxl:hiddenLabel*).

Apéndice C. Tecnologías y herramientas

En este apéndice se describen las tecnologías y herramientas utilizadas para el desarrollo y gestión de este proyecto.

A continuación se presenta un listado que contiene todas las tecnologías y herramientas utilizadas.

Desarrollo del proyecto	Gestión del proyecto
PostgreSQL y PostGIS	GitHub
GeoKettle	Dropbox
MapServer	Visio
Apache Jena	
Apache Jena Fuseki	
Java	
JavaScript	
Spring Boot	
Apache Maven	
OpenLayers	
Bootstrap	

Tabla 10: Tecnologías y herramientas

C.1. Tecnologías y herramientas para el desarrollo del sistema

C.1.1. PostgreSQL y PostGIS

PostgreSQL [18] y **PostGIS** [19] son las tecnologías utilizadas para la creación y gestión de la base de datos de la aplicación.

PostgreSQL es un sistema gestor de bases de datos de código abierto, distribuido bajo licencia BSD, que comenzó a desarrollarse hace más de 16 años, a lo largo de estos años las características más destacables son: estabilidad, potencia, robustez, facilidad de administración e implementación de estándares. Funciona muy bien con grandes volúmenes de datos y soporta una alta concurrencia de usuarios accediendo de forma simultánea al sistema. PostgreSQL tiene una arquitectura cliente/servidor, utiliza multiprocesos en lugar de multihilos, garantizando la estabilidad y seguridad del sistema.

PostGIS es una extensión de PostgreSQL que permite trabajar con bases de datos geoespaciales, desarrollada por la empresa Refraction Research [20] y distribuido bajo licencia Pública General de GNU. Esta extensión incorpora el soporte necesario para trabajar con objetos geográficos incorporando consultas de ubicación que pueden ejecutarse mediante SQL. También ofrece muchas otras características.

C.1.2. GeoKettle

GeoKettle [21] es utilizado para el tratamiento y carga de información en la base de datos geoespacial. GeoKettle es una herramienta de ETL desarrollada por Spatialytics [22] en el año 2011. Esta herramienta permite la integración de las diferentes fuentes de datos geoespaciales, para la creación y edición de los almacenes de datos geoespaciales .GeoKettle permite la extracción y transformación de los datos, permitiendo así corregir posibles errores en la información, realizar un proceso de cleansing, modificar la estructura de los datos, y cargar los datos en los correspondientes almacenes de datos, o generar ficheros de salida que contienen los datos en diversos formatos.

C.1.3. MapServer

MapServer [23] es la herramienta utilizada para la creación del WMS propio de la aplicación. Mapserver es una plataforma para la publicación de datos geoespaciales y generar aplicaciones web cartográficas. Se desarrollo a mediados de los 90 en la Universidad de Minnesota, esta publicado bajo una licencia tipo MIT y funciona en los principales sistemas operativos.

C.1.4. Apache Jena

Apache Jena [24] es la tecnología utilizada para trabajar con la información en formato RDF. Apache Jena es un framework de java que permite el desarrollo de aplicaciones de web semántica. Inicialmente fue desarrollado por HP Labs en el 2009 y posteriormente pertenece a Apache Software Foundation [25].

Este framework proporciona una serie de librerías java que permite trabajar con formatos de web semántica (RDF, RDFS, RDFa, OWL y SPARQL). Jena incluye un motor de inferencia basado en reglas para llevar a cabo un razonamiento basado en ontologías OWL y RDFS, y diversas estrategias de almacenamiento para almacenar tripletas RDF en memoria o en disco.

C.1.5. Apache Jena Fuseki

Apache Jena Fuseki [26] es el servidor de SPARQL utilizado en la aplicación. Apache Jena Fuseki es un servidor de SPARQL integrado dentro del framework Jena. Puede funcionar como un servicio del sistema operativo, como una aplicación web Java (archivo WAR), y como un servidor independiente. Proporciona seguridad (usando Apache Shiro) y tiene una interfaz de usuario para el seguimiento y la administración del servidor.

Proporciona la SPARQL1.1 protocolos para la consulta y actualización de almacenes RDF.

Fuseki está integrado con TDB, componente de Jena, proporcionando así una capa de almacenamiento persistente robusta, transaccional, e incorpora Jena text query y Jena spatial query, para realizar las consultas a los almacenes de datos.

Se puede utilizar para proporcionar el motor de protocolo para otros sistemas de consulta y almacenamiento RDF.

C.1.6. Java

Java [27] es un lenguaje de propósito general, localizado en el paradigma de los lenguajes orientados a objeto, diseñado por Sun Microsystems posteriormente adquirida por Oracle Corporation [28] en 1995. Java no es lo mismo que javascript, que se trata de una tecnología sencilla que se usa para crear páginas web y solamente se ejecuta en el explorador. En el caso de este proyecto se utiliza Java 1.8.

C.1.7. JavaScript

JavaScript es un lenguaje de programación interpretado, que solamente se ejecuta en el lado del navegador, normalmente se utiliza para la creación de páginas web.

C.1.8. Spring Boot

Spring Boot [29] es la tecnología utilizada para la implementación del LMS. Spring Boot es parte de Spring IO Platform, un framework que facilita el desarrollo de proyectos y su posterior testeo, fue desarrollado por Rod Johnson en 2002 aunque el lanzamiento de la versión 1.0 fue en 2004.

Spring Boot permite crear aplicaciones Java que se pueden iniciar usando java-jar o mediante la generación de un fichero war y su posterior despliegue. También se puede trabajar mediante línea de comandos, facilitando así el uso de script para la puesta en marcha de las aplicaciones.

C.1.9. Apache Maven

Apache Maven [30] es una herramienta para crear y gestionar proyectos basados en Java. Originalmente se creó para intentar simplificar el proceso de construcción del proyecto Jakarta Turbine. Con esto se buscaba una forma estándar para construirlos proyectos, definir en qué consistían, publicar información fácilmente y compartir archivos JAR a través de varios proyectos.

Maven se basa en el concepto de Project Object Model (POM), para describir el proyecto y sus dependencias de otras librerías y componentes externos, y el orden de construcción de los elementos. Maven incluye un motor que permite descargar plugins dinámicamente de diversos repositorios.

C.1.10. Openlayers

Openlayers [31] es la tecnología utilizada para la creación de la parte geoespacial en el cliente. Openlayers es una librería de código abierto, implementada en JavaScript , publicada bajo licencia BSD. Originalmente fue desarrollado por MetaCarta en 2006, pero en 2007 este proyecto paso a formar parte de los proyectos de Open Source Geospatial Foundation (OSGeo). Esta librería permite la visualización de mapas interactivos en los navegadores web. Con la API que ofrece OpenLayers, también

permite fácilmente la integración de servicios de mapas (WMS), servicio de features (WFS), mapas comerciales (OSM, Google Maps, etc.), mapas vectoriales, etc.

C.1.11. Bootstrap

Bootstrap [32] es la tecnología utilizada para la implementación de la parte visual del cliente (apariencia, formularios, etc.). Bootstrap es un framework de código abierto para la creación y desarrollo de sitios y aplicaciones web. Ofrece plantillas HTML y CSS con diferentes componentes (botones, menús, etc.) además de incorporar extensiones de JavaScript. El origen de Bootstrap fue en la primera "Semana de Hackeo" (Hackweek) de Twitter." (Octubre 2010) y posteriormente en agosto del 2011, Twitter lo liberó a como código abierto.

C.2. Tecnologías y herramientas para la gestión del proyecto

C.2.1. GitHub

GitHub [33] es un servicio de alojamiento de repositorios Gitbasado en la Web, que ofrece control de versiones y gestión de código fuente (funcionalidades propias de Git), y también añade sus propias características distribuidas. GitHub proporciona una interfaz gráfica basada en Web y de escritorio, así como la integración móvil, además de permitir trabajar mediante línea de comandos. También proporciona control de acceso y varias funciones de colaboración como de seguimiento de errores, peticiones de características, gestión de tareas, y wikis para cada proyecto. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

Github se ha utilizado como repositorio principal del proyecto.

C.2.2. Dropbox

Dropbox [34] es un servicio de alojamiento en la nube, que permite almacenar archivos multiplataforma. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre ordenadores y compartir archivos y carpetas con otros usuarios y dispositivos móviles. Existen versiones gratuitas y de pago, cada una de las cuales tiene opciones variadas.

Dropbox se ha utilizado para mantener un directorio de trabajo compartido con el director del proyecto, en el cual compartir documentos, y a su vez como plataforma de almacenamiento para tener una copia de seguridad del proyecto durante su desarrollo.

C.2.3. Visio

Microsoft Visio es un software de dibujo vectorial para Microsoft Windows. Las herramientas que lo componen permiten realizar diagramas de oficinas, diagramas de bases de datos, diagramas de flujo de programas, UML, y más, que permiten iniciar al usuario en los lenguajes de programación.

Visio se ha utilizado para el desarrollo de los diagramas desarrollados a lo largo del desarrollo del proyecto, y también para generar los diagramas que aparecen en la documentación de este.

Apéndice D. Gestión del proyecto

En esta sección se presentan algunos detalles del proceso de gestión del proyecto. Se explica el tiempo empleado en la realización del proyecto, en base a diferentes criterios, incluye el diagrama de Gantt del desarrollo del proyecto. También se ha añadido las historias de usuario obtenidas en la especificación del proyecto, de las cuales se han obtenido los requisitos y funcionalidades que debe satisfacer y poseer el proyecto.

D.1. Tiempo empleado

A lo largo del desarrollo del proyecto se ha llevado de forma paralela un seguimiento de las horas empleadas en las diferentes fases del mismo. Con este seguimiento se han podido obtener una serie de gráficos, sobre los tiempos necesarios para cada una de las fases y el tiempo total de desarrollo del proyecto.

Fases del proyecto:

- **Aprendizaje**: en esta fase se engloban las horas dedicadas a la lectura de manuales de tecnologías utilizadas para el desarrollo del proyecto, así como a la realización de tutoriales, y diferentes implementaciones de pruebas para ver el funcionamiento de todas las tecnologías implicadas en el desarrollo e implementación del proyecto.
- **Análisis y diseño**: en esta tarea están englobadas las horas dedicadas a la discusión de los requisitos y funcionalidades que deben cumplir e implementar el sistema, así como las decisiones de diseño de la arquitectura a seguir para la implementación del mismo. También están las horas dedicadas a la toma de decisiones de diseño del cliente web del sistema.
- **Pruebas**: aquí se encuentran englobadas las horas dedicadas a las pruebas tanto del sistema completo, como pruebas de módulos del sistema de forma independiente.

- **Documentación**: son todas las horas dedicadas a generar la documentación asociada al proyecto desarrollado.
- **Reuniones**: son las horas dedicadas a las reuniones con el director del proyecto, para discutir decisiones del diseño del proyecto, así como para consultas y resolver dudas sobre el mismo.
- **Implementación**: horas dedicadas a la implementación del proyecto

En la siguiente tabla (Tabla 16) se encuentra un resumen de las horas dedicadas a cada una de las tareas descritas anteriormente:

Tarea	**Horas dedicadas**
Aprendizaje	135
Análisis y diseño	84
Pruebas	76
Documentación	145
Reuniones	20
Implementación	242

Tabla 11: Tabla de horas dedicadas al desarrollo del proyecto

Como se puede ver en la tabla el gran grueso de las horas se divide entre la documentación e implementación del sistema, aunque las horas dedicadas al análisis y diseño del sistema también son considerables, ya que estas horas son las más importantes para la posterior implementación, además de ello las horas de aprendizaje también son elevadas debido al desconocimiento de las tecnologías utilizadas para el desarrollo del proyecto.

El total de horas dedicado al desarrollo de este proyecto ha sido de 702 horas.

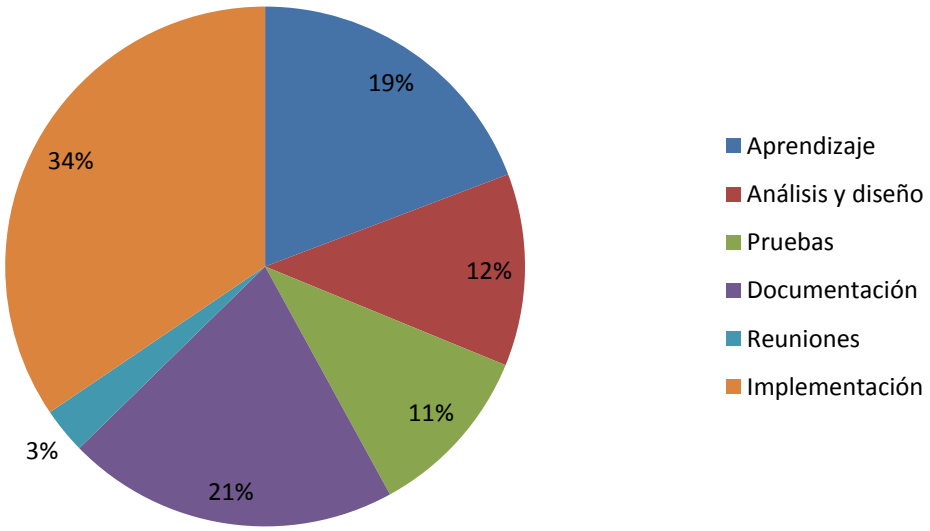

Figura D.1: Horas de trabajo

D.2. Planificación desarrollo del proyecto

En esta sección se presentan una tabla (tabla 11) que representan la planificación seguida durante el desarrollado el proyecto.

Tarea	May 15	Jun 15	Jul 15	Ago 15	Sep 15	Oct 15	Nov 15	Dic 15	Ene 16	Feb 16	Mar 16	Abr 16	May 16	Jun 16
Aprendizaje	■	■	■		■	■	■	■						
Análisis y diseño					■	■	■							
Implementación								■	■	■	■	■	■	
Pruebas										■	■	■	■	
Documentación											■	■	■	
Reuniones	■	■			■	■	■	■	■	■	■	■	■	■

Tabla 12: Planificación del proyecto

En la tabla 11 se puede ver la planificación del proyecto en base a una serie de tareas. En los primeros meses se puede apreciar que las horas de trabajo están dedicadas al aprendizaje de las tecnologías que se utilizan en el proyecto y reuniones con el director. En los meses posteriores se observa que se sigue con la tarea de aprendizaje y se empieza con el análisis y diseño, una vez finaliza esta etapa se inicia la fase de

implementación y pruebas del proyecto, para finalmente empezar la tarea de documentación. En la tabla se puede observar que las fases de implementación y pruebas se solapan en el tiempo, esto es debido a que paralelamente a la implementación se iban desarrollando las pruebas. La tarea de reuniones es continua a lo largo de todo el desarrollo ya que se realizaban reuniones periódicamente (al menos mensualmente) y la comunicación con el director del proyecto era continua. La tarea de aprendizaje también supone un gran peso en el número de horas de la realización del proyecto, debido al desconocimiento inicial de las tecnologías utilizadas, y la necesidad de leer diferentes manuales, realización de numerosos tutorial para algunas tecnologías específicas debido a la complejidad de esta, y la necesidad de tener un conocimiento fluido de las librerías utilizadas.

D.3. Historias de usuario

Para la planificación del análisis del proyecto, se crearon una serie de historias de usuario, de forma que se pudieron obtener los requisitos y las funcionalidades que debía satisfacer y tener el proyecto.

Con estas historias de usuario se describen las diferentes posibilidades de uso de la aplicación, desde el punto de vista de diferentes posibles usuarios potenciales de la aplicación.

Los roles que se han tenido en cuenta para el desarrollo de las historias de usuario son:

Rol	**Usuario final**
Descripción	Utilizar el cliente web (demo) de la aplicación
Breve historia	Voy a buscar información de un POI[10] en el mapa, para ver la breve descripción que muestra, y si es lo que esperaba, accederé a la información detallada de este. Voy a buscar un POI de interés en el mapa y voy a editar su información, después verificare que mi edición se ha llevado a cabo, y comprobaré que aparece el punto editado en el mapa.

[10] POI: Punto de interés (Point of Interest)

Rol	**Evaluador GIS**
Descripción	Realizar pruebas sobre el LMS para evaluarlo desde el ámbito geográfico
Breve historia	Voy a realizar las peticiones que se muestran en el manual de usuario para verificar que todas ellas se pueden realizar sin problemas. Voy a realizar numerosas peticiones para comprobar que el sistema no falla con una carga de peticiones elevada y sigue funcionando correctamente. Voy a ver las funcionalidades que me ofrece el mapa, y observar así las tecnologías utilizadas en el WMS.

Rol	**Evaluador Web Semántica**
Descripción	Realizar pruebas sobre el LMS para evaluarlo desde el ámbito de la web semántica
Breve historia	Voy a realizar las peticiones que se muestran en el manual de usuario para verificar que todas ellas se pueden realizar sin problemas. Voy a realizar numerosas peticiones para comprobar que el sistema no falla con una carga de peticiones elevada y sigue funcionando correctamente. Voy a realizar preguntas sobre el módulo de SPARQL para comprobar su seguridad.

Rol	**Investigador**
Descripción	Utilizar la aplicación en su investigación e incluir nuevas funcionalidades a esta
Breve historia	Voy a realizar las peticiones que se muestran en el manual de usuario para verificar que todas ellas se pueden realizar sin problemas. Voy a comprobar el tipo de tecnología utilizada en el desarrollo de la aplicación para ver su compatibilidad con el proyecto en el que estoy trabajando. Voy a utilizar la aplicación dada añadiendo una serie de funcionalidades para adaptarla al proyecto en el que estoy trabajando.

En la siguiente tabla (tabla 17) se muestran las historias de usuarios generadas a partir de los requisitos del sistema (ver apéndice A sección A.1).

ID	Requisito	Historia de usuario
1	**RF-1**	Como usuario final quiero poder ver un mapa en la aplicación
1000	**RF-1**	Como usuario de la aplicación web quiero poder ver un mapa procedente de un WMS en la aplicación
1001	**RF-1**	Como usuario de la aplicación web quiero poder desplazarme por el mapa
2	**RF-2**	Como usuario de la aplicación web quiero poder hacer un zoom in/out en el mapa
3	**RF-3**	Como usuario de la aplicación web quiero poder desplazar el mapa para poder visualizar partes que no aparezcan inicialmente
4	**RF-4**	Como usuario de la aplicación web quiero poder visualizar varias capas en el mapa
5	**RF-5**	Como usuario de la aplicación web quiero obtener información de un punto
5000	**RF-5**	Como usuario de la aplicación web quiero pasar el cursor por un punto y obtener una breve descripción
5001	**RF-6**	Como usuario de la aplicación web quiero visualizar información detallada al seleccionar un punto y clicar “más información”
7	**RF-7**	Como usuario final quiero poder editar la información sobre un punto
7001	**RF-8**	Como usuario final quiero que la aplicación utilice diferentes iconos para diferentes tipos de POI dibujados en el mapa
7002	**RF-8**	Como usuario de la aplicación web quiero ver el punto original y el editado
8	**RNF-2**	Como evaluador GIS quiero tener disponible un manual que me describa cada una de las peticiones que se pueden realizar al API
8000	**RNF-2**	Como evaluador GIS quiero tener un manual de usuario con ejemplos de todas las peticiones que permite realizar la aplicación

Tabla 13: Historias de usuario

Apéndice E. Manual de uso del cliente

Este manual describe el uso de la aplicación Linked Map Service, detalla las funcionalidades de la aplicación, especificaciones y requerimientos mínimos para su correcto funcionamiento.

- Instalación y configuración del sistema

  En esta sección se describe los pasos a seguir para la correcta instalación y configuración del sistema.

- Primer uso de la aplicación

  En esta sección se describe como ejecutar la aplicación en un primer momento

- Operaciones de consulta

  En esta sección se describe todas las funcionalidades que proporciona el sistema en lo que se conoce como "modo lectura". Ofreciendo ejemplos de todas las posibilidades que la aplicación ofrece al usuario.

- Operaciones de edición

  En esta sección se describe todas las funcionalidades que proporciona el sistema en lo que se conoce como "modo escritura". Ofreciendo al usuario ejemplos de todas las posibilidades que la aplicación ofrece en la edición y actualización de la aplicación.

E.1. Instalación y configuración del sistema

Para la instalación del sistema es necesario descargar Mapserver para desplegar el servicio de mapas, PostgreSQL con la extensión de PostGIS, para tener soporte para la

base de datos, y por último la aplicación que se encuentra el repositorio de Github del proyecto.

Para descargar Mapserver basta con ir a la página oficial, a la sección de descargas[11] y seleccionar la que se corresponda según el sistema operativo con el que se esté trabajando. Una vez descargado se descomprime el fichero. En el fichero de configuración httpd.congf verificamos que el puerto por defecto es el 80.

PostgreSQL se puede descargar de la página oficial[12], seleccionar la versión apropiada para el sistema operativo con el que se esté trabajando, y seguir los pasos de la instalación. Al final instalar la extensión de PostGIS. Tras terminar la instalación aparecerá pgAdmin que es el programa que permite gestionar la base de datos. Se puede optar por utilizar los script que están en el repositorio de la base de datos, o directamente usar el enlace al back-up de la misma y cargarla directamente en el sistema.

A continuación se descarga el fichero ".war" y se despliega en la máquina. Para esta operación se puede usar Tomcat[13]. Cuando termina el despliegue del WAR en el Tomcat, ya está la aplicación lista para ser utilizada. O se puede optar por descargar los fuentes de la aplicación y generar su propio ".war" y desplegarlo. En el repositorio online se puede encontrar un tutorial más detallado de cómo configurar la aplicación.

E.2. Primera ejecución de la aplicación

Asumiendo que la aplicación esta correctamente instalada y configurada (ver sección E.1), se va a proceder a su ejecución.

Para ello se ha de abrir el navegador web, e introducir en la barra de direcciones la siguiente URL: `http://localhost:8080/`. Tras acceder a la dirección anterior aparecerá en el navegador la web principal de la aplicación (figura E.1) y ya tenemos la aplicación lista para trabajar con ella.

---

[11] http://mapserver.org/download.html

[12] http://www.postgresql.org/download/

[13]http://tomcat.apache.org/

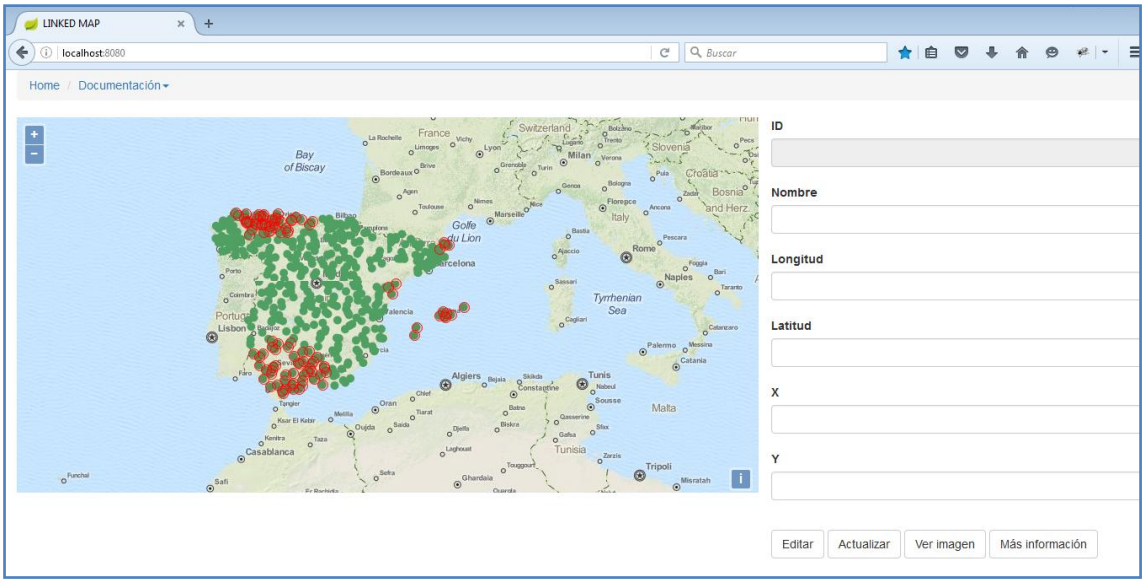

Figura E.1: Pantalla inicio de la aplicación

En la pantalla inicial se puede ver una serie de formularios y botones que permiten trabajar con la aplicación, una barra de menú superior y un mapa en el que se están visualizando las siguientes capas de información: capa OSM, que sirve la capa del mundo ofrecida por OpenStreetMap, capa WMS, representa la información almacenada en la base de datos del sistema, y por último la capa LMS, que permite visualizar de forma gráfica la información RDF del sistema.

En este punto ya se puede proceder a realizar consultas, y editar información.

La barra de menú superior (figura E.2) permite volver o recargar la página principal de la aplicación haciendo clic en el botón "Home".

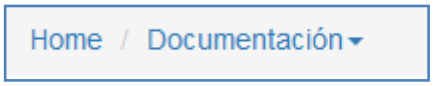

Figura E.2: Barra menú

El botón "Documentación" ofrece un desplegable que contiene un listado de los diferentes módulos que forma el sistema completo (figura E.3), y permite acceder al código fuente y documentación sobre cada módulo, redireccionando al repositorio de Github donde se encuentra el proyecto.

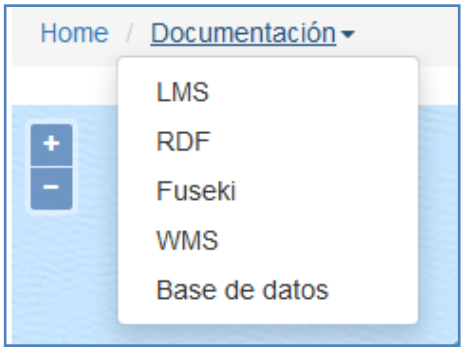

Figura E.3: Menú documentación

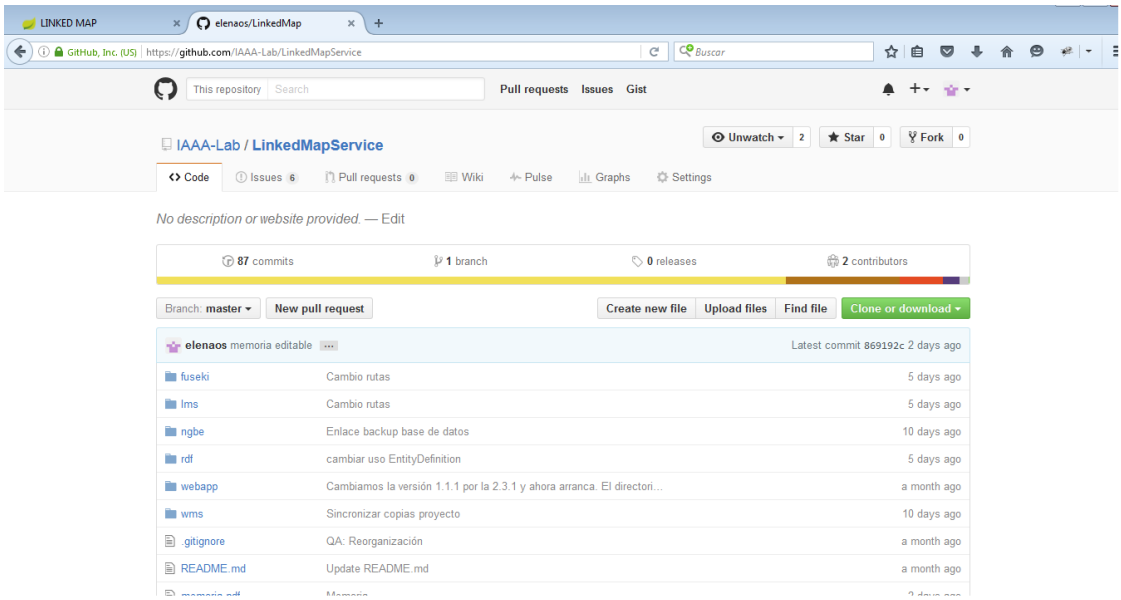

Figura E.4: Repositorio Github del proyecto

E.3. Operaciones de consulta

Esta aplicación proporciona una serie de operaciones de consulta de información (modo lectura) que se listan a continuación:

- Navegación y visualización en el mapa.
- Petición de secciones del mapa, obteniendo la imagen ofreciendo servicio de visualización.
- Peticiones de información extendida sobre los diferentes recursos contenidos en el sistema y visualización de esta.

E.3.1. Navegación y visualización en el mapa

El cliente de la aplicación ofrece un visor de mapas, en el cual podemos realizar diferentes operaciones.

1. Desplazar mapa: el cliente permite desplazar la sección que se está visualizando del mapa, haciendo clic y arrastrando sobre este. Esto tiene como resultado el desplazamiento de la sección a visualizar dentro del visor de mapas.
2. Zoom in/out: al hacer clic en el botón del zoom, se obtiene una visualización más cercana o lejana del mapa, según si el botón seleccionado es "Zoom in" o "Zoom out".
3. Visualización de las diferentes capas del sistema: se puede visualizar una capa OSM, ofrece una vista de Open Street Map, una capa servida por el propio servicio de mapas del sistema y una capa generada por el LMS que representa los recursos almacenados en el almacén RDF.
4. Breve descripción del punto: en el visor de mapa se puede observar que hay una serie de puntos destacados sobre el mapa que se está visualizando (necesario tener activa capa LMS), al pasar el cursor del ratón por encima de estos aparece una ventana emergente, en la que se muestra una breve descripción del punto sobre el que está el cursor.

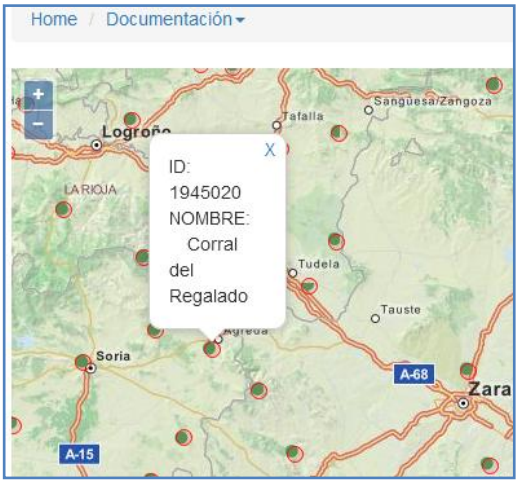

Figura E.5: Venta emergente que contiene breve descripción

E.3.2. Petición secciones del mapa

Una de las peticiones que ofrece el sistema es la consulta de una sección del mapa, obteniendo como resultado una imagen de la sección solicitada.

Para ello el usuario debe posicionar el mapa en la sección que quiere, o seleccionar un punto que este contenido en la sección deseada, y una vez se encuentre en ella hacer clic en el botón “Ver Imagen”.

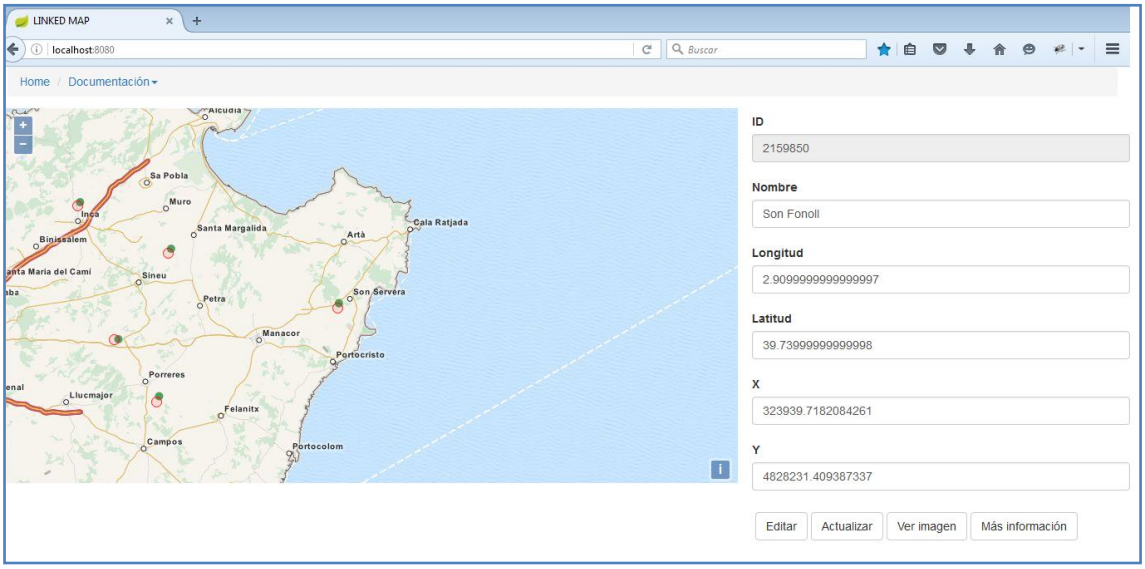

Figura E.6: Petición imagen

La petición que se le envía al LMS es del siguiente tipo:

POST `http://localhost:8080/imagenWMS`

Esta petición tiene como parámetros asociados las coordenadas actuales de la zona visualizada en el mapa. De esta forma cuando la petición es tratada por el LMS genera la petición *GetMap()* con las coordenadas de la zona que se quiere obtener la imagen. Tras realizar la llamada al WMS devuelve la respuesta al cliente. Ahora se puede visualizar el resultado en el navegador, además de la ruta de la petición del servicio de mapas. El resultado que obtenido tras realizar la petición es de la siguiente forma:

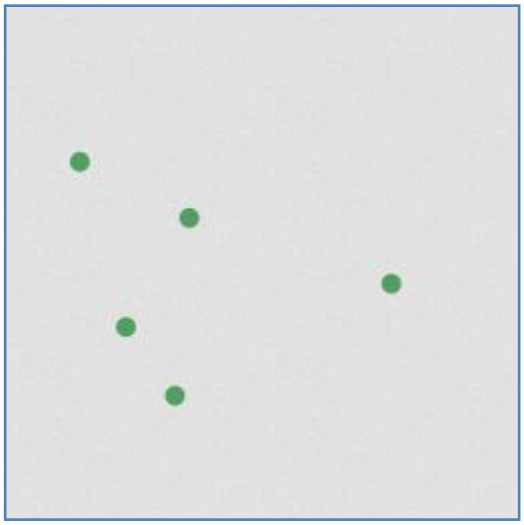

Figura E.7: Resultado petición ver imagen (fondo transparente)

El resultado que muestra la figura E.7, corresponde a la imagen asociada a la sección visualizada en la figura E.6, en la figura E.7, se pueden observar las *features* que existen en la zona seleccionada. La imagen muestra el fondo blanco, pero en realidad es un fondo transparente, ya que de esta manera se puede usar el resultado obtenido con programas GIS, permitiendo superponer esta capa encima de otras con las que se esté trabajando.

E.3.3. Petición información extendida de un recurso

El sistema ofrece una petición para obtener toda la información sobre un recurso que se encuentra en el sistema. Para realizar este tipo de petición basta con hacer clic sobre el punto del que se quiere obtener la información, y posteriormente darle al botón "Más información".

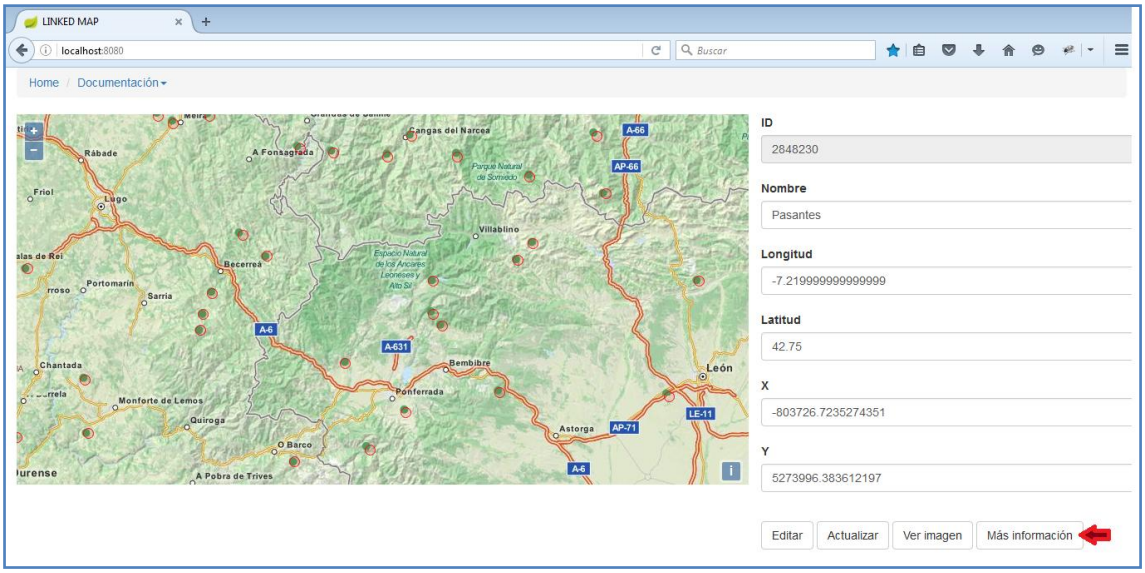

Figura E.8: Petición más información

La petición que se le envía al LMS es del siguiente tipo:

`http://localhost:8080/informacion`

Esta petición tiene como parámetro el id del recurso del que se quiere obtener más información. El resultado que obtenido tras realizar la petición es de la siguiente forma:

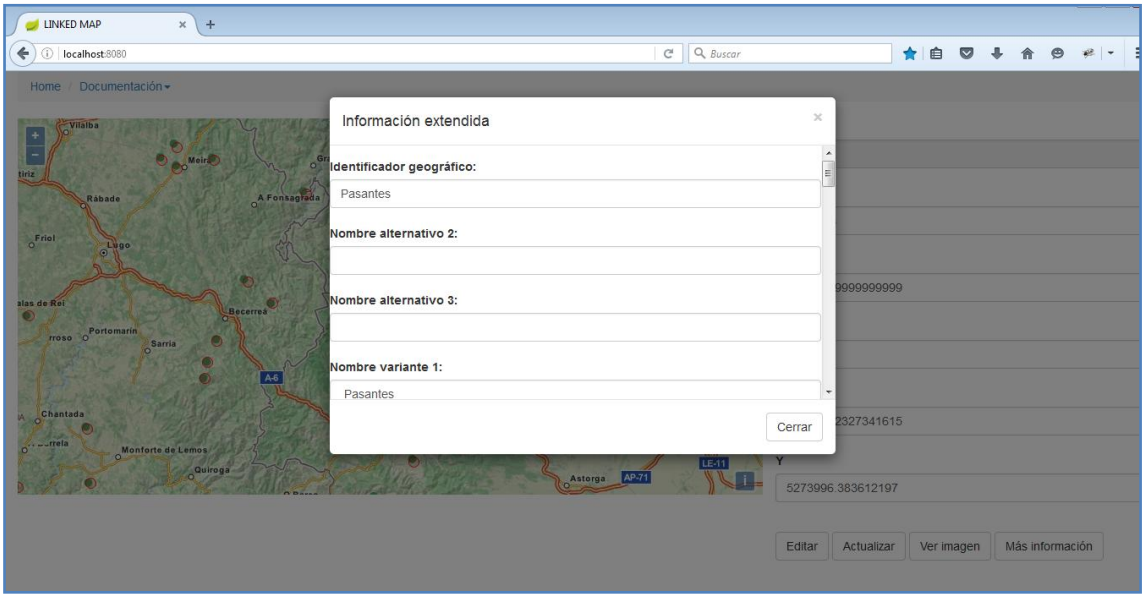

Figura E.9: Resultado petición más información

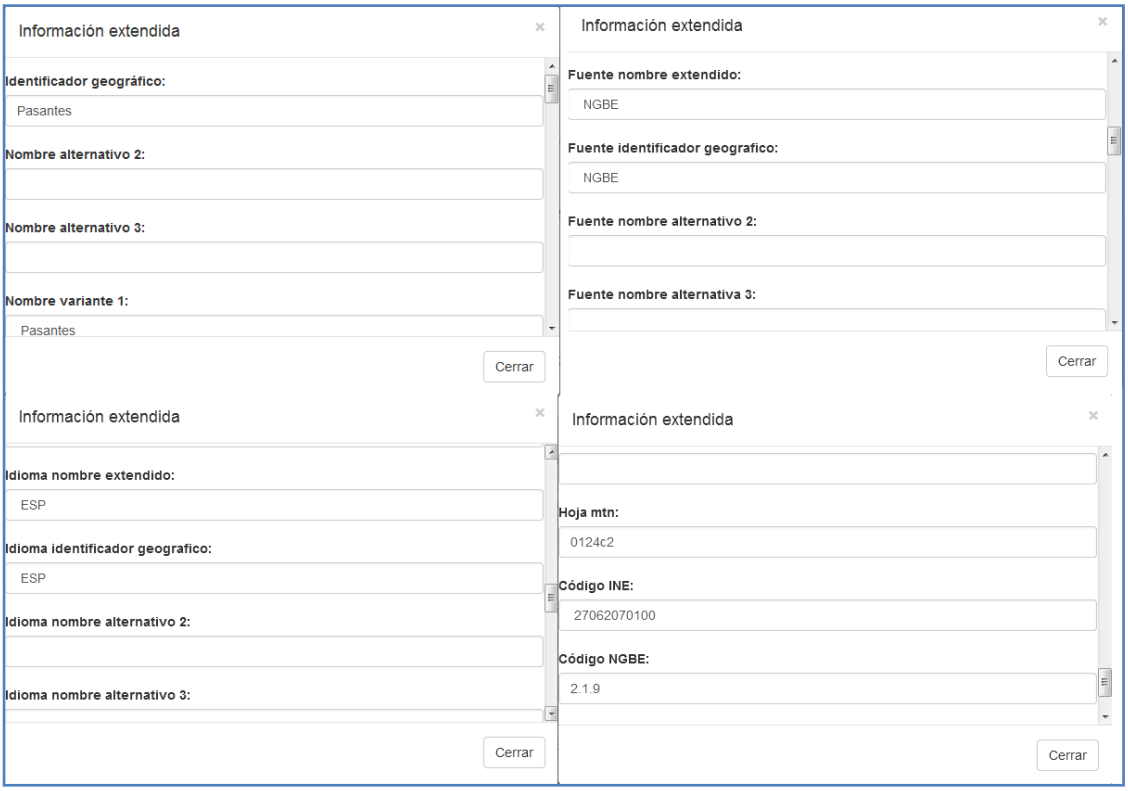

Figura E.10: Resultado petición más información

En la figura E.9 y E.10 se muestra los resultados obtenidos tras darle al botón “Más información”. Se muestra la información completa que el sistema posee sobre el punto seleccionado en el formulario que aparece, en caso de que el sistema no contenga información de alguno de los campos este se mostrará vacio. De esta forma resultara más fácil editar la información en caso de que se desee.

E.4. Operaciones de edición

El sistema permite la edición de información tanto de manera temporal, como de manera permanente, es por ello que existen dos peticiones diferenciadas.

- Petición de edición: permite editar la información asociada a un recurso (excepto el id), y visualizar las modificaciones en el cliente, de manera “temporal”.
- Petición de actualización: permite actualizar la información asociada a un recurso en los datos fuentes del sistema, es decir, modifica los datos de los almacenes de datos del sistema.

E.4.1. Petición edición

La petición de edición que proporciona el sistema, permite modificar la información asociada a un recurso, tanto la información asociada a la posición en el mapa (coordenadas x-y, longitud, latitud), como la información de formato "texto", es decir los campos que proporcionan información asociada al recurso (nombre, fuente, etc.).

Para ejecutar este tipo de petición es necesaria previamente a ver seleccionado un punto, hay que tener en cuenta que si no se ha realizado una petición de información detallada sobre un recurso (ver sección E.3.3), el recurso editado solo tendrá como información asociada la información básica, es decir, el nombre y la posición en el mapa. Una vez con la información cargada en el cliente, se puede proceder a modificarla.

Al hacer clic en el botón editar, se realizan las modificaciones realizadas sobre la información. En el caso de que se haya modificado la posición del punto, aparecerá un nuevo icono en el mapa, el cual nos permite ver el punto original y el editado, en caso de que solo se haya modificado información de tipo texto el nuevo icono aparece sobre el punto original.

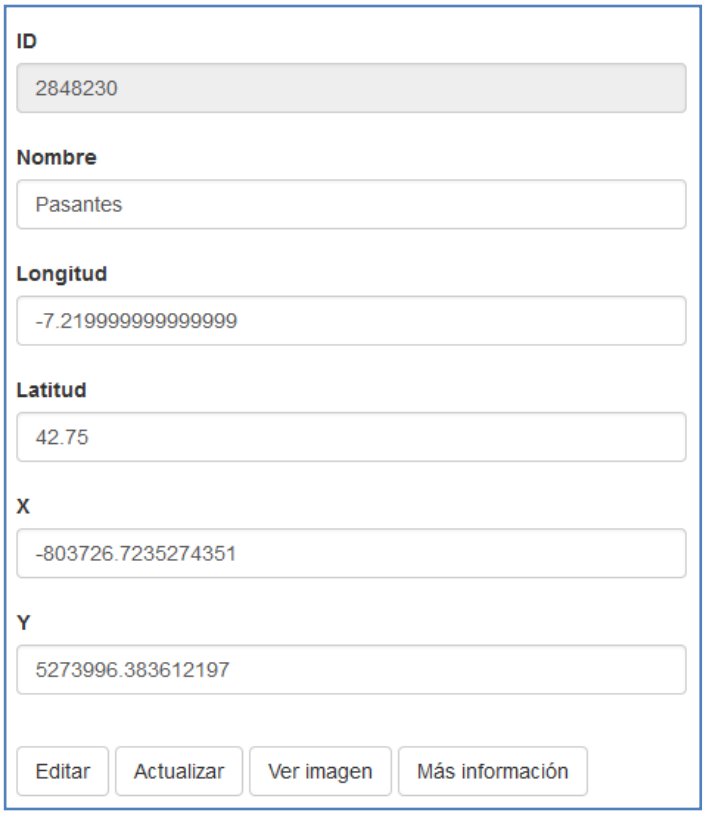

Figura E.11: Información punto original

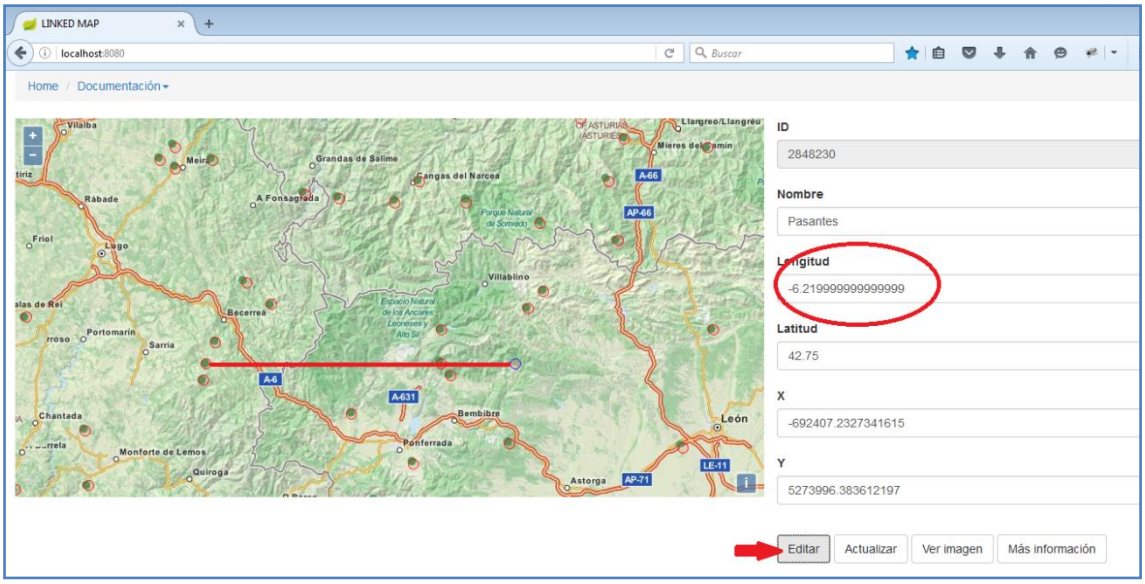

Figura E.12: Petición edición

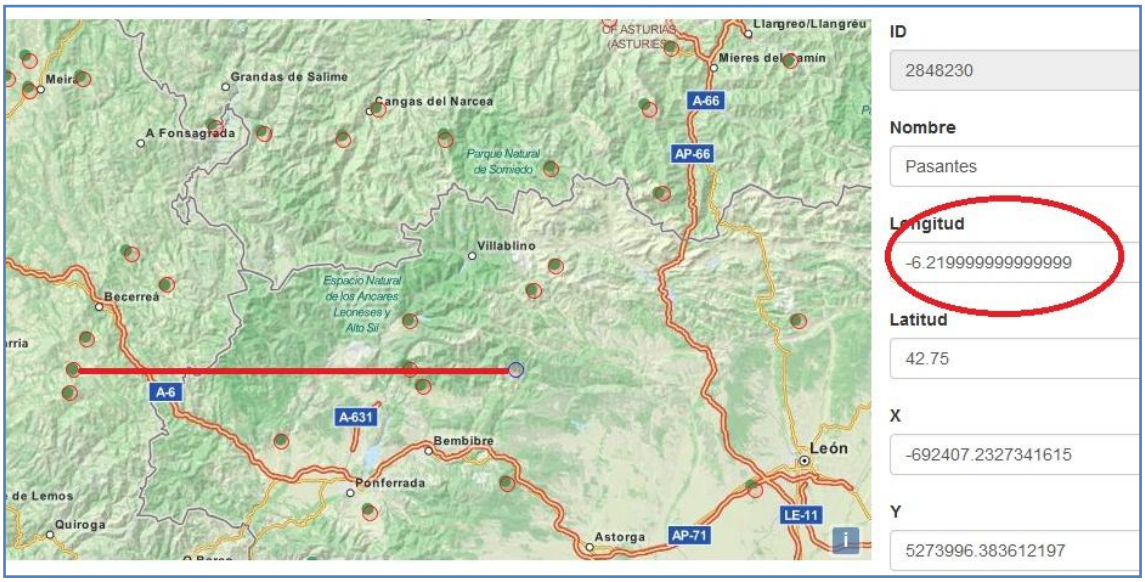

Figura E.13 Petición edición detallada

La figura E.11 se muestra la información original del punto, en la figura E.12 se muestra el resultado de la petición de edición y en la figura E.13 se puede ver con más detalle el resultado, la línea roja señala el punto en la posición original y el nuevo punto editado. En este caso se ha modificado la longitud del punto lo que ha generado en el visor de mapa que el punto se visualice en otra posición es por ello que el mapa se puede visualizar el punto original y el punto editado que aparece en color azul en la nueva posición. Es este nuevo punto generado es el que contiene la información editada.

En este caso dado que es una edición temporal el encargado de realizar las operaciones necesarias para realizarlas es el cliente, por tanto no se envía ninguna petición que al LMS.

El cliente es el encargado de crear una nueva *feature* temporal, en la que estén los valores indicados por el usuario. Esta petición realiza modificaciones "temporales" porque los cambios que se están visualizando en el cliente, no se almacenan en el sistema, es decir, si reiniciamos la aplicación no se podrá visualizar la modificación realizada, porque no ha sido guardada. Este tipo de petición proporciona al usuario la posibilidad de ver una serie de cambios sobre los recursos, y si es lo que él esperaba decidir guardarlos o no.

La petición de edición permite la edición de múltiples puntos, se muestra un ejemplo de ello en la figura E.14 (las líneas rojas están de ayuda para ver la correspondencia de puntos originales con puntos editados).

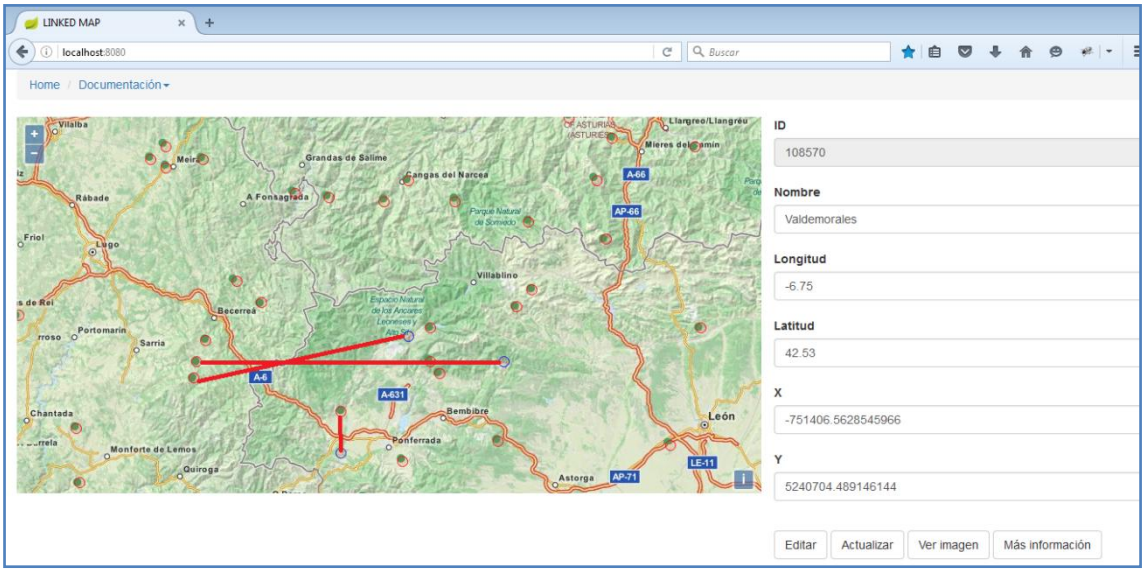

Figura E.14: Varios puntos editados

E.4.2. Petición actualización

Esta petición modifica los datos fuente del sistema. Si se realiza una petición de actualización al reiniciar la aplicación o en ejecuciones posteriores la aplicación conservara los cambios realizados.

Para realizar una petición de actualización es necesario previamente haber realizado una petición (o varias) de edición (ver sección E.4.1). Basta con darle al botón "actualizar".

En este caso se envían varias peticiones al LMS ya que debe actualizar los datos en los diferentes almacenes del sistema:

Para actualizar la base de datos, se envían peticiones de tipo PUT

`http://localhost:8080/service/bd/resource/{id}`

Donde {id} es el id correspondiente al punto a editar. Se manda una petición PUT por cada punto que se tenga que editar en la base de datos

Para actualizar la capa del LMS la petición enviada es:

`http://localhost:8080/service/updateRDFSpatial/{id}`

Esta petición se encarga de actualizar los recursos RDF en el dataset espacial, para ello primero actualiza el recurso concreto, y posteriormente actualiza el índice que permite que el sistema soporte las peticiones espaciales sobre información RDF. Esta petición también gestiona la parte de actualizar la visualización en el cliente, es decir, de eliminar los puntos editados, y validarlos, permitiendo que el cliente tenga una vista actualizada de los puntos que aparecen en el mapa.

Para actualizar los datos en el almacén RDF que contiene la información completa, se hace de forma similar a las peticiones anteriores, mediante peticiones PUT

`http://localhost:8080/service/updateRDF/{id}`

Donde {id} es el id correspondiente al recurso RDF a editar. Se manda una petición PUT por cada punto que se tenga que editar en el almacén RDF. Se generan los cambios necesarios sobre el recurso y se procede a actualizarlo en el almacén RDF que contiene el servicio de SPARQL del sistema.

El resultado que se obtiene tras realizar la petición es de la siguiente forma:

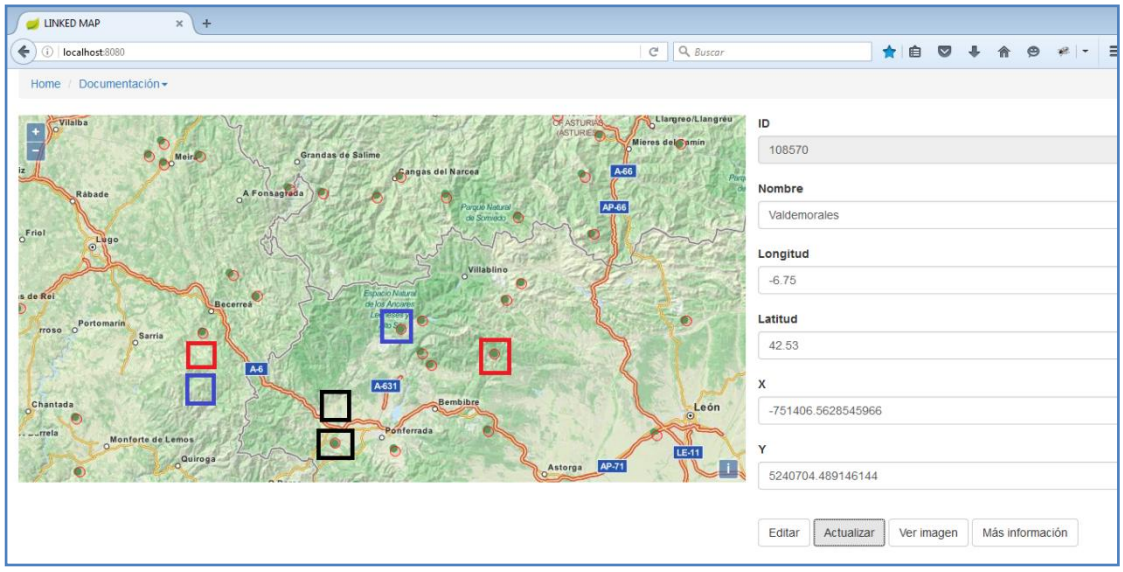

Figura E.15: Petición actualización

Como se puede observar en la figura E.15 la imagen muestra el mapa con los datos actualizados y validados, tanto en la capa del LMS como la capa servida por el servicio de mapas. Los cuadrados vacios es la posición inicial que tenían los puntos y los cuadrados con los puntos actualizados, en la siguiente figura (E.16) se puede apreciar más claramente la actualización de los puntos. Aunque a nivel del cliente no se puede ver, la información esta actualizada en todos los almacenes de datos del sistema.

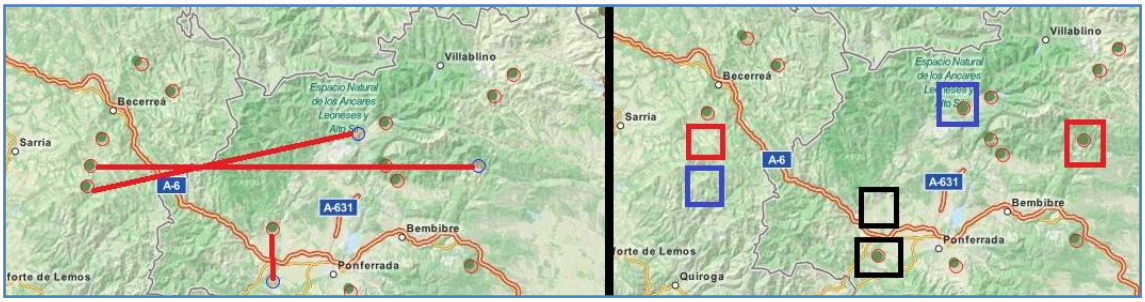

Figura E.16: Comparación puntos originales vs actualizados

Apéndice F. Especificación de la API

A continuación se especifican las funciones implementadas mediante Spring-Boot para el intercambio de información con el cliente, se incluyen también las funciones de test realizadas con JUnit.

F.1. Funciones interacción y procesado interfaz del servicio de mapas

getMap: actúa como proxy inverso del servicio de mapas. Permite procesar la petición enviada por el cliente, redireccionarla al servicio de mapas y devuelve como resultado la imagen correspondiente a la petición del servicio de mapas

```
 /**
 * Redireccion WMS
 **/

@RequestMapping(value="/service/getMap",method=RequestMethod.GET,produ
ces = "image/png")
@ResponseBody public synchronized
 byte[] getMap(HttpServletRequest request)
```

getInfoWMS: genera una petición del servicio de mapas con los parámetros enviados por el cliente y devuelve la información de la función getFeatureInfo() del servicio de mapas

```
/**
 * Funcion redireccion WMS PETICION:GetFeatureInfo
**/

@RequestMapping(path = "infoWMS", method = RequestMethod.POST)
@ResponseBody public
 String getInfoWMS(@RequestParam(value = "txtId") String id,
                   @RequestParam(value = "longitud") String xmin,
                   @RequestParam(value = "latitud") String ymin,
                   @RequestParam(value = "infoURL") String url)
```

getImagenWMS: genera una petición del tipo getMap() y la envía al servicio de mapas devuelve la imagen asociada a los parámetros pasados por el cliente

```
/**
* Funcion redireccion WMS PETICION: GetMap
**/

@RequestMapping(path = "imagenWMS", method = RequestMethod.POST)
@ResponseBodypublic
 byte[] getImagenWMS(@RequestParam(value = "txtId") String id,
                     @RequestParam(value = "longitud") String xmin,
                     @RequestParam(value = "latitud") String ymin,
                     @RequestParam(value = "infoURL") String url)
```

Actualizar: Realiza la actualización en la base de datos geoespacial del sistema del punto pasado como parámetro con los valores de las propiedades asociados al mismo

```
/**
* Función actualizar BD
**/

@RequestMapping(path = "service/bd/resource/{id}", method =
RequestMethod.PUT)
@ResponseBody public
 String actualizar(@RequestBody String punto)
```

parseJSON:obtener los valores de un JSON en un ArrayList para facilitar su posterior uso.

```
/** Metodo parse JSON de respuesta del WMS**/

@RequestMapping(value="parseGetInfo",method=RequestMethod.POST)
@ResponseBody public
ArrayList<String> parseJSON(@RequestParam(value="infoURL")String url)
```

F.2. Funciones interacción y procesado interfaz de web semántica

getMap: genera una petición similar a la de servicio de mapas pero para el módulo RDF solicitando los recursos contenidos en los parámetros y devuelve una lista de ellos.

```
/**
* Funcion getMap RDF
**/

@RequestMapping(path = "/service/getMap", method = RequestMethod.GET,
produces = "application/rdf")
@ResponseBody public synchronized
 String getMap(HttpServletRequest request)
```

updateRDF: actualiza en el almacén de datos RDF el recurso especificado por el parámetro de entrada con los valores correspondientes.

```
/**
* Función actualizar almacén RDF
*/

@RequestMapping(path = "service/updateRDF/resource/{id}", method =
RequestMethod.PUT)
@ResponseBody public

String updateRDF(@RequestBody String parametros)
```

updateSpatialRDF: reindexa y actualiza el dataset espacial de la aplicación

```
/**
*  Funcion actualizar el datasetSpatia sobre el que se lanzan
*las consultas espaciales
**/

@RequestMapping(path = "service/updateRDFSpatial/resource/{id}",
method = RequestMethod.PUT)
@ResponseBody public

 String updateSpatialRDF(@RequestBody String parametros)
```

infoRDF: devuelve la información asociada al recurso especificado que se encuentra en el almacén RDF

```
/**
* Funcion peticion informacionRDF
**/

@RequestMapping(path = "infoRDF", method = RequestMethod.POST)
@ResponseBody public
String infoRDF(@RequestParam(value = "txtId") String id)
```

Informacion: devuelve la información asociada al recurso especificado que se encuentra en el almacén RDF

```
/**
* Funcion peticion informacion recuros
**/

@RequestMapping(path = "informacion", method = RequestMethod.GET)
@ResponseBody public

String informacion(@RequestParam(value = "txtId") String id)
```

F.3. Funciones de test

Funciones de test interfaz servicio de mapa	
getMapTest()	Función de test para la función getMap(), comprueba que la imagen devuelta por la función es la misma que se obtendría si se realizará la petición directamente al WMS
getInfoTest()	Función de test para la función getInfoWMS(), comprueba que la información devuelta por la función es la misma que se obtendría al realizar la petición directa al WMS
imagenWMSTest()	Función de test para la función getImagenWMS(), comprueba que la imagen devuelta por la función es la misma que se obtendría si se realizará la petición directamente al WMS
actualizarBDTest()	Función de test para la función actualizar(), comprueba que se realiza la actualización de forma correcta en la base de datos

Funciones de test interfaz web semántica	
RDFServiceTest()	Función de test para la función init() de la clase RDFController, comprueba que se crea e indexa correctamente el dataset espacial al arrancar la aplicación
selectResource()	Función de test para verificar que se ha cargado correctamente la información en el servidor RDF
borrarRecurso()	Función de test para verificar que la configuración del servicio de SPARQL soporta la escritura y elimina un recurso
insertarRecurso()	Función de test para verificar que la configuración del servicio de SPARQL soporta la escritura e inserta un recurso
updateRDF()	Función de test para la función updateRDF(), comprueba que elimina e inserta el recurso correctamente, por tanto se actualiza
cargarDataset()	Función de test para comprobar que el servicio de SPARQL soporta operación de actualizar el dataset con el que está trabajando
actualizarSpatialDataset()	Función de test para la función updateSpatialRDF(), comprueba que actualiza el recurso y este sigue indexado correctamente para su posterior uso
testDescribe()	Función de test para la función informacion(), comprueba que devuelve la información asociada al recurso seleccionado

Glosario

WMS Servicio de Mapas

RDF Marco de Descripción de Recursos

OGC Open Geoespatial Consortium

OSM Open Street Map

SOA Arquitectura Orientada a Servicios

GIS Sistema de Información Geográfica

WFS Servicio Web de Features

WCS Servicio de Cobertura Web

SOS Sensor Observation Service

CSW Web Catalogue Service o Servicio de catálogo

URI Identificador de Recursos Uniforme

LMS Linked Map Service

API Application Programming Interface

POM Project Object Model

SPARQL SPARQL Protocol and RDF Query Language

Referencias

[1] Proyecto Linked Map http://linkedmap.unizar.es/
[Fecha de acceso 1 Junio 2016]

[2] RDF https://www.w3.org/RDF/
[Fecha de acceso 1 Junio 2016]

[3] Ordenance Survey https://www.ordnancesurvey.co.uk/
[Fecha de acceso 1 Junio 2016]

[4] OGC http://www.opengeospatial.org/
[Fecha de acceso 1 Junio 2016]

[5] SOA https://www.w3.org/TR/ws-arch/
[Fecha de acceso 1 Junio 2016]

[6] Servicios OGC http://www.opengeospatial.org/standards/common
[Fecha de acceso 1 Junio 2016]

[7] Web Feature Service http://www.opengeospatial.org/standards/wfs
[Fecha de acceso 1 Junio 2016]

[8] Web Coverage Service
http://www.opengeospatial.org/standards/wcs
[Fecha de acceso 1 Junio 2016]

[9] Sensor Observation Service
http://www.opengeospatial.org/standards/sos
[Fecha de acceso 1 Junio 2016]

[10] Catalogue Service http://www.opengeospatial.org/standards/cat
[Fecha de acceso 1 Junio 2016]

[11] Web Map Service http://www.opengeospatial.org/standards/wms
[Fecha de acceso 1 Junio 2016]

[12] SPARQL https://www.w3.org/TR/sparql11-service-description/
[Fecha de acceso 1 Junio 2016]

[13] F. J. Lopez-Pellicer, A. J. Florczyk, J. Nogueras-Iso, P. R. Muro-Medrano, and F. J. Zarazaga-Soria, “Exposing CSW Catalogues as *Linked Data*,” in Lecture Notes in Geoinformation and Cartography, M. Painho, M. Y. Santos, and H. Pundt, Eds. Springer Berlin Heidelberg, 2010,pp. 183–200

[14] Noménclator Básico de España
http://www.ign.es/ign/layoutIn/actividadesToponimia.do
[Fecha de acceso 1 Junio 2016]

[15] GeoSPARQL http://www.opengeospatial.org/standards/geosparql
[Fecha de acceso 1 Junio 2016]

[16] SKOS-XL https://www.w3.org/TR/skos-reference/skos-xl.html
[Fecha de acceso 1 Junio 2016]

[17] Dublin Core http://dublincore.org/
[Fecha de acceso 1 Junio 2016]

[18] PostgreSQL http://www.postgresql.org.es
[Fecha de acceso 1 Junio 2016]

[19] PostGIS http://postgis.net/
[Fecha de acceso 1 Junio 2016]

[20] Refraction Search www.refractions.net
[Fecha de acceso 1 Junio 2016]

[21] GeaoKettle http://www.spatialytics.org/projects/geokettle/
[Fecha de acceso 1 Junio 2016]

[22] Spatialytics http://www.spatialytics.com/
[Fecha de acceso 1 Junio 2016]

[23] Mapserver http://mapserver.org/es/
[Fecha de acceso 1 Junio 2016]

[24] Jena https://jena.apache.org
[Fecha de acceso 1 Junio 2016]

[25] Apache https://www.apache.org/
[Fecha de acceso 1 Junio 2016]

[26] Apache Jena Fuseki http://jena.apache.org/documentation/fuseki2/
[Fecha de acceso 1 Junio 2016]

[27] Java https://www.java.com/es/
[Fecha de acceso 1 Junio 2016]

[28] Oracle http://www.oracle.com
[Fecha de acceso 1 Junio 2016]

[29] Spring Framwork http://spring.io/docs/reference
[Fecha de acceso 1 Junio 2016]

[30] Apache Maven http://maven.apache.org/
[Fecha de acceso 1 Junio 2016]

[31] Openlayers http://openlayers.org/
[Fecha de acceso 1 Junio 2016]

[32] Bootstrap http://getbootstrap.com/
[Fecha de acceso 1 Junio 2016]

[33] Github
https://guides.github.com/activities/hello-world/#what
[Fecha de acceso 1 Junio 2016]

[34] Dropbox https://www.dropbox.com/tour/0
[Fecha de acceso 1 Junio 2016]