Titulación: Ingeniería Informática

Centro de Gestión de Contenido En RDF

Alumno: Sergio Ibáñez Fraile
Director: Jesús Bermúdez de Andrés

Proyecto Fin de Carrera, julio 2015

ÍNDICE

1. INTRODUCCIÓN

La primera idea de la realización de este proyecto, fue concebida por la necesidad de tener un sistema por el cual se pudieran cambiar datos de una aplicación, en un sistema móvil, a través de una página web. Sin embargo al conocer la potencia que tiene RDF para ser muy escalable terminó siendo un sistema de gestión de contenido general en RDF.

Este sistema de gestión se ha realizado para ser lo más simple posible para un usuario, de tal manera que con solo 2 click en la página web y rellenando un formulario simple, pudiera tener una base de datos sin muchos conocimientos sobre la gestión de las mismas. Aunque claramente no es un sistema potente como si fuera una base de datos en Oracle, por citar un ejemplo, sirve para poder agregar, modificar y eliminar datos con sencillez. Así este sistema de gestión da una posibilidad muy sencilla de realizar tus propias bases de datos.

Además aunque tiene un motor SQL para la gestión interna de almacenamiento, la salida de los datos es en RDF/XML con lo que podría ser compatible con un sistema más amplio como *Oracle Database Semantic Technologies*

Este CMS también tendrá un sistema de seguridad basado en usuario y contraseña. Para que la edición del contenido sea accesible solo a usuarios con acceso, mientras que la exportación de los datos será pública, y podrá ser accesible por cualquier usuario mediante una URI. EL contenido que se podrá crear tiene 4 tipos que se explicaran en los puntos sucesivos:

1. Términos de búsqueda
2. Propiedades globales
3. Formularios de instancias
4. Instancias

El formato del HTML que se usará para la visualización de los distintos formularios, será HTML5 y CSS3 en un formato responsive para poder ser visualizado en dispositivos móviles.

Asimismo este proyecto podría ser ampliable en muchos sentidos, ya que es un sistema muy general de gestión de datos, y como se explicara en el DOP

2. ANÁLISIS DE REQUISITOS

Para la realización de este proyecto, vamos a necesitar tecnologías, herramientas, hardware y software específico. Al pensar en una aplicación web nos viene a la cabeza que necesitamos un servidor y servicio web, y después programación con la que utilizar ese servicio. Además como estamos hablando de almacenar datos, por medio de un CMS, necesitamos tecnología con la cual almacenarlos. Y encima querer que sea fácil, debe de ser fácil la forma de almacenarlos.

Además para hacer este tipo de proyecto, necesitamos un programa de edición que sea sencillo, y que sea compatible con el lenguaje de programación elegido. Es por eso que al ir estudiando los requisitos se ha valorado usar lo siguiente:

Requisitos de tecnologías

- Para el nivel de transporte de los datos : XML y XMLS
- Para el nivel de modelado de los datos: RDF y RDFS
- Presentación de los datos por pantalla: HTML5, CCS3 y JavaScript

Requisitos de software

- Transporte de información Web entre usuario y servidor: Apache
- Estructuras para guardar los datos: SQL / MySQL
- Programación del servidor Web: PHP
- Programación en el cliente: jQuery

Requisitos de aplicaciones

- Edición de la programación : SublimeText
- Edición del Diseño : Adobe Photoshop
- Modelado de la base de datos : PhpMyAdmin
- Transferencia entre servidores: Filezilla Client (FTP)
- Navegador de Internet : Firefox, Internet Explorer y Chrome
- Servidores: WAMP y LAMP

Requisitos de hardware

- Ordenador personal: Para hacer los cambios
- Servidor externo en 1and1: Para los backups y las versiones “estables”

2.1 Requisitos de tecnologías

He englobado en este grupo los requisitos más teóricos, en definitiva los no tienen que ver con un software específico. Por lo tanto no hay que confundir con lo que se ha denominado requisitos de software.

2.1.1 XML/XMLS

El formato de documentos XML, que son las siglas de e**X**tensible **M**arkup **L**anguage, y desarrollado por el W3C (World Wide Web Consortium), es un subconjunto de los lenguajes de marcado generalizado estándar, que se usa para la organización y etiquetado de documentos.

Muchos de los desarrolladores de aplicaciones para internet usan este formato de documento porque es una forma para almacenar datos legible para el humano con un simple vistazo, además porque a diferencia de otros lenguajes de marcas, XML da soporte a bases de datos, lo que hace que sea muy útil para el desarrollo de aplicaciones como este proyecto.

Pero XML en su origen no era para hacer aplicaciones de Internet, sino que era un estándar para el intercambio de información estructurada entre distintas plataformas. Así que se puede usar tanto para bases de datos, como hojas de cálculo. Sin embargo, seguro que el más conocido de los subconjuntos de los documentos XML, es el subconjunto que se usa para la realización de páginas web, el HTML. De este tipo de lenguaje se hablara un poco más adelante en el apartado HTML5, ya que ha sufrido varias versiones.

Las ventajas del XML son muchas, pero quizás las más importantes son las que se enumeran a continuación:

- Es fácilmente procesable, por su estructura, lo puede procesar prácticamente igual un computador y una persona humana. Por lo que mejora la compatibilidad entre aplicaciones.
- La separación total entre contenido (datos) del formato de presentación (diseño)
- Es extensible, incluso después de haberse diseñado y puesto en producción, se puede extender agregando solo etiquetas nuevas.
- Se puede utilizar en cualquier alfabeto desde los derivados del latín a los derivados del chino.

La tecnología XML busca dar solución al problema de expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. Entonces se tiene un árbol de nodos de información.

Estos nodos son llamados elementos y se los señala mediante etiquetas. Una etiqueta consiste en una marca hecha en el documento, que señala una porción de éste como un elemento. Un pedazo de información con un sentido claro y definido. Las etiquetas tienen la forma <nombre>, donde nombre es el nombre del elemento que se está señalando.

Ejemplo de XML

Un documento XML debe de estar bien formado, aunque no sea válido, es decir la sintaxis de un documento XML debe de ser correcta, aunque no se ajuste a ninguna definición de tipo de documento del estándar SGML. Un ejemplo de un XML bien formado es el siguiente:

```
<?xml version="1.0" encoding="UTF-8" ?>
<envio>
        <!--Esto es un mensaje entre bares -->
        <bar nombre=”Arrate”>
                <propietario nombre=”Pepe” apellidos=”González García” />
                <tematica tema=”pub”/>
        </bar>
        <bar nombre=”Txurrut”>
                <propietario nombre=”Sergio” apellidos=”Ibañez Fraile” />
                <tematica tema=”Bar de copas”/>
        </bar
        <![CDATA[ este texto: <mensaje></mensaje> no es reconocido como una etiqueta o marca]]>
        <Mensaje>
                <Remitente>
                        <Nombre> Pepe González García </Nombre>
                </Remitente>
                <Destinatario>
                        <Nombre>Guadalupe Ibañez Perrino</Nombre>
                </Destinatario>
                <Texto>
                        <Asunto>Cambio de hora</Asunto>
                        <Parrafo>No te olvides de cambiar el turno en el bar</Parrafo>
                </Texto>
        </Mensaje>
</envio>
```

Este podría ser el ejemplo del resultado de una aplicación para que mandar mensajes entre bares de una ciudad, como se ve no pertenece a ningún tipo de documento estándar pero si está bien formado. Los documentos bien formados son aquellos que cumplen con todas las definiciones básicas de formato y pueden, por lo tanto, analizarse correctamente por cualquier analizador sintáctico que cumpla con la norma. Se separa esto del concepto de validez que se explica más adelante. Las reglas que tienen que seguir son las siguientes:

Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados. En nuestro ejemplo se ve que *Mensaje* está bien cerrado y anida dentro de sí *Remitente, Destinatario y Texto*. Las tabulaciones no cuentan más que para un orden visual.

- Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, solo pueden tener un elemento inicial. En nuestro ejemplo el nodo raíz es *envio*
- Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles. Como se puede ver en el atributo *nombre* de la etiqueta *propietario* de nuestro ejemplo
- El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML. Así que *envio* es diferente de Envio o EnVio
- Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen alguna característica en común.
- Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos «entendibles» por las personas.

Las partes que corresponden a un documento XML son a grandes rasgos, el prólogo y el cuerpo. El prólogo es una parte opcional aunque recomendada, utilizada para realizar declaraciones, mientras que el cuerpo, también conocido como elemento documento o elemento raíz, es el elemento principal del documento, que contiene a todos los demás. Dentro del cuerpo, podemos encontrar elementos, atributos, PCDATA, CDATA, y espacios en blanco. Esta parte a diferencia del prólogo es obligatoria.

Elementos

Son las etiquetas, componentes básicos de un documento XML. El identificador de un elemento debe comenzar siempre con una letra o guion bajo, y puede contener letras, números, puntos, guiones o guiones bajos. Los elementos se delimitan por medio de etiquetas formadas por el identificador. Todos los elementos deben tener las etiquetas de inicio y de fin, a no ser que se trate de un elemento vacío. En nuestro ejemplo:

<propietario nombre=”Pepe” apellidos=”González García” />

<tematica tema=”pub”/>

El contenido de un elemento puede estar formado por otros elementos, datos formados por caracteres, referencias a otras entidades o una mezcla de los anteriores. Los elementos también pueden contener atributos, que proporcionan meta información acerca de los datos de los elementos.

El contenido formado por caracteres puede ser cualquier texto que no sea considerado una marca o etiqueta (contenido textual de los elementos, valores de atributos o un literal de cadena ["dato" o 'dato']). Con las excepciones del carácter menor que (<) y ampersand (&), que no pueden pertenecer al contenido, pues tienen un significado especial en el marcado, pero pueden ser empleados usando secuencias de escape (“<” y “&” respectivamente).

Atributos

Un elemento puede contener atributos que le doten de información adicional, estos no son considerados parte del contenido de un documento y se utilizan para asociar pares nombre-valor a los elementos. Nunca podrá haber más de un atributo con el mismo nombre en un elemento determinado.

El valor de los atributos debe ir siempre entre comillas (simples o dobles) y puede estar compuesto por cadenas de caracteres y su orden dentro del elemento es indiferente. Siempre son declarados en las marcas o etiquetas de apertura del elemento, o en las etiquetas de los elementos vacíos. Siguiendo con nuestro ejemplo:

<tematica tema=”Bar de copas”/>

La diferencia básica entre un atributo y un elemento es si el dato que está representando es sencillo o complejo.

Secciones CDATA

Pueden aparecer en cualquier zona del documento en la que puedan hallarse datos formados por caracteres. Se utilizan para introducir texto que de otra manera sería reconocido como marcado.

Espacios en blanco

Son espacios en blanco para XML los siguientes caracteres: Espacio en blanco, Tabulador, Avance de línea, Retorno de carro. Y los analizadores XML deben eliminar todos los espacios en blanco de las etiquetas y de los valores de los atributos, y convertir todos los caracteres de fin de línea en caracteres de avance de línea.

Comentarios

Los comentarios aparecen entre “<!–” y “–>”, pudiendo hallarse en cualquier lugar del documento XML. <!-- Esto es un comentario -->

Como se ha dicho antes un documento XML debe de estar bien formado, aunque no sea válido, para este proyecto se van a usar XML's válidos. Para que un documento sea válido, debe de estar bien formado a nivel sintáctico y respetar unas limitaciones impuestas por un esquema. Y un esquema es un documento creado en un lenguaje que sí está basado en XML, y cuya finalidad es indicar que contenido debe de tener el documento XML yen que orden. Es decir que actúa como una plantilla del documento. En el caso de los XM, el esquema usado es XMLS o XML Schema, que es un lenguaje de esquema utilizado para describir la estructura y las restricciones de los contenidos de los documentos XML de una forma muy precisa, no solo a nivel sintáctico sino a nivel semántico, que al final es lo que se busca al hacerlo por RDF.

Aunque en el siguiente apartado hablaremos de RDF y RDFS, hay que hacer una similitud con que RDF es un "subconjunto" de XML, y RDFS es un "subconjunto" de XMLS. Por lo tanto digamos que RDF contiene los datos, mientras que RDFS contiene las reglas del documento RDF. Pero no es un subconjunto propiamente dicho, ya que XML modela datos, mientras RDF modela metadatos. Además XML falla en la escalabilidad de los datos puesto que el orden de los elementos es antinatural y su mantenimiento es muy difícil y costoso, por el contrario, RDF permite la interoperabilidad entre aplicaciones y proporcionar una infraestructura que soporte actividades de metadatos

Como en el proyecto hay que utilizar un parser de XML, y en internet hay infinidad de ellos para PHP. Es más el propio PHP o JavaScript en modo nativo tiene la posibilidad de "parsear" los documentos XML. Sin embargo indagando más por internet se encontró una clase de PHP que no solo analiza sintácticamente el XML, sino que además tiene funciones para RDF muy interesantes, esta clase es EasyRdf de la página web http://www.easyrdf.org/.

2.1.2 Resource Description Framework (RDF)

RDF es un modelo estándar para el intercambio de datos en la Web. RDF tiene características que facilitan la fusión de datos, incluso si sus esquemas básicos difieren, y soporta específicamente la evolución de esquemas en el tiempo sin necesidad cambiar todos los datos a ser cambiado. Además proporciona interoperabilidad entre aplicaciones que intercambian información legible por máquina en la Web. RDF destaca por la facilidad para habilitar el procesamiento automatizado de los recursos Web. Conjuntamente da una compatibilidad entre diversos sistemas de metadatos. Un ejemplo de metadatos, es la información extra que guarda una fotografía de una cámara digital: la cámara con la que se ha hecho o la hora cuando se hizo.

Aunque originalmente fue diseñado como un modelo de datos para estos metadatos, Ha empezado ser usado como un método general para la descripción conceptual de la información que se implementa en los recursos web.

Además como el RDF, en general, está enfocado para establecer una forma de describir objetos con total independencia del sistema operativo o del motor de la base de datos donde se guarda, es perfecto para establecerlo como un modelo de datos multiplataforma. Lo que hace que puedas compartir datos con sistemas de dispositivos móviles, que es uno de las motivaciones que nos llevó a sacar este proyecto sobre gestión de contenido. Todo este modelo está basado en lo que se denomina, modelado basado un URI's, siendo una URI un identificador de recursos uniforme o en inglés Uniform Resource Identifier, que es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

Para continuar hablando de RDF, hay que hablar de los diferentes elementos que componen este modelo de metadatos: Recurso, propiedad, valor y sentencia. Estos elementos son los componentes básicos del RDF:

- ***Recurso*:** Un recurso es cualquier "cosa" que pueda ser identificada inequívocamente, aunque habitualmente suele ser en internet. Es por ello que los identificadores son válidos tanto para fuera como dentro de la red. Como ejemplo puede ser que una Uri identifica desde una página WEB, un bar, o una persona como se explicara un poco en el ejemplo FOAF.
- ***Propiedad*:** Las propiedades son un tipo especial de recurso que describen relaciones entre recursos. Normalmente es tan importante el nombre de la propiedad como su valor, ya que recordemos que son datos de los datos. Unos ejemplos de nombre "edad", "título", como recurso las propiedades también se identifican por URI's.
- ***Valor*:** es la representación que toma la propiedad en cuestión.

El último elemento que vamos a describir, no es propiamente un elemento sino un conjunto de elementos, en diversos sitos lo llaman **enunciado** o **sentencia,** básicamente es una tripleta entre recurso, propiedad y un valor. También son conocidas sus partes como sujeto, predicado y objeto.

Como hemos visto antes la sentencia es una tripleta entre los tres objetos esenciales del RDF, y por lo tanto se puede representar todo RDF mediante un grafo entre sujeto y objeto unido por un predicado.

Figura 1 –Grafo RDF simple

El grafo anterior representa la forma de gráfica con nodos y flechas de las sentencias RDF. Los recursos se suelen representar por óvalos, las propiedades son las flechas o arcos etiquetados que unen los nodos objeto. Entonces estos objetos son nodos representados por óvalos (si es un recurso) o por un rectángulo (si es un literal). Esta suele ser la representación típica en internet sobre RDF. Haciendo un ejemplo con este grafo: "El proyecto está hecho por Sergio Ibáñez Fraile"

Figura 2 –Ejemplo de grafo RDF simple

Como se ha dicho antes si en este grafo Sergio Ibáñez Fraile en vez de ser un literal seria otro recurso la representación sería distinta. Los literales son los últimos nodos del grafo, por hacer un ejemplo: "El proyecto está hecho por Sergio Ibáñez" y "Sergio Ibáñez tiene la edad de 38 años", o sea un grafo que tiene dos sentencias o tripletas. Por lo tanto una agrupación de grafos se puede ver como un grafo más grande, por lo tanto un conjunto finito de triples RDF es un grafo RDF.

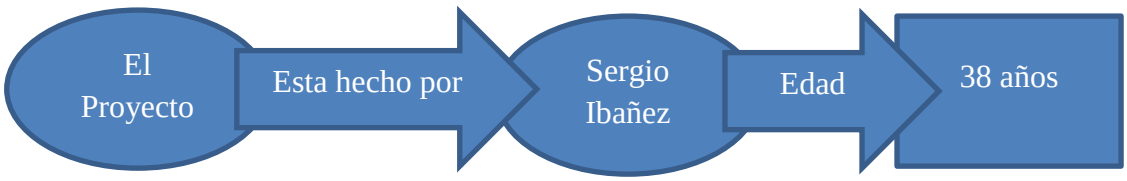


Figura 3 –Ejemplo de grafo RDF compuesto

Podría ser un grafo con un nodo en blanco, si ese recurso también puede es anónimo, y por eso no tiene una URI asociada y se representa mediante un óvalo sin etiquetar. La sentencia siguiente describe "El persona cuyo email es sergio.ibanezfraile@gmail.com y tiene de nombre Sergio es el autor del proyecto CRGRDF" y se representa mediante el grafo:

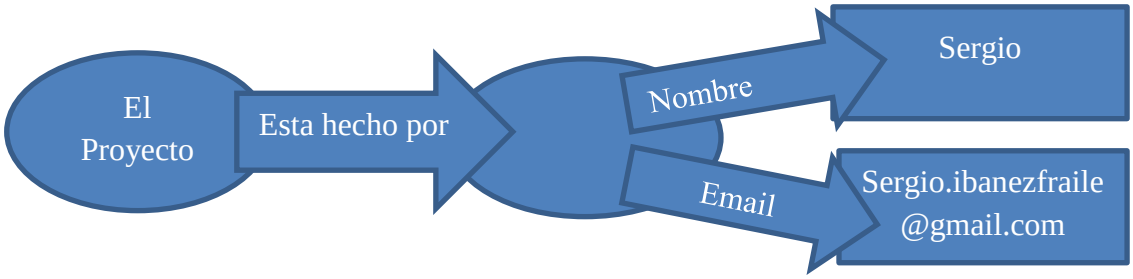


Figura 4 –Ejemplo de grafo RDF compuesto con grafo anónimo

Los predicados son también recursos no anónimos, y por tanto están referenciados por URIs y pueden tener sus propias propiedades, para hacer un ejemplo podemos usar el predicado Esta hecho por del ejemplo anterior para realizar la siguiente descripción: "Esta hecho por tiene rango un estudiante de la UPV/EHU".

Figura 5 –Ejemplo de grafo RDF de propiedades

Como se puede concluir lógicamente después de ver los grafos anteriores Sergio Ibáñez es un estudiante de la UPV/EHU que ha hecho el proyecto CGCRDF. Ya que en el grafo de la Figura 3 el proyecto CGCRDF está hecho por Sergio Ibáñez Fraile, y está hecho es un estudiante de la UPV/EHU como se ve en la figura 5. Además siendo un estudiante de la UPV se le pueden agregar más propiedades que dependan del predicado o del sujeto. Esto hace que RDF sea muy extensible, escalable y versátil.

Como estamos hablando de que vamos a usar RDF en este proyecto necesitamos que RDF, al revés que el castellano o inglés, sea procesado por máquina. Para hacer esta clase de declaraciones sean adaptables a un procesamiento por máquina, se necesitan 2 cosas:

- Un sistema de identificadores procesable por máquina para identificar un sujeto, predicado u objeto en una declaración, sin posibilidad de confusión con un identificar similar que debe usarse por alguien en la Web, o sea las URI's

- Un lenguaje procesable por máquina para representar estas declaraciones e intercambiar la información entre máquinas. O sea el XML, aunque hay muchos más lenguajes, es el usado por mi proyecto y en otro apartado hablaremos del RDF/XML

Para aclarar un poco el tema de las URI's y diferenciarlas de las URL's que son más conocidas por los exploradores de internet, decir que una URL, localizador de recursos uniforme, hace referencia a recursos que pueden variar en el tiempo y la URI es más completo que la URL porque tiene un fragmento. El esquema de una URI es el siguiente:

- *Esquema*: nombre que se refiere a una especificación para asignar los identificadores, más conocido como el protocolo de acceso al recurso. Por ejemplo ftp o http:, que este último es seguramente el más conocido por los internautas
- *Autoridad:* elemento jerárquico que identifica la autoridad de nombres, como por ejemplo //www.ehu.eus.
- *Ruta:* Información usualmente organizada en forma jerárquica, que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres, como por ejemplo /es/web/informatika-fakultatea. Pueden tener dentro de la ruta una extensión como la de un fichero.
- *Consulta:* Información con estructura no jerárquica (usualmente pares "clave=valor") que identifica al recurso en el ámbito del esquema URI y la autoridad de nombres. El comienzo de este componente se indica mediante el carácter '?', y es totalmente opcional mientras que los anteriores son obligatorios.
- *Fragmento:* Permite identificar una parte del recurso principal, o vista de una representación del mismo. El comienzo de este componente se indica mediante el carácter '#'.

Un ejemplo completo de una URI , puede ser el siguiente, aunque ahora existe una corriente denominada URL amigables que reescriben las URI's para que nos sean todavía más fáciles de leer:

Uri completa
http://www.ehu.eus/es/web/informatika-fakultatea/consultar.php?proyecto=CGCRDF#sergio
Uri amigable
http://www.ehu.eus/es/web/informatika-fakultatea/consultar/proyecto/CGCRDF/sergio

Aunque se acostumbra llamar URL a todas las direcciones web, URI es un identificador más completo y por eso es recomendado su uso en lugar de la expresión URL ya que una URI permite incluir en la dirección una subdirección, determinada por el "fragmento" y esto es muy interesante para el proyecto. Como nota dos URI's **pueden** representar el mismo recurso pero una URI **no puede** representar dos recursos.

2.1.2.1 RDF Schema(RDFS)

RDFS proporciona un vocabulario de modelado de datos para los datos RDF. Un lenguaje primitivo de ontologías que proporciona los elementos básicos para la descripción de vocabularios Siendo el término ontología en informática la formulación de un exhaustivo y riguroso esquema conceptual dentro de uno o varios dominios dados; con la finalidad de facilitar la comunicación y el intercambio de información entre diferentes sistemas y entidades.

Una de las mejores características del RDFS, es que está escrito en el propio RDF, es decir que tiene la misma estructura y sintaxis que ese modelo de datos. Y esto mismo hace que sea extensible y que cada persona pueda desarrollar sus propios esquemas RDF.

En este proyecto opte por el uso del RDFS, en concreto la 1.1 del 2014, como un lenguaje para almacenar la estructura de los metadatos. Porque los formularios y términos, que se iban a usar para este proyecto, no se sabían bien que estructura o forma iban a tener. Y por las propiedades anteriores se llegó a la idea de que como requisito se necesitaría un formato parecido al RDFS.

Los esquemas RDF podrían desentonar con las definiciones del tipo de documentos de XML y los XMLS, de los que hemos hablado anteriormente. A diferencia de estos últimos, que dan restricciones específicas en la estructura de un documento XML, un esquema RDF proporciona información sobre la interpretación de una sentencia dada en un modelo de datos RDF. Mientras un esquema XML puede utilizarse para validar la sintaxis de una expresión RDF/XML, aunque hablaremos después de este formato. Un esquema sintáctico sólo no es suficiente para los objetivos de RDF, ni para los de este proyecto. Los Esquemas RDF pueden también especificar restricciones que deben seguirse por estos modelos de datos. El porqué de que no sea suficiente un parser sintáctico es que necesitamos tener ciertas nociones de los elementos semánticos, ya que por ejemplo necesitamos saber si una instancia está relacionada con un formulario. Pero de esto hablaremos más adelante.

El esquema RDF definido en este proyecto es una colección de recursos RDF que puede utilizarse para describir propiedades de otros recursos RDF (incluyendo propiedades) que definen vocabularios RDF de aplicación específica. El núcleo del vocabulario del esquema se define en un namespace, denominado aquí informalmente 'rdfs', e identificado por la URI de referencia http://www.w3.org/2000/01/rdf-schema#.

RDFS no proporciona un vocabulario sobre aplicaciones orientadas a clases, sino que provee de mecanismos para especificar que tales clases y propiedades son parte de un vocabulario y de cómo se espera su relación. También permite definir a los recursos como instancias de una o más clases. Además permite que las clases puedan ser organizadas en forma jerárquica.

Listado clases y propiedades RDFS

Las siguientes listas representan las clases y las propiedades que posee el esquema de RDF, según el esquema RDFS 1.1, hecho el 25 de Febrero de 2014:

Clases

- *rdfs:Resource*
- *rdfs:Class*
- *rdfs:Literal*
- *rdf:langString*
- *rdfs:Datatype*
- *rdf:HTML*
- *rdf:XMLLiteral*
- *rdf:Property*
- rdfs:Container
- rdf:Statement
- rdf:Bag
- rdf:Seq
- rdf:Alt
- rdfs:ContainerMembershipProperty
- rdf:List

Propiedades

- *rdf:type*
- *rdfs:domain*
- *rdfs:range*
- *rdfs:subClassOf*
- *rdfs:subPropertyOf*
- *rdfs:label*
- *rdfs:comment*
- rdfs:member
- rdf:first
- rdf:rest
- rdfs:seeAlso
- rdfs:isDefinedBy
- rdf:value
- rdf:subject
- rdf:predicate
- rdf:object

Las clases y las propiedades de estas listas están incluidas todas en los recursos de RDFS, es decir toda clase de RDF es un recurso de RDF. Es por ello que verdaderamente todas las clases de RDF son subclases de la súper clase rdfs:Resource. En las anteriores listas se han puesto en cursiva las clases y propiedades más relevantes de RDFS (Core-RDFS), pero para entender mejor cada una de las clases y propiedades de RDF, vamos a explicar cada una de ella, ya que en el proyectos se usan casi todas.

Clases RDFS

Las clases definidas por RDFS pueden representar casi cualquier categoría de cosas, como páginas web, gente, tipos de documentos, conceptos abstractos, etc. Las clases son descritas por recursos rdfs:Class y rdf:Resource, y por propiedades como rdf:type y rdfs:subClassOf. Voy a describir dos términos que vamos a usar continuamente en las definiciones de los elementos de rdfs, que son instancia y subclase.

- Instancia: Recurso perteneciente a una clase
- Subclase: Clase que hereda todas las propiedades de su clase padre

rdfs:Resource

Como ya he dicho anteriormente, todo es un recurso, luego todos los elementos que se describen en RDF se denominan recursos, y por ello son instancias de la clase *rdfs:Resource*. Todas las demás clases son subclases de esta clase. *rdfs:Resource* es una instancia de *rdfs:Class*, porque es una clase.

rdfs:Class

Esta es la clase de recursos que son clases en RD, en concreto todos los recursos que van a estar en esta lista. *rdfs Class* es una instancia de *rdfs:Class*, puede sonar a raro pero a la vez lógico pensar que una clase pertenece al grupo de clases de RDF. Como se ve todos los elementos clase son *rdfs:Class*. Esto es lo que usaremos en el proyecto para distinguir una clase o plantilla de una instancia que no será clase, sino recurso general.

rdfs:Literal

La clase *rdfs:Literal* es la clase de los literales, valores como cadenas(strings), números enteros(integers), números decimales(float), etc ... Los propios valores de propiedad, tales como cadenas de texto son ejemplos de literales RDF. *rdfs:Literal* es una instancia de *rdfs:Class* y *rdfs:Literal* es una subclase de *rdfs:Resource* .

rdfs:Datatype

rdfs:Datatype es la clase que especifica los tipos de datos de RDF. Todas las instancias de *rdfs:Datatype* corresponden con el modelo RDF de un tipo de datos, que se describe en las especificaciones de conceptos de RDF. *rdfs: Datatype* es a la vez una instancia de *rdfs:Class* . Y Cada instancia de *rdfs:Datatype* es una subclase de rdfs: Literal y una subclase de *rdfs:Class*. Para ver los tipos de datos que hay en RDF, habría que fijarse en los que hay en el esquema XML, porque son los mismo más dos tipos especiales. El recurso *rdfs:Datatype* es perfecto para guardar en cada plantilla de nuestro proyecto el tipo de datos que es.

Tabla de tipos de datos RDF compatible con tipos XSD

	rdfs:Datatype	Descripción
Tipos Básicos	xsd:string	Cadena de caracteres (No todos los Unicodes)
	xsd:boolean	True o false (Si/no)
	xsd:decimal	Número decimal
	xsd:integer	Número entero
Números decimales	xsd:double	Número en coma flotante (64 bit de precisión)
	xsd:float	Número en coma flotante (32 bit de precisión)
Fecha y Hora	xsd:date	Fecha en formato (2015-05-20)
	xsd:time	Hora en formato (14:50:30.56)
	xsd:dateTime	Fecha y hora con zona horaria
	xsd:dateTimeStamp	Fecha y hora sin zona horaria
Duraciones Y partes de la fecha	xsd:gYear	Año de la fecha
	xsd:gMonth	Mes de la fecha
	xsd:gDay	Dia de la fecha
	xsd:gYearMonth	Año y mes de la fecha
	xsd:gMonthDay	Mes y dia de la fecha
	xsd:duration	Duración de tiempo
	xsd:yearMonthDuration	Duración de tiempo (Año y mes)
	xsd:dayTimeDuration	Duración de tiempo días y hora
Números con limites	xsd:byte	8 bit con signo + o -
	xsd:short	16 bit con signo + o -
	xsd:int	32 bit con signo + o -, diferente del integer
	xsd:long	64 bit con signo + o -
	xsd:unsignedByte	8 bit sin signo
	xsd:unsignedShort	16 bit sin signo
	xsd:unsignedInt	32 bit sin signo
	xsd:unsignedLong	64 bit sin signo
	xsd:positiveInteger	Entero mayor que 0
	xsd:nonNegativeInteger	Entero mayor o igual a 0
	xsd:negativeInteger	Entero menor que 0
	xsd:nonPositiveInteger	Entero menor o igual 0
Datos binarios	xsd:hexBinary	Hexadecimal
	xsd:base64Binary	Base 64
Varios tipos XSD	xsd:anyURI	URIs y IRIs absolutas o relativas
	xsd:language	Tags lingüísticos
	xsd:normalizedString	String normalizados
	xsd:token	Token
	xsd:NMTOKEN	XML NMTOKENs
	xsd:Name	XML Names
	xsd:NCName	XML NCNames

rdf: LangString

La clase *rdf:LangString* es la clase de los valores de cadena de lenguaje de etiquetado. Básicamente representa con un literal string, en que idioma esa. Estos elementos ya está implementados en el propio XML, por medio de xml:lang. Ejemplos: Español;"es", Ingles;"en", etc… *rdf:LangString* es una instancia de *rdfs:Datatype* y una subclase de rdfs:Literal .

rdf:HTML (Esta sección es no normativa).

La clase *rdf:HTML*, es uno de los tipos especiales de *rdfs:Datatype*, es la clase de los valores literales de HTML. Es decir con este tipo de datos sabemos que lo que contiene este recurso es puro HTML. *rdf:HTML* es una instancia de *rdfs:Datatype* y una subclase de *rdfs:Literal* .

rdf:XMLLiteral (Esta sección es no normativa).

Lo mismo que ocurre con la clase *rdf:HTML*, este es otro tipo especial de *rdfs:Datatype*, si el anterior representaba código HTML dentro del recuso, este representa código XML. El anterior y este recurso son los dos tipos especiales que se añaden a los que XML posee como se dijo en *rdfs:Datatype. rdf:XMLLiteral* es una instancia de *rdfs:Datatype* y una subclase de *rdfs:Literal* .

rdfs:Property

Este recurso y clase de RDF, es la segunda clase clasificadora, es decir si rdfs:Class dice si un recurso es una clase, esta clase dice si el recurso es una propiedad dentro de RDF. Es por ello que todas las propiedades de RDF son instancias de rdfs:Property lógicamente. *rdfs:Property* es una instancia de *rdfs:Class*.

Estas clases son las que verdaderamente se usan en el proyecto, en un apartado posterior de esta sección se explicaran las clases y propiedades extras que da RDFS, para la implementación del esquema. En concreto los contenedores, colecciones y la expansión de vocabulario de RDFS.

Propiedades RDFS

En el apartado anterior se han definido las clases de RDFS, y como se ha dicho en este apartado vamos a definir todas las posibles instancias que posee la clase *rdfs:property* en el esquema de RDF 1.1. Además de escribir clases de recursos, también queremos escribir propiedades específicas .En RDFS las propiedades se describen usando la clase *rdfs:Property* con las propiedades definidas *rdfs:domain, rdfs:range* y *rdfs:subPropertyOf.* Estas tres últimas propiedades son las más importantes para definir una propiedad en sí misma. En programación cuando se habla de propiedad siempre está asociada a una clase, en RDF la propiedad es independiente a las clases únicamente tiene como valor *rdf:type rdfs:Property.* Básicamente las propiedades son unas relaciones entre los recursos objetos y los recursos sujetos. Como notación la primera letra de las propiedades va en minúscula y la de las clases en mayúscula

rdfs:type

Esta propiedad define a que clase pertenece el recurso en cuestión, por ejemplo si un sujeto es una propiedad, su rdfs:*type* será *rdfs:Property*, mientras que si es una clase será *rdfs:Class*. Pero en verdad lo que asegura es el tipo de clase con el que se relaciona el recurso. El dominio o rdfs:domain de rdfs:type es un recurso rdfs:Resource, y su rango o rdfs:range es un rdfs:Class. Un ejemplo:

ex:Persona rdf:type rdfs:Class
ex:Local rdf:type rdfs:Class
ex:camarero rdf:type rdfs:Property

rdfs:range

Esta propiedad indica que los valores de la propiedad en cuestión está tomando valores de una clase en concreto. Es decir si la propiedad P tiene como valor rdfs:range C, quiere decir que la que lo valores de la propiedad P están incluidos en la clase C. Por definir una ejemplo:

ex:Persona rdf:type rdfs:Class
ex:camarero rdf:type rdfs:Property
ex:camarero rdf:range ex:Persona

Como se ve la propiedad camarero de por ejemplo un local puede tomar como valor, aquellos que estén definidos en la clase persona. Esto en el proyecto es muy interesante para poder definir que valores pueda tomar cada propiedad del formulario. Recordar que los tipos de datos son subclases de rdfs:Datatype, luego puede definirse aquí también que tipo de datos puede tomar una propiedad.

Normalmente o solo hay un rango o ninguno definido para una propiedad, aunque puede haber más de 1 definicion de rdfs:range. Si no hay ningún rango no nos da información extra sobre la propiedad, si hay un rango únicamente nos dice que la propiedad toma valores de esa clase, si hubiera más de uno dice que una propiedad toma valores de instancias que pertenezcan a todas las clases del rango. El rango de rdfs:range es rdfs:Class y el dominio *rdfs:property*

rdfs:domain

Esta propiedad define la clase de los recursos que aparecen como sujetos las propiedades. Siguiendo un poco con el ejemplo anterior podríamos definir lo siguiente:

ex:Local rdf:type rdfs:Class
ex:camarero rdf:type rdfs:Property
ex:camarero rdf:domain ex:Local

Esto lo que hace es definir que una propiedad camarero tiene como clase a la que es aplicada Local. Así como en el caso de los rangos, una propiedad puede definirse para varios dominios. En caso de que no se defina ningún dominio entonces la propiedad es como una propiedad global sin clases. Si solo está definida para un dominio, se entenderá que esa propiedad se aplica sólo a las instancias de esa clase. En caso de que tuviera más de dos dominios, se entiende como que cualquier recurso que tenga esa propiedad es instancia de todas las clases especificadas por sus dominios. El rango de esta propiedad es *rdfs:Class* y su dominio es *rdfs:property*

rdfs:subClassOf

Esta propiedad es usada para decir que todas las instancias de una clase son instancias de otra clase, tal y como se podría usar para programación en cualquier lenguaje, asi mismo establece que las propiedades de una clase “padre”, pasan también a ser parte de una clase “hijo”. Siguiendo con el ejemplo:

ex:Local rdf:type rdfs:Class
ex:Bar rdf:type rdfs:Class
ex:Taberna rdf:type rdfs:Class
ex:Bar rdf:subClassOf ex:Local
ex:Taberna rdf:subClassOf ex:Local

Como se ve en el ejemplo Bar y taberna son clases “hijas” de la clase “padre” Local, si además lo juntamos con lo anterior como definimos camarero como propiedad de local, camarero pasaría a ser propiedad también de Taberna y Bar. El rango de esta propiedad es *rdfs:Class* y su dominio es también *rdfs:Class*

subPropertyOf

Esta propiedad se utiliza para indicar que todos los recursos relacionados por una propiedad "padre" también están relacionados con esta propiedad "hijo". En sí mismo es el mismo grado de herencia que posee la propiedad *rdfs:subClassOf* solo que para propiedades. El rango y el dominio de esta propiedad es *rdfs:Property*. También puede usarse para definir grados en una propiedad, como podemos ver en el siguiente ejemplo:

ex:camarero rdf:type rdfs:Property
ex: maitre rdf:type rdfs:Property
ex:maitre rdfs:subPropertyOf rdfs:camarero

rdfs:label

Esta propiedad es usada para proporcionar un nombre legible para las personas humanas, sobre todo para los recursos cuyo nombre sea bastante diferente, aunque normalmente se suele usar un nombre en el recurso bastante legible. Por eso también esta propiedad suele ser opcional. El dominio de la propiedad es rdfs:Resource y su rango es rdfs:Literal, normalmente suele ser de la subclase string

ex:maitre rdf:type rdfs:Property
ex:maitre rdfs:label "maître"

rdfs:comment

Lo mismo que la propiedad anterior, es usada para proporcionar un nombre legible, esta es para dar una breve descripción sobre el recurso legible para los humanos. También esta propiedad suele ser opcional. El dominio de la propiedad es rdfs:Property y su rango es rdfs:Literal, normalmente también suele ser de la subclase string

ex:maitre rdf:type rdfs:Property
ex:maitre rdfs:comment "Es el camarero principal de un local"

Clases y propiedades contenedoras en RDFS

Los contenedores escritos en RDF son recursos que se utilizan para representar todo tipo de colecciones. El mismo recurso puede aparecer en un recipiente más de una vez. La única diferencia básica con los términos que se usan en programación es que un recipiente puede estar contenido en sí mismo. Dentro de estos contenedores tenemos 3 tipos de contenedores, por lo que tenemos tres clases que perteneces a una súper clase.

rdfs: Container

Esta es la súper clase de contenedores, por lo que todas dependen de ella misma.

rdf:Bag

Un RDF:Bag se utiliza para indicar que este tipo de contenedor está destinado a ser un contenedor desordenado. Normalmente no hay diferencia entre las otras dos subclases, sin embargo a nivel humano es útil saber sin están desordenados u ordenados. Como ya se ha dicho antes es subclase de *rdfs:Container*

rdf:Sec

Un rdfs:Sec se utiliza para indicar que el orden indicado del contenedor es un orden numérico, dentro de las propiedades miembros de este contenedor. Esto es la diferencia sobre utilizar *rdfs:Bag* como recurso contenedor. Esta clase también e s subclase de *rdfs:Container*

rdf:Alt

Un rdf:Alt como contenedor se utiliza para indicar que el procesamiento de este recipiente será seleccionar uno de las propiedades miembros.

rdfs:ContainerMembershipProperty

Esta clase tiene como instancias las propiedades rdf:_1, rdf:_2, rdf:_3, etc.… que se utiliza como propiedad de numeración. Esta clase es subclase de *rdfs:Property* y cada instancia de esta clase es una subpropiedad de rdfs:member. Por lo tanto dado un contenedor C y una tripleta de la forma C rdf: _NNN O, donde nnn es la representación decimal de un número entero mayor que cero sin ceros a la izquierda, afirma que O es un miembro de la categoría C.

rdfs:member

Esta propiedad es una instancia de la clase rdfs:Property y que es un super-propiedad de todas las propiedades de miembros de contenedores, es decir, cada propiedad de miembros recipiente tiene una rdfs: subPropertyOf relación a la propiedad rdfs:member . Eldominio de *rdfs:member* es *rdfs:Resource* . El rango de esta propiedad es *rdfs:Resource*.

Colecciones en RDFS

Una colección de RDF representa una lista de elementos, este tipo de representación será familiar para aquellos con experiencia de Lisp y lenguajes de programación similares. La única diferencia frente a las listas de programación, es que una lis de RDF no requiere que tenga un primer elemento, ni que solo haya un primer elemento. Estas particularidades que tiene RDF se debe a intenta englobar lo más posible todo los tipos de datos.

rdf: List

Esta clase es una instancia de la clase rdfs:Class y que es usada para crear listas o estructuras similares a listas.

rdf:first

Esta propiedad es una instancia de la clase rdf:Property y que se puede utilizar para construir descripciones de listas y otras estructuras de listas similares. Siendo una tripleta formada por el siguiente ejemplo:

L rdf:first O

Afirma que existe una relación de primera elemento entre L y O. El dominio de *rdfs:first* es *rdfs:Resource* . El rango de esta propiedad es *rdfs:Resource*.

rdf:rest

Esta propiedad es una instancia de la clase rdfs:Property y se puede utilizar para construir descripciones de listas y otras estructuras de listas similares. Siendo una tripleta formada por el siguiente ejemplo:

L rdf: rest O

Afirma que existe una relación entre O y el resto de la lista de L. El dominio de *rdfs:rest* es *rdfs:List* . El rango de esta propiedad es también *rdfs:List*.

rdf:nil

El recurso rdf:nil es una instancia de rdf:List que se puede utilizar para representar una lista vacía u otra estructura-lista similar. La tripleta siguiente:

L rdf:rest rdf:nil

Establece que el resto de L es vacío. También puede indicarse mediante el rdf:first como propiedad.

Propiedades de utilidad en RDFS

Las siguientes clases de utilidad y propiedades están definidas en los espacios de nombres fundamentales.

rdfs:seeAlso

rdfs:seeAlso es una instancia de *rdf:Property* que se utiliza para indicar un recurso que podría proporcionar información adicional sobre el recurso sujeto. El dominio de rdfs:seeAlso es rdfs: Resource. Y el rango de rdfs:seeAlso es rdfs: Resource.

rdfs: isDefinedBy

rdfs:isDefinedBy es una instancia de *rdf:Propery,* que se utiliza para indicar que un recurso define otro recurso. Esta propiedad se puede utilizar para indicar que un recurso RDF define un vocabulario RDF. El dominio es *rdfs:Resource,* y el rango es *rdfs:Resource.*

rdf: value

rdf:value es una instancia de *rdf:Propery,* que se utiliza para dar valores a un recurso estructurado. La propiedad *rdf:value* no tiene sentido por sí misma. A pesar de la falta de especificación formal del significado de esta propiedad, hay un valor en la definición que para fomentar el uso de un idioma común en los ejemplos de este tipo. El dominio de esta propiedad es *rdfs:Resource* y el rango es también *rdfs:Resource*

Como conclusión, tras el estudio de todas las clases y propiedades de RDF, solo con las del núcleo sería suficiente para el proyecto actual. Además las clases RDF ya tiene la estructura que se necesita para este proyecto.

2.1.2.2 RDF/XML

Ya he analizado la estructura de datos que voy a usar para este proyecto, sin embargo necesito una forma para representar estos datos. Aunque recordemos que los grafos RDF son los que verdaderamente representan las estructuras y datos de RDF, necesitamos alguna forma de serializarlos. Hay diferentes formas de serializar las estructuras RDF y sus datos, y suele ser bastante fácil la conversión entre estos formatos de serialización.

- Notation3 (N3)
- Turtle
- RDF/JSON
- N-Triples
- RDF/XML

Notation3 o N3

Notation3 o N3 es una forma abreviada de serialización, que no sigue el estándar de estructura XML de modelos de metadatos en RDF. Está diseñado pensando en la legibilidad por parte de humanos. N3 es mucho más compacto y fácil de leer que la notación RDF/XML, sin embargo el modelo XML está más extendido.

N3 tiene varias características que van más allá de una serialización de RDF, como el soporte para reglas basadas en RDF. Todas la serializaciones se basan en tripletas, como se puede ver en el siguiente ejemplo de un recurso hecho en FOAF, este ejemplo lo voy a serializar en los distintos formatos:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.example.com/sibanez#me>
  a foaf:Person ;
  foaf:name "Sergio Ibañez" ;
  foaf:nick "Sergio.Ibanez" ;
  foaf:homepage <http://example.com/sibanez/> ;
  foaf:dateOfBirth "1977-05-05"^^xsd:date ;
  foaf:currentProject [
    a foaf:Proyect ;
    foaf:name "Centro de gestión de contenidos para aplicaciones"
  ] .
```

Como notación FOAF es Friend Of A Friend, literalmente "Amigo de un Amigo". Y es una ontología legible para las máquinas que describe a las personas, sus actividades y sus relaciones con otras personas y objetos. El ejemplo anterior describe que:

- Sergio Ibáñez es una Persona
- Sergio Ibáñez tiene como usuario Sergio. Ibáñez
- Sergio Ibáñez tiene como web http://example.com/sibanez/
- Sergio Ibáñez tiene como fecha de nacimiento el 05-05-1977
- Sergio Ibáñez tiene un proyecto actual
- Su proyecto actual se llama “Centro de gestión de contenidos para aplicaciones”

Turtle

Turtle es la misma notación que N3, solo que esta estrictamente hecha para RDF, lo que hace que sea un subconjunto de N3. Este tipo de serialización es muy parecido a SPARQL, y también se basa en tripletas para guardar los metadatos. En verdad todas las serializaciones se basan en tripletas. Como se ve en el siguiente ejemplo Turtle es N3 para RDF por lo tanto igual en este caso

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://www.example.com/sibanez#me>
  a foaf:Person ;
  foaf:name "Sergio Ibañez" ;
  foaf:nick "Sergio.Ibanez" ;
  foaf:homepage <http://example.com/sibanez/> ;
  foaf:dateOfBirth "1977-05-05"^^xsd:date ;
  foaf:currentProject [
    a foaf:Proyect ;
    foaf:name "Centro de gestión de contenidos para aplicaciones"
  ] .
```

RDF/JSON

JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. Este sistema es muy poco legible para el humano, ya que básicamente se usa para la transmisión entre aplicaciones, ya que es muy compacto.

```
{"http:\/\/www.example.com\/sibanez#me":{"http:\/\/www.w3.org\/199
9\/02\/22-rdf-syntax-
ns#type":[{"type":"uri","value":"http:\/\/xmlns.com\/foaf\/0.1\/Person
"}],"http:\/\/xmlns.com\/foaf\/0.1\/name":[{"type":"literal","value":"
Sergio
Iba\u00f1ez"}],"http:\/\/xmlns.com\/foaf\/0.1\/nick":[{"type":"literal
","value":"Sergio.Ibanez"}],"http:\/\/xmlns.com\/foaf\/0.1\/homepage":
[{"type":"uri","value":"http:\/\/example.com\/sibanez\/"}],"http:\/\/x
mlns.com\/foaf\/0.1\/dateOfBirth":[{"type":"literal","value":"1977-05-
05","datatype":"http:\/\/www.w3.org\/2001\/XMLSchema#date"}],"http:\/\
/xmlns.com\/foaf\/0.1\/currentProject":[{"type":"bnode","value":"_:gen
id1"}]},"_:genid1":{"http:\/\/www.w3.org\/1999\/02\/22-rdf-syntax-
ns#type":[{"type":"uri","value":"http:\/\/xmlns.com\/foaf\/0.1\/Proyec
t"}],"http:\/\/xmlns.com\/foaf\/0.1\/name":[{"type":"literal","value":
"Centro de gesti\u00f3n de contenidos para aplicaciones"}]}}
```

N-Triples

Esta es la representación más sencilla porque se basa en puras tripletas de RDF, es decir tiene un sujeto, relación y objeto, y se repite siempre así.

```
<http://www.example.com/sibanez#me>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
 <http://xmlns.com/foaf/0.1/Person> .
<http://www.example.com/sibanez#me>
<http://xmlns.com/foaf/0.1/name>
"Sergio Iba\u00F1ez" .
<http://www.example.com/sibanez#me>
<http://xmlns.com/foaf/0.1/nick>
"Sergio.Ibanez" .
<http://www.example.com/sibanez#me>
<http://xmlns.com/foaf/0.1/homepage>
<http://example.com/sibanez/> .
<http://www.example.com/sibanez#me>
<http://xmlns.com/foaf/0.1/dateOfBirth>
"1977-05-05"^^<http://www.w3.org/2001/XMLSchema#date> .
<http://www.example.com/sibanez#me>
<http://xmlns.com/foaf/0.1/currentProject> _:genid1 .
_:genid1 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://xmlns.com/foaf/0.1/Proyect> .
_:genid1 <http://xmlns.com/foaf/0.1/name>
"Centro de gesti\u00F3n de contenidos para aplicaciones" .
```

RDF/XML

Esta es la serialización digamos mixta, es una media entre N3 y N-Triples. RDF/XML como su nombre indica representa un grafo RDF, por medio del estándar XML. Además este formato es muy extendido y fácil de reconocer por usuarios habituales de internet.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/">

<foaf:Person rdf:about="http://www.example.com/sibanez#me">
   <foaf:name>Sergio Ibañez</foaf:name>
   <foaf:nick>Sergio.Ibanez</foaf:nick>
   <foaf:homepage rdf:resource="http://example.com/sibanez/"/>
   <foaf:dateOfBirth
rdf:datatype="http://www.w3.org/2001/XMLSchema#date">
          1977-05-05
   </foaf:dateOfBirth>
   <foaf:currentProject>
          <foaf:Proyect>
                 <foaf:name>
                        Centro de gestión de contenidos para
aplicaciones
                 </foaf:name>
          </foaf:Proyect>
   </foaf:currentProject>
</foaf:Person>
</rdf:RDF>
```

Hay más formatos de serialización para RDF, sin embargo estos son los más extendidos e importantes. Y tras el estudio de los formatos, se llegó a la conclusión que los dos últimos formatos eran los más adecuados para el proyecto, siendo RDF/XML el que se usa para la visión por pantalla y para el envío entre aplicaciones, mientras que se usaría N-Triples para las posibles concatenaciones, ya que son muy fáciles de hacer con ese formato.

2.1.3 Web

La decisión de utilizar un entorno Web al proyecto fue dada por el entorno actual de la programación, ya que los programas que antes se ejecutaban de forma local en los propios ordenadores, ahora se ejecutan en las famosas "nubes". Básicamente una nube es un entorno Web al que se le ha dado una forma de introducir datos, ya sean fichero como en el caso de Google Drive o Dropbox, datos comerciales como puede ser en Microsoft Dynamics CRM Online o datos en RDF como puede ser en este proyecto.

Además el entorno de un gestor de contenido online es de sobra conocido por la mayoría de los usuarios, sobre todo aquellos usuarios que usan sistemas de gestión WEB como Joombla (Gestor de contenido de artículos Web) o Prestahop (Gestor comercial de venta por internet).

Otra ventaja que se le ve a este sistema es que toda la estructura RDF, la que se forma con este proyecto, seria accesible por medio de internet. Además estas estructuras RDFS que se construyeran, podrían compatibilizar con esquemas estándar RDF que ya existen en internet, como por ejemplo el FOAF (para personas), VCARD (Para contactos) o BILBO (Para bibliografías)… Aunque en este proyecto solamente se usaran RDFs locales o internos.

Siguiendo con las ventajas, otra más, es efecto de aplicaciones responsives, es decir aquellas que funcionan tanto en un servidor torre, como en un dispositivo móvil como puede ser un teléfono. Esta situación se está dando en las distintas web empresariales de muchos dominios, por lo tanto el paso lógico es cada vez ir haciendo más aplicaciones en red y menos locales. Esto también ha hecho que tome la decisión de usar HTML5 y CSS3, que son los estándares comunes usados para aplicaciones en formato responsive. Así mismo se toma esta decisión de usar estos estándares, porque los exploradores web poco a poco se están estandarizando a seguir HTML5 y CSS3. Por supuesto para el correcto funcionamiento tendrá que ser HTML5 y CSS3 válido.

Como es una aplicación web, que aunque está colgada en internet, no es para un uso comercial, nos dará un poco igual el uso de metatags. Solamente usaremos los indispensables para el buen funcionamiento de la página, y una buena construcción de la misma, pero no para el posicionamiento SEO.

La única desventaja de hacerlo en WEB, es que obligatoriamente se necesita una conexión a internet, porque aunque para hacer los testeos se puede usar un servidor local, para tener este proyecto en producción se necesita tener un servidor conectado a internet, y mínimo un usuario también con este tipo de conexión.

En el siguiente apartado no explicaremos todo el HTML, ni el CSS. Sino que explicaremos las ventajas de usar HTML5 contra HTML4, y CSS3 contra CSS2. Ya que si usamos las ventajas de HTML5 para mejorar el proyecto, no solo estéticamente sino funcionalmente también. En definitiva usaremos HTML5 para el contenido y CSS3 para el diseño de la aplicación web.

2.1.3.1 HTML

Aunque es de sobra conocido por los usuarios de internet paso a explicar una síntesis de lo que es HTML. HTML, que son las siglas de HyperText Markup Language, hace referencia al lenguaje de marcado usado para la realización de páginas y aplicaciones web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código HTML. Es un estándar a cargo de la W3C, y que para que sea válido debe seguir la misma sintaxis que tiene XML.

Al igual que RDF, El lenguaje HTML tiene como filosofía de desarrollo el uso de referencias externas. Es decir, que para añadir un elemento externo a la página, ya sea un elemento multimedia, otra página, o un script completo, este no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de este elemento mediante texto, prácticamente igual que como se haría en RDF. De este modo, la página web sólo contiene texto y es el navegador web el que une todos los elementos y visualizar la página final. Además si se hace con estándares válidos, la teoría indica que debería verse lo mismo, aunque en la práctica no ocurre así, pero ya se están estandarizando cada vez más los exploradores.

Aunque a lo largo del tiempo ha habido diferentes versiones, en las cuales se han incorporado y suprimido diversas características, con el fin de hacerlo más rápido, eficiente y entendible, además de facilitar el desarrollo de páginas web compatibles con distintos navegadores y plataformas, efecto de hacer todo responsive. Hay ciertas características o etiquetas que se han mantenido a lo largo de estas versiones para que haya cierta compatibilidad entre los distintos navegadores web. Las etiquetas comunes son las siguientes:

Etiquetas principales del código HTML

- <html></html>: Define el inicio y el fin del documento HTML, lo que le indica al navegador que lo que viene a continuación debe ser interpretado como código HTML. Como se ve es un lenguaje estructurado igual al XML., pues to que este es el elemento raíz del documento. E igual que el XML antes de este elemento raíz tiene una línea de código que dice exactamente qué tipo de documento es , en este caso es la línea con la instrucción DOCTYPE

- <head></head>: Contiene la cabecera del documento HTML, y esta cabecera suele contener información sobre el documento que no se muestra directamente al usuario como, pero que usa el para tener más información sobre la página. En esta parte se meten los metadatos también.
- <body>: La parte principal del documento. Esta es la parte del documento que se muestra en el navegador al usuario. En definitiva es la web en sí misma, porque todo el contenido de la web está aquí incluido.

Etiquetas principales de la etiqueta head

- <title></title>: Es el título de la página, además de que un buscador al estilo Google lo usa como referencia del contenido de la web, este título aparece en la barra que hay encima de la ventana del explorador.
- <script></script>: Inserta un script completo en una web, sea en JavaScript o ecmascript definido por el atributo type, o llama a uno mediante src que contendrá la url del script. Se recomienda incluir el tipo MIME en el atributo type, en el caso de JavaScript text/javascript, aunque en HTML5 esto está por defecto.
- <link></link>: Es un link para vincular el sitio a hojas de estilo o iconos. Tiene un atributo para generar el vínculo que es href y type para decir que tipo de link es.
- <style></style>: Estilos internos de la página, pero es mucho mejor normalmente usar <link>.
- <meta>: Para metadatos como la autoría o la licencia, aunque es este proyecto se le dan importancia de otra manera.

Etiquetas principales de la etiqueta body

- <h1></h1> a <h6></h6>: Títulos del documento con diferente relevancia siendo <h1> el más relevante y <h6> el menos. Tal como ocurre en un documento formal, normalmente solo hay un título h1 general.
- : Define un párrafo o trozo de texto.
- <table></table>: Forma dentro del documento una tabla. Esta tabla define sus filas con la etiqueta <tr></tr>y sus celdas dentro de la fila con la etiqueta <td></td>. Además en la tablas puede haber encabezados de forma opcional con la etiquera <th></th>.
- <a>: hipervínculo o enlace, dentro o fuera del sitio web. Debe definirse el parámetro del enlace por medio de una url absoluta o relativa en el atributo href.
- : Agrupación de etiquetas HTML, normalmente denominadas capa.
- <img></img>: imagen. Requiere del atributo src, que indica la ruta en la que se encuentra la imagen. Por accesibilidad, se suele poner un atributo alt="texto alternativo por si no hay imagen".

- /ul> y : Etiquetas para generar listas en el documento HTML desordenada y ordenadamente respectivamente. Cada uno de los ítem de las listas se encierran en las etiquetas .
- Además también hay etiquetas que dan diseño al contenido, sin embargo normalmente es mejor usar CSS. o texto en negrita; <i></i> o <para el texto en cursiva; <s></s> o <del></del> para el texto tachado; y <u></u> para el texto subrayado.

El HTML, como lenguaje de tags, requiere que sus etiquetas se abran y se cierren, al igual que en el caso de XML. En algunos casos se puede cerrar con el elemento "/>" si es un elemento vacío, como puede ocurrir en una imagen, y en otros casos se debe cerrar con el elemento </nombredelelemento> como puede ocurrir en una tabla. Ilustrándolo con ejemplos:

```
<img src=http://www.ejemplo.com/miimagen.jpg alt="mi imagen"/>
<table>
    <tr><td>celda fila 1 columna 1</td><td>celda fila 1 columna 2</td></tr>
    <tr><td>celda fila 2 columna 1</td><td>celda fila 2 columna 2</td></tr>
</table>
```

Hemos dicho que la etiqueta <script> está definida entre las etiquetas incluidas en el head, sin embargo eso no es cierto del todo, ya que se puede definir en cualquier parte del documento web, siempre y cuando siga los estándares definidos por W3C. Ya que aunque la sintaxis es como la de un XML, para que un documento HTML se vea correctamente debe de seguir unas reglas asociadas a este lenguaje.

En definitiva, la pregunta es ¿Por qué usar HTML5?, básicamente nos da estos beneficios sobre estándares anteriores como HTML4.0 o HTML4.1

- Es código más simplificado, luego carga más rápido en el navegador.
- Las páginas y los elementos que contienen, se ven perfectamente en todos los navegadores. La gran mayoría de los navegadores de los teléfonos Smartphone y las tabletas, son compatibles con HTML5, si posees uno de estos dispositivos puedes comprobarlo, accediendo con él a la siguiente página: Detector e identificador de dispositivos móviles
- Los plugins, widgets y botones que ofrecen los desarrolladores de las redes como Facebook, Twitter y otras, escritos en HTML 5 funcionan excelentemente, con más opciones que los clásicos en XHTML o que los iframes.
- Es posible insertar directamente videos en las páginas sin tener que acudir a los iframes o usar la etiqueta object.

- HTML 5 incluye etiquetas orientadas principalmente a los buscadores, para facilitarles comprender el contenido de las páginas, lo que nos beneficia, por ejemplo: header, footer, article, nav, etc.
- Permite la geolocalización del usuario.
- Otras de las razones es el empleo del micro formato en las páginas web, que algunos son totalmente incompatibles con otros lenguajes por lo que no validan correctamente a no ser que se use HTML5. estar una misma página web.

2.1.3.2 Diferencias entre HTML5 y HTML4/XHTML

Cuando nos referimos al html4, normalmente nos referimos al estándar de HTML4 y XHTML , puesto que XHTML contiene también las mismas etiquetas que HTML4, solo se diferencian en la sintaxis, ya que en HTML no se obliga a cerrar los elementos mientras que en XHTML sí. Vamos a enumerar las principales diferencias entre los dos estándares:

Estructuración del contenido web

Antes en HTML4 la única estructuración que había era básicamente los títulos desde <h1> hasta <h6>. Sin embargo en HTML5 se han puesto nuevos elementos con el fin de saber que es la cabecera o el pie de página del contenido web. Estos elementos no solo le dan una estructura sino que le dan un significado semántico ya que divide el contenido en distintas zonas. Estas etiquetas son:

<header> : Es el equivalente a la cabecera de la página web. Contiene el título principal del contenido WEB, que puede ser un logo de empresa.

<nav>: Contiene los enlaces (barra de navegación) externos o internos de la página. Puede haber más de un <nav> porque puede haber más de un menú de navegación

<section>: Es una gran caja que sirve para mostrar grandes bloques de contenido de la página. Puede contener diferentes subapartados de diferentes temas normalmente de tipo <article>.

<article>: Es una caja independiente de contenido que puede estar contenida (o no) dentro de un <section>. Normalmente utilizada para contenidos no demasiado extensos.

<aside>: Define un bloque de contenido relacionado de manera indirecta con el contenido principal, pero que no es esencial para la compresión del mismo.

<footer>: Equivale al pie de página de un apartado concreto (<section>, <article>...) o de la página web en general.

Separación entre diseño y contenido web

En HTML5 se han dejado obsoletos elementos y etiquetas referidas puramente al diseño. Etiquetas como <font>, que se usa para la selección de una fuente, o , que se usa para centrar contenido, han sido eliminadas para usar las hojas de estilos escritas en CSS3 para dar diseño. Además gracias al CSS3, se pueden realizar acciones de diseño que antes no se podían hacer más que con parches de JavaScript, por lo tanto la separación entre diseño y contenido es necesaria hasta en la programación. Aunque se hablara un poco más de estos cambios en el apartado siguiente de CSS3.

Inserción de elementos multimedia

Debido a que en la actualizas todos los exploradores van sobre dispositivos capaces de reproducir videos o sonidos, en HTML5 se han agregado etiquetas para poder incrustar en el código HTML directamente videos y sonidos. Estas etiquetas son < video>, para la inserción de videos, y <audio> para los sonidos. El tratamiento de estos elementos prácticamente es igual que el de una imagen, es decir se carga el video o el sonido a través del atributo src, y el navegador se encarga de poner en la web un reproductor multimedia. Como nota decir que no se pueden reproducir todos los formatos de video, sino que en video solo pueden ser los códec de MP4, WebM y Ogg, y en audio los formatos Mp3, Wav y Ogg. Luego si se quiere reproducir otro formato hay que reformatearlos a los que reproduce el navegador de forma estándar.

Elementos Input inteligentes de formularios

Otra de las grandes diferencias de HTML5 respecto a los anteriores es lo denominados Inputs o entradas inteligentes. En los estándares de antes, cuando realizabas un Input para que un usuario de internet metiera datos, se tenía que poner *type*=”text”, para que metiera texto, y después poner medio de un script se tenía que identificar si esa entrada era correcta. Ahora gracias el nuevo estándar de HTML5, además del tipo *button o checkbox,* el *type* no solo tiene porque ser text. Puede ser cualquiera de los tipos siguientes:

- Color: Te genera un control para elegir un color en RGB
- Date: Una fecha valida
- Datetime: Una fecha y hora valida
- Email: Un email valido
- Month: Un mes con su año validos
- Number: un numero valido
- range: Un rango de enteros valido
- tel: Un teléfono valida
- time: Una hora
- url: una url valida
- week: Un día de la semana valido

También gracias a su atributo *required,* puedes obligar a un usuario a que rellene los campos necesarios para el envío de un formulario. Además con su otro atributo *pattern* puede rellenarse con una expresión regular, para que compruebe el contenido con esa expresión regular. Y todo ello sin ningún script, lo hace el navegador si sigue el estándar HTML5. Aunque estos inputs no están totalmente incluidos en los diferentes navegadores a día de hoy, sin embargo el estándar obligara a que puedan ponerse en cualquier formato HTML5.

2.1.3.3 Diferencias entre CSS3 y CSS

Como ya hemos dicho antes CSS o la hoja de estilos, es un formato de fichero para dar diseño a un documento escrito en HTML o XHTML, aunque CSS3 está muy ligado a HTML5. Normalmente cuando se habla de CSS2 se habla de su revisión CSS2.1, ya que corrigió muchos errores que había en su primera versión. Todas las versiones de CSS tienen muchas de sus características de diseño en común, como el fondo o el color y tamaño de la fuente. Sin embargo CSS3 tiene algunas funcionalidades extras que mejoran su rendimiento.

Sombras y bordes redondeados

Aunque esto parezca algo que ya se ha visto desde hace años, la verdad es que antes de la aparición de CSS3, había muchos tipos de parches para hacer esos bordes redondeados o sombras. Todos ellos eran una amalgama de script e imágenes, que lo que hacían era dar contenido extra a una parte del documento que era solo para diseño. Como en HTML5 y CSS3 se pretende separar por completo el diseño del contenido, se ha añadido unos atributos a los documentos de CSS para que se pueda redondear y sombrear todos los elementos de HTML.

Consulta de medios

En CSS3 se ha agregado lo que se denomina media querys, esto es una de las principales razones por las cuales se pretende usar CSS3. Estas querys lo que hacen es consultar medidas del dispositivo, anchura, altura, etc... Siendo esto perfecto para realizar una aplicación en formato responsive, ya que se puede consultar la medida de un dispositivo y hacer la presentación e3n pantalla en función de las medidas. Esto no solo es para las medidas del dispositivo ya que también pueden hacerse consultas sobre el navegador que se está usando para optimizar su presentación.

Efectos básicos

En CSS3 se han agregado funcionalidades de efectos básicos como hacer aparecer una lista de botones si se pone el puntero en cierto sitio de la web. Anterior mente a CSS3 todo esto se hacía por medio de JavaScript, pero al tener estos efectos ya no son necesarios conocimientos de JavaScript para el procesado de estos efectos.

Carga más rápida

CSS3 carga más rápido en los navegadores porque no va en bloque sino que lo hace por medio de "módulos" es decir va como por trozos, y hace que el contenido ya se cargue mientras se le está dando diseño. Sin embargo hay una desventaja actualmente en CSS3 respecto a CSS 2.1, y es que CSS3 es relativamente nuevo, con lo que tiene cierta inestabilidad como ocurre con el HTML5, los navegadores no están 100% preparados para este formato. Sin embargo los beneficios que da bien valen las posibles "inestabilidades" que produce, ya que se tendera a que los navegadores soporten 100% CSS3 y HTML5.

2.1.3.4 JavaScript

El lenguaje JavaScript, en internet abreviado comúnmente como JS, es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, dinámico, muy débilmente tipado. La ultima característica tiene sus ventajas e inconvenientes, debido a que al ser débilmente tipado se puede agregar un entero a un float sin tener que intercambiar el tipo, pero hace que muchos desarrolladores de software al intentar atajar, caigan en errores comunes y se desarrolle software de funcionamiento ineficiente.

Se utiliza principalmente en su forma del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Aunque existe una forma de JavaScript del lado del servidor. También se puede extender su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (widgets), etc...

El lenguaje JavaScript se diseñó con una sintaxis similar al C o PHP, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados para nada y tienen semánticas y propósitos muy diferentes. Recordemos que Java está fuertemente tipado, por ejemplo.

Todos los navegadores modernos, de manera opcional, interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). Sin embargo cada explorador tiene "su interprete JavaScript", lo que hace que tengan diferentes operaciones y funciones que unos navegadores reconocen y otros no. Haciendo una similitud existen diferentes dialectos como ocurriría con el Euskera, sin embargo hay comunidades de desarrollo que se dedican a unificar dichos dialectos en un idioma común. Si en el Euskera es el Batua en JavaScript sería el jQuery, del cual hablaremos en un apartado posterior

JavaScript normalmente depende del entorno en el que se ejecute (por ejemplo, en un navegador web) para ofrecer objetos y métodos por los que los scripts pueden interactuar con el "mundo exterior". De hecho, depende del entorno para ser capaz de proporcionar la capacidad de incluir o importar scripts. Por ejemplo, en HTML por medio del tag <script>, como ya comentamos en el apartado HTML.

2.2 Requisitos de software

Una vez decidido las tecnologías generales que se van a ir usando al proyecto, debemos tener un soporte software con el cual usarlas. En el mercado hay multitud de softwares que interactúan con HTML, CSS, RDF... etc... Toda esta multitud de softwares vienen implícitos a la época que vivimos de realizar todo en internet. En la actualidad sin internet parece que se pararía el mundo.

Por ver un poco unos ejemplos vamos a definir qué servicios necesitamos para la realización del proyecto. Como se trata una aplicación para usuarios de internet, necesitamos un servicio que haga de transporte de información entre este usuario y un servidor en internet. Entre los distintos servicios hay dos principales, IIS para plataformas basadas en Windows y Apache para plataformas basadas en Unix. Debido que el proyecto lo queremos hacer con el menor coste posible y Apache es software libre pero es incompatible con la GPL, puede usarse sin problemas.

Otra necesidad es la de guardar los datos, por lo que necesitamos un servicio que pueda guardar estructuras de datos, en definitiva bases de datos. Estas estructuras que necesitamos están escritas en RDF hay muchas posibilidades. Una de las posibilidades es utilizar archivos de texto planos para guardar esta información, que dado que XML es un formato de archivo, es muy sencillo querer almacenar datos XML como un archivo plano. Sin embargo, presentan los inconvenientes asociados a la gestión de ficheros de este tipo: falta de concurrencia, comprobaciones de integridad, atomicidad, recuperación, seguridad… La otra es utilizar sistemas de gestión de bases de datos nativas a XML, como Oracle Berkeley DB XML, y su lenguaje de consultas XQuery, que en un principio parece interesante. Pero al final la decisión la vuelve a marcar un poco el coste y el balancear coste con efectividad. Por lo que se optó por un sistema de gestión de base de datos relacional, que sea gratuito. En el caso que nos ocupa usamos uno de los más extendidos MySQL con su lenguaje de consulta SQL. Esta decisión no solo vino marcada por el coste sino que también las bases de datos relacionales están muy extendidas, lo que hace que los sistemas de gestión de bases de datos relacionales estén mucho más desarrolladas y depuradas en la actualidad que los sistemas XML nativos., aunque se están poniendo al día, debido precisamente al termino web semántica, que tiene mucho que ver con estructuras RDF.

Una vez ya decidido que se iba a usar Apache y MySql, el paso de en qué programar en el servidor viene dado es PHP, ya que Apache, MySql y PHP vienen dados de la mano en casi todos los proyectos. Si miramos estos 3 servicios veremos que su origen viene de plataformas Unix, pero en la actualidad contamos ya con lo mismo para plataformas Windows. Esta es una de las diferencias del LAMP y el WAMP. En los siguientes apartardos explicaremos un poco cada servicio para dar una instrucción al diseño del proyecto.

2.2.1 Apache

Apache es un servicio HTTP , o servicio web que procesa una aplicación, en este caso el proyecto actual, del lado del servidor, realizando entre el servidor y el usuario, generando una respuesta que recibirá un aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo. Generalmente se usa el protocolo HTTP para estas comunicaciones.

El protocolo HTTP o HyperText Transfer Protocol (Protocolo de transferencia de hipertexto) es el método más común de intercambio de páginas web en la world wide web. Como queremos realizar una aplicación web necesitamos de este protocolo que transfiere información entre servidores y clientes. Existe una variantes que es más segura denominada HTTPS o HyperText Transfer Protocol Secure, sin embargo en un principio no se ve necesario más seguridad para el proyecto , puesto que no son informaciones de pago o sensibles.

Como ya hemos dicho Apache es un poderoso servidor web, cuyo nombre proviene de la frase inglesa "a patchy server" y es completamente libre, ya que es un software Open Source. Una de las ventajas más grandes de Apache, es que es un servidor web multiplataforma, IIS no lo es, es decir, puede trabajar con diferentes sistemas operativos y mantener su excelente rendimiento. Además desde el año 1996, es el servidor web más popular del mundo, debido a su estabilidad y seguridad. Apache sigue siendo desarrollado por la comunidad de usuarios desarrolladores que trabaja bajo la tutela de Apache Software Foundation.

El funcionamiento de un servidor Apache es el que se presenta en el siguiente diagrama, bueno en general un servidor HTTP funciona como el diagrama que se presenta a continuación:

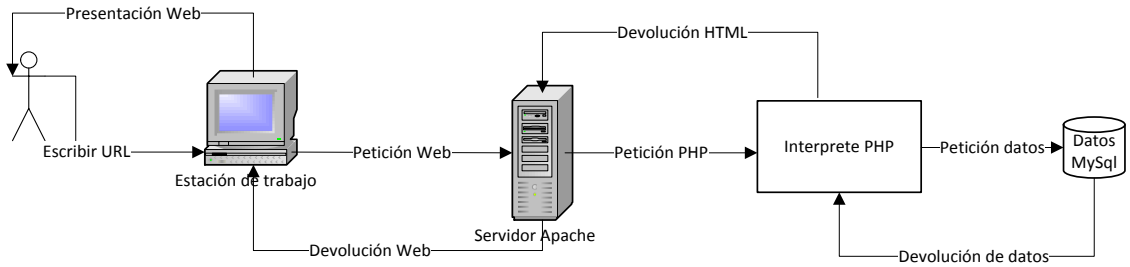


Figura 6 –Esquema de una petición a servidor Apache

El funcionamiento es el siguiente, un usuario abre su navegador y escribe una URL o pincha un enlace, el ordenador personal por medio de traducir esa URL a Ip enlaza con un servidor, por medio de una petición web. Este servidor comprueba que tipo de archivo es el que pide y si lo puede servir. Si es un archivo PHP le dice a su interprete PHP que lo necesita a través de la petición PHP. Este intérprete ejecuta el script en PHP, y pide datos a las bases de datos pertinentes por medio de peticiones de datos, normalmente en SQL. Una vez que el intérprete ha ejecutado todo el código script devuelve un código HTML que el servidor Apache recoge. El servidor apache entonces empieza a devolver el código HTML al ordenador personal y este lo interpreta visualmente en la pantalla para el usuario. Este caso es si se pediría un documento PHP que no tuviera nada más que el propio PHP. Sin embargo como se ha visto en el apartado HTML una Web no solo se constituye de ese código HTML, sino que tiene distintos links de otras páginas, o links a archivos CSS, etc... En este caso el servidor Apache se encarga de ir mirando en la petición Web hay enlaces a distintos archivos, dando a el intérprete PHP lo que están en PHP para que le devuelva el código HTML correspondiente.

Una de las mejores ventajas que tiene Apache es que es muy modular, o sea que tiene un núcleo básico con el cual hace el funcionamiento del diagrama anterior, pero tiene módulos independientes que amplían y mejoran su funcionalidad. Algunos de ellos son muy interesantes, como por ejemplo el mod_rewrite o módulo de reescritura de direcciones. Este módulo generalmente es utilizado para transformar páginas dinámicas, como las que están escritas en PHP, en páginas estáticas HTML. Esto se hace para engañar a los navegantes o a los motores de búsqueda, con url amigables y tener así esas direcciones compresibles para los usuarios.

PHP se puede instalar como un module del Apache, aunque normalmente no es así ya que cuando PHP es usado como un módulo de Apache, hereda los permisos del usuario de Apache (generalmente los del usuario "nobody"). Este hecho representa varios impactos sobre la seguridad y las autorizaciones. Por ejemplo, si se está usando PHP para acceder a una base de datos, a menos que tal base de datos disponga de un control de acceso propio, se tendrá que hacer que la base de datos sea asequible por el usuario "nobody". Esto quiere decir que un script malicioso podría tener acceso y modificar la base de datos, incluso sin un nombre de usuario y contraseña.

En definitiva Apache tiene multitud de módulos configurables por el administrador del servidor. Es de código abierto por lo que es gratuito. También ya existe en multitud de plataformas, lo que lo hace multi-plataforma. Es extensible ya que se puede hasta recompilar si fuera necesario su núcleo. Pero lo más importante es que es muy muy popular, lo que hace fácil conseguir ayuda y soporte de distintos usuarios o foros. Es por todo esto lo que es una buena elección para este proyecto

2.2.2 MySQL

Como se ha dicho anteriormente, se ha decidido que en el proyecto se usara una base de datos relacional, es por ello que hemos buscado un sistema de gestión de bases de datos relacionales, en este caso MySQL. MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario, muy pero que muy extendido por los usuarios de internet. Entre los sitios web, grandes y populares, por los que es usado tenemos Wikipedia, Google, Facebook, Twitter, Flickr y YouTube. Como se ve es muy extendido, esto al igual que Apache hace que sea fácil encontrar tanto soporte como ayuda por parte de terceros. MySQL al igual que Apache tiene una licencia GNU GPL.

MySQL está desarrollado casi por completo en ANSI C, al igual que Apache, pero la interactuación con las bases que datos que crea se hace por medio de SQL en la mayoría de las veces. Es por ello que hay multitud de aplicaciones que hacen uso de esto, para no tener que usar la consola de comandos de MySQL. Estas aplicaciones pueden ser programadas en C, C++, C#, Pascal, Delphi, Java, Lisp, Perl, PHP, Python, etc.… cada uno de estos utiliza una interfaz de programación de aplicaciones específica. También existe una interfaz ODBC, llamado MyODBC que permite a cualquier lenguaje de programación que soporte ODBC comunicarse con las bases de datos MySQL. También se puede acceder desde el sistema SAP, lenguaje ABAP. Sin embargo como ya se ha decidido usar PHP, usaremos una aplicación construida en PHP para la administración de la base de datos, en este caso el PhpMyAdmin de la cual hablaremos al final de este apartado.

En la figura 6, hemos visto que una aplicación escrita en PHP como nuestro proyecto, pide a la base de datos relacional, escrita en MySQL, datos por medio de una petición de datos. Esta petición normalmente se hace en SQL, aunque PHP tiene algunos mecanismos para interactuar directamente con una base de datos escrita en MySQL, como por ejemplo saber que ultimo identificador se ha insertado en la base, normalmente se prepara una consulta en SQL y a posteriori se ejecuta contra la base de datos, dando un resultado ya se a una tupla, un entero como resultado, un resultado vacío o error con su identificador de error. Este resultado se devuelve a través de una devolución de datos, que normalmente esta en PHP representado por un objeto de programación o una estructura de datos, donde se recoge el resultado, para después poder ser usada en el sitio correspondiente de nuestra aplicación.

En definitiva MySQL es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido. Velocidad, integridad, flexibilidad y gratuito, perfecto para el proyecto que nos ocupa.

2.2.2.1 SQL

Para entender bien lo que es MySQL hay que conocer su lenguaje de consultas. El lenguaje de consulta estructurado o SQL (por sus siglas en inglés Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Una de sus características es el manejo del álgebra y el cálculo relacional que permiten efectuar consultas con el fin de recuperar, de forma sencilla, información de bases de datos, así como hacer cambios en ellas.

Debido a que hay numerosos manuales tanto offline como online, tampoco nos vamos a explayar mucho en lo que es SQL, porque nos daría páginas para otro proyecto. Sin embargo vamos a destacar las operaciones de manipulación de datos fundamentales que tiene SQL, y que vamos a usar en nuestro proyecto sobre RDF. Estas son las siguientes:

SELECT

La sentencia SELECT nos permite consultar los datos almacenados en una tabla de la base de datos, estos datos pueden estar en multitud de formatos pero lo más interesante es que posee un tipo llamado memo en el cual se puede almacenar textos de gran tamaño.

INSERT

Una sentencia INSERT de SQL agrega uno o más registros a una (y sólo una) tabla en una base de datos relacional. Las cantidades de columnas y valores deben ser iguales. Si una columna no se especifica, le será asignado el valor por omisión. Los valores especificados (o implícitos) por la sentencia INSERT deberán satisfacer todas las restricciones aplicables.

UPDATE

Una sentencia UPDATE de SQL es utilizada para modificar los valores de un conjunto de registros existentes en una tabla. Al igual que INSERT las restricciones aplicables para los valores deben de cumplirse.

DELETE

Una sentencia DELETE de SQL borra uno o más registros existentes en una tabla. Todas las operaciones tienen la posibilidad de agregarse una condición WHERE para saber si se tienen que seleccionar editar o borrar. En el caso de INSERT se usar para copiar filas.

2.2.2.2 PhpMyAdmin

La aplicación web PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de servidores MySQL en las aplicaciones web. Actualmente esta herramienta puede crear y eliminar bases de datos; crear, eliminar y modificar la estructura de las tablas y vistas; añadir, eliminar y editar campos de las tablas y vistas; en definitiva ejecutar cualquier sentencia SQL. También puede administrar claves en campos, administrar privilegios de usuarios desde root a simple visualizador, gestionar disparadores y procedimientos almacenados, exportar datos en varios formatos. Y una cosa muy importante es que está disponible en 62 idiomas entre ellos el español.

2.2.3 PHP

Como ya hemos hablado en apartados anteriores, ya hemos decidido utilizar Apache MySQL, entonces lo único que nos falta si vemos la figura 6, es "algo" que interactúe entre los datos que tenemos guardaos y la página web escrita en HTML. En este punto es donde entra en juego un intérprete PHP para hacer esa función.

PHP es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. Es decir, en lugar de usar muchos comandos para mostrar HTML, como sería en lenguajes como C o Perl, las páginas de PHP contienen HTML con código incrustado que hace "algo". El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP", aunque en la actualidad se intenta separar completamente la parte PHP con la parte HTML, haciendo tantas inclusiones de archivos como sean necesarias para este fin.

Lo que distingue a PHP de algo del lado del cliente como JavaScript es que el código es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script, aunque no se sabrá el código subyacente que era. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué se tiene debajo de la manga.

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes.

Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo. PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy, lo que ha atraído el interés de múltiples sitios con gran demanda de tráfico, como Facebook, para optar por el mismo como tecnología de servidor. Como anotación recordemos que PHP puede ser un módulo de Apache o no, ya que puede ser estar desvinculado a ser un servicio aparte. Si ejecutas Apache y PHP en Windows se ve claramente la diferencia ya que hay una aplicación que ejecuta PHP.

En definitiva PHP es un lenguaje extendido, flexible y muy rápido y eficaz. Aunque por el contrario tiene algunas deficiencias, como por ejemplo que no es altamente tipado, y cuando se ejecuta aplicaciones orientadas a objetos no las compila, sino que las trata de incrustar en el código.

2.2.4 jQuery

Como se ha mencionado en el apartado JavaScript, necesitamos un sistema para unificar todas las posibles interpretaciones del JavaScript en distintos navegadores. La necesidad de tener JavaScript en este proyecto es porque para la aceleración y mejor funcionamiento del proyecto se utilizara la técnica AJAX.

jQuery es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. JQuery consiste en un único fichero JavaScript que contiene las funcionalidades comunes de DOM, eventos, efectos y AJAX. La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o jQuery().

Una de las principales ventajas es que funciona independientemente del navegador sobre el que se visualiza la aplicación Web. Al servir para todos, el desarrollador tiene menos trabajo, menos código, menos espacio, menos problemas… Otra ventaja que tiene jQuery, es existen cientos y cientos de plugings, incluidos los que pueden manipular RDF. Adaptables al tipo de proyecto que estés desarrollando. Aunque debido a que es una biblioteca, se tiene que aprender términos que no existen en JavaScript nativamente, por regla general, podemos considerar su sintaxis liviana, teniendo en cuenta que la sencillez y poca extensión de código es fundamental para los desarrollos. Además volviendo a lo que llevamos dando vueltas en este proyecto, su licencia es GPL, lo que hace que no tenga coste extra el usar jQuery.

2.2.4.1 Ajax

Para dar fluidez al proyecto, utilizaremos un sistema AJAX, acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en los navegadores mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones. Ajax es una tecnología asíncrona, los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el funcionamiento de la página. JavaScript es el lenguaje interpretado en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML. En jQuery hay una instrucción simple para realizar comunicaciones en AJAX, lo que reduce significativamente la complejidad de hacer una llamada AJAX.

2.3 Requisitos de Aplicaciones

He englobado en este grupo los requisitos de aplicaciones necesarias, para manejar el software del apartado anterior y las tecnologías indicadas en el apartado 2.1. No voy a explicar cada una de las aplicaciones, ya que cada una de ellas tiene un manual con el cual se podría investigar todo el funcionamiento. Sino que voy a hacer una breve descripción de ellas y por qué usarlas en este proyecto.

2.3.1 SublimeText

El programa SublimeText, es un editor de texto plano, escrito en C, sencillo pero muy potente, debido a que vamos a usar PHP en el proyecto necesitamos algún tipo de editor para poder escribir el código. Este editor además tiene muchos módulos adicionales referentes al código PHP, que permite acelerar el trabajo cuando se está editando. Sublime Text se distribuye de forma gratuita pero no tiene una licencia GPL, sin embargo sí que se puede hacer uso de él de forma ilimitada. Otra ventaja que tiene el Sublime Text, es que te permite agrupar tus diferentes ficheros de código PHP en proyectos y grupos de trabajo. Y por si fuera poco es multi plataforma.

2.3.2 Adobe Photoshop

Adobe Photoshop es un editor de gráficos vectoriales y rasterizados. Aunque el uso fundamental de este programa es el retoque fotográfico,

2.3.3 PhpMyAdmin

En el apartado de MySql ya hemos hablado del PhpMyAdmin, solamente destacar que para el proyecto ha sido necesario, como sistema de depuración, y observación de la base de datos. Debido a que las sentencias SQL hechas para el proyecto son ejecutadas directamente con código PHP, no a través de PhpMyAdmin.

2.3.3 Filezilla Client

Este programa está especializado en hacer transferencias de archivos FTP, lo usaremos para poder subir el proyecto actual del ordenador local, a un servidor web para que tenga un servicio hacia el exterior. Ya que todas las depuraciones se hacen en un ordenador local. También tiene licencia GPL.

2.3.4 Firefox, Internet Explorer y Chrome

Usaremos distintos navegadores Web para observar si la aplicación web tiene bien hecho el diseño responsive. Ya que cada navegador tiene sus peculiaridades respecto a este asunto del diseño responsive. Además usaremos tanto los formatos de ordenador como de móvil.

2.3.5 WAMP y LAMP

Las siglas WAMP indican lo siguiente, Windows Apache MySQL PHP server, mientras que las de LAMP indican lo mismo solo que bajo plataforma Linux. Utilizaremos WAMP en el ordenador local para poder hacer las distintas depuraciones y optimizaciones del código PHP, ya que bajo Windows la instalación de este tipo de servidor es muy rápida y sencilla. Sin embargo para el servidor de producción usaremos LAMP, porque al fin y al cabo el WAMP es una simulación del LAMP. Sin embargo el proyecto debería de funcionar en las dos plataformas por igual con la misma configuración, solamente que con un poco más de velocidad y menos gasto de recursos en el sistema LAMP.

Debido a lo que hemos dicho en los distintos apartados de que el servidor Apache, el servicio MySQL y el lenguaje PHP tienen licencia GPL, WAMP también tiene el mismo tipo de licencia. Es decir que se da de forma gratuita, y puede instalarse en cualquier ordenador personal. Para la instalación de Linux suele tener algo más de complicación por lo tanto en el anexo sobre la instalación del proyecto se hablara de la instalación de WAMP.

Solo indicar que para un mejor aprovechamiento para RDF de las URIS, en el anexo de instalación se indicara como se hace una Virtual Host, o directorio virtual dentro de WAMP. Todo ello para no tener que manejar IP's o utilizar la dirección http://localhost tan común, y poder utilizar algo más acorde con una dirección web real.

2.4 Requisitos de hardware

No vamos a destacar el uso de ningún hardware especial, ya que con un ordenador personal para hacer los cambios de código y emulación del mismo, tendríamos suficiente para ver el proyecto en funcionamiento. Sin embargo como se quiere también utilizar como aplicación web real, necesitaremos un servidor externo que haga algún tipo de hosting, en este caso nos hemos decidido por un servidor externo en 1and1, para los backups y las versiones "estables"

3. ANÁLISIS DE FACTIBILIDAD

Para determinar si este proyecto es factible, hay que revisar varios puntos de este proyecto. Como por ejemplo si requiere excesivo conocimiento técnico para el creador del mismo, o si representa un gran gasto económico para el mismo.

Nivel técnico

Debido a que en la actualidad la realización de páginas web o aplicaciones web, está a la orden del dia hay muchos foros al respecto. Además con la experiencia del creador del proyecto sobre páginas web, se tienen más recursos sobre ello.

Por otra parte RDF y XML está muy bien documentado en las páginas de W3C. Al mismo tiempo XML no es nada desconocido debido a su uso a nivel popular, tanto en páginas web como en aplicaciones.

Nivel económico

A nivel económico, dentro de todos los requisitos, se ha intentado usar todo lo posible con GPL, o licencia de uso público. Para que el desarrollo del proyecto fuera gratuito. El único software que es de pago es el de Adobe, pero ya se posee licencia luego no hay problema

Nivel de complejidad

Al igual que el nivel técnico, el desarrollo de este proyecto presenta ciertos retos, sin embargo debido a la experiencia del creador con entornos de desarrollo en PHP, se puede determinar que habrá poco problemas para ir avanzando en el proyecto.

Tiempo

Como solo se puede determinar a priori el tiempo necesario, por proyectos o aplicaciones similares, se puede determinar que si es factible en el tiempo realizarlo

Tras el estudio de los puntos necesarios para saber si es factible, se puede determinar que el proyecto lo es. Ya que el tiempo, conocimiento y economía están dentro de los márgenes que pueden determinarse como factibles.

4. DOCUMENTO DE OBJETIVOS DEL PROYECTO

El proyecto Centro de Gestión de Contenido en formato RDF, consiste en el desarrollo de un CMS (Content management system), para poder crear, modificar y eliminar contenido, que después podría alimentar desde una web a una app de un móvil. Como su nombre indica la exportación de ese contenido será en formato RDF.

Este CMS también tendrá un sistema de seguridad basado en usuario y contraseña. Para que la edición del contenido sea accesible solo a usuarios con acceso, mientras que la exportación de los datos será pública, y podrá ser accesible por cualquier usuario mediante una URI. Además el formato del HTML que se usará para la visualización de los distintos formularios, será HTML5 y CSS3 en un formato responsive para poder ser visualizado en dispositivos móviles.

En el documento de objetivos del proyecto se expondrá tanto los objetivos a alcanzar en el proyecto, como las motivaciones del mismo, además de los pasos, denominados hitos, para la realización del mismo. También se expondrán cada paso del alcance del proyecto, como la metodología y la planificación.

4.1 Objetivos y motivaciones del proyecto

4.1.1 Objetivos del proyecto

El objetivo de este proyecto, es realizar una interfaz para usuarios, una interfaz de programación de aplicaciones (API) y los procesos independientes necesarios que controlarán una estructura que contendrá datos en RDF, donde se alojará el contenido, que después se usará para distinto tipo de aplicaciones, desde una web a una aplicación móvil para distintos fines. Esta interfaz se realizará en un sistema WAMP (Windows + Apache + MySQL + PHP) para el desarrollo, mientras que se pondrá en un LAMP (Linux + Apache + MySQL + PHP) cuando este en producción, esta decisión se ha tomado por el hecho de ser gratuito y muy flexible.

El gestor de contenido será una aplicación web informática que creará, editará, gestionará y exportará contenido digital que será enviado a distintas aplicaciones. Este sistema de gestión utilizará el formato RDF/XML o XML semántico para la exportación del contenido, que alojara una base de datos del servidor. Para que haya un sencillo manejo para los usuarios, se utilizará un API para poder exportar los datos de la base.

El hacer este gestor permitirá que los usuarios, que no tengan conocimientos de programación o diseño de aplicaciones, gestionar su contenido como si de una página web se trataran. Ya que el uso de los Blog's o de las aplicaciones web informáticas, como Joombla o Wordpress, están extendidos. Y es más sencillo construir el contenido con una aplicación muy parecida a estas últimas herramientas mencionadas.

Para poder llegar al objetivo final será necesario cumplir los siguientes hitos:

- Definir el modelo de base de datos para almacenar la información de la plataforma
- Construir la interfaz de usuario para la manipulación del contenido
- Realizar un API para la exportación del contenido en formato RDF/XML
- Desarrollar el sistema de seguridad
- Testear todo lo anterior, para comprobar el correcto funcionamiento.
- Desarrollo continuo de la documentación

4.1.2 Motivaciones del proyecto

Las motivaciones por las cuales he decidido realizar este proyecto, se pueden clasificar en tres motivaciones fundamentalmente:

- La primera es la extensión de las aplicaciones móviles en el mercado y la navegación, a día de hoy, muy extendida en dispositivos móviles.
- La segunda es la abundancia de CMS's en el mercado para páginas web, pero ninguna para APP's, o si las hay desde luego no son de bajo coste. En la actualidad, ya las app's para móviles son tan extendidas, como una página web. Es por ello que es necesario algún tipo de mecanismo de bajo coste, con el que se pueda modificar su contenido sin tener grandes conocimientos de informática.
- La ultima motivación, es referido al RDF, en la actualidad se usa poco el sistema RDF con fines comerciales, y la idea de realizar este proyecto es poder darle un fin comercial al RDF. Más adelante explicaré con algún ejemplo cual es el fin comercial de este CMS.

Por ultimo respecto a la motivaciones, tengo que explicar que la idea de este proyecto salió, de la necesidad de una asociación de bares, de dar a conocer a sus clientes las fiestas de fines de semana que se organizaban en los distintos bares, por lo que aparte de las motivaciones anteriores esta es una motivación extra, ya que como he dicho antes es intentar dar un fin comercial a el RDF.

4.2 Alcance del proyecto

Para llegar al objetivo que hemos propuesto para este proyecto, hace falta que se efectúen las siguientes fases de proyecto enumeradas a continuación:

4.2.1 Gestión del proyecto

Esta fase se llevará a cabo durante todo el proceso que dure el proyecto, verificando que se cumple la planificación estimada. Además de testear cada uno de los puntos en los que se divide este objetivo. También en esta fase se incluyen las diferentes reuniones sean online o físicas con el tutor para verificar que se van cumpliendo los objetivos. Además de los distintos intercambios de comunicación vía email con el tutor.

4.2.2 Análisis y obtención de requerimientos

Esta fase es la inicial de este proyecto, ya que los requerimientos son necesarios también para ver la viabilidad de este proyecto. Si este proyecto se determinara como no viable, no habría más fases de proyecto, puesto que se pararía. Esta fase comprende muchas subfases, sin embargo las fundaméntales se comprenden los siguientes puntos:

- El estudio del formato RDF/XML que se hace para la total compresión de sus estándares actuales, ya que es el formato que se usará en todo el proyecto para el guardado y la exportación del contenido de la base de datos. Como se puede ver por en nombre mismamente, necesitaremos dos estudios diferentes, el estudio de los tipos de formatos XML y el estudio de los tipo de formatos RDF. En este punto del proyecto la principal fuente de lectura será internet.
- Obtener vía internet la información necesaria, sobre el software LAMP (Linux + Apache + MySQL + PHP) o WAMP (Windows + Apache + MySQL + PHP), para tenerlo accesible. Debido a que se va a usar WAMP y LAMP como plataformas de desarrollo, habrá que tener las instalaciones de este software preparadas en ordenadores, para poder trabajar con ellas. Miraremos también lo relacionado a WAMP y directorios o hosting virtuales, porque a nivel local se trabajará con Windows, y para minimizar errores siempre hay que tratar no usar direcciones del tipo http://localhost.
- Estudio del nuevo formato de HTML denominado HTML5 y del nuevo formato de CSS llamado CSS3. Estos dos formatos presentan algunas diferencias importantes respecto a sus predecesores. Estas diferencias sobre todo van relacionadas con hacer una aplicación responsive para una plataforma web.

- Instalación del software necesario a nivel local para el desarrollo del proyecto. Esta instalación local comprende tanto los editores de texto e imágenes, como navegadores y servidor virtual local WAMP para la ejecución del proyecto. También incluye la búsqueda de un servidor web, en un registrante de internet, para la instalación del proyecto en modo definitivo. El software más básico necesario será una estructura WAMP y un editor de texto potente.
- Cuando ya se tenga tanto el servidor WAMP como el servidor LAMP, será necesario un sistema de comunicación entre estos dos servidores por medio de FTP y un software que puede gestionar estos FTP's. Además llegados a este punto será necesario planear un sistema de backups para asegurar el proyecto, ya sea a nivel de FTP o a nivel de nube.
- Como última subfase de esta fase, revisaremos que ya tendremos la información y el software necesario, tanto a nivel local como a nivel de servicio web, para las siguientes fases de diseño e implementación. Además viendo los requerimientos y el alcance del proyecto se decidirá si este proyecto es viable.

4.2.2 Diseño e implementación de la estructura de datos

Esta segunda fase es la que empezara a dar la primera visión sobre el tipo de datos y estructura a usar, además al transcurso de esta fase se ira decidiendo si se usa programación orientada a objetos para PHP o no. La tiene los siguientes puntos:

- Independientemente si se usa una programación orientada a objetos o no en PHP, hará falta un análisis de los objetos necesarios para el proyecto. Con estos objetos o clases se podrá realizar el posterior diseño e implementación del proyecto. Este análisis se hará un poco a grandes rasgos para ir viendo las diferentes necesidades de programación
- Realización de un esquema UML con las posibles clases del anterior análisis, para la posterior implementación de la estructura de datos. Dependiendo de los requisitos UML que se vayan "dibujando" se decidirá en que sistema se guardara los estos datos, porque en RDF hay varios sistemas. En este punto también se irá sacando el esquema Entidad/Relación, para utilizar esas clases para la posterior implementación en PHP. En esta fase, también iremos comparando si haremos un sistema procedural de PHP o por clases, esto nos lo dará si los procesos de PHP serían muy complicados, ya que normalmente en PHP la programación por clases es más fácil pero más lenta. Sin embargo PHP poco a poco está avanzando para ser un sistema más estructurado a la programación a objetos.

- Tras la elección del motor que guardara los datos del proyecto, se pasará a la implementación de la base de datos en ese motor. Aunque el servidor de datos es MySQL en un principio, este tipo de servidor tiene varios motores para guardar los datos. Por lo tanto, con el esquema Entidad/Relación que ha sido elaborado previamente, se realizara la implementación de la estructura que guardará los datos.
- Al término de esta fase, se tendrá una base de datos con lo necesario para poder almacenar el contenido. También obtendremos un esquema UML y otro de Entidad/Relación que usaremos en las siguientes fases.

4.2.3 Diseño e implementación de la interfaz de usuario

Esta tercera fase es la encargada de dar una interfaz que interactuara con los datos guardados en la base de datos anteriormente construida. Recordando que el diseño de esta interfaz tiene que ser limpio, para poder usar RDF sin tener conocimientos de este formato. Se divide en las siguientes subfases:

- Diseño gráfico de la interfaz, para poder realizar una visualización responsive. Todos estos diseños se realizaran mediante Adobe Photoshop y capturas de pantalla. Además se guardaran unos JPG para visualizarlos en los distintos navegadores.
- Estudio y si fuera necesario creación de algoritmos, para implementaciones en PHP distintas a lo que sería una simple traducción a un formulario web. Implementación en PHP la interfaz gráfica, para poder realizar operaciones con la base de datos.
- Creación de plantillas y datos con esta nueva interfaz para el testeo de la misma con distintos ejemplos. Además se pueden utilizar distintos usuarios beta testadores para comprobar la facilidad del uso de la interfaz
- Al acabar esta fase, se obtendrá una interfaz preparada para interactuar con la base de datos, que tendrá entre sus operaciones:
    - Introducir datos con plantillas implementadas en PHP. Estos datos se podrán introducir por un medio sencillo de formularios web, que serán guardados en la base de datos implementada. Estos datos los denominaremos instancias de los formularios.
    - Edición de los datos a través de las plantillas en PHP. La edición será la modificación de los datos previamente insertados en el punto anterior en la base de datos
    - Eliminación de contenido a través de plantillas en PHP. Sin embargo se estudiara cual es la mejor forma de eliminación puesto que RDF puede mantener datos aunque no tengan coherencia.

- Creación de plantillas para poder usarlos en los puntos anteriores. Estas plantillas a las que nos referimos constantemente son lo que denominaremos formularios, términos o propiedades globales, que a su vez generaran instancias
- Visualización de las plantillas existentes y sus instancias, tanto para poder listarlas como para poder editarlas o eliminarlas, según las opciones anteriores.
- Ayudas de los distintos complementos de la interfaz, aunque en un principio se va a intentar un diseño lo más simple posible, en el caso de necesitar alguna ayuda extra se pondrá. Además otro tipo de ayuda es por ejemplo hacer zoom o poner en ventanas apartes objetos para la mejor edición de los mismos. Estos zoom se suelen llamar ligth box.

4.2.4 Diseño e implementación de la interfaz de aplicaciones (API)

Esta cuarta fase será para poder crear un servicio web por el cual otra aplicación, ya sea web o móvil, pueda extraer de la base de datos contenido en RDF/XML. Este servicio se realizara mediante una llamada web a un procedimiento PHP. Esta subfase Se divide en las siguientes etapas:

- Estudio y realización de algoritmos si fueran necesarios, ya que a priori pueden ser muy simples, y no sería necesaria más que la implementación propia del PHP.
- Implementación en PHP la interfaz de aplicaciones, para poder realizar exportaciones desde la base de datos en formato RDF/XML.
- Estudio sobre si sería posible con condiciones simples, enviar también las exportaciones en JSON u otros formatos. Ya que llegados a este punto podría ser interesante poder enviar en otros formatos.
- Testeo de la interfaz de aplicaciones con distintos ejemplos. Al término de esta fase se obtendrá una API, que podrá exportar contenido de la base de datos a través del formato RDF/XML.

4.2.5 Diseño e implementación de un sistema de seguridad

Esta fase es la encargada de dar una interfaz de usuario y contraseña, para dotar de seguridad a todo el sistema, e impedir accesos no autorizados a la edición de la base de datos. Esta fase a su vez se divide en varias subfases:

- Integración de las tablas del esquema UML.
- Implementación en PHP de los formularios necesarios para agregar, editar, eliminar y acceso de usuarios. También de la generación de token para utilizarlos con la API.
- Testeo del sistema de seguridad, tanto en la interfaz de usuario como en la interfaz de aplicaciones.

4.2.6 Testeo general

Esta fase es la encargada de verificar que la integración de las distintas fases da como resultado el objetivo deseado. En caso de estar todo correcto se iría a la fase correspondiente para determinar el posible error y depurarlo. Debido a que se hacen testeos durante las fases previas, no debería de haber muchos fallos que depurar. Así mismo haremos que unos usuarios estándares de ofimática nos den una valoración de la facilidad de uso del proyecto.

4.2.7 Documentación

Aunque se creará documentación durante todo el proceso hasta conseguir el objetivo del proyecto, es en esta última fase donde se unificara toda la documentación. Así que en esta última fase se recopilará toda la documentación creada, y se integrará con la memoria del proyecto. Si se puede se intentará la creación de manuales online en formato PDF, si con las ayudas no fuera suficiente. Por lo tanto este punto será opcional, ya que se harán ayudas online para todas las interfaces.

4.3 Metodología de trabajo

Las fases que definiremos a continuación las hará el alumno Sergio Ibañez Fraile, en su ordenador de casa, con una media diaria de 5 horas de trabajo excepto si hubiera alguna contingencia descrita en los factores de riesgo. También se llevaran a cabo ciertas reuniones con el tutor, ya sea físicamente o vía online, aunque también podrá verse el avance del proyecto vía internet en el dominio cgcapp.sandbox.com.es

También se mantendrá un contacto vía email o incluso teléfono si fuera necesario, para concretar aspectos del proyecto. Las fases seguidas con esta metodología empezaran por el análisis de requerimientos, después se pasará a la obtención de los mismos, comprobando en todo momento que los mismos están disponibles.

Después se pasara al diseño y la implementación de la base de datos, para la posterior verificación de la misma. Una vez que los testeos de la misma son correctos, pasaremos al diseño e implantación de la interfaz de usuario, sino procederemos a la revisión de los errores para depurar la base de datos.

El siguiente paso que sería el diseño e implantación de la interfaz de usuario, haciéndose testeos al final de la implementación de la interfaz de usuario. Una vez verificados los testeos se pasará al siguiente paso que sería el diseño y la implementación de la interfaz de aplicaciones (API), en caso de no ser verificados se depurará la interfaz de usuario.

Se continuará con diseñando e implementado el API, que tras acabar la implementación darán paso a una serie de testeos con datos, tanto internos a nivel de red local preparados, como externos que en este caso serán datos públicos gubernamentales. En caso de estar todos correctos se pasara al siguiente paso, que será la producción del sistema de seguridad (Usuario/Contraseña/Token), sino pasaremos a la depuración de la API.

Tras la implementación del sistema de seguridad, haremos unos testeos para comprobar que todo está correcto en este sistema de seguridad. Una vez finalizados correctamente los testeos, realizaremos unos testeos globales para que veamos si toda la plataforma funciona correctamente.

Tras los últimos testeos globales pasaremos a la creación de los manuales y la documentación final de la memoria, para comprobar si lo planeado se equiparará a lo real. En todo momento tras la instalación de requerimientos, se irá subiendo a un posible dominio elegido, seguramente en este caso será el dominio cgcapp.sandbox.com.es, para la verificación online de la situación del proyecto. También decir que la documentación del proyecto se ira generando mientras se va avanzando en el proyecto, aunque en la última fase se recopilará toda la información para generar su memoria.

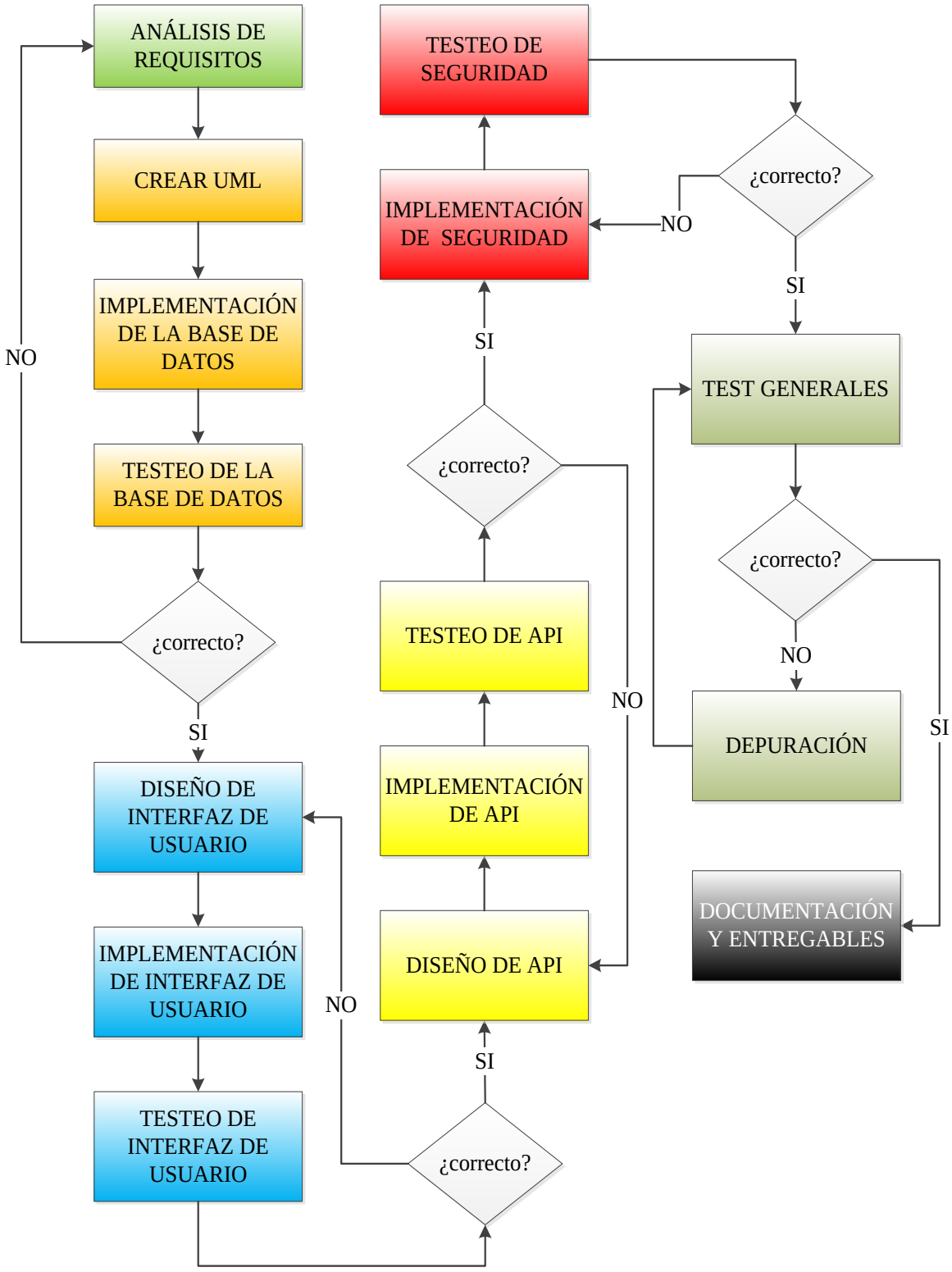


Figura 7 –Diagrama de la metodología de trabajo

Tras el análisis de requisitos comprobaremos que herramientas necesitaremos para la realización de este proyecto, sin embargo a priori debido a que el proyecto va referido a PHP, necesitaremos una maquina servidor donde pueda utilizarse PHP y MySQL. En concreto usaremos la última versión de servidor WAMP, que podamos descargar, para el desarrollo de la aplicación. Sin embargo, como la mayoría de las aplicaciones web, utilizaremos un servidor remoto para poner en producción la aplicación y ver si en internet funciona, este servidor normalmente es un servidor LAMP.

También usaremos un editor de texto avanzado para la edición del cogido PHP necesario. Y para la edición de imágenes y diseños usaremos Adobe Photoshop. Como además necesitaremos un sistema de comunicación entre el dominio de producción y nuestro servidor local, usaremos cuentas FTP para la comunicación. Así podemos grabar los archivos en el servidor remoto, para el posterior uso en producción.

Las herramientas para esta metodología de trabajo serán, a ser posible, con licencias GPL. Estas licencias hacen que las herramientas sean gratuitas, disminuyendo el coste del proyecto.

Por ultimo dentro de la metodología, cabe destacar que haremos diariamente backups tanto de la documentación como del código fuente del proyecto para no perder ningún dato. Además si utilizamos varios ordenadores con WAMP, debido a que por falta de tiempo quizás se necesite a largo plazo un portátil, usaremos un sistema de sincronía entre los dos WAMP's para que siempre este la última versión del proyecto en ambos ordenadores, un ejemplo es el Synthink como herramienta de sincronización.

4.4 Planificación del proyecto

4.4.1 Análisis y obtención de requerimientos

Como ya se ha dicho esta es la fase inicial en la que se intentara recopilar toda la información necesaria, para poder diseñar e implementar todo el proyecto.

Nombre de la fase	Día de inicio	Dia final	Horas
Estudio del formato RDF/XML	30-04-2015	05-05-2015	12 Horas
Obtener vía internet la información LAMP/WAMP	05-05-2015	05-05-2015	2 horas
Estudio del HTML5 y CSS3 (Responsive)	06-05-2015	07-05-2015	6 horas
Planear un sistema de backups	08-05-2015	08-05-2015	1 hora
Gestion de proyecto (fase 1): **Crear documentación** **Reuniones** **Emails** **Comprobar si se cumple el plan**	30-04-2015	10-05-2015	 3 horas 6 horas 1 hora 1 hora
			32 horas

4.4.2 Diseño e implementación de la base de datos

Segunda fase en la que se realizara la base de datos para almacenar todo el contenido, empezada desde 0.

Nombre de la fase	Día de inicio	Dia final	Horas
Análisis de los objetos	12-05-2015	16-05-2015	14 Horas
Realización de un esquema UML	12-05-2015	16-05-2015	10 horas
Implementación de la base de datos	16-05-2015	17-05-2015	6 horas
Testeo de la base de datos	17-05-2015	17-05-2015	1 hora
Gestión de proyecto (fase 2) : **Crear documentación** **Reuniones** **Emails** **Comprobar si se cumple el plan**	12-05-2015	17-05-2015	 3 horas 1 horas 1 hora 1 hora
			37 horas

4.4.3 Diseño e implementación de la interfaz de usuario

Tercera fase en la que se realizara la interfaz que un usuario podrá manejar para la modificación del contenido de la base de datos. También estará empezada desde 0 y con las pautas que tiene el diseño UML.

Nombre de la fase	Día de inicio	Dia final	Horas
Diseño gráfico del interfaz	19-05-2015	22-05-2015	14 Horas
Implementación en PHP	21-05-2015	29-05-2015	25 horas
Creación de plantillas en PHP	29-05-2015	30-05-2015	10 horas
Testeo de la interfaz	23-05-2015	30-05-2015	2 horas
Gestion de proyecto (fase 3) : **Crear documentación** **Reuniones** **Emails** **Comprobar si se cumple el plan**	19-05-2015	30-05-2015	 3 horas 1 horas 1 hora 1 hora
			57 horas

4.4.4 Diseño e implementación de la interfaz de aplicaciones (API)

Cuarta fase en la que se realizara la herramienta por la cual se podrá importar y exportar contenido en formato RDF/XML.

Nombre de la fase	Día de inicio	Dia final	Horas
Estudio y realización de algoritmos	02-06-2015	03-06-2015	10 Horas
Implementación en PHP	03-06-2015	04-06-2015	10 horas
Creación de la exportación JSON (*opcional*)	05-06-2015	05-06-2015	1 hora
Testeo de la API	05-06-2015	05-06-2015	2 horas
Gestión de proyecto (fase 4) : **Crear documentación** **Reuniones** **Emails** **Comprobar si se cumple el plan**	01-06-2015	06-06-2015	 3 horas 1 horas 1 hora 1 hora
			29 horas

4.4.5 Sistema de seguridad

La quinta fase en la dotará a la herramienta de un sistema de seguridad para proteger el contenido de la base de datos, por medio de usuario y contraseña.

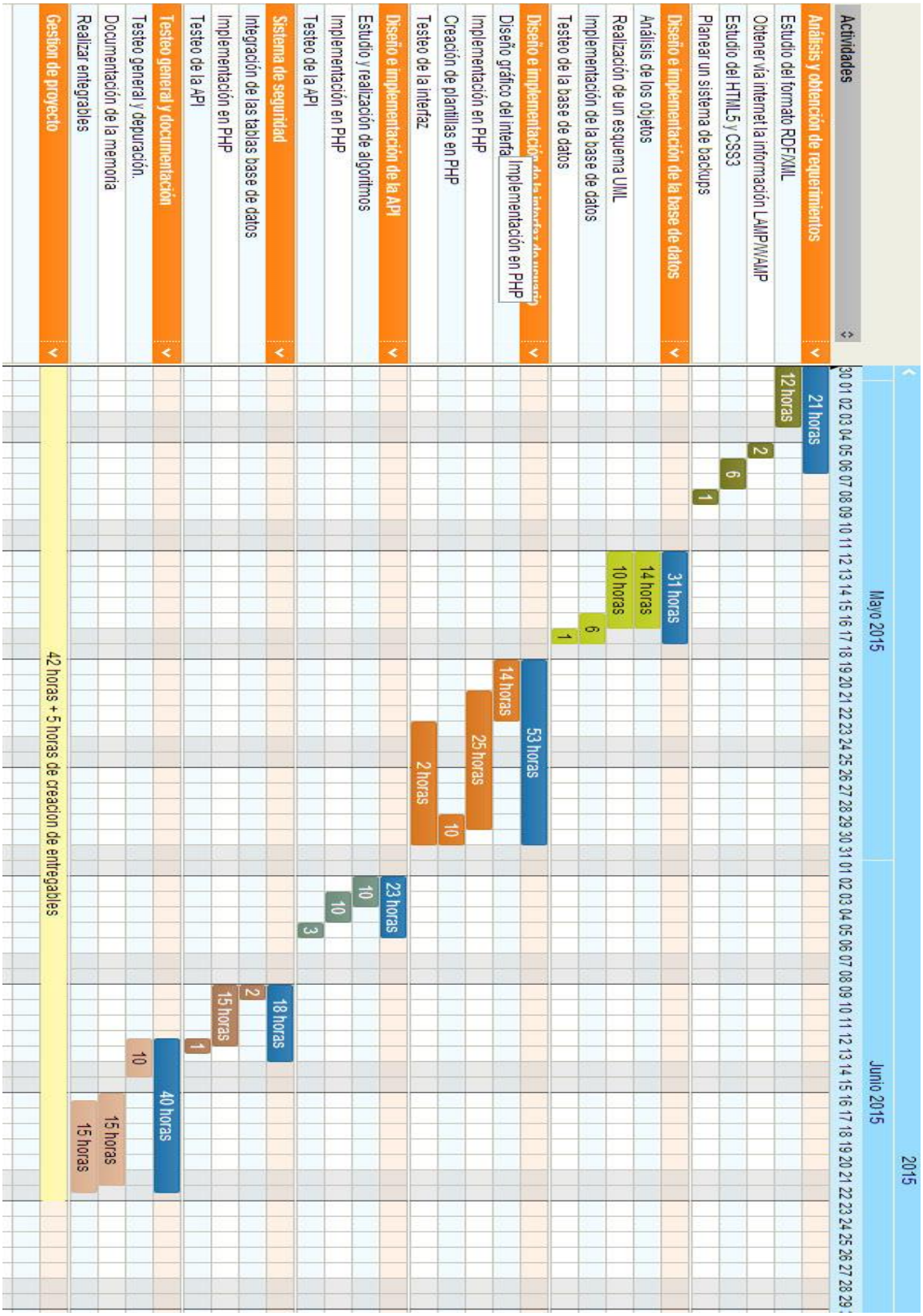
Nombre de la fase	Día de inicio	Dia final	Horas
Integración de las tablas base de datos	09-06-2015	09-06-2015	2 horas
Implementación en PHP	09-06-2015	12-06-2015	15 horas
Testeo de la API	12-06-2014	12-06-2014	1 hora
Gestión de proyecto (fase 5) : **Crear documentación** **Reuniones** **Emails** **Comprobar si se cumple el plan**	09-05-2014	12-06-2014	 3 horas 1 horas 1 hora 1 hora
			24 horas

4.4.5 Testeo general y documentación

La última fase será para poder verificar todas las demás partes del proyecto, y en caso de dar correcto los casos de uso, realizar la integración de la distinta documentación con la memoria.

Nombre de la fase	Día de inicio	Dia final	Horas
Testeo general y depuración.	12-06-2015	14-06-2015	10 Horas
Creación de manuales	16-06-2015	21-06-2015	15 horas
Documentación de la memoria	16-06-2015	21-06-2015	15 horas
Realizar entegrables	12-05-2015	17-05-2015	5 horas
			45 horas

4.5 Diagrama de Gannt



4.6 Factores de riesgo

Debido a la situación laboral, los días planificados podrían ser algo distintos, ya que una entrevista de trabajo o la posibilidad de un trabajo, podría hacer variar mis horas diarias dedicadas al proyecto, incluso no puede ser horas sino días. El plan de contingencia ante una entrevista de trabajo, seria básicamente trasladar todo el proyecto a un día más tarde, si fuera un cambio en la situación laboral, habría que ver en qué fase del proyecto nos quedaríamos, y preparar otra planificación sabiendo que horas se tienen disponibles

También tendremos en cuenta la posibilidad de una leve enfermedad que podría alterar las horas diarias dedicadas al proyecto. Es por ello que ya en la planificación se añadirá un cierto tiempo prudencial para estos menesteres. La enfermedad puede ser tanto del alumno de este proyecto, como del tutor por lo tanto hay que tener en cuenta ambas partes. El plan de contingencia ante un a enfermedad, seria básicamente apuntar el punto del proyecto donde nos quedaríamos y esperar a seguir cuando el alumno se reponga. Si fuera el tutor podríamos continuar con el proyecto, pero sin las reuniones o comunicaciones correspondientes

Otro factor de riesgo a tener en cuenta es la falta de internet, aunque hay partes del proyecto en las que no se necesita internet, también hay partes en las que son necesarias visitar páginas web. Sin embargo no hace falta preparar un plan de contingencia debido a que simplemente intercambiando fases del proyecto que no necesiten internet, con aquellas que lo necesiten debería bastar.

El último factor de riesgo a tener en cuenta seria la posibilidad que el hardware se estropeara. Es por ello que se planificaran distintos backups para poder recambiar el hardware y poder recuperarlo. El plan de contingencia es reinstalar el software y reponer todo le código necesario para seguir con el proyecto. Este plan de contingencia también es válido en el caso de que haya un problema de virus o se pierda el código. Además de la reinstalación se ajustara las fechas apropiadamente.

Además debido al corto plazo que se tiene hasta la presentación de la memoria, podría verse un poco más alargado y presentarse algo más tarde. Si esto ocurriera se debería replanificar el proyecto, sabiendo que horas hay disponibles.

5. RESULTADOS

En esta parte de la documentación, explicaremos las distintas partes de diseño e implementación de todo el proyecto. El diseño incluye tanto las distintas clases encontradas, puesto que vamos a hacer una programación orientada a objetos todo lo posible, hasta la interfaz gráfica. Incluyendo las fases por las que ha pasado la implementación del proyecto.

En la parte de diseño expondremos pondremos primeramente los casos de uso para el usuario clásico de internet, debido a que es una aplicación web. Los casos de uso se diferenciarán entre los distintos elementos de la aplicación, pasando de los más generales a los más particulares de cada elemento.

Después de los casos de uso se presentarán unos diseños de interfaz, que primeramente se hicieron en Adobe Photoshop, aunque en para la memoria se han añadido diferentes capturas de pantalla del explorador, para mayor ajuste con la realidad en el diseño. Aunque en esta fase de diseño se hizo también el diagrama UML, se presentara posteriormente en la zona de implementación

Tras la documentación de la parte de diseño, pasaremos a la documentación que concierne a la implementación, donde se presentarán varios puntos tal y como se listan a continuación:

- Tecnologías usadas para el proyecto: Aunque se habló de ellas en el análisis de requerimientos se expondrá cuales se usaron.
- Arquitectura de la aplicación, dando a conocer las diferentes capas del proyecto.
- Diagrama UML, donde se enumeraran las clases.
- Estructura de la base de datos, aunque como se expondrá se usa más para optimizar la velocidad que para el propio almacenamiento.
- Descripción de las clases presentadas antes en el diagrama UML.
- Diagrama de secuencia de los casos de uso más necesarios y comunes del proyecto.
- Breve descripción de las pruebas realizadas.

.Una de las cosas a tener en cuenta en el diagrama UML, es que las clases Users, Login, han sido reutilizadas de otro proyecto y EasyRDF ha sido descargada de internet, y no han sido construidas para este proyecto. Es por ello que no se detalla todas las funciones de esta clase, sino simplemente las que se usan.

5.1 DISEÑO (CASOS DE USO)

Aquí se pondrán todos los casos de uso que utilizará el usuario que se usan en el proyecto, estableciendo un orden como el siguiente:

- Casos de uso generales
- Casos de uso gestión de formularios
- Casos de uso gestión de términos
- Casos de uso gestión de propiedades
- Casos de uso gestión de instancias
- Casos de uso gestión de usuarios

Lo que hemos denominado tipos básicos de RDF, son los siguientes, debido que para el proyecto no nos hacen falta todos los dataypes de RDF: string, integer, nonNegativeInteger, nonPositiveInteger, decimal, boolean, date, time y datetime. Aunque en el proyecto los hemos denominado con lenguaje común.

5.1.1 Casos de uso generales

En cuanto se entra a internet y a la dirección web en la que se instalará la aplicación se abrirá el formulario de entrada al sistema, si es que no está el usuario logueado. Ya que si estuviera logueado no le haría falta entrar a este formulario. Lo que si sucede es que si el usuario no está logueado, y entra en cualquiera de las diferentes páginas, le volverá a este caso de uso. Todos los casos de uso se usarán en el back-end de la aplicación, excepto el login que se usa en front-end.

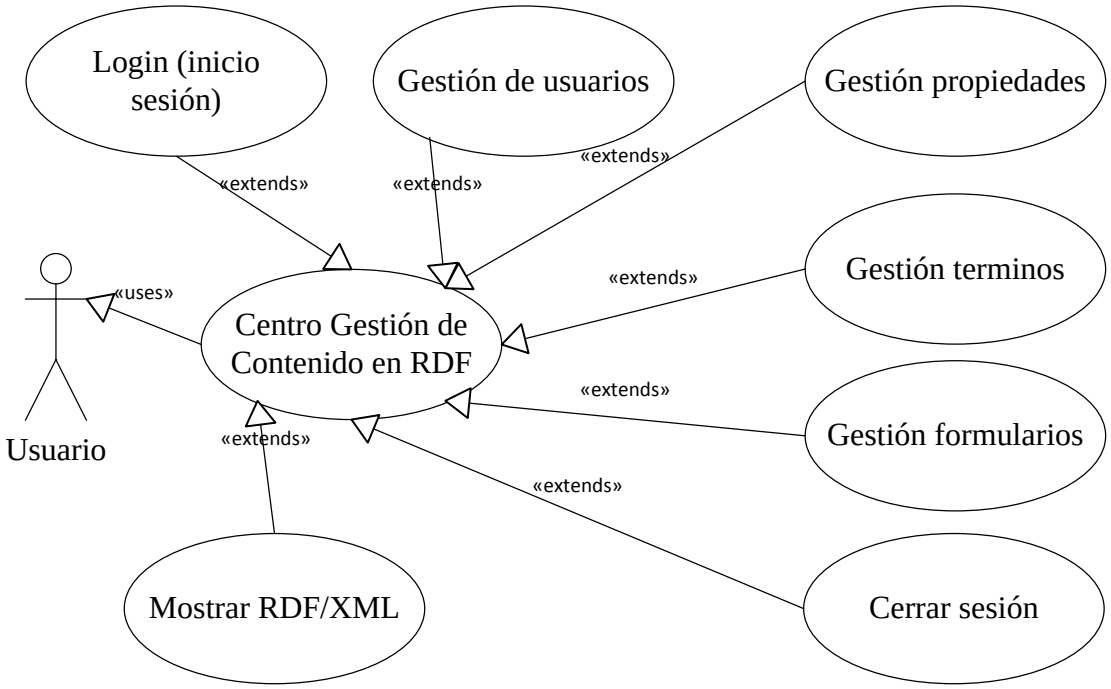


Figura 8 –Diagrama casos de uso generales

5.1.1.1 Login

El acceso al proyecto está restringido a usuarios que estén dados de alta en el sistema. Para poder acceder al sistema el usuario deberá autenticarse mediante su nombre de usuario y contraseña.

- Actores: Usuario del CGCRDF
- Precondiciones: El usuario no está autenticado en el sistema.
- Flujo de eventos:
    1. El usuario accede a la plataforma mediante un navegador web.
    2. La plataforma muestra un formulario web solicitando los datos de acceso.
    3. El usuario introduce los datos.
    4. Si los datos son correctos la plataforma concede el acceso en caso contrario nuestra un mensaje de error y vuelve al paso 2.
- Postcondiciones: El usuario se mantiene autenticado hasta que cierre la sesión o el navegador.

5.1.1.2 Cerrar sesión

En cualquier momento el usuario puede dar al botón *"Cerrar sesión"*, para poder cerrar la sesión dentro del proyecto.

- Actores: Usuario del CGCRDF
- Precondiciones: El usuario está autenticado en el sistema.
- Flujo de eventos:
    1. Al activar este evento, se pedirá al usuario la confirmación de la salida.
    2. El usuario confirma la salida.
    3. La plataforma cierra la sesión del usuario redireccionando a la interfaz de acceso.
- Postcondiciones: El usuario sale del sistema.

5.1.1.3 Gestión de formularios

En cualquier momento el usuario puede dar al botón *"Formularios"*, para poder acceder a la parte con la cual podrá: Añadir, editar y eliminar formularios entre otras cosas..

- Actores: Usuario del CGCRDF
- Precondiciones: El usuario está autenticado en el sistema.
- Flujo de eventos:
    1. Simplemente es un acceso a la parte de gestión de formularios.
- Postcondiciones: El usuario va a la zona de gestión de formularios.

5.1.1.4 Gestión de terminos

En cualquier momento el usuario puede dar al botón *"Términos"*, para poder acceder a la parte con la cual podrá: Añadir, editar y eliminar términos.

- Actores: Usuario del CGCRDF
- Precondiciones: El usuario está autenticado en el sistema.
- Flujo de eventos:
    1. Simplemente es un acceso a la parte de gestión de términos.
- Postcondiciones: El usuario va a la zona de gestión de términos.

5.1.1.5 Gestión de propiedades

En cualquier momento el usuario puede dar al botón *"Propiedades"*, para poder acceder a la parte con la cual podrá: Añadir, editar y eliminar propiedades.

- Actores: Usuario del CGCRDF
- Precondiciones: El usuario está autenticado en el sistema.
- Flujo de eventos:
    1. Simplemente es un acceso a la parte de gestión de propiedades.
- Postcondiciones: El usuario va a la zona de gestión de propiedades.

5.1.1.6 Gestión de usuarios

En cualquier momento el usuario puede dar al botón *"Usuarios"*, para poder acceder a la parte con la cual podrá: Añadir, editar y eliminar usuarios entre otras cosas.

- Actores: Usuario del CGCRDF
- Precondiciones: El usuario está autenticado en el sistema y es administrador.
- Flujo de eventos:
    1. Simplemente es un acceso a la parte de gestión de usuarios.
- Postcondiciones: El usuario va a la zona de gestión de usuarios.

5.1.1.7 Mostrar RDF/XML

En cualquier menú de listados el usuario tendrá la opción de ver el RDF/XML de los tipos. Seleccionándolo y pulsando el enlace. *"Mostrar el RDF/XML"*

- Actores: Usuario del CGCRDF
- Precondiciones: El usuario está autenticado en el sistema. Y estar en el menú Formularios, Términos, Propiedades o en el listado de instancias.
- Flujo de eventos:
    1. Al pulsar el enlace se muestra el RDF/XML del tipo seleccionado.
- Postcondiciones: Hace un display del RDF/XML del tipo seleccionado.

5.1.2 Gestión de formularios

Este evento de la aplicación se lanza cuando, después de haber logeado uno entra en la opción formularios dentro del menú derecho de navegación. También si estando logeado se pone en la barra de direcciones, http://HOST/ modules/plantilla/formularios/formularios.php, donde HOST es la dirección web donde se ha instalado la aplicación.

Mostrando una lista de los formularios que están creados por la aplicación. Además de mostrar el botón de añadir un nuevo formulario,. Las diferentes opciones que tiene cada formulario se pueden realizar de dos opciones diferentes, seleccionando el formulario, que hace que muestre las opciones en forma de botón debajo de la lista, o mediante los enlaces que salen al lado de cada formulario.

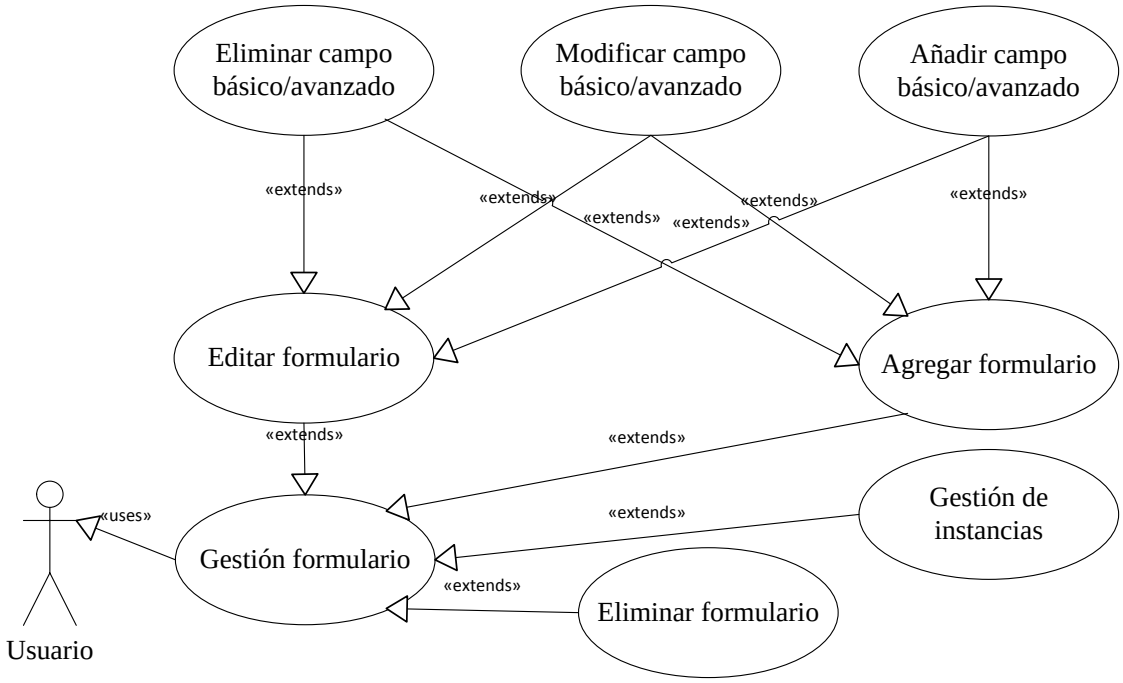


Figura 9 –Diagrama casos de uso gestión formularios

5.1.2.1 Agregar formulario

El evento se lanza pulsando el botón de *"Añadir nuevo formulario"*, dentro de las opciones que salen dentro de la página listado de formularios.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Formularios
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá rellenar con diferentes campos
  - Obligatoriamente el usuario debe rellenar el campo Nombre
  - Opcionalmente se puede rellenar un comentario en el formulario
  - Opcionalmente puede pulsar el botón añadir *"Añadir nuevo campo básico"*. Y entonces se disparara el evento añadir campo básico.
  - Opcionalmente puede pulsar el botón añadir *"Añadir nuevo campo avanzado"*. Y entonces se disparara el evento añadir campo avanzado.
  - Al final para guardar hay que dar al botón crear formulario.
- Postcondiciones: Se guarda en la base de datos y se en formato RDF.

5.1.2.2 Editar formulario

El evento se lanza pulsando el botón de *"Modificar formulario seleccionado"* cuando hay un formulario seleccionado. Hay una segunda forma de hacerlo pulsando el enlace con el icono de un lápiz al lado del formulario que se quiere editar, dentro de la página listado de formularios de búsqueda generales

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Formularios
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá modificar con diferentes campos, estos campos están rellenados con los valores guardados del formulario. Además saldrán los campos que posee el formulario, estos se podrán agregar, modificar o eliminar con las opciones de caso de uso.
  - Opcionalmente el usuario puede cambiar el campo Nombre o comentario.
  - Opcionalmente puede pulsar el botón *"Añadir nuevo campo básico"*. Y entonces se disparara el evento añadir campo básico.
  - Opcionalmente puede pulsar el botón añadir *"Añadir nuevo campo avanzado"*. Y entonces se disparara el evento añadir campo avanzado.
  - Opcionalmente puede pulsar el botón en forma de lápiz encima de cada campo, y entonces se disparara el evento modificar campo básico.
  - Opcionalmente puede pulsar el botón en forma de cubo de la basura encima de cada campo, y se disparara el evento eliminar campo básico.
  - Al final para guardar hay que dar al botón actualizar formulario.
- Postcondiciones: Se guarda en la base de datos y se en formato RDF

5.1.2.3 Eliminar formulario

El evento se lanza pulsando el botón de *"Eliminar formulario seleccionado"* cuando hay un formulario seleccionado. Hay una segunda forma de hacerlo pulsando el enlace con el icono de un cubo de basura al lado del formulario que se quiere eliminar, dentro de la página listado de formularios de búsqueda generales

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Formularios
- Flujo de eventos:
  - Cuando se dispara este caso, se le pide al usuario que confirme si quiere eliminar este formulario. Si se acepta se sigue el flujo sino se aborta.
  - Borra de la base el formulario. También se borran los subformularios.
- Postcondiciones: Formulario eliminado de la base de datos, NO BORRA LAS INSTANCIAS

5.1.2.4 Añadir campo básico

El evento se lanza pulsando el botón *"Añadir nuevo campo básico"*, que aparece dentro de la creación o edición de un término o subtérmino. Añade al formulario un campo de un tipo RDF básico, estos campos después podrá darse valor en la creación de instancias. Los campos de un término los heredan sus subtérminos

- Actores: Usuario del CGCRDF
- Precondiciones: Estar logueado y en la creación o edición de un formulario
- Flujo de eventos:
  - Cuando se dispara el evento sale un formulario con 3 campos:
    - Nombre del campo: Obligatoriamente hay que rellenarlo con un texto.
    - Tipo del campo: Se selecciona un tipo básico en una lista desplegable
    - Breve comentario: opcionalmente se puede poner un texto aqui
  - El usuario pulsa el botón *"Añadir campo"*.
  - Si el usuario pulsa *"Cancelar"* se vuelve al gestión de formularios.
- Postcondiciones: Se inserta 1 campo en la lista de campos del término.

5.1.2.5 Modificar campo básico

El evento se lanza pulsando el enlace con forma de lápiz, que hay en la parte superior de cada campo del formulario, pudiendo modificar el actor cualquiera de los 3 valores del campo.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar logueado, en la creación o edición de un término, tener al menos un campo el formulario en cuestión y dar en básico en la segunda alerta.
- Flujo de eventos:
  - Cuando se dispara el evento sale salen dos alertas:
    - La primera alerta es para la confirmación de la modificación
    - La segunda alerta es para saber si la modificación es para hacer un campo básico o avanzado, además igual que en la alerta anterior se puede cancelar.
  - Después de la segunda alerta sale un formulario con 3 campos:
    - Nombre del campo: Obligatoriamente hay que rellenarlo con un texto.
    - Tipo del campo: Se selecciona un tipo básico en una lista desplegable
    - Breve comentario: opcionalmente se puede poner un texto aquí
  - En los campos inicialmente estarán los valores guardados por el caso Añadir campo básico o editado por este caso.
  - El usuario pulsa el botón *"Actualizar campo"*.
  - Si el usuario pulsa *"Cancelar"* se vuelve al gestión de formularios.
- Postcondiciones: Se modifica 1 campo en la lista de campos del término.

5.1.2.6 Eliminar campo básico

El evento se lanza pulsando el enlace con forma de cubo de basura, que hay en la parte superior de cada campo del formulario, haciendo que el actor pueda eliminar un campo de un término.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar logueado, en la creación o edición de un término y tener al menos un campo el formulario en cuestión
- Flujo de eventos:
  - Cuando se dispara el evento sale un mensaje de alerta diciendo si esta seguro de la eliminación del campo.
  - En caso de confirmar la eliminación se quita de la lista de campos el campo seleccionado.
- Postcondiciones: Elimina el campo seleccionado de la lista de campos del término.

5.1.2.7 Añadir campo avanzado

El evento se lanza pulsando el botón *"Añadir nuevo campo externo"*, que aparece dentro de la creación o edición de un formulario. Añade al formulario un campo de un tipo , estos campos después podrá darse valor en la creación de instancias.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar logueado y en la creación o edición de un formulario
- Flujo de eventos:
  - Cuando se dispara el evento sale un formulario con 3 campos:
    - Nombre del campo: Obligatoriamente hay que rellenarlo con un texto.
    - Tipo de campo: Se selecciona con 3 pestañas y una lista desplegable.
    - Breve comentario: opcionalmente se puede poner un texto aquí
  - El usuario selecciona una de las 3 pestañas encima del campo tipo de datos que representan los diferentes tipos que hay en el sistema, que he pasado a denominar campos externos, siendo 3 tipos:
    - Formularios: Que representan las diferentes instancias de los diferentes formularios con un lista desplegable.
    - Términos: Que representan los términos que se han metido en el sistema con una lista desplegable. En la pestaña términos se pueden seleccionar subtérminos de un término de la lista despegable, con los botones de subtérminos o termino padre.
    - Propiedades, que representas las propiedades que se me han metido en el sistema.
  - El usuario pulsa el botón *"Añadir campo"*.
  - Si el usuario pulsa *"Cancelar"* se vuelve al gestión de formularios.
- Postcondiciones: Se inserta 1 campo en la lista de campos del formulario.

5.1.2.8 Modificar campo avanzado

El evento se lanza pulsando el enlace con forma de lápiz, que hay en la parte superior de cada campo del término, pudiendo modificar el actor cualquiera de los 3 valores del campo.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar logueado, en la creación o edición de un término, tener al menos un campo el formulario en cuestión y dar en avanzado en la segunda alerta.
- Flujo de eventos:
  - Cuando se dispara el evento sale salen dos alertas:
    - La primera alerta es para la confirmación de la modificación
    - La segunda alerta es para saber si la modificación es para hacer un campo básico o avanzado, además igual que en la alerta anterior se puede cancelar.
  - Después de la segunda alerta sale un formulario con 3 campos:
    - Nombre del campo: Obligatoriamente hay que rellenarlo con un texto.
    - Tipo de campo: Se selecciona con 3 pestañas y una lista desplegable.
    - Breve comentario: opcionalmente se puede poner un texto aquí
  - Los campos nombre del campo y comentario se rellenan con los datos que están guardados en el formulario en cuestión, sin embargo el tipo no se carga porque puede ser que se quiera modificar a un campo básico o avanzado.
  - El usuario selecciona una de las 3 pestañas encima del campo tipo de datos que representan los diferentes tipos que hay en el sistema, que he pasado a denominar campos externos, siendo 3 tipos:
    - Formularios: Que representan las diferentes instancias de los diferentes formularios con un lista desplegable.
    - Términos: Que representan los términos que se han metido en el sistema con una lista desplegable. En la pestaña términos se pueden seleccionar subtérminos de un término de la lista despegable, con los botones de subtérminos o termino padre.
    - Propiedades, que representas las propiedades que se me han metido en el sistema.
  - El usuario pulsa el botón *"Actualizar campo"*.
- Postcondiciones: Se modifica 1 campo en la lista de campos del término.

5.1.2.9 Eliminar campo avanzado

El evento se lanza pulsando el enlace con forma de cubo de basura, que hay en la parte superior de cada campo del término, haciendo que el actor pueda eliminar un campo de un término.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar logueado, en la creación o edición de un formulario y tener al menos un campo el formulario en cuestión
- Flujo de eventos:
  - Cuando se dispara el evento sale un mensaje de alerta diciendo si esta seguro de la eliminación del campo.
  - En caso de confirmar la eliminación se quita de la lista de campos el campo seleccionado.
- Postcondiciones: Elimina el campo seleccionado de la lista de campos del término.

5.1.2.10 Gestión de instancias

El evento se lanza pulsando el botón de *"Ver Intancias del formulario"* cuando hay un formulario seleccionado. También el evento se lanza pulsando el enlace con forma de ojo, que hay al lado derecho de las uris de los formularios.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar logueado, en la gestión de formularios.
- Flujo de eventos:
  - Comprueba si el formulario tiene instancias.
  - Devuelve una lista con las instancias del formulario en cuestión.
- Postcondiciones: Devuelve una lista de instancias.

5.1.3 Gestión de términos

Este evento de la aplicación se lanza cuando, después de haber logeado uno entra en la opción términos dentro del menú derecho de navegación. También si estando logeado se pone en la barra de direcciones, http://HOST/ modules/plantilla/terminos/terminos.php, donde HOST es la dirección web donde se ha instalado la aplicación.

Mostrando una lista de los términos que están creados por la aplicación. Además de mostrar el botón de añadir un nuevo termino, mostrando un enlace con forma de "+" si posee subtérminos. Las diferentes opciones que tiene cada término se pueden realizar de dos opciones diferentes, seleccionando el término, que entonces le salen las opciones en forma de botón debajo de la lista, o mediante los enlaces que salen al lado de los términos.

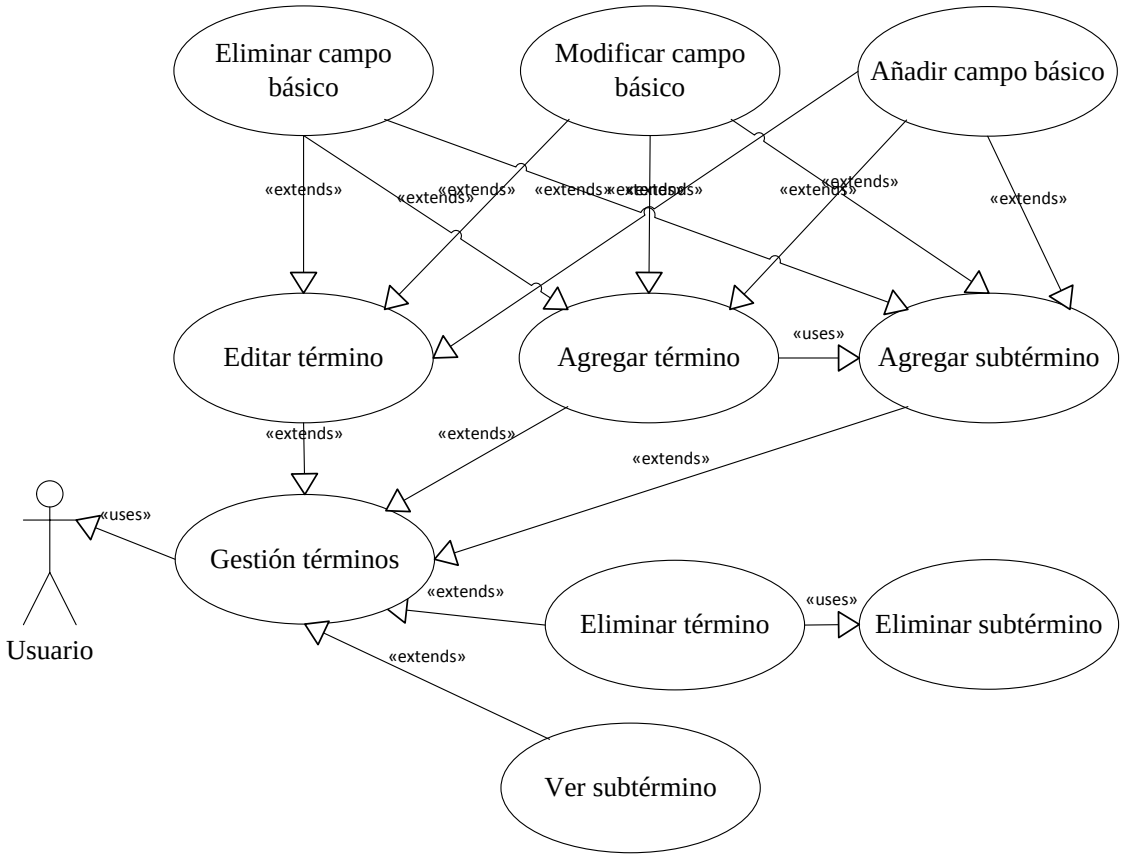


Figura 10 –Diagrama casos de uso gestión términos

5.1.3.1 Agregar termino

El evento se lanza pulsando el botón de *"Añadir nuevo término"* , dentro de las opciones que salen dentro de la página listado de términos de búsqueda generales

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Términos
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá rellenar con diferentes campos
  - Obligatoriamente el usuario debe rellenar el campo Nombre
  - Opcionalmente se puede rellenar un comentario en el termino
  - Opcionalmente puede pulsar el botón añadir *"Añadir nuevo campo básico"*. Y entonces se disparara el evento añadir campo básico.
  - Al final para guardar hay que dar al botón crear término.
- Postcondiciones: Se guarda en la base de datos y se en formato RDF.

5.1.3.2 Editar termino

El evento se lanza pulsando el botón de *"Modificar término seleccionado"* cuando hay un término seleccionado. Hay una segunda forma de hacerlo pulsando el enlace con el icono de un lápiz al lado del término que se quiere editar, dentro de la página listado de términos de búsqueda generales

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Términos
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá modificar con diferentes campos, estos campos están rellenados con los valores guardados del término. Además saldrán los campos que posee el término, estos se podrán agregar, modificar o eliminar con las opciones de caso de uso.
  - Opcionalmente el usuario puede cambiar el campo Nombre o comentario.
  - Opcionalmente puede pulsar el botón *"Añadir nuevo campo básico"*. Y entonces se disparara el evento añadir campo básico.
  - Opcionalmente puede pulsar el botón en forma de lápiz encima de cada campo, y entonces se disparara el evento modificar campo básico.
  - Opcionalmente puede pulsar el botón en forma de cubo de la basura encima de cada campo, y entonces se disparara el evento eliminar campo básico.
  - Opcionalmente puede pulsar el botón añadir *"Añadir nuevo campo básico"*. Y entonces se disparara el evento añadir campo básico. Al final debe guardar el botón crear botón agregar término
  - Al final para guardar hay que dar al botón actualizar término.
- Postcondiciones: Se guarda en la base de datos y se en formato RDF

5.1.3.3 Eliminar termino

El evento se lanza pulsando el botón de *"Eliminar término seleccionado"* cuando hay un término seleccionado. Hay una segunda forma de hacerlo pulsando el enlace con el icono de un cubo de basura al lado del término que se quiere eliminar, dentro de la página listado de términos de búsqueda generales

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Términos
- Flujo de eventos:
  - Cuando se dispara este caso, se le pide al usuario que confirme si quiere eliminar este término. Si se acepta se sigue el flujo sino se aborta.
  - Borra de la base el término. También se borran los subtérminos.
- Postcondiciones: Termino y subtérminos eliminados de la base de datos

5.1.3.4 Ver subtérminos

El evento se lanza pulsando el botón con forma de "+", que aparece al lado de términos que contiene subtérminos, y despliega una sub lista con los subtérminos. Este caso de uso sirve también para ocultar los subtérminos si se pulsa cuando esta con la forma "-" y la lista desplegada.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Términos
- Flujo de eventos:
  - Cuando se dispara el caso trae de la base los subtérminos que contiene.
  - Crea una lista y la despliega colocando botones encada termino que tiene subtérminos
  - En el caso de estar con forma "-" la oculta.
- Postcondiciones: Termino y subtérminos eliminados de la base de datos

5.1.3.5 Agregar subtérmino

El evento se lanza pulsando el botón de *"Añadir subtérmino al seleccionado"* , dentro de las opciones que salen dentro de la página listado de términos de búsqueda generales. Tambien pulsando el "+" cercano al término al que se quiere agregar.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Términos
- Flujo de eventos:
  - Es el mismo flujo que Agregar termino con la excepción que hay un parámetro más que indica quien es el término padre.
  - Al final para guardar hay que dar al botón crear término.
- Postcondiciones: Se guarda en la base de datos y se en formato RDF.

5.1.3.6 Añadir campo básico

El evento se lanza pulsando el botón *"Añadir nuevo campo básico"*, que aparece dentro de la creación o edición de un término o subtérmino. Añade al termino un campo de un tipo RDF básico, estos campos después podrá darse valor en la creación de instancias. Los campos de un término los heredan sus subtérminos

- Actores: Usuario del CGCRDF
- Precondiciones: Estar logueado y en la creación o edición de un termino
- Flujo de eventos:
  - Cuando se dispara el evento sale un formulario con 3 campos:
    - Nombre del campo: Obligatoriamente hay que rellenarlo con un texto.
    - Tipo del campo: Se selecciona un tipo básico en una lista desplegable
    - Breve comentario: opcionalmente se puede poner un texto aqui
  - El usuario pulsa el botón *"Añadir campo"*.
- Postcondiciones: Se inserta 1 campo en la lista de campos del término.

5.1.3.7 Modificar campo básico

El evento se lanza pulsando el enlace con forma de lápiz, que hay en la parte superior de cada campo del término, pudiendo modificar el actor cualquiera de los 3 valores del campo.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar logueado, en la creación o edición de un término y tener al menos un campo el termino en cuestión
- Flujo de eventos:
  - Cuando se dispara el evento sale un formulario con 3 campos:
    - Nombre del campo: Obligatoriamente hay que rellenarlo con un texto.
    - Tipo del campo: Se selecciona un tipo básico en una lista desplegable
    - Breve comentario: opcionalmente se puede poner un texto aquí
  - En los campos inicialmente estarán los valores guardados por el caso Añadir campo básico o editado por este caso.
  - El usuario pulsa el botón *"Actualizar campo"*.
- Postcondiciones: Se modifica 1 campo en la lista de campos del término.

5.1.3.8 Eliminar campo básico

El evento se lanza pulsando el enlace con forma de cubo de basura, que hay en la parte superior de cada campo del término, haciendo que el actor pueda eliminar un campo de un término.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar logueado, en la creación o edición de un término y tener al menos un campo el termino en cuestión
- Flujo de eventos:
  - Cuando se dispara el evento sale un mensaje de alerta diciendo si esta seguro de la eliminación del campo.
  - En caso de confirmar la eliminación se quita de la lista de campos el campo seleccionado.
- Postcondiciones: Elimina el campo seleccionado de la lista de campos del término.

5.1.4 Gestión de propiedades

Este evento de la aplicación se lanza cuando, después de haber logeado un usuario entra en la opción términos dentro del menú derecho de navegación. También si estando logeado se pone en la barra de direcciones la siguiente dirección, http://HOST/modules/plantilla/propiedades/propiedades.php, donde HOST es la dirección web donde se ha instalado la aplicación.

Mostrando una lista de los términos que están creados por la aplicación. Además de mostrar el botón de añadir un nueva propiedad, con enlaces al lado de cada propiedad. Las diferentes opciones que tiene cada propiedad se pueden realizar de dos opciones diferentes, seleccionando la propiedad, que entonces le salen las opciones en forma de botón debajo de la lista, o mediante los enlaces que salen al lado de las propiedades.

Las propiedades, en sí mismo son muy parecidas a los campos de los formularios y los términos, salvo que no están asignados a ningún formulario o término. Es por ello que hemos denominados a estas propiedades globales, ya que son accesibles desde la creación y edición de formularios. Esta explicado en el caso de uso de campos avanzados.

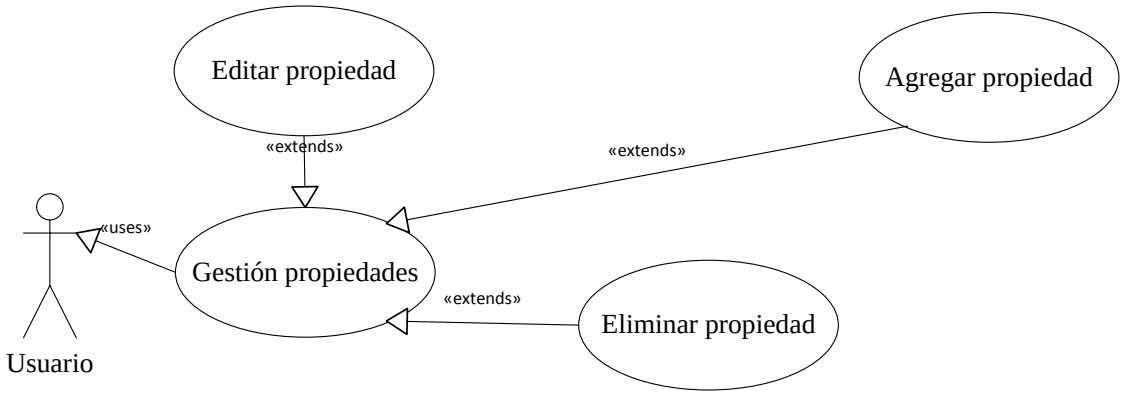


Figura 11 –Diagrama casos de uso gestión propiedades

5.1.4.1 Agregar propiedad

El evento se lanza pulsando el botón de *"Añadir nueva propiedad"*, dentro de las opciones que salen dentro de la página listado de propiedades, denominadas globales.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Propiedades
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá rellenar con diferentes campos
  - Obligatoriamente el usuario debe rellenar el campo Nombre, y se debe de elegir un tipo básico de RDF de una lista desplegable.
  - Opcionalmente se puede rellenar un comentario en la propiedad.
  - Al final para guardar hay que dar al botón crear propiedad.
- Postcondiciones: Se guarda en la base de datos y se en formato RDF.

5.1.4.2 Editar propiedad

El evento se lanza pulsando el botón de *"Modificar propiedad seleccionado"* cuando hay una propiedad seleccionada. También puede hacerse pulsando el enlace con el icono de un lápiz al lado de la propiedad, dentro del listado de propiedades.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Propiedades
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá modificar con diferentes campos, que están rellenados con los valores guardados de la propiedad.
  - Opcionalmente el usuario puede cambiar el campo Nombre o comentario.
  - Opcionalmente el usuario puede cambiar el campo tipo de datos.
  - Al final para guardar hay que dar al botón actualizar propiedad.
- Postcondiciones: Se guarda en la base de datos y se en formato RDF

5.1.4.3 Eliminar propiedad

El evento se lanza pulsando el botón de *"Eliminar término seleccionado"* cuando hay una propiedad seleccionada. Hay una segunda forma de hacerlo pulsando el enlace con el icono de un cubo de basura al lado de la propiedad

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú Propiedades
- Flujo de eventos:
  - Cuando se dispara este caso, se le pide al usuario que confirme si quiere eliminar esta propiedad. Si se acepta se sigue el flujo sino se aborta.
  - Borra de la base la propiedad.
- Postcondiciones: Propiedad eliminada de la base de datos

5.1.5 Gestión de instancias

Este evento de la aplicación se lanza cuando, después de haber logeado un usuario entra en la opción formularios dentro del menú derecho de navegación y después pulsar ver instancias. También si estando logeado se pone en la barra de direcciones la siguiente dirección, http:// HOST /modules/plantilla/instancias/instancias.php?uri=URI-FORM, donde HOST es la dirección web donde se ha instalado la aplicación y URI-FORM la uri codificada del formulario del que queremos ver las instancias

Mostrando una lista de las instancias que están creados en la aplicación relacionadas con el formulario. Además de mostrar el botón de añadir un nueva instancia, con enlaces al lado de cada URI de las instancias. Las diferentes opciones que tiene cada instancia se pueden realizar de dos opciones diferentes, seleccionando la instancia, que entonces le salen las opciones en forma de botón debajo de la lista, o mediante los enlaces que salen al lado de las instancias.

Las instancias, son verdaderamente los datos que usaran a posteriori las aplicaciones o usuarios externos al proyecto, ya que los formularios los usaremos como esqueleto para estas instancias.

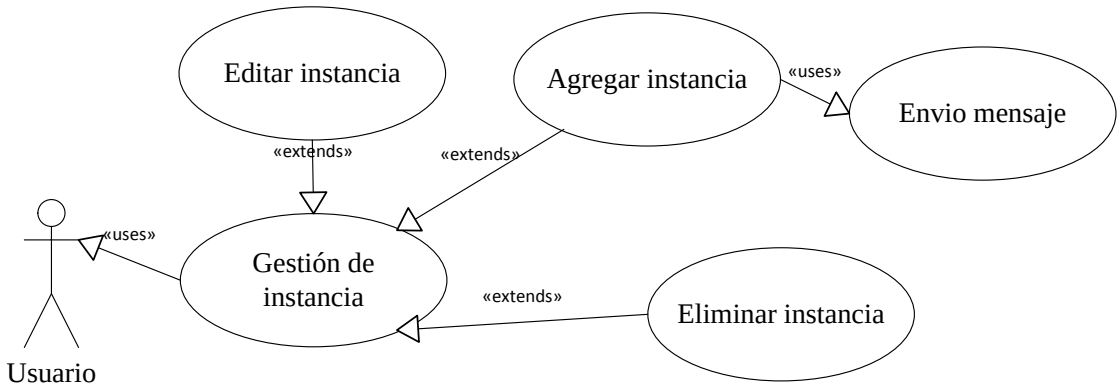


Figura 11 –Diagrama casos de uso gestión instancias

5.1.5.1 Agregar instancia

El evento se lanza pulsando el botón de *"Añadir una nueva instancia"*, dentro de las opciones que salen dentro de la página listado de instancias relacionadas con un formulario. Estas instancias son los valores que se dan a los esquemas que se han hecho con los casos de uso de los formularios.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y en el menú formularios y dentro del caso de uso gestión de instancias.
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá rellenar con diferentes campos
  - Obligatoriamente el usuario debe rellenar el campo Nombre.
  - Opcionalmente se puede rellenar un comentario en la propiedad.
  - Rellenar los siguientes campos de datos es opcional, puesto que pueden rellenarse en una posterior edición o quedarse vacío. Pero los campos siguientes se adecuaran a los datos que van a meterse de esta forma:
    - String: Saldrá un cuadro de texto en el cual se podrá meter cualquier cadena de caracteres que el usuario necesite.
    - Integer: Saldrá un cuadro de texto en el cual se podrá meter cualquier cadena de caracteres con la expresión regular $[0\text{-}9]^{*}$.
    - nonNegativeInteger: Saldrá un cuadro de texto en el cual se podrá meter cualquier cadena de caracteres con la expresión regular correspondiente a $[-]^{?}[0\text{-}9]^{*}$, y que el valor sea mayor o igual a 0.
    - nonPositiveInteger: Saldrá un cuadro de texto en el cual se podrá meter cualquier cadena de caracteres con la expresión regular $[0\text{-}9]^{*}$, y que el valor sea menor o igual a 0.
    - decimal: Saldrá un cuadro de texto en el cual se podrá meter cualquier cadena de caracteres con la expresión regular correspondiente a $[-]^{?}[0\text{-}9]^{*}[.]^{?}[0\text{-}9]^{*}$, y que el valor sea menor o igual a 0.
    - Boolean: Saldrá una lista desplegable con sí o no o vacio.
    - Date: Saldra un cuadro de texto con la máscara de esta expresión regular $[0\text{-}9]^{+}$ {2} [/]$[0\text{-}9]^{+}${2} [/]$[0\text{-}9]^{+}$ $[0\text{-}9]^{+}$ {4}
    - Time: Saldrá un cuadro de texto con la máscara de esta expresión regular $[0\text{-}9]^{+}$ {2} [:]$[0\text{-}9]^{+}${2
    - Datetime: Saldrá un cuadro de texto con la mezcla de tipo data más el tipo time
  - Al final para guardar hay que dar al botón crear instancia.
- Postcondiciones: Se guarda instancia en la base de datos y en formato RDF.

5.1.5.2 Editar instancia

El evento se lanza pulsando el botón de *"Modificar la instancia seleccionada"*, dentro de las opciones que salen dentro de la página listado de instancias relacionadas con un formulario. Estas instancias son los valores que se dan a los esquemas que se han hecho con los casos de uso de los formularios. Puede usarse el enlace en forma de lápiz.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado, en el menú formularios y dentro del caso de uso gestión de instancias.
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá editar con diferentes campos, los campos están rellenados con los datos guardados en la instancia.
  - Opcionalmente el usuario puede editar el campo Nombre.
  - Opcionalmente se puede editar un comentario en la propiedad.
  - Editar los siguientes campos de datos es opcional, puesto que pueden rellenarse en una posterior edición o quedarse vacío. Pero los campos siguientes se adecuaran a los datos que van a meterse de esta forma:
    - String: Saldrá un cuadro de texto en el cual se podrá meter cualquier cadena de caracteres que el usuario necesite.
    - Integer: Saldrá un cuadro de texto en el cual se podrá meter cualquier cadena de caracteres con la expresión regular $[0\text{-}9]^{*}$.
    - nonNegativeInteger: Saldrá un cuadro de texto en el cual se podrá meter cualquier cadena de caracteres con la expresión regular correspondiente a $[-]^{?}[0\text{-}9]^{*}$, y que el valor sea mayor o igual a 0.
    - nonPositiveInteger: Saldrá un cuadro de texto en el cual se podrá meter cualquier cadena de caracteres con la expresión regular $[0\text{-}9]^{*}$, y que el valor sea menor o igual a 0.
    - decimal: Saldrá un cuadro de texto en el cual se podrá meter cualquier cadena de caracteres con la expresión regular correspondiente a $[-]^{?}[0\text{-}9]^{*}[.]^{?}[0\text{-}9]^{*}$, y que el valor sea menor o igual a 0.
    - Boolean: Saldrá una lista desplegable con sí o no o vacio.
    - Date: Saldra un cuadro de texto con la máscara de esta expresión regular $[0\text{-}9]^{+}$ {2} [/]$[0\text{-}9]^{+}${2} [/]$[0\text{-}9]^{+}$ $[0\text{-}9]^{+}$ {4}
    - Time: Saldrá un cuadro de texto con la máscara de esta expresión regular $[0\text{-}9]^{+}$ {2} [:]$[0\text{-}9]^{+}${2
    - Datetime: Saldrá un cuadro de texto con la mezcla de tipo data más el tipo time
  - Al final para guardar hay que dar al botón actualizar instancia.
- Postcondiciones: Se guarda instancia en la base de datos y en formato RDF.

5.1.5.3 Eliminar instancia

El evento se lanza pulsando el botón de *"Eliminar la instancia seleccionada"* cuando hay una instancia seleccionada. Hay una segunda forma de hacerlo pulsando el enlace con el icono de un cubo de basura al lado de la instancia.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado, en el menú Formularios y en gestión de instancias
- Flujo de eventos:
  - Cuando se dispara este caso, se le pide al usuario que confirme si quiere eliminar esta isntancia. Si se acepta se sigue el flujo sino se aborta.
  - Borra de la base la instancia.
- Postcondiciones: Propiedad eliminada de la base de datos

5.1.5.4 Envio mensaje

El evento se lanza al guardar una instancia ya sea creando o editando una instancia, además de actualizar la instancia envía si tiene el checkbox *"Enviar cambios a la APP"* activado.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar creando o editando una instancia.
- Flujo de eventos:
  - Cuando se dispara este caso, la aplicación prepara un formulario para mandar un máximo de 255 caracteres.
  - Usa un token para preparar un formulario.
  - Manda ese formulario a través de un API que está en una URL
- Postcondiciones: Manda un formulario, Gracias a un proyecto de notificaciones Push mandara una notificación Push a móviles.

5.1.6 Gestión de usuarios

Este evento de la aplicación se lanza cuando, después de haber logeado un usuario entra en la opción Usuarios dentro del menú derecho de navegación. También si estando logeado se pone en la barra de direcciones la siguiente dirección, http://HOST/modules/users/users.php, donde HOST es la dirección web donde se ha instalado la aplicación .

Mostrando una lista de los usuarios que están registrados en la aplicación. Dentro de esta lista hay dos opciones, un botón en el que pone *"Añadir usuario"* que lanzara el caso de uso agregar usuario, y también enlaces en forma de lápiz por cada usuario listado que lanza editar usuario.

Los usuarios pueden ser de dos tipos: administradores o usuarios simples, los administradores pueden agregar y editar usuarios por medio del caso de uso gestión de usuarios, mientras que los usuarios simples pueden ver su información en el caso de uso Mi perfil.

Por defecto siempre hay un usuario root que no se puede eliminar, tantos los usuarios simples como administradores, pueden usar los casos de uso anteriores de formularios, términos, propiedades e instancias. Por lo que la diferencia es solo a nivel de administrar usuarios

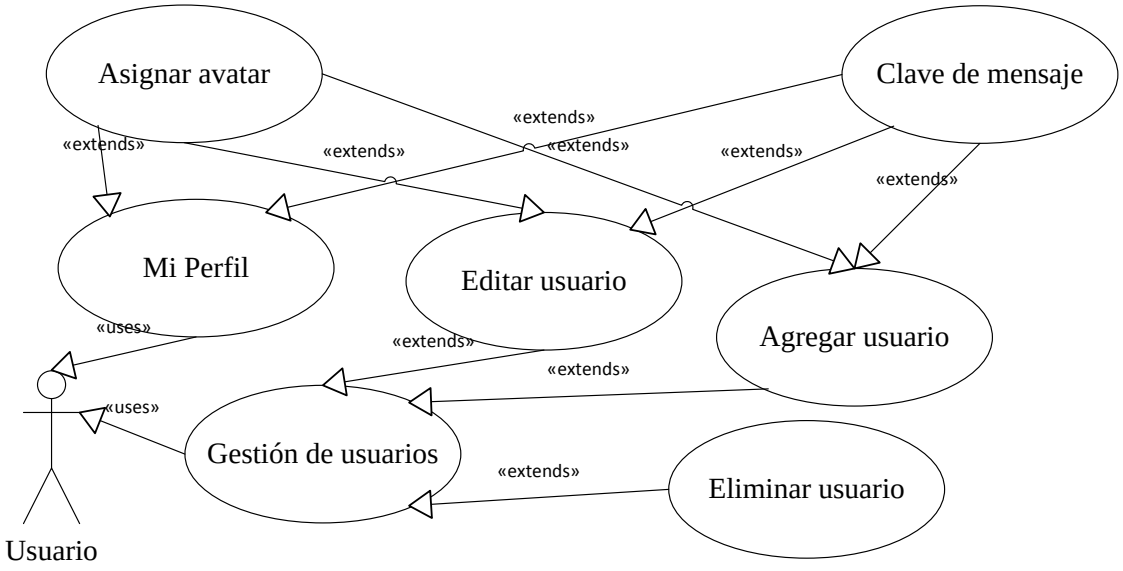


Figura 11 –Diagrama casos de uso gestión usuarios

5.1.6.1 Agregar usuario

El evento se lanza pulsando el botón de *"Añadir nuevo usuario"*, esto le permite al administrador añadir un usuario.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado, en el menú Usuarios y que tiene que ser root
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá rellenar con diferentes campos
  - El usuario introduce el nombre, email, nombre de usuario, contraseña, idioma. Opcionalmente se puede rellenar un token para los mensajes push.
  - Al final para guardar hay que dar al botón *"Crear usuario"*.
- Postcondiciones: Se guarda usuario en la base de datos.

5.1.6.2 Editar usuario

El evento se lanza pulsando el enlace con el icono de un lápiz al lado de la información de usuarios en la lista de usuarios.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado, en el menú Usuarios y que tiene que ser root
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá rellenar con diferentes campos, estos campos están rellenados con los datos del usuario guardado.
  - El usuario puede editar el nombre, email, nombre de usuario, contraseña, idioma y el token para los mensajes push.
  - Al final para guardar hay que dar al botón *"Modificar usuario"*.
- Postcondiciones: Se guarda usuario en la base de datos.

5.1.6.3 Eliminar usuario

El evento se lanza pulsando el enlace con el icono de un cubo de basura al lado de la información de usuarios en la lista de usuarios.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado, en el menú Usuarios y que tiene que ser root
- Flujo de eventos:
  - Cuando se dispara este caso, se le pide al usuario que confirme si quiere eliminar este usuario. Si se acepta se sigue el flujo sino se aborta.
  - Borra de la base el usuario.
- Postcondiciones: Usuario eliminado de la base de datos

5.1.6.4 Mi perfil

El evento se lanza pulsando el enlace *"Mi perfil"*, del menú derecho de navegación. Este enlace sale en sustitución de *"Usuarios"* cuando en vez de un usuario administrador entra un usuario simple.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado y no tiene que ser root
- Flujo de eventos:
  - Le sale un formulario que el usuario podrá editar, con diferentes campos. Estos campos tiene los datos del usuario actual.
  - El usuario puede editar el nombre, email, nombre de usuario, contraseña, idioma y el token para los mensajes push.
  - Al final para guardar hay que dar al botón *"Modificar usuario"*.
- Postcondiciones: Se guarda usuario en la base de datos.

5.1.6.5 Asignar avatar

Permite asignar un avatar a un usuario para mostrarlo en la interfaz de la aplicación.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado, en el menú Usuarios o en la edición del perfil. Puede ser administrador o usuario simple.
- Flujo de eventos:
  - El usuario hace clic sobre la opción "Subir avatar".
  - La plataforma abre el sistema de ficheros local.
  - El usuario selecciona la imagen de su disco local y pulsa el botón aceptar.
  - La plataforma hace una copia de la imagen y la almacena en el alojamiento web en donde está instalada.
- Postcondiciones: Se guarda la imagen como avatar en la aplicación.

5.1.6.6 Clave mensaje

Si el usuario edita el token para los mensajes push, esta se grabara para el envio de los mensajes push.

- Actores: Usuario del CGCRDF
- Precondiciones: Estar dentro del sistema logueado, en el menú Usuarios o en la edición del perfil. Puede ser administrador o usuario simple.
- Flujo de eventos:
  - El usuario tiene un valor en el token para los mensajes push.
  - Pulsar el botón "Crear usuario" o "Modificar usuario".
- Postcondiciones: Usuario tiene un token para el envío de mensajes.

5.2 DISEÑO INTERFAZ GRÁFICA

En ese apartado se presentara algunos diseños hechos para la aplicación de proyectos, estos diseños son responsives, es decir que para distintas medidas de pantallas salen distintos diseños. La interfaz gráfica ha sido diseñada para la navegación natural, es decir suponiendo que las pantallas más grandes son ordenadores y las más pequeñas son móviles

El diseño de la interfaz es a pantalla completa y se han definido tres adaptaciones globales a tipos de dispositivos: ordenadores de escritorio, tablets y dispositivos móviles con unas resoluciones medias mayores a 1024 pixeles, entre 800 y 1024 pixeles y menores de 800 pixeles de ancho respectivamente.

El diseño está hecho primeramente con archivos de photoshop y posteriormente se han pasado a HTML5 y CSS3 válidos

5.2.1 Interfaz gráfica de Login

Pantalla de login

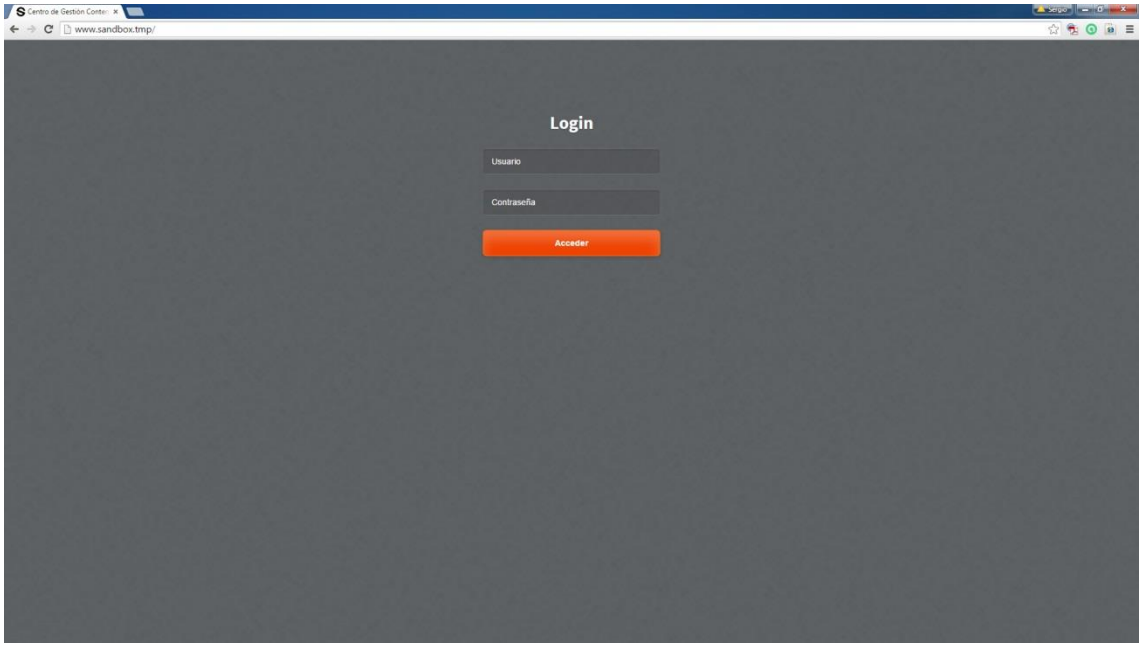


Figura 12 –Interfaz gráfica login

5.2.2 Interfaz gráfica de Entrada

EL menú lateral es responsive y donde están los enlaces y la zonade trabajo donde se muestran los datos

Figura 13 –Interfaz gráfica general

5.2.3 Interfaz gráfica de Cerrar sesión

Cierre de sesion

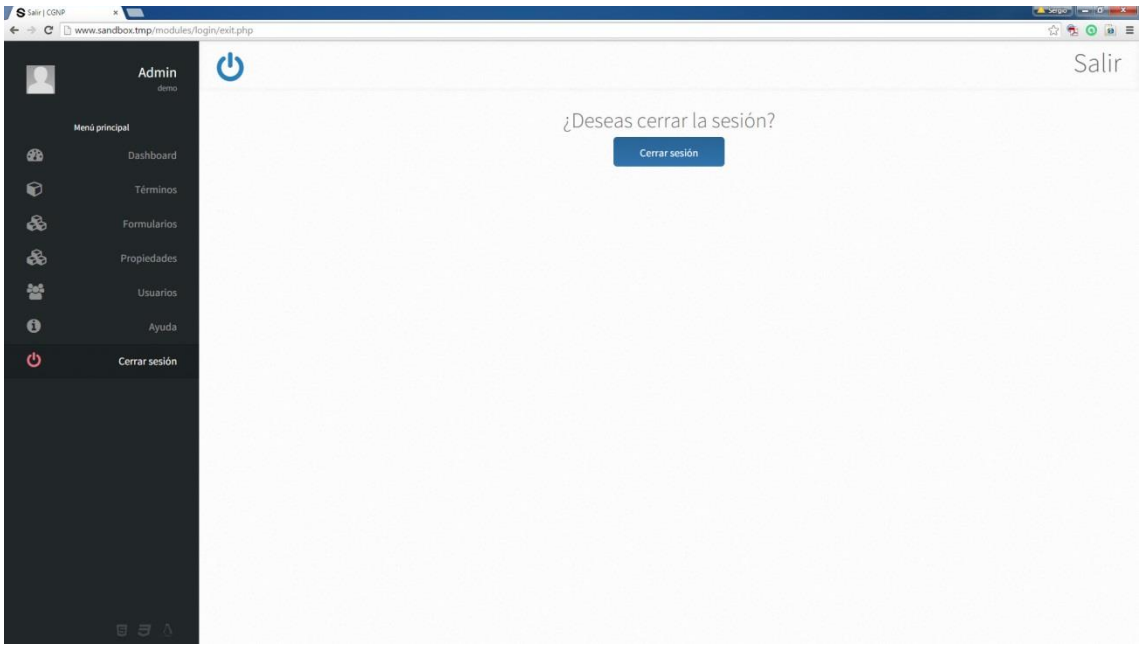


Figura 13 –Interfaz gráfica cierre sesión

5.2.4 Interfaz gráfica de Terminos

Listando términos , caso de uso *"gestión términos"*

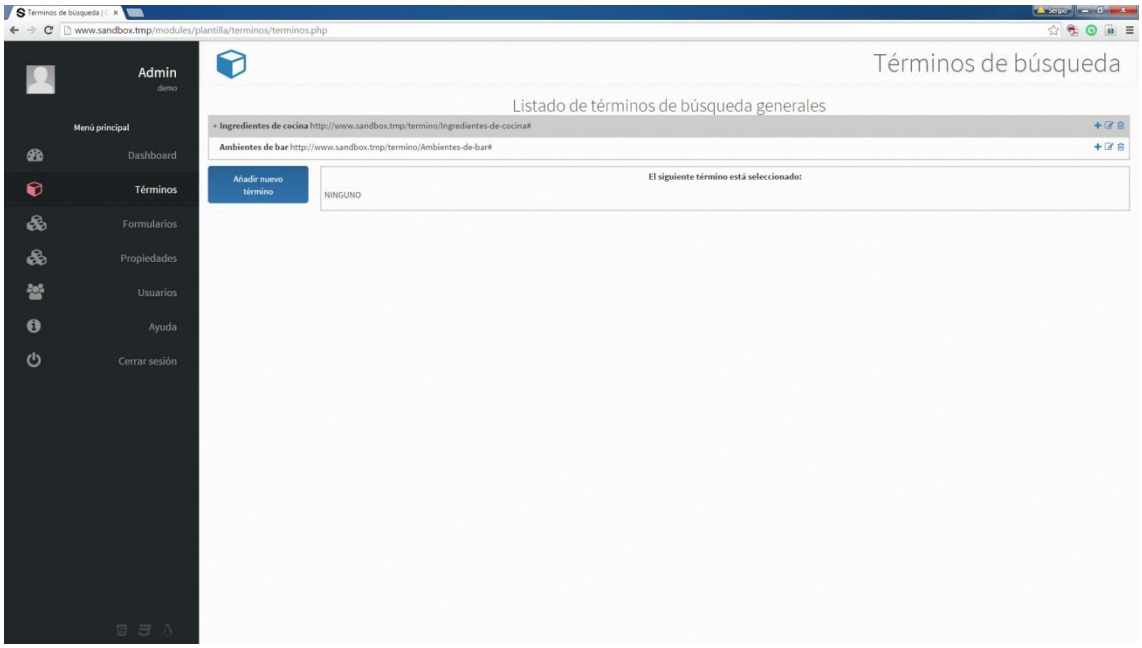


Figura 14 –Interfaz gráfica gestión de terminos

Listando subtérminos , caso de uso *"ver subtérminos"*

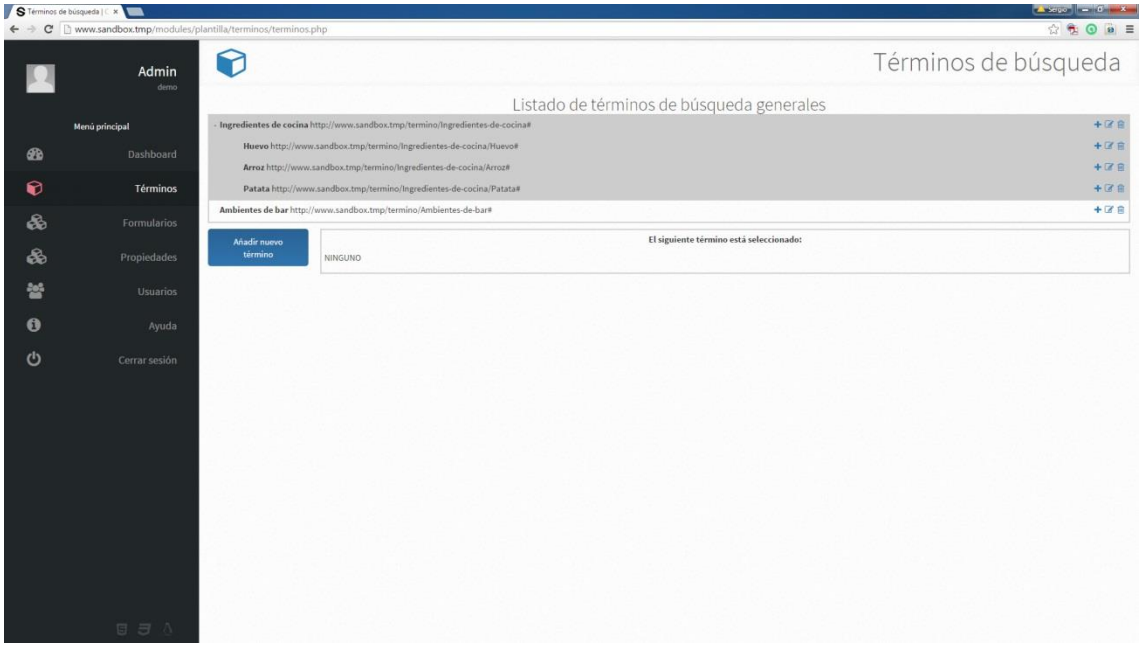


Figura 15 –Interfaz gráfica gestión de subtérminos

Añadiendo nuevos términos o subtérminos, caso de uso *"Agregar término"*

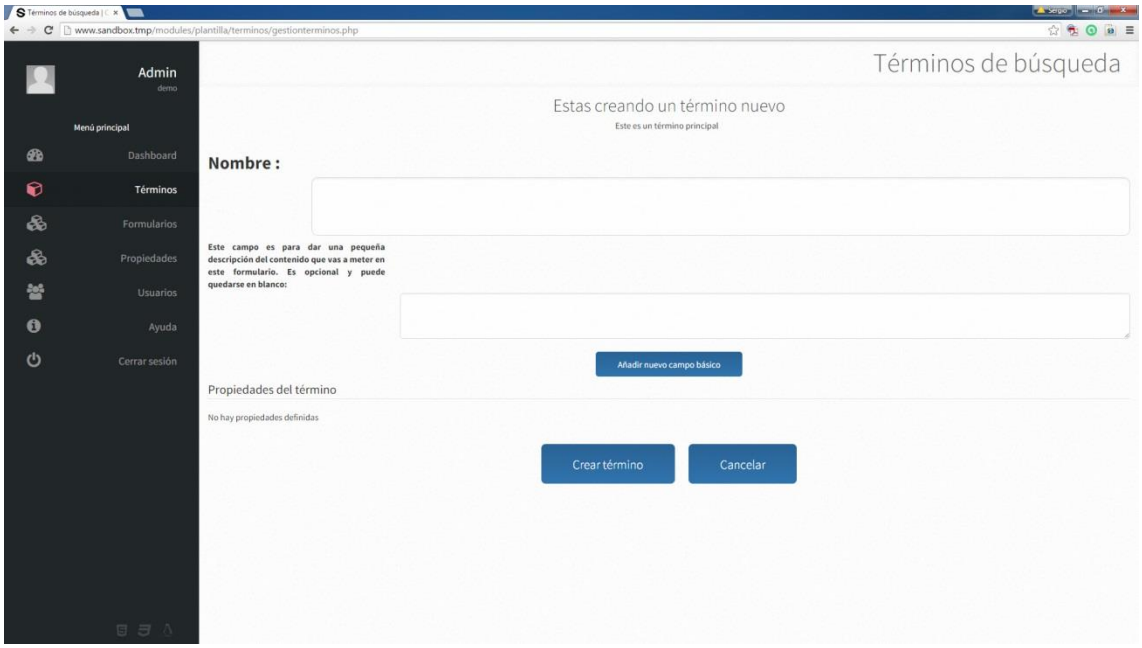


Figura 16 –Interfaz gráfica crear términos

Editando términos o subtérminos, caso de uso *"Modificar término"*

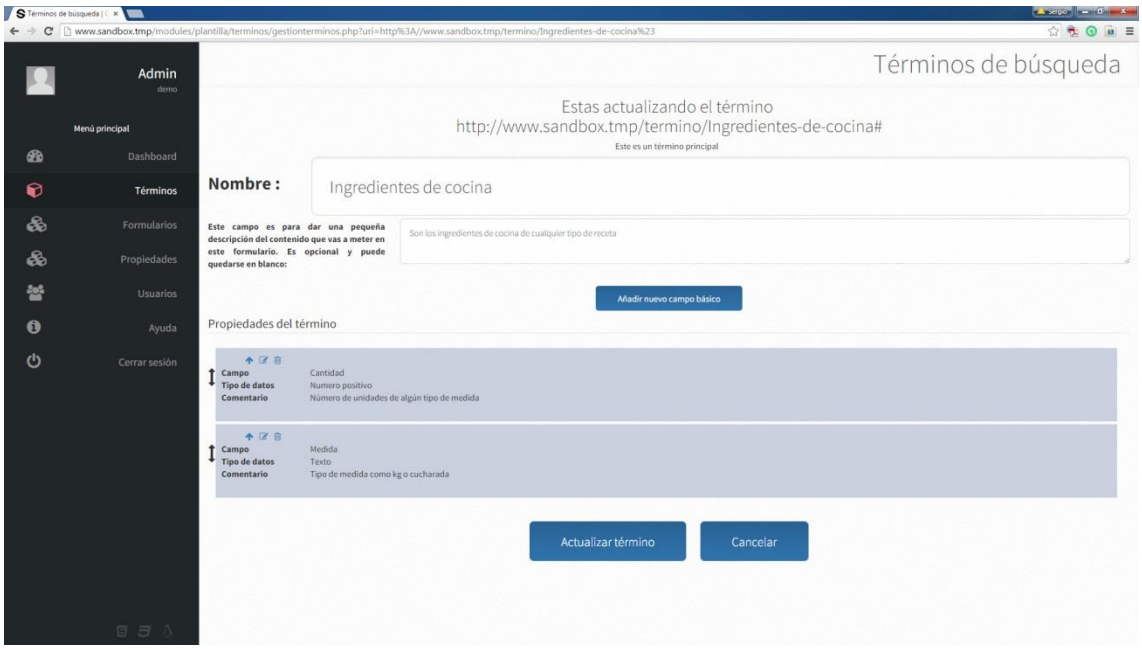


Figura 17 –Interfaz gráfica modificando términos

Añadiendo nuevos campos básicos, caso de uso *"Añadir campo básico"*

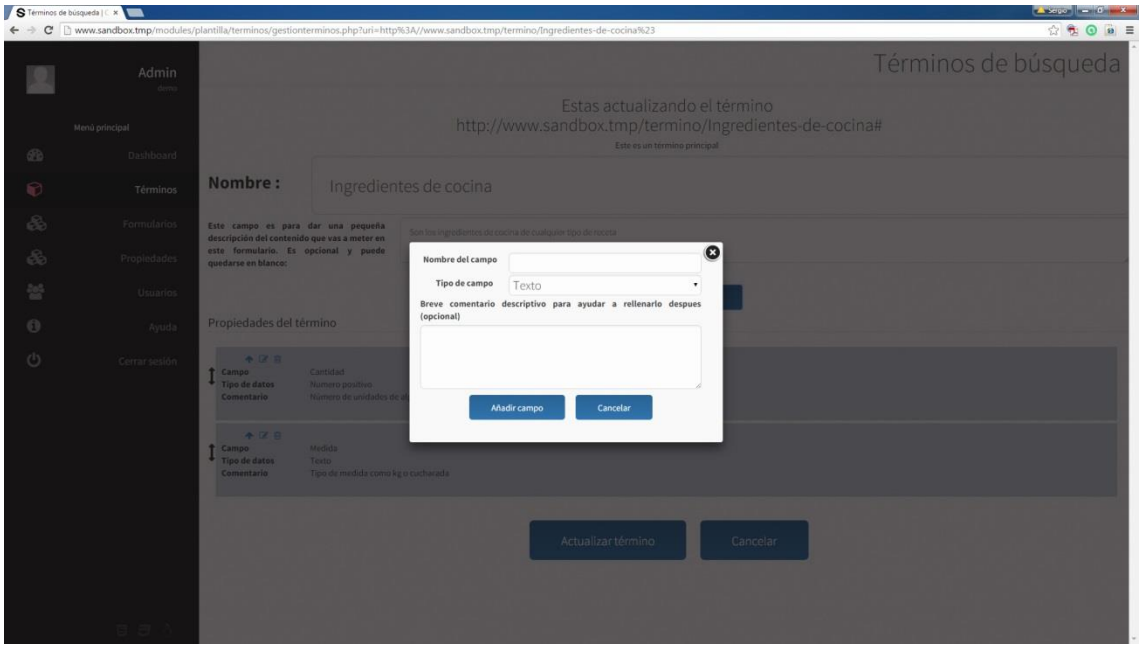


Figura 18 –Interfaz gráfica creando campo básico

Editando campos básicos, caso de uso *"Modificar campo básico"*

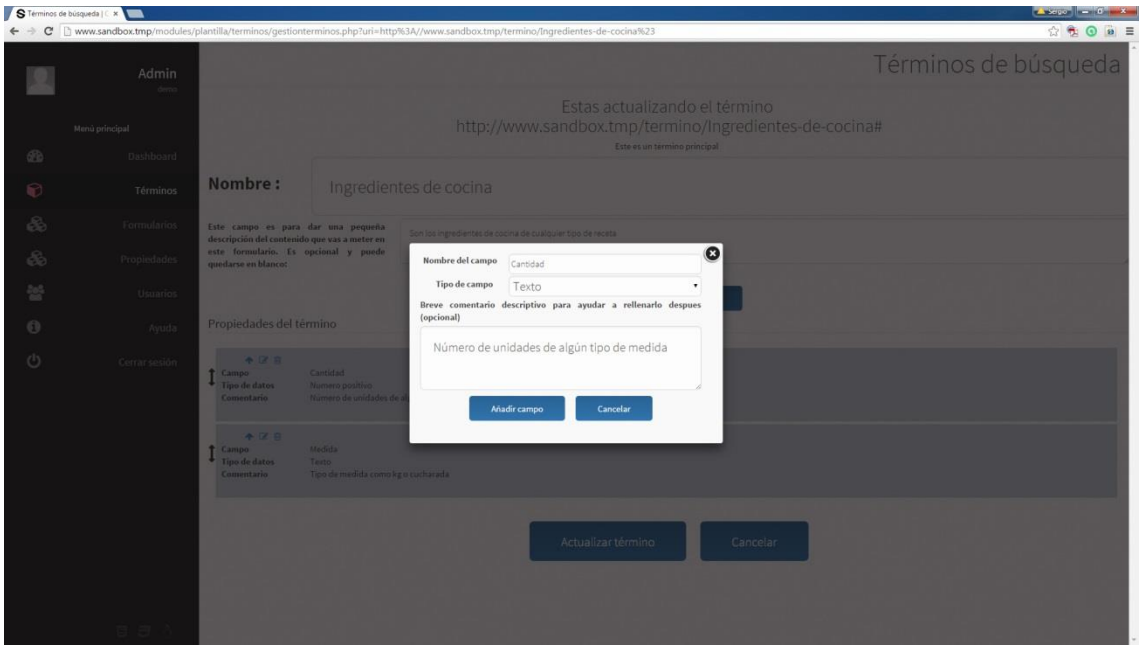


Figura 19 –Interfaz gráfica modificando campo básico

Respuesta al guardar un termino

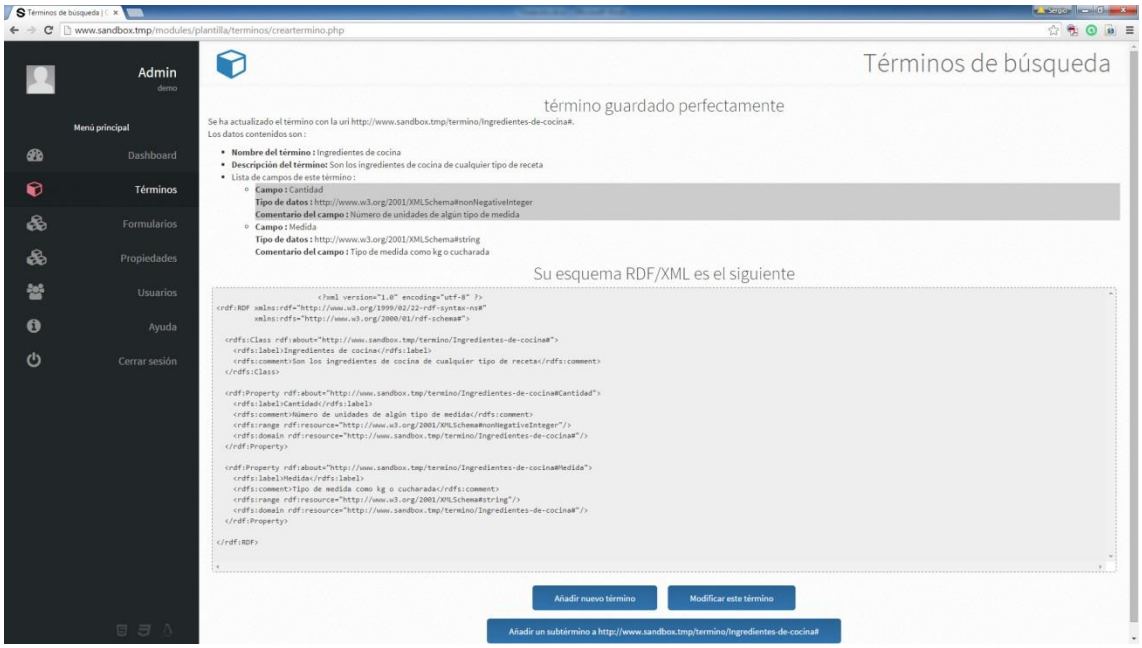


Figura 20 –Interfaz gráfica de los resultados rdf/xml

Segundas opciones, mostrando botones cuando se selecciona un término en gestión de terminos

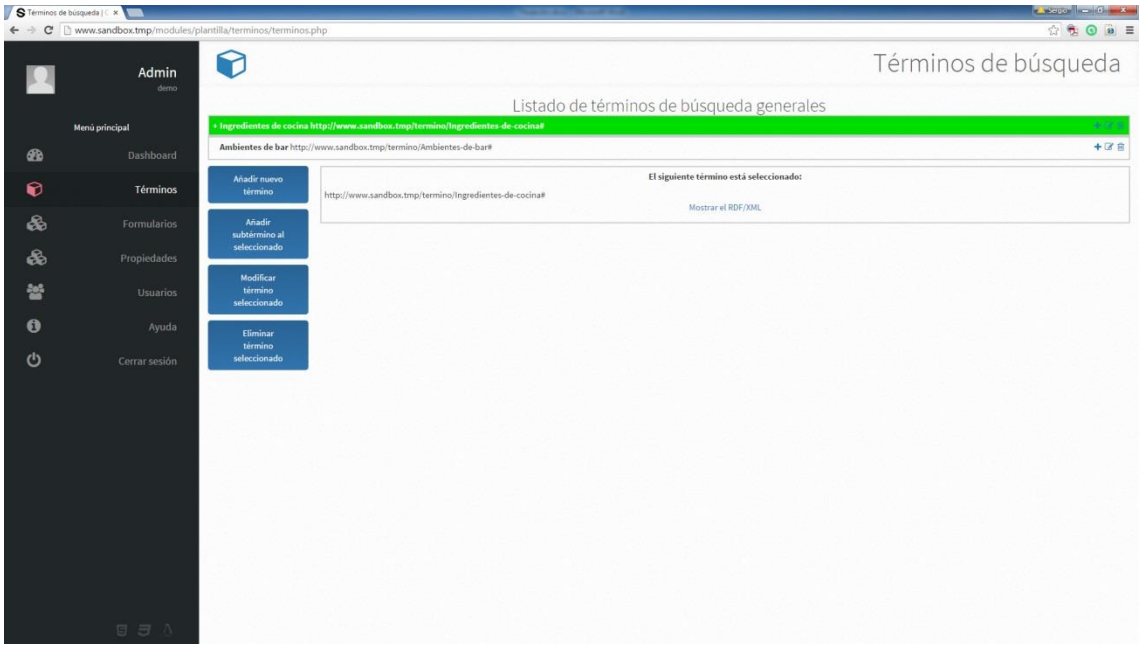


Figura 20 –Interfaz gráfica seleccionando término en gestión de terminos

5.2.5 Interfaz gráfica de Formularios

Listando formularios , caso de uso *"Gestión formularios"*

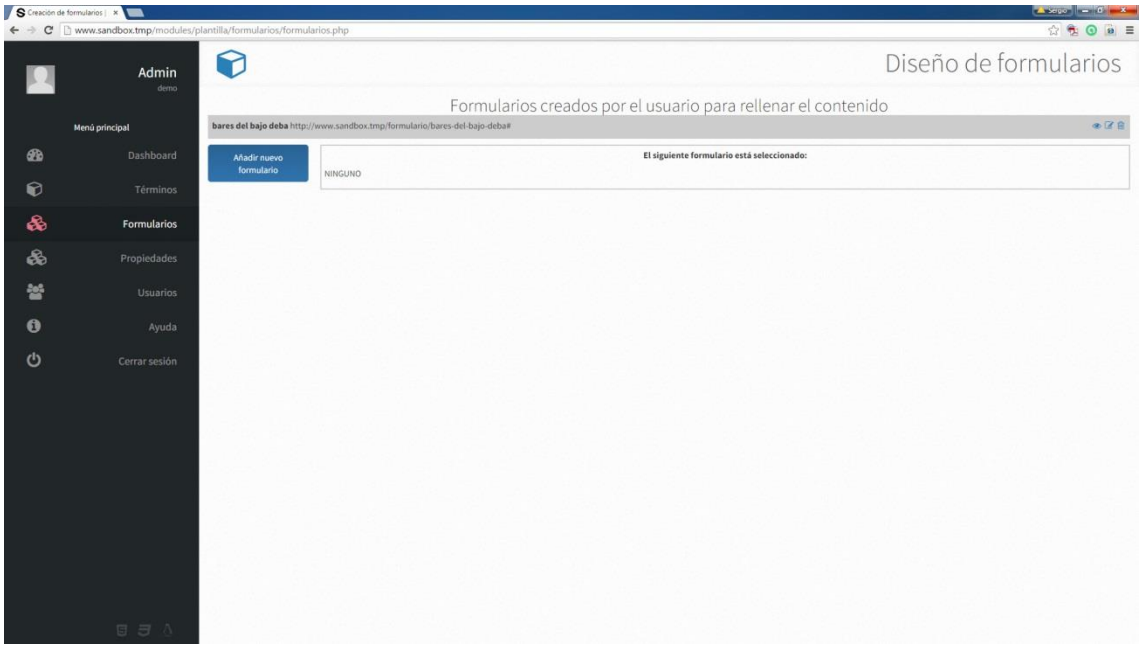


Figura 21 –Interfaz gráfica gestión de formularios

creando formulario , caso de uso *"Agregar formulario"*

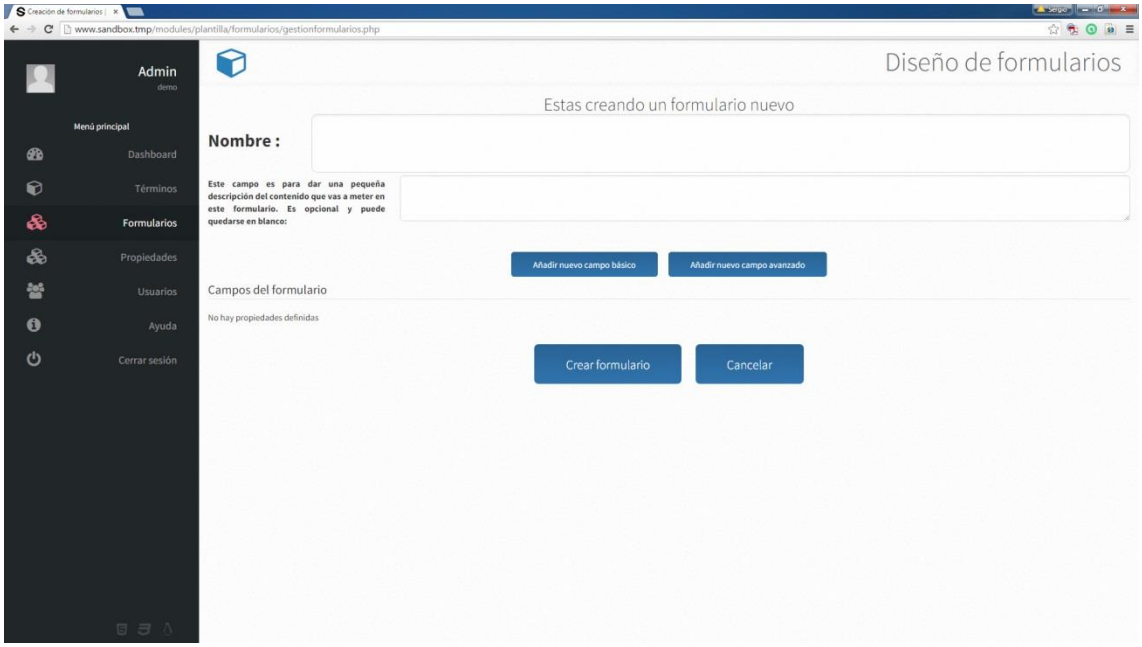


Figura 22 –Interfaz gráfica agregar formulario

Modificando formularios , caso de uso *"Editar formularios"*

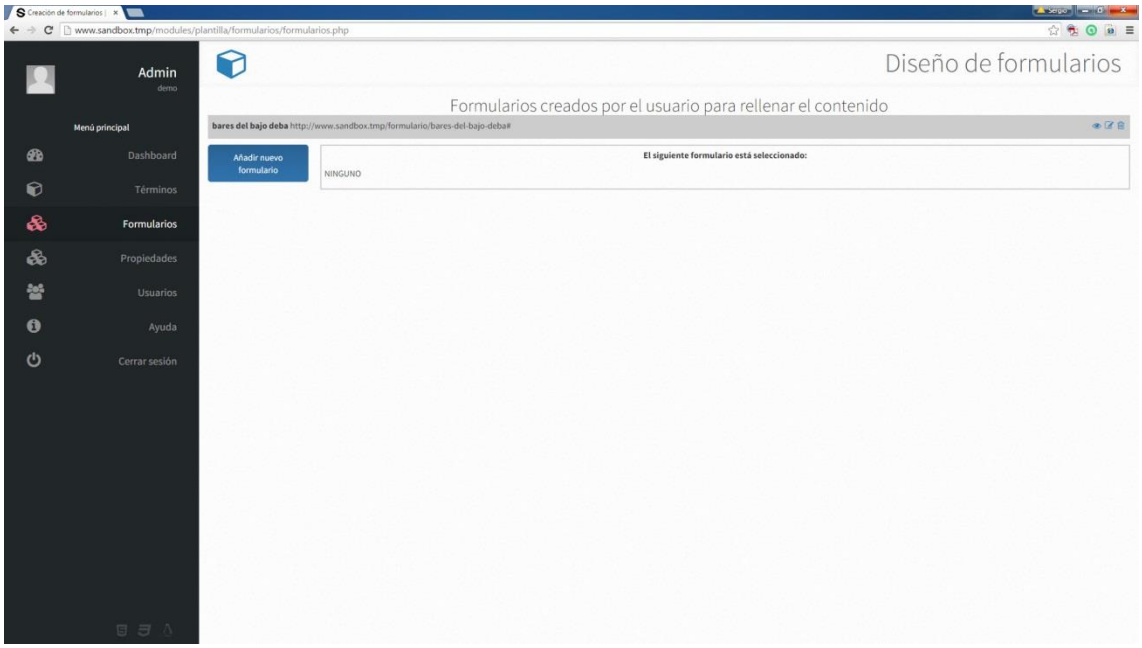


Figura 23 –Interfaz gráfica gestión de formularios

Agregando campo avanzado al formulario , caso de uso *"Agregar campo avanzado"*

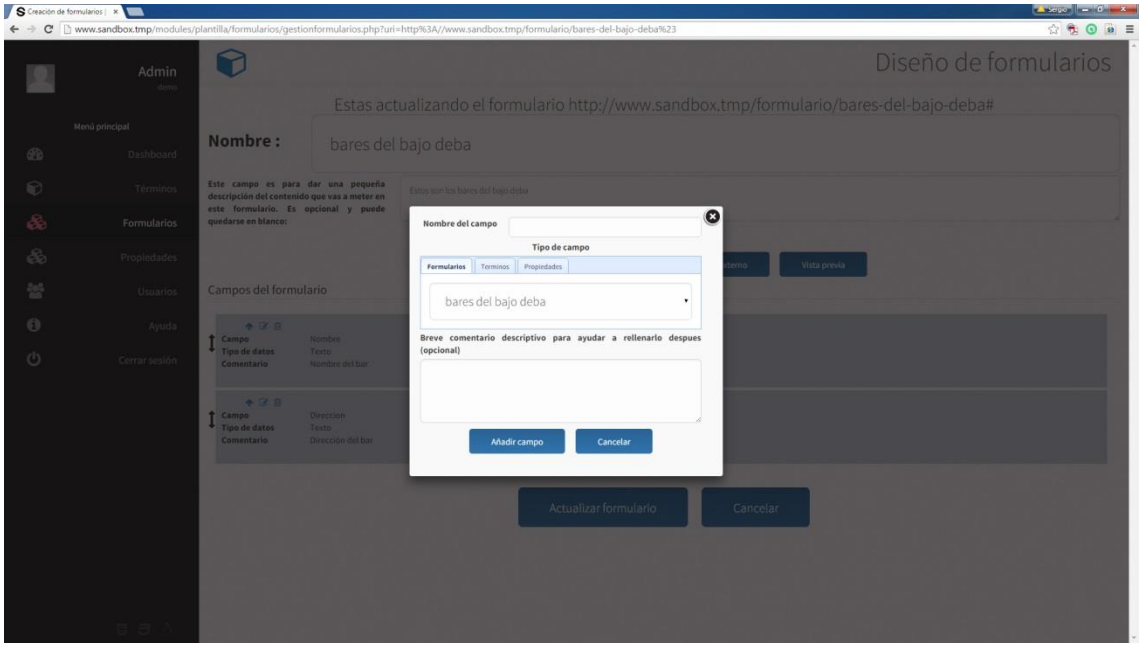


Figura 24 –Interfaz gráfica agregar campo avanzado

5.3 IMPLEMENTACION

En ese apartado se presentara algunos diseños hechos para la aplicación de proyectos, estos diseños son responsives, es decir que para distintas medidas de pantallas salen distintos diseños.

5.3.1 Tecnologías usadas

Para el desarrollo del proyecto se han usado gran variedad de tecnologías que incluyen las empleadas para implementar las capas de presentación, la lógica de negocio, el acceso a datos y la infraestructura del servidor web

5.3.1.1 Presentación

Para la presentación de los casos de uso, formularios y controles, se empleado la tecnología HTML5, que agrupa las tecnologías HTML5, CSS3 y JavaScript. En nuestra implementación hemos usado JavaScript a través de la librería jQuery.

5.3.1.2 Lógica de negocio

La lógica de negocio de las funcionalidades del backend de ha implementado con PHP como lenguaje de programación. Este PHP que se ha usado es el del servidor WAMP o LAMP

5.3.1.3 Acceso a datos

Para almacenar los datos de la aplicación se ha utilizado el sistema de gestión de bases de datos relacional MySQL. Esta aplicación esta instalada en los servidores WAMP y LAMP

5.3.1.4 Infraestructura web

Para que el desarrollo pueda ser accesible a través de la web se ha empleado el servidor Apache, que será el encargado de la comunicación entre el navegador y de la aplicación.

5.3.1.5 Sublime Text

Se ha utilizado el editor de texto avanzado Sublime Text para el desarrollo de las capas de presentación y la lógica de negocio del back end, dado que su flexibilidad permite su uso para editar tanto HTML, CSS, JavaScript y código PHP.

5.3.1.6 PhpMyAdmin

Se ha empleado PhpMyAdmin para la creación y manipulación de la estructura de la base de datos MySQL que emplea la plataforma. Esta aplicación viene preinstalada en los servidores WAMP. La mayoria de los servidores LAMP tambien lo tienen integrado

5.3.1.7 Filezilla

Se ha utiliza este cliente FTP para transferir la implementación realizada al servidor web en donde se aloja la plataforma. Esta aplicación se ha instalado en la plataforma Windows

5.3.1.8 Adobe Photoshop

Se ha utilizado Adobe Photoshop para realizar retoques fotográficos a diversos elementos de la interfaz gráfica del back end. Este es el unico programa que no tiene una licencia GPL

5.3.2 Arquitectura del sistema

Para la implementación del proyecto se han utilizado dos capas o lógicas diferentes. La primera es la capa de presentación que es con la que puede interactuar el usuario, y la segunda es la capa de lógica de negocio que es la que hace que el proyecto interactúe con la base de datos

La lógica de presentación se compone de un conjunto de archivos HTML, estructurados utilizando la última versión del estándar HTML5. Para el diseño se ha usado un archivo CSS, codificado en CSS, que se inserta en la cabecera de los documentos HTML. Tambien tanto para el diseño como para la funcionalidad de la aplicación, se ha usado la librería jQuery de javascript, que se ejecuta en el cliente. Esta librería también se mete en la cabecera de los documentos HTML

La capa de lógica de negocio agrupa al conjunto de clases creadas en documentos PHP, siguiendo las reglas de programación orientado a objetos (POO), que se encargan de implementar la funcionalidad del sistema, y de la creación de contenido HTML dinámico, el acceso a la base de datos. Tambien se encarga de la respuesta en RDF/XML a las direcciones web por medio de la API.

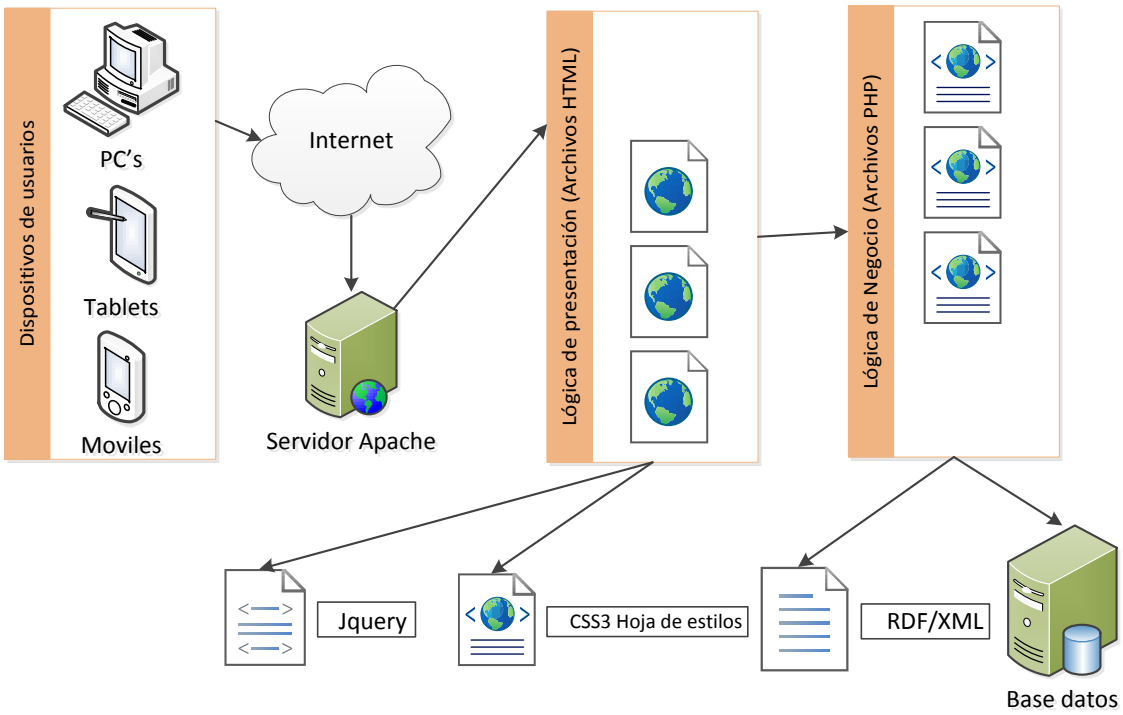


Figura 25 –Arquitectura del sistema

5.3.3 Diagrama UML

Para la implementación del proyecto se han utilizado las siguientes clases

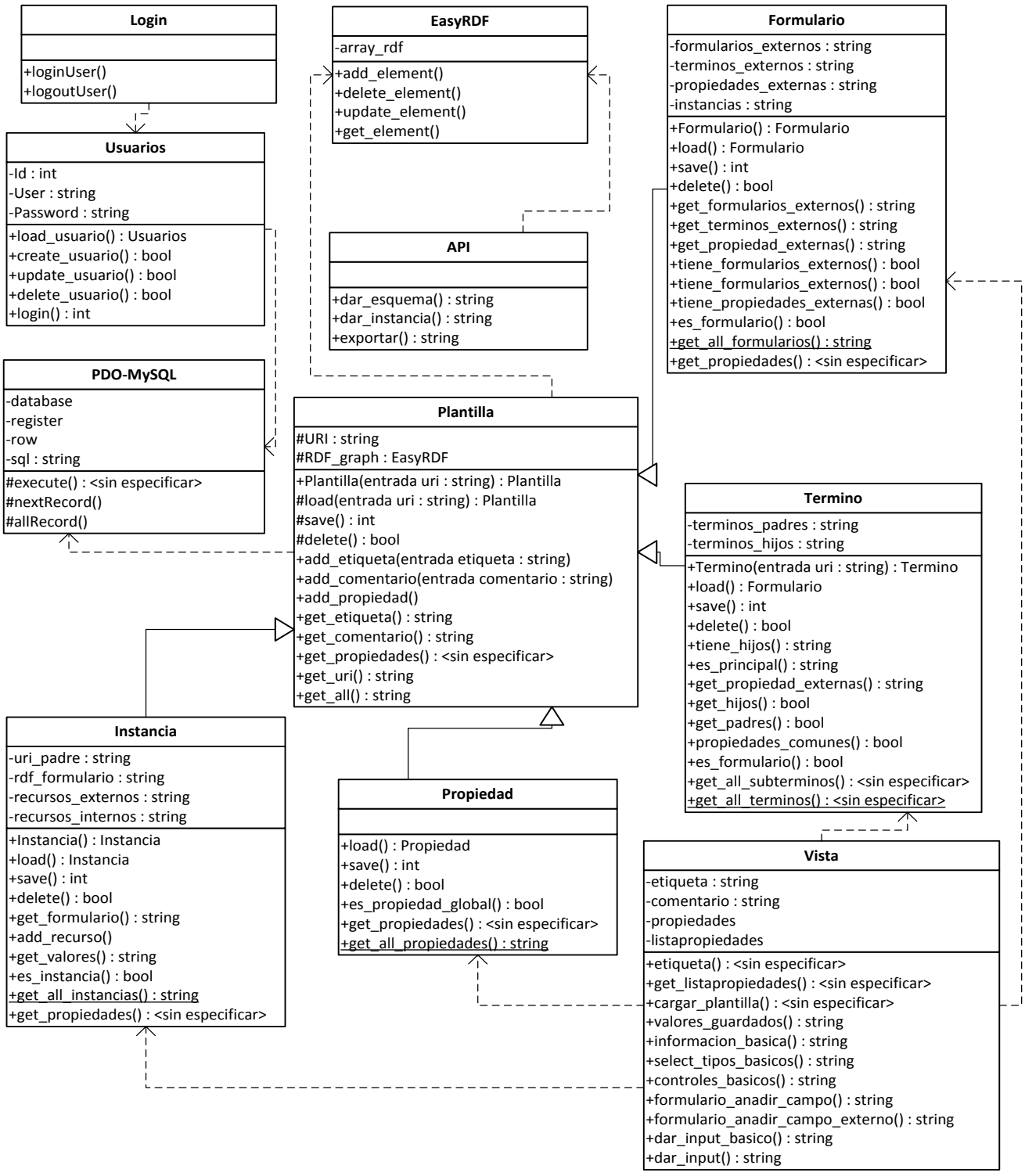


Figura 26 –UML.

5.3.4 Base de datos

Para el proyecto se podrían haber usado ficheros de texto, puesto que la información se guarda toda en un campo de la base de datos en formato n-triples de RDF. Sin embargo para acelerar las consultas a estos datos, se ha utilizado tablas de bases de datos en MySQL. Estas tablas tienen relaciones entre sí para poder acelerar el acceso de datos en el proyecto

Todas las tablas se guardan en UTF-8 como codificación de caracteres, además se guardan sin case sensitive, es decir que no diferencia entre mayúsculas y minúsculas al comparar por medio de las sentencias SQL.

5.3.4.1 Tabla Plantillas

Esta tabla guarda toda la información relacionada con la clase plantilla y sus subclases,excepto instancias que se ha dividido por motivos de velocidad y almacenamiento.

- id_plantilla (int): Clave principal de la tabla.
- Etiqueta (varchar): Etiqueta principal de la plantilla
- Uri_plantilla (varchar): Uri de la plantilla
- ntriples_RDF (longtext) : Tratado como el fichero de texto de n-triples
- fecha_creacion (datetime): fecha de cuando se crea las n-triples
- fecha_modificacion (datetime): fecha de cuando se modifica las n-triples
- formulario_final (boolean) : Si se trata de una subclase termino
- clase_principal (boolean) : Si se trata de una subclase termino y no tiene subtérminos
- propiedad_global (boolean) : Si se trata de una subclase propiedad

5.3.4.2 Tabla Instancias

Esta tabla guarda toda la información relacionada con la subclase de plantilla instancias.

- Id (int): Clave principal de la tabla.
- Etiqueta (varchar): Etiqueta principal de la instancia
- Uri (varchar): Uri de la instancia
- Uri_formulario (varchar) : De que instancia proviene
- ntriples_RDF (longtext) : Tratado como el fichero de texto de n-triples
- fecha_creacion (datetime): fecha de cuando se crea las n-triples
- fecha_modificacion (datetime): fecha de cuando se modifica las n-triples

5.3.4.3 Tabla n-Plantillas

Esta tabla guarda la relacion n que marca de quien es subtérmino cada termino del proyecto, la misma información que esta aquí esta en el RDF del campo ntriples_RDF del apropiado termino. Sin embargo por velocidad tambien se plasma en esta tabla. Aunque en le proyecto se usa solo que un termino pueda tener 0 o 1 padres, la programacion y la estructura de la tabla permite tener n padres.

- Id_linea (int): Clave principal de la tabla.
- uri_plantilla_padre (varchar): Uri del termino padre
- uri_plantilla_hijo (varchar): Uri del subtérmino que tiene como padre el término de uri_plantilla_padre

5.3.4.4 Tabla formulario-formulario

Esta tabla guarda la relacion n que marca que formularios son campos avanzados de otros formularios.

- Id (int): Clave principal de la tabla.
- uri_formulario (varchar): uri del formulario
- uri_esquema_externo (varchar): uri del formulario que es campo avanzado del formulario indicado en uri_formulario

5.3.4.5 Tabla formulario-termino

Esta tabla guarda la relacion n que marca que términos son campos avanzados de otros formularios.

- Id (int): Clave principal de la tabla.
- uri_formulario (varchar): uri del formulario
- uri_esquema_externo (varchar): uri del término que es campo avanzado del formulario indicado en uri_formulario

5.3.4.6 Tabla formulario-propiedad

Esta tabla guarda la relacion n que marca que propiedades son campos avanzados de otros formularios.

- Id (int): Clave principal de la tabla.
- uri_formulario (varchar): uri del formulario
- uri_esquema_externo (varchar): uri del propiedad que es campo avanzado del formulario indicado en uri_formulario

5.3.4.7 Tabla instancia-termino

Esta tabla guarda la relacion n que marca que términos son recurso de una instancia.

- Id (int): Clave principal de la tabla.
- uri_instancia (varchar): uri del formulario.
- uri_termino (varchar): uri del término.

5.3.4.8 Tabla instancia-instancia

Esta tabla guarda la relacion n que marca que instancias son recursos de otras instancias.

- Id (int): Clave principal de la tabla.
- uri_formulario (varchar): uri del formulario
- uri_esquema_externo (varchar): uri del propiedad que es campo avanzado del formulario indicado en uri_formulario

5.3.4.9 Tabla users

Esta tabla guarda la información de todos los usuarios de la aplicación.

- ID (int): Clave principal de la tabla.
- Name (varchar): nombre del ususario
- Email (varchar) : email del usuario
- Username (varchar) : Nick del usuario
- Password (varchar) : Password del usuario
- Lang (int) : Lenguaje por defecto de la aplicación para este usuario
- Avatar (varchar) : fichero de imagen que tiene el usuario dentro de la ruta /user/img.
- Admin (boolean): True si es root
- Active (boolean): True si está activo
- PushID (varchar) : Código para enviar mensajes push
- CreationDate (datetime) : Fecha de creación del usuario
- UpdateDate (datetime) : Fecha de modificación del usuario

5.3.4.10 Tabla lang

Esta tabla guarda la información los lenguajes de la aplicación.

- ID (int): Clave principal de la tabla.
- Code (varchar): Codigo iso del lenguaje
- Description (varchar) : nombre del lenguaje
- Orderfield (varchar) : Orden de aparición dentro del select

5.3.5 Descripción de las clases

En este apartado se explicaran las distintas clases realizadas en el apartado UML, detallando las que componen la lógica de negocio e interfaz gráfica. La clase EasyRDF no ha sido realizada en este proyecto, sino que ha sido utilizada de la página web www.easyrdf.org. Debido a la extensión de esa clase no se comentara entera sino las funciones que se usan, pero básicamente usa un array asociativo como estructura de datos. Tampoco se explicara la clase POO de PHP ya que es muy extensa pero se dará una breve descripción de los métodos usados a través de una subclase que se ha llamado mysql.

5.3.5.1 Plantilla

La plantilla es una clase que engloba todos los posibles esquemas RDF que puede generar el proyecto, además de tener los atributos comunes entre los diferentes elementos de este proyecto. La clase plantilla es el núcleo de las demás clases.

Atributos

protected **uri_plantilla**: Indica la uri de la plantilla
protected **RDF_graph**: Guarda un arrray asociativo que corresponde con el grafo RDF de la plantilla
protected **esclase**: Indica si es una tipo de recurso general o una clase;
private **database**: Enlace con la base de datos;

Métodos

public __construct
Parametros:

- $uri(string): Uri de la plantilla
- $etiqueta(string[]): Opcional, etiqueta principal de la plantilla, se puede poner después con el método add_etiquetas
- $comentario(string[]): Opcional, comentario de la plantilla, se puede poner después con el método add_comentarios
- $propiedades(array): Opcional, como es un array de objetos es mejor usar la función add_propiedad
- $subclases(string[]): Opcional, uri de las subclases, se puede poner después con el método add_subclase
- $esclase (bolean): Opcional, true si es clase, false si es un recurso general, por defecto está en false.

Return:

Constructor de la clase plantilla, además en caso haber parámetros opcionales usa las funciones que se han puesto en cada atributo. También sanea la uri de caracteres raros.

public add_etiquetas
Parametros:

- $etiquetas(string): Etiqueta o etiquetas que se quieren añadir, porque puede ser un array de strings
- $uri(string): Opcional, puede ser vacío

Return:
Si $uri esta vacio añade un rdfs:label con valor de $etiquetas a la estructura de datos de la plantilla actual, sino busca a ver si hay alguna plantilla con esa uri para añadir el rdfs:label. Si es array $etiquetas repite el proceso por cada uno de los valores.

public add_comentarios
Parametros:

- $comentarios(string): Comentario o comentarios que se quieren añadir, porque puede ser un array de strings
- $uri(string): Opcional, puede ser vacío

Return:
Si $uri esta vacio añade un rdfs:comment con valor de $comentarios a la estructura de datos de la plantilla actual, sino busca a ver si hay alguna plantilla con esa uri para añadir el rdfs:comment. Si es array $etiquetas repite el proceso por cada uno de los valores

public add_subclases
Parametros:

- $ subclases (string): Subclase o subclases que se quieren añadir, porque puede ser un array de strings
- $uri(string): Opcional, puede ser vacío

Return:
Si $uri esta vacio añade un rdfs:subClassOf con valor de $etiquetas a la estructura de datos de la plantilla actual, sino busca a ver si hay alguna plantilla con esa uri para añadir el rdfs:subClassOf. Si es array $etiquetas repite el proceso por cada uno de los valores y solamente añade si la plantilla es una clase.

public add_propiedades
Parametros:

- $etiqueta(string): etiqueta de la propiedad
- $cometario(string): Opcional, puede ser vacío
- $rango: Opcional, puede ser vacío
- $dominio: Opcional, puede ser vacío
- $subpropiedad: Opcional, puede ser vacío
- uri: Opcional, puede ser vacío

Return:
Si $uri esta vacio añade un rdf:property con valor de $etiqueta a la estructura de datos de la plantilla actual, sino busca a ver si hay alguna plantilla con esa uri para añadir el rdf:property.

public load
Parametros:

- $uri(string): uri de la plantilla

Return:
Carga en una clase plantilla lo que está guardado en la base de datos con esa uri.

public save
Return:
Guarda la clase plantilla actual en la base de datos con esa uri.. Devuelve el id de la inserción o del update si es correcto, false en caso contrario.

public delete
Parametros:

- $uri(string): uri de la plantilla

Return:
Borra de la base de datos la clase plantilla con la uri del parámetro.

public get_uri
Return:
Devuelve la uri de la plantilla.

public get_etiqueta
Return:
Devuelve la etiqueta principal de la plantilla.

public get_comentario
Return:
Devuelve un comentario de la plantilla.

public get_propiedades
Return:
Devuelve un array con todas las rdf:Property de la plantilla cada elemento del array tiene esta forma: {"uri","etiqueta","comentario","rango"}

public get_all
Return:
Serializa la plantilla en RDF/XML

public RDF_graph()
Return:
Devuelve el array asociativo de RDF_graph

public es_plantilla
Parametros:

- $uri(string): uri de la plantilla

Return:
Dice si hay una plantilla con esa uri

public Sanear_uri
Parametros:

- $uri(string): uri de la plantilla

Return:
Devuelve uri totalmente saneada por medio de filtros PHP

5.3.5.2 Termino

Termino es una extensión de plantilla y guarda lo que en el proyecto hemos denominado términos. .

Atributos

private **terminos_padres**: Array que contiene los términos padre este término puede ser vacio, en el proyecto solo hemos usado 0 o 1 solo padre.
private **terminos_hijos**: Array que tiene los términos hijo este término puede ser vacío

Métodos

public __construct
Parametros:

- $uri(string): Uri de la plantilla
- $etiqueta(string[]): Opcional, etiqueta principal de la plantilla, se puede poner después con el método add_etiquetas
- $comentario(string[]): Opcional, comentario de la plantilla, se puede poner después con el método add_comentarios
- $propiedades(array): Opcional, como es un array de objetos es mejor usar la función add_propiedad
- $subclases(string[]): Opcional, uri de las subclases, se puede poner después con el método add_subclase

Return:
Constructor de la clase término, además en caso haber parámetros opcionales usa las funciones que se han puesto en cada atributo. También sanea la uri de caracteres raros. Básicamente llama al constructor de plantilla, pero pone en los arrays los subtérminos.

public load
Parametros:

- $uri(string): uri del termino

Return:
Carga en una clase termino lo que está guardado en la base de datos con esa uri.

public save
Return:
Guarda la clase termino actual en la base de datos con esa uri. También pone en la base de datos los flags necesarios. Devuelve el id de la inserción o del update si es correcto, false en caso contrario.

public delete
Parametros:

- $uri(string): uri del término, también borra de la base datos los subtérminos

Return:
Borra de la base de datos la clase término con la uri del parámetro.

public tienehijos
Return:
Dice si el término actual tiene subtérminos el término actual.

public categoriaprincipal
Return:
Dice si el término no tiene términos padre

public get_hijos
Return:
Devuelve terminos_hijos

public get_padres
Return:
Devuelve terminos_padres

public propiedades_comunes
Return:
Devuelve todas los recursos rdf:Property desde el termino actual hasta los términos que no tengan padres.

public get_all_terminos
Return:
Devuelve todas las uris y etiquetas de los términos de la base de datos

public get_all_subterminos
Parametros:

- $uri(string): uri del término, también borra de la base datos los subtérminos

Return:
Devuelve todas las uris y etiquetas de los términos de la base de datos que son subtérminos del termino cuya uri se indica en el parametro

5.3.5.3 Formulario

Formulario es una extensión de plantilla y guarda lo que en el proyecto hemos denominado formulario, que son los que después se rellenaran con instancias. Todos los atributos son arrays con URIs, pero están separados por aceleración del sistema. .

Atributos

private **formularios_externos**: Array que contiene los formularios relacionados
private **terminos_externos**: Array que tiene los términos relacionados
private **propiedades_externas**: Array que contiene las propiedades relacionados
private **instancias_formulario**: Array que tiene las instancias relacionadas

Métodos

public __construct
Parametros:

- private $uri(string): Uri de la plantilla
- private $etiqueta(string[]): Opcional, etiqueta principal de la plantilla, se puede poner después con el método add_etiquetas
- private $comentario(string[]): Opcional, comentario de la plantilla, se puede poner después con el método add_comentarios
- private $propiedades(array): Opcional, como es un array de objetos es mejor usar la función add_propiedad
- private $subclases(string[]): Opcional, uri de las subclases, se puede poner después con el método add_subclase

Return:
Constructor de la clase formulario, además en caso haber parámetros opcionales usa las funciones que se han puesto en cada atributo. También sanea la uri de caracteres raros. Básicamente llama al constructor de plantilla, pero pone en los arrays los datos si hubiera opcionales.

public load
Parametros:

- $uri(string): uri del formulario

Return:
Carga en una clase formulario lo que está guardado en la base de datos con esa uri.

public save
Return:
Guarda la clase formulario actual en la base de datos con esa uri. También pone en la base de datos los flags necesarios. . Devuelve el id de la inserción o del update si es correcto, false en caso contrario..

public delete
Parametros:

- $uri(string): uri del formulario, NO BORRA LAS INTANCIAS

Return:
Borra de la base de datos la clase formulario con la uri del parámetro.

public tiene_instancias
Return:
True si tiene instancias, false si no tiene instancias relacionadas con el formulario.

public get_instancias
Return:
Devuelve el array de instancias.

public tiene_formulario_externos
Return:
True si tiene formularios relacionados, false si no tiene formularios relacionados con el formulario.

public get_formulario_externos
Return:
Devuelve el array de formularios.

public tiene_terminos_externos
Return:
True si tiene terminos relacionados, false si no tiene terminos relacionados con el formulario.

public get_terminos_externos
Return:
Devuelve el array de formularios

public tiene_propiedades_externas
Return:
True si tiene propiedades relacionadas, false si no tiene propiedades relacionadas con el formulario.

public get_propiedades_externas
Return:
Devuelve el array de propiedades

public es_formulario_global
Parametros:

- $uri(string): uri de la plantilla

Return:
Dice si hay un formulario en la base de datos con esa uri

public get_all_formularios
Return:
Devuelve todas las uris y etiquetas de los formularios de la base de datos

public get_propiedades
Return:
Llama al get_propiedades de plantilla, y a cada valor del array le añade un nuevo elemento a cada valor que hemos denominado “tipo”. Dependiendo en uqe array esta le añade “formulario”,”termino”,”propiedades” o “simple”

5.3.5.4 Propiedades

Propiedades es una extensión de plantilla y guarda lo que en el proyecto hemos denominado propiedades.

Métodos

public __construct
Parametros:

- $uri(string): Uri de la plantilla
- $etiqueta(string[]): Opcional, etiqueta principal de la plantilla, se puede poner después con el método add_etiquetas
- $comentario(string[]): Opcional, comentario de la plantilla, se puede poner después con el método add_comentarios
- $propiedades(array): Opcional, como es un array de objetos es mejor usar la función add_propiedad
- $subclases(string[]): Opcional, uri de las subclases, se puede poner después con el método add_subclase

Return:
Constructor de la clase propiedades, además en caso haber parámetros opcionales usa las funciones que se han puesto en cada atributo. También sanea la uri de caracteres raros. Básicamente llama al constructor de plantilla, pero solamente con el primer elemento de cada array.

public load
Parametros:

- $uri(string): uri de la propiedad

Return:
Carga en una clase propiedad lo que está guardado en la base de datos con esa uri.

public save
Return:
Guarda la clase propiedad actual en la base de datos con esa uri. También pone en la base de datos los flags necesarios. . Devuelve el id de la inserción o del update si es correcto, false en caso contrario.

public delete
Parametros:

- $uri(string): uri de la propiedad.

Return:
Borra de la base de datos la clase propiedad con la uri del parámetro.

public es_propiedad_global
Parametros:

- $uri(string): uri de la propiedad

Return:
Dice si hay una propiedad en la base de datos con esa uri

public get_all_propiedades
Return:
Devuelve todas las uris y etiquetas de los propiedades de la base de datos

public get_propiedades
Return:
Llama al get_propiedades de plantilla, y solo devuelve el primer termino

5.3.5.5 Instancias

Instancias es una extensión de plantilla y guarda lo que en el proyecto hemos denominado instancias, que son los datos que el usuario de la aplicación quiere guardar, usando como esqueleto del formulario lo guardado en las clases formularios.

Atributos

private **uri_formulario_padre**: Uri del formulario que tiene de esqueleto la instancia
private **RDF_graph_formulario**: Array que representa el RDF del formulario
private **recursos_externos**: Array que contiene las uris de resources que están fuera de este RDF
private **recursos_internos**: Array que contiene las uris de resources que están dentro de este RDF

Métodos

public __construct
Parametros:

- private $etiqueta(string): etiqueta principal de la instancia, es obligatoria porque luego forma la uri de la instancia con esta etiqueta.
- private $uri_formulario(string): Uri del formulario que usa como esqueleto, esta es la primera parte de la uri de la instancia
- private $comentario(string[]): Opcional, comentario de la plantilla, se puede poner después con el método add_comentarios
- private $valores(array): Opcional, como es un array de objetos que da valores a la instancia , se puede añadir despues con función add_recurso

Return:
Constructor de la clase instancia, además en caso haber parámetros opcionales usa las funciones que se han puesto en cada atributo. También sanea la uri de caracteres raros. Básicamente llama al constructor de plantilla, pero pone en los arrays los datos si hubiera opcionales.

public add_recurso
Parametros:

- private $valor(objeto): Array asociativo de una forma en concreto
- private $uri (string): Uri del recurso
- private $uri_destino(string): Opcional, Destino donde se quiere añadir el recurso.

Return:
Añade un tipo rdfs:Resource con la uri del parámetro $uri al recurso cuya uri destino es $uri_destino, si está en blanco lo añade al atributo uri_plantilla, porque lo hereda de plantilla. Sino busca en la estrutura RDF actual si hay una $uri_destino.

public load
Parametros:

- $uri(string): uri de la instancia

Return:
Carga en una clase instancia lo que está guardado en la base de datos con esa uri.

public save
Return:
Guarda la clase instancia actual en la base de datos con esa uri. También pone en la base de datos los flags necesarios. Devuelve el id de la inserción o del update si es correcto, false en caso contrario.

public delete
Parametros:

- $uri(string): uri de la instancia.

Return:
Borra de la base de datos la clase instancia con la uri del parámetro.

public get_formulario
Return:
Devuelve la uri del formulario que usa la instancia de esqueleto.

public get_valores
Parametros:

- $uri_origen(string): Opcional, uri de la instancia.
- $uri_plantilla(string): Opcional, uri de la instancia.

Return:
Devuelve un array de valores con los tipos rdfs:resource que tiene la instancia.

public es_instancia
Parametros:

- $uri(string): uri de la instancia

Return:
Dice si hay una instancia en la base de datos con esa uri

5.3.5.6 Vista

Vista es una clase estatica que da la mayoría del HTML que se necesita dinámicamente dependiendo de los parámetros.

Atributos

private **etiqueta**: Etiqueta de la plantilla que se está pintando, para no usar todo el rato get_etiqueta de la plantilla.
private **comentario**: Comentario de la plantilla que se está pintando, para no usar todo el rato get_comentario de la plantilla.
private **propiedades**: lista de propiedades de la plantilla para no usar todo el rato get_propiedades
private **$listapropiedades**: donde se guarda el HTML que luego se va a pintar. Se denomina lista porque normalmente empieza con el tag <ul> de html

Métodos

public etiqueta
Parametros:

- private $rango(string): uri que representa un rdfs:range

Return:
Traduce el rdfs:range al lenguaje por defecto en este caso español. Ejemplo xsd:string a texto o , xsd:integer a entero. Si la uri es una uri que no representa un rango, intenta buscar en las plantillas y devuelve get_etiqueta de plantilla.

public get_listapropiedades
Return:
Devuelve listapropiedades

public cargar_plantilla
Parametros:

- private $uri(string): uri que representa una plantilla

Return:
Rellena los atributos necesarios, para no llamar una y otra vez a get_etiquetas. Es parecido a lo que sería un constructor, pero como la clase es estática, no se le puede hacer un constructor.

public valores_guardados
Parametros:

- private $uri(string): uri que representa una plantilla

Return:
Hace carga_plantilla y además devuelve el HTML correspondiente a la lista de propiedades, en concreto los campos que tiene una plantilla con formato correspondiente..

public informacion_basica
Return:
Devuelve el HTML correspondiente a un bloque que contiene etiqueta y el comentario. Si los parámetros etiqueta y comentario están inicializados mete esos valores en los valores de los tag input de HTML.

public select_tipos_basicos
Return:
Devuelve el HTML correspondiente a un bloque select con los tipos de datos básicos de RDF. Como está comentado anteriormente no uso más que ciertos tipos como básicos

public controles_basicos
Return:
Devuelve el HTML correspondiente a un bloque de botones.

public formulario_anadir_campo
Return:
Devuelve HTML correspondiente al formulario que se usa para añadir campo basico.

public formulario_anadir_campo_externo
Return:
Devuelve HTML correspondiente al formulario que se usa para añadir campo avanzado

public formulario_anadir_campo_externo
Return:
Devuelve HTML correspondiente al formulario que se usa para añadir campo avanzado

public dar_input_basico
Parametros:

- private $rango(string): uri que representa un rdfs:range
- private $valor(string): string para insertar en un parámetro value de un input
- private $nombre(string): string para insertar en un parámetro name de un input

Return:
Devuelve el HTML de un input, que mas se ajuste al rdfs:range, además rellena el value y el name si no son blancos.

5.3.5.7 Api

Api es la clase que exporta un elemento de la base de datos a cualquier tipo de formato RDF, gracias a la serialización del array asociativo. Esta clase es la que se usa para poder comunicarse con las aplicaciones exteriores.

Atributos

private **database**: link a la base de datos

Métodos

public dar_esquema
Parametros:

- private $uri(string): uri que representa una plantilla
- private $format(string): formato en que se serializa el RDF

Return:
Devuelve la plantilla serializada en el formato que se pide, solamente devuelve el RDF de la plantilla.

public dar_instancia
Parametros:

- private $uri(string): uri que representa una plantilla
- private $format(string): formato en que se serializa el RDF

Return:
Devuelve la instancia serializada en el formato que se pide, solamente devuelve el RDF de la instancia.

public exportar
Parametros:

- private $uri(string): uri que representa una plantilla
- private $format(string): formato en que se serializa el RDF

Return:
Esta es la función que exporta todo, es decir si metes una plantilla devuelve el RDF de la plantilla y de las instancias asociadas a esa plantilla..

5.3.5.8 MySql

MySql es la clase realizada para usar la clase nativa de PHP de comunicación con bases de datos (PDO). Se ha hecho esta clase para tener más ordenado las llamadas a la base de datos.

Atributos

private **pdo**: Enlace con la base de datos.
private **statement**: Guardado de la respuesta a la sentencia SQL.
private **record**: Tupla actual

Métodos

public __construct
Return:
Conecta con la base de datos con los parámetros guardados en la clase.

public execute
Parametros:

- private $sql(string): sentencia SQL
- private $parameters (string): array de valores a usar

Return:
Ejecuta la sentencia sql con la conexión de la base de datos hecha en pdo.

public nextRecord
Parametros:

- private $style: Opcional, indica como devoler el array de respuesta de la sentencia SQL. Por defecto esta en asociativo el array

Return:
Devuelve una tupla si statement tiene valor .

public allRecord
Parametros:

- private $style: Opcional, indica como devoler el array de respuesta de la sentencia SQL. Por defecto esta en asociativo el array

Return:
Devuelve todas las tuplas si statement tiene valor

5.3.5.9 User

La clase user es la típica clase de administrar los usuarios de una base de datos, en internet hay muchos tipos de esas clases para PHP.

Atributos

private **username**: nick del usuario.
private **id**: id del usuario dentro de la base.
private **admin**: booleano para saber si es admin
private **password**: password del usuario.
private **avatar**: string del path del avatar.

Métodos

public __construct : construye la clase usuario
public loadUser: carga el usuario
public loadUserList: devuelve la lista de usuarios si es administrador
public create : crea un usuario con los datos apropiados en la base de datos
public update : modifica un usuario con los datos apropiados en la base de datos
public delete : borra un usuario de la base de datos
public updateAvatar: modifica en la base de datos el path del avatar del usuario.

5.3.5.10 Login

Login es la clase que se asegura de que el usuario este en el sistema o no, además de comprobar si un usuarios es adminsitrador o no.

Atributos

private **username**: nick del usuario.
private **id**: id del usuario dentro de la base.
private **admin**: booleano para saber si es admin

Métodos

public login
Parametros:

- private $user(string): usuario
- private $password (string): contraseña

Return:
Si existe un usuario con esa contraseña rellena el id y el admin. Además da un mensaje de que existe un usuario.

public checkLogin()
Return:
Este método se encarga comprobar si la sesión en la plataforma esta activa o no.

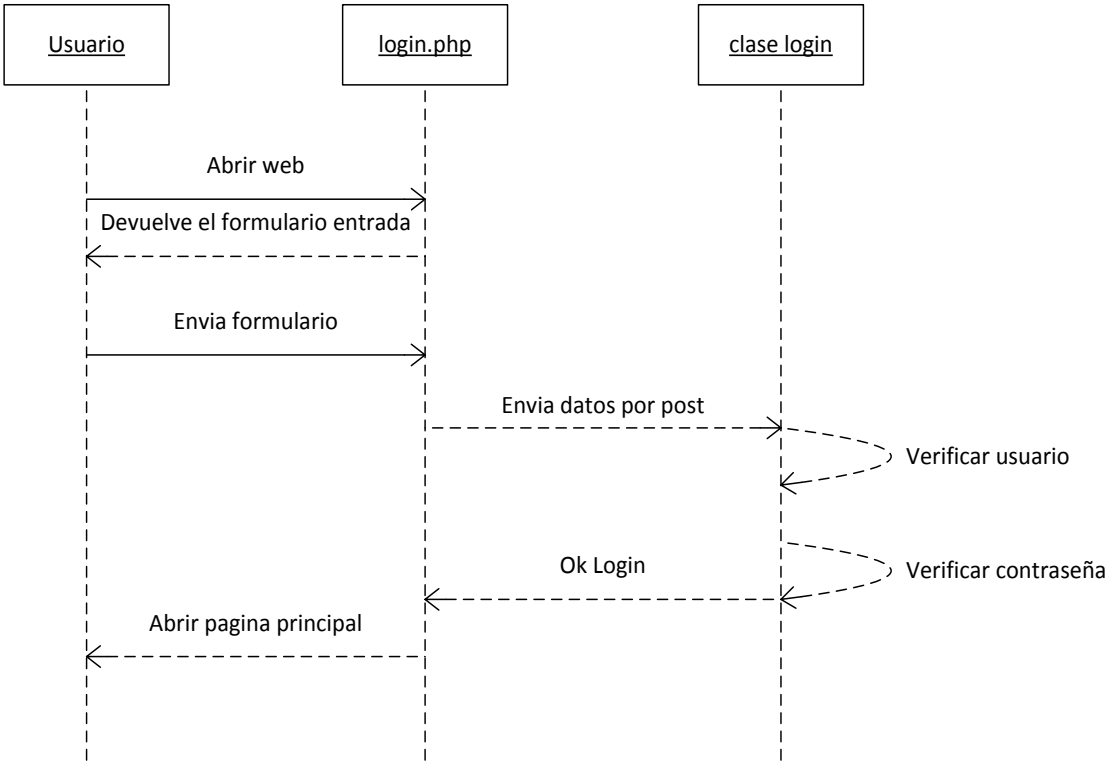
public isAdmin
Return:
Método para saber si el usuario es administrador o no

public logout
Return:
Método para cerrar la sesión en la plataforma

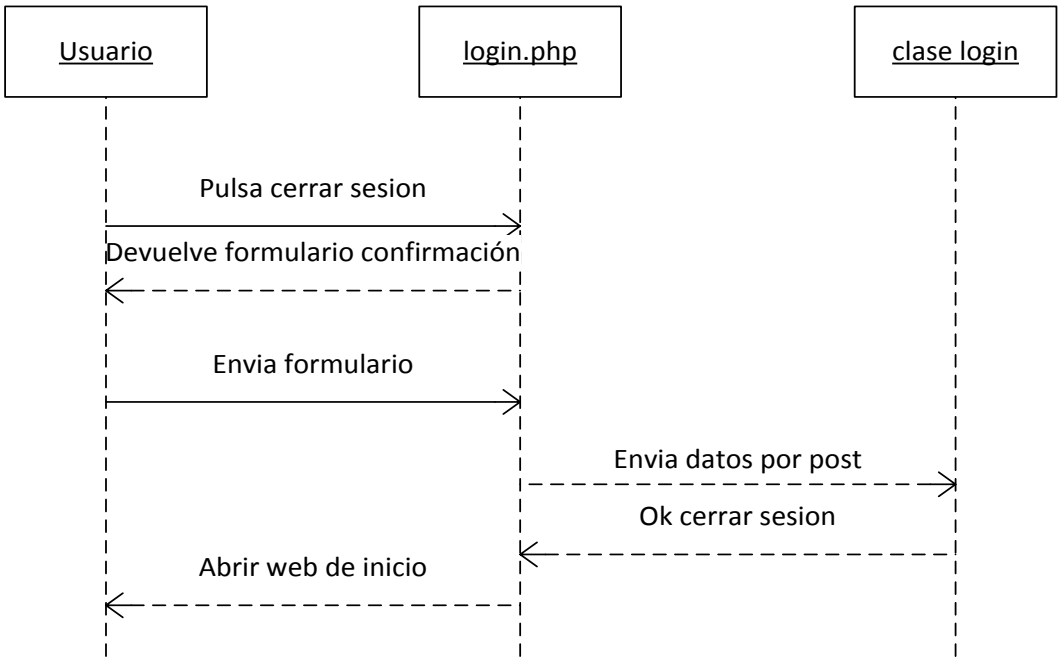
5.3.6 Diagrama de secuencias

En este apartado se explicara como interactúa cada caso de uso con las diferentes clases. Los diagramas de secuencias se han puesto por orden de aparición en la aplicación. Algunos casos de uso no tienen un diagrama de secuencia porque son solo una instrucción dentro de la aplicación, como por ejemplo envio mensaje. También se han metido los diagramas de secuencias de los elementos más importantes para este proyecto.

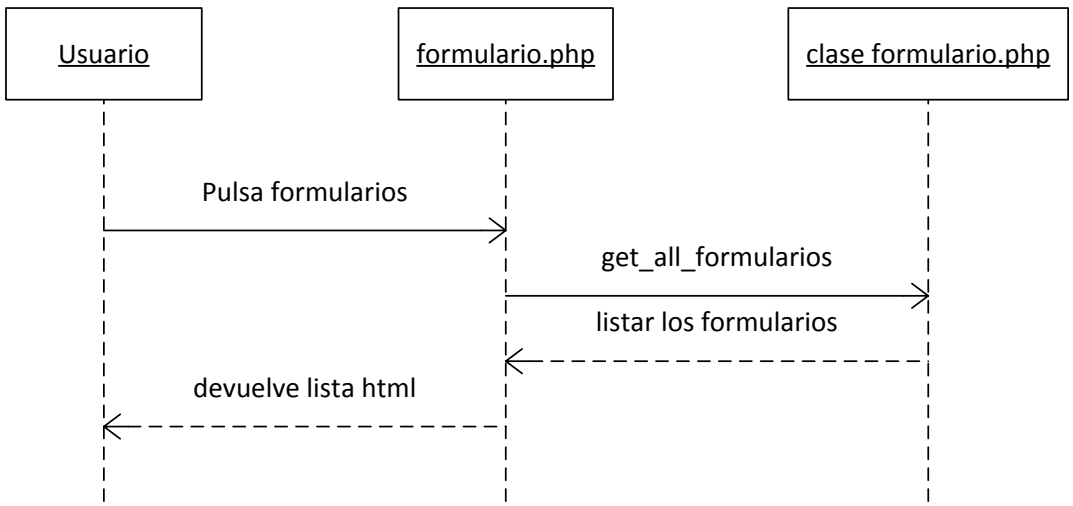
5.3.6.1 Login



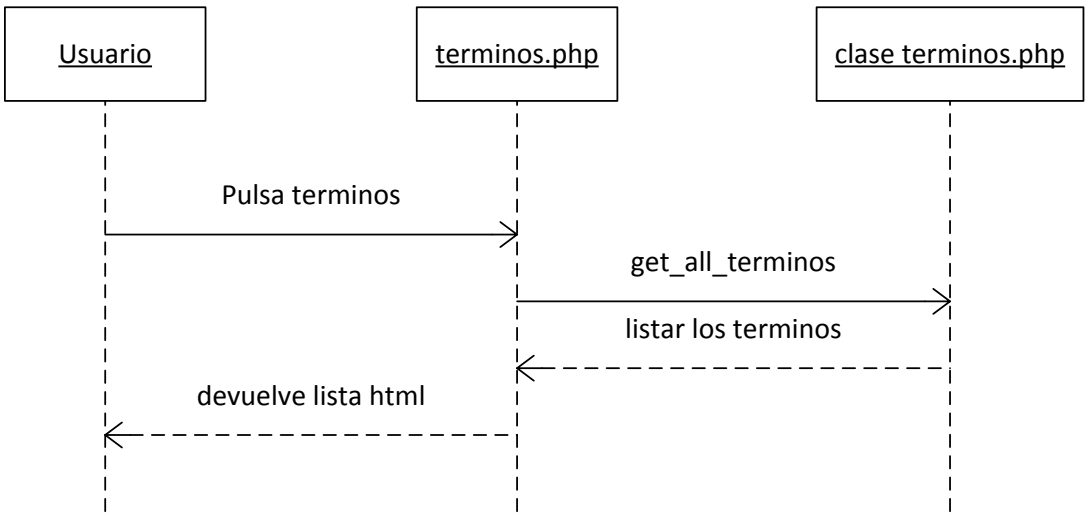
5.3.6.2 Cerrar sesión



5.3.6.3 Gestión de formularios

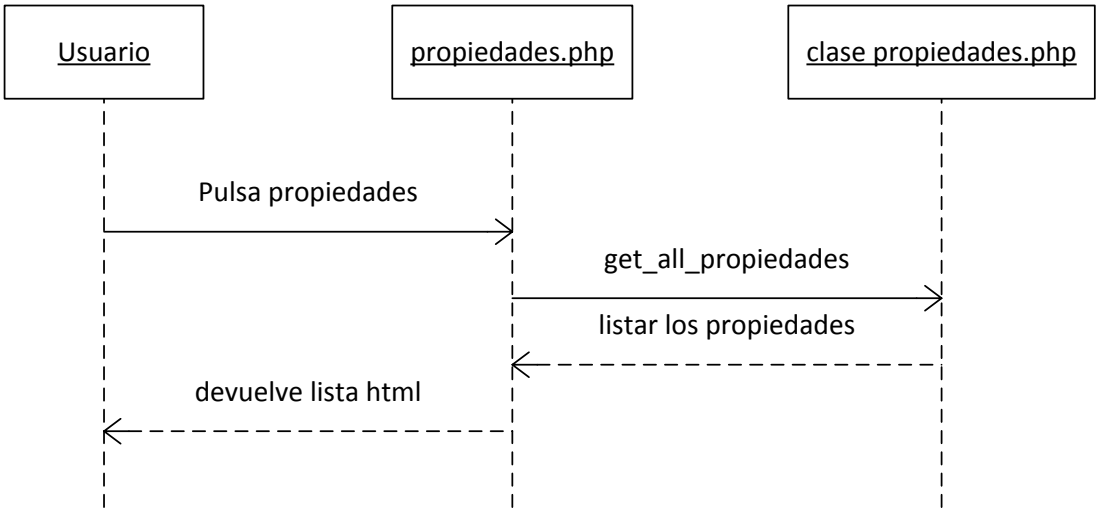


5.3.6.4 Gestión de terminos



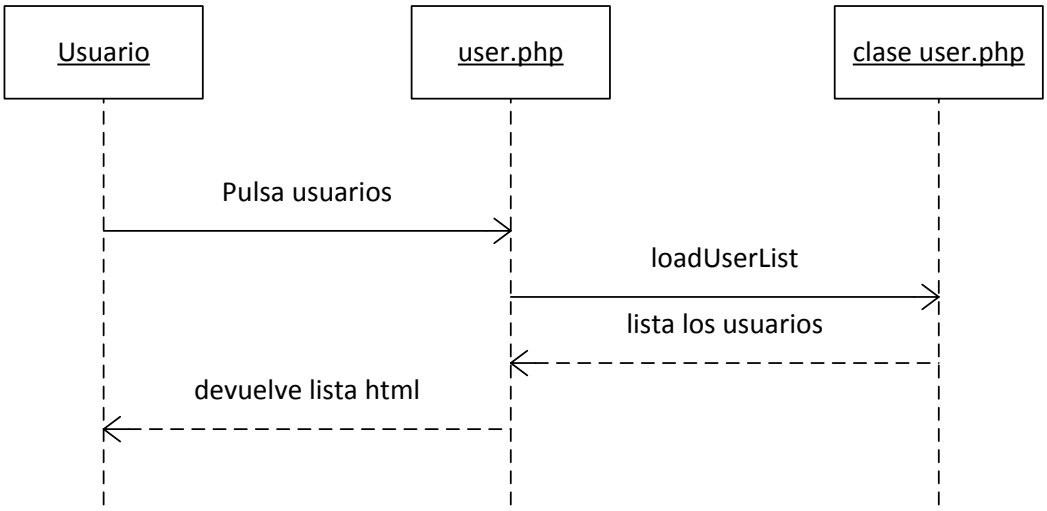
.

5.3.6.5 Gestión de propiedades



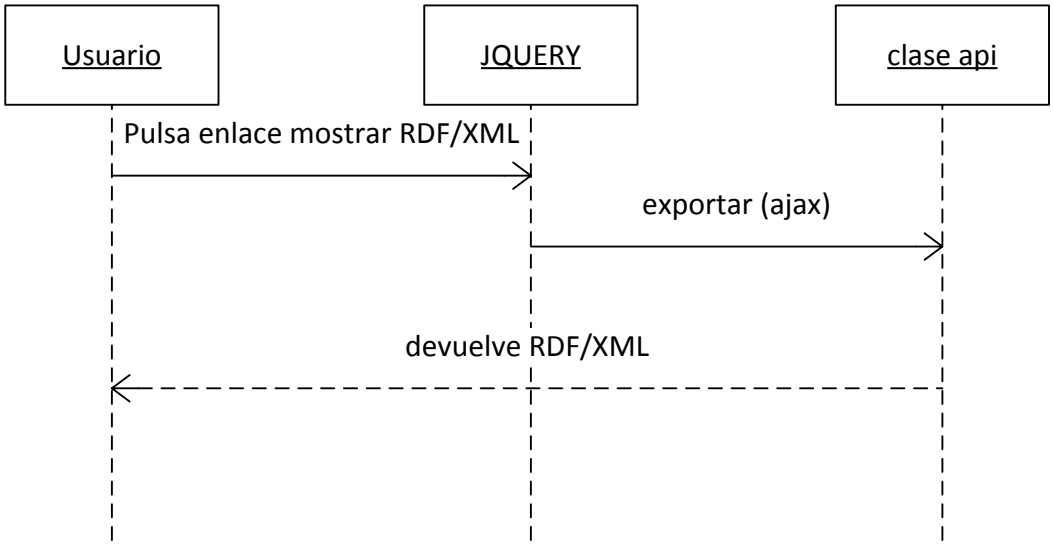
.

5.3.6.7 Gestión de usuarios



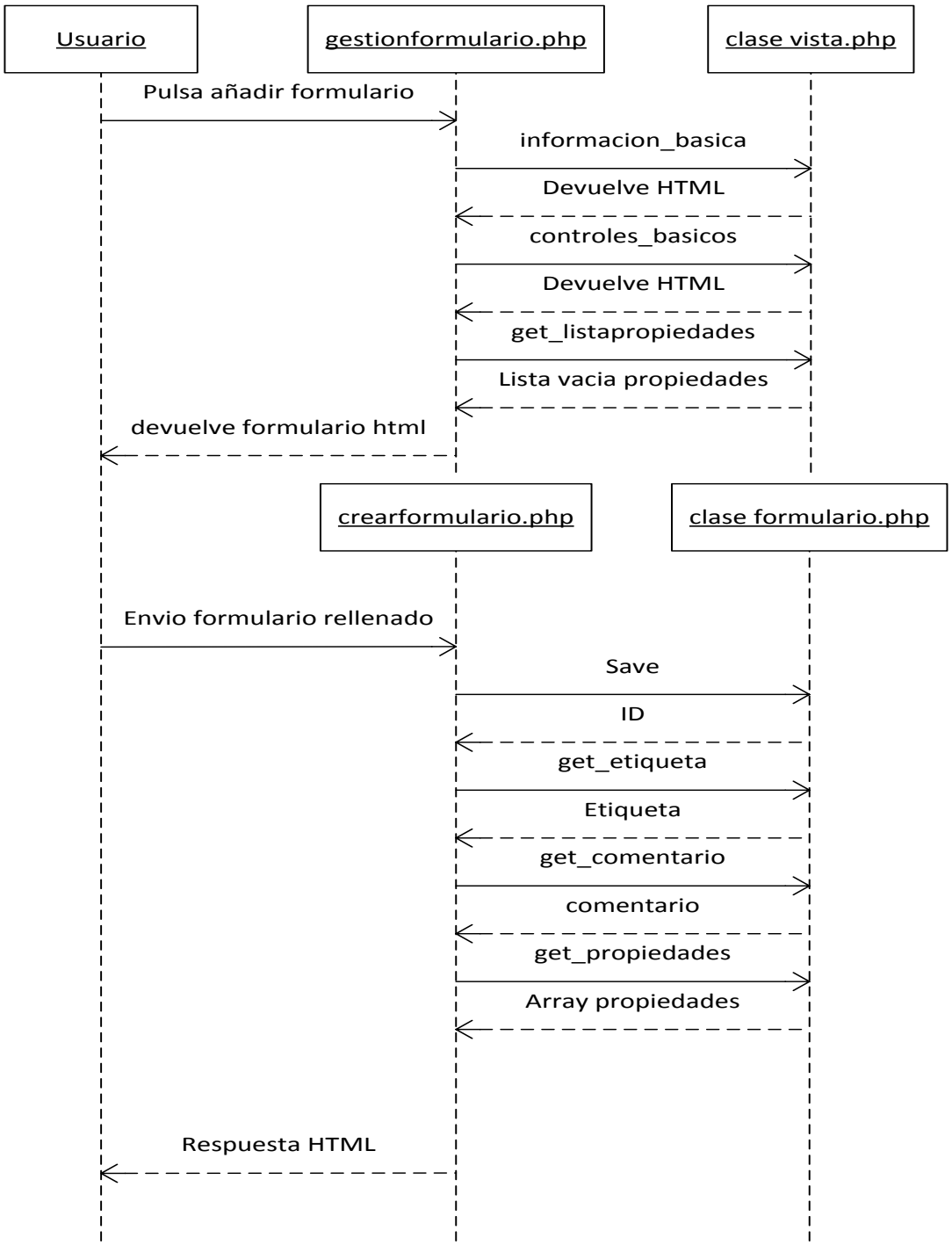
.

5.3.6.8 Mostrar RDF/XML

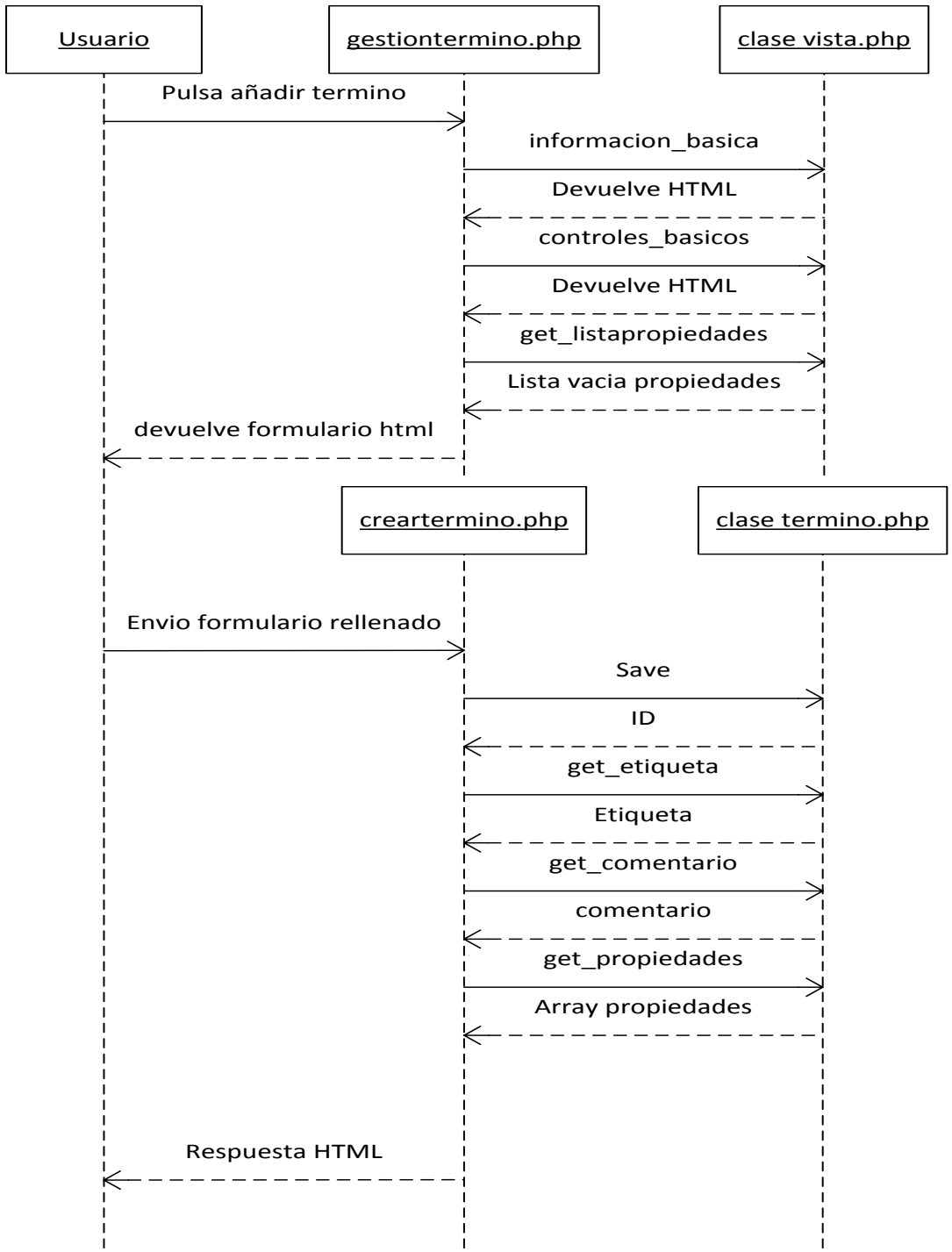


.

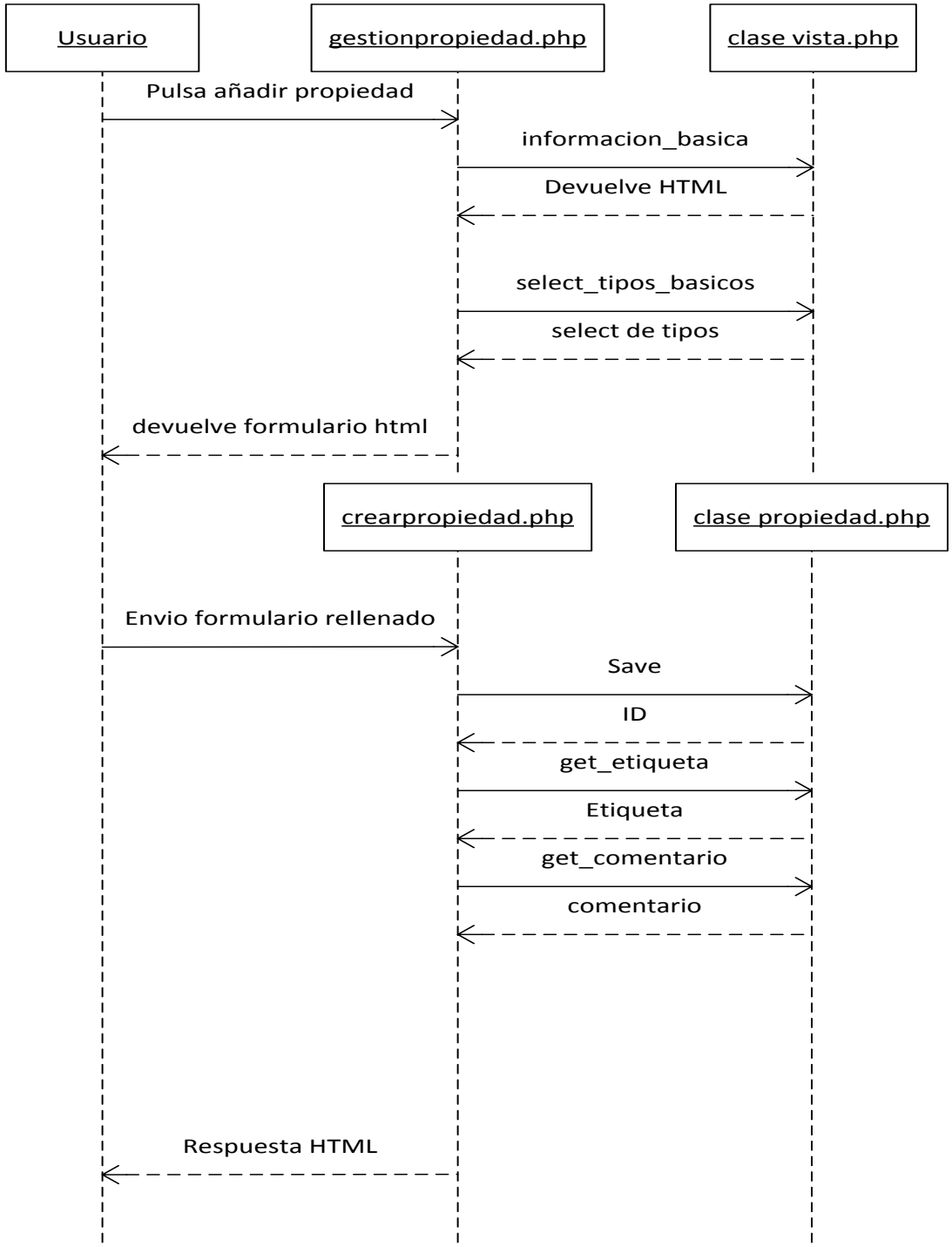
5.3.6.9 Agregar formulario



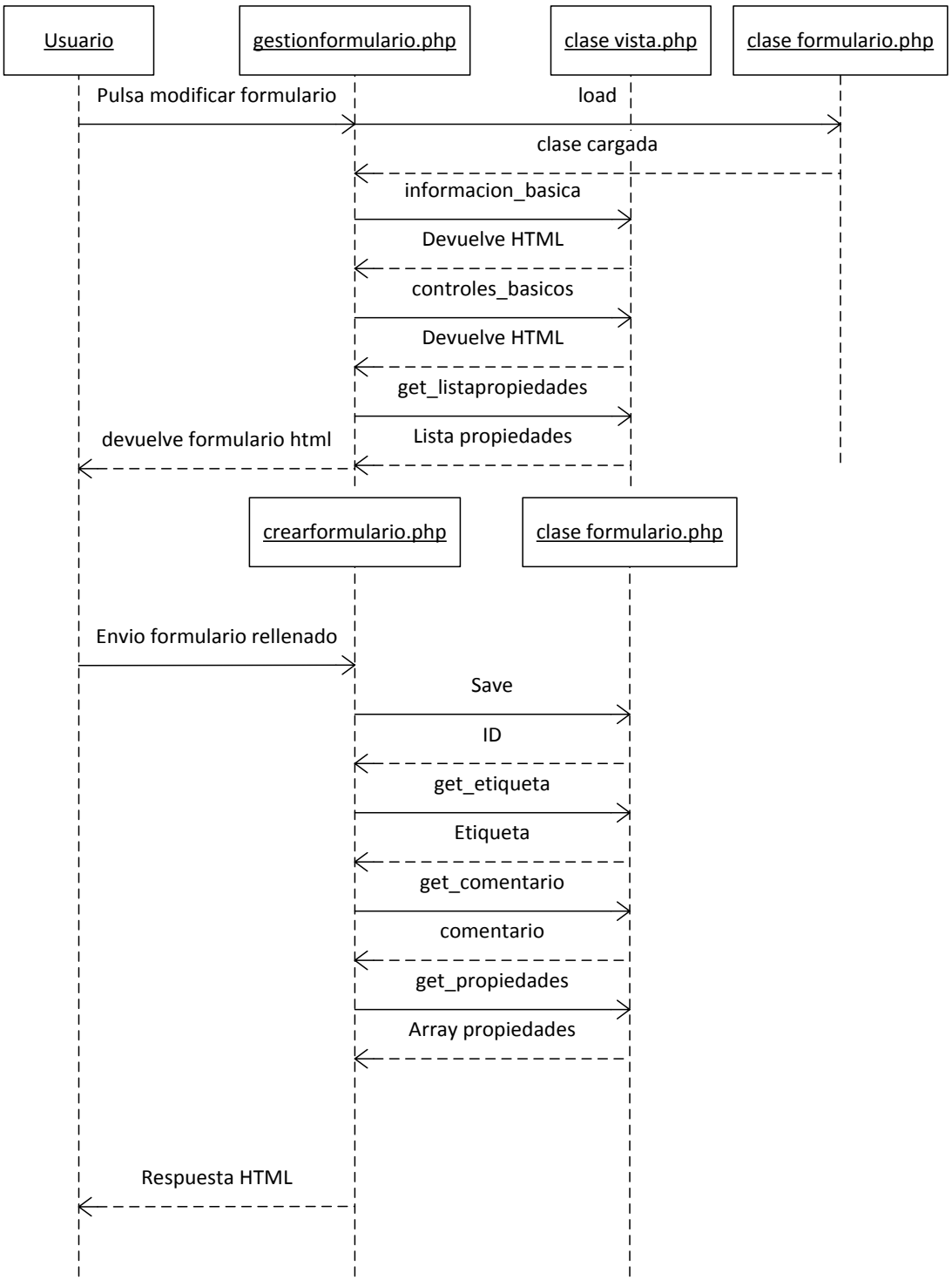
5.3.6.10 Agregar termino



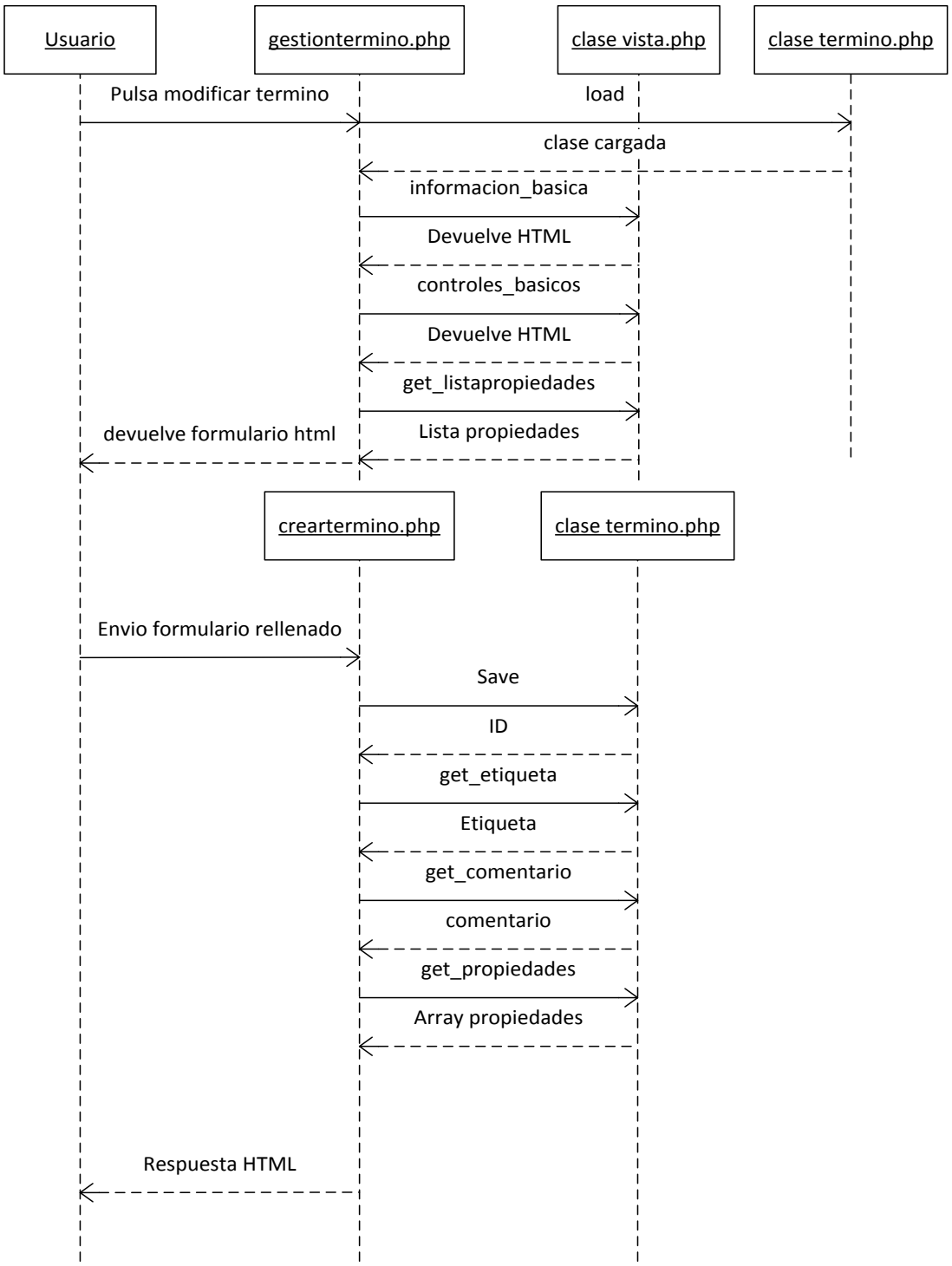
5.3.6.11 Agregar propiedad



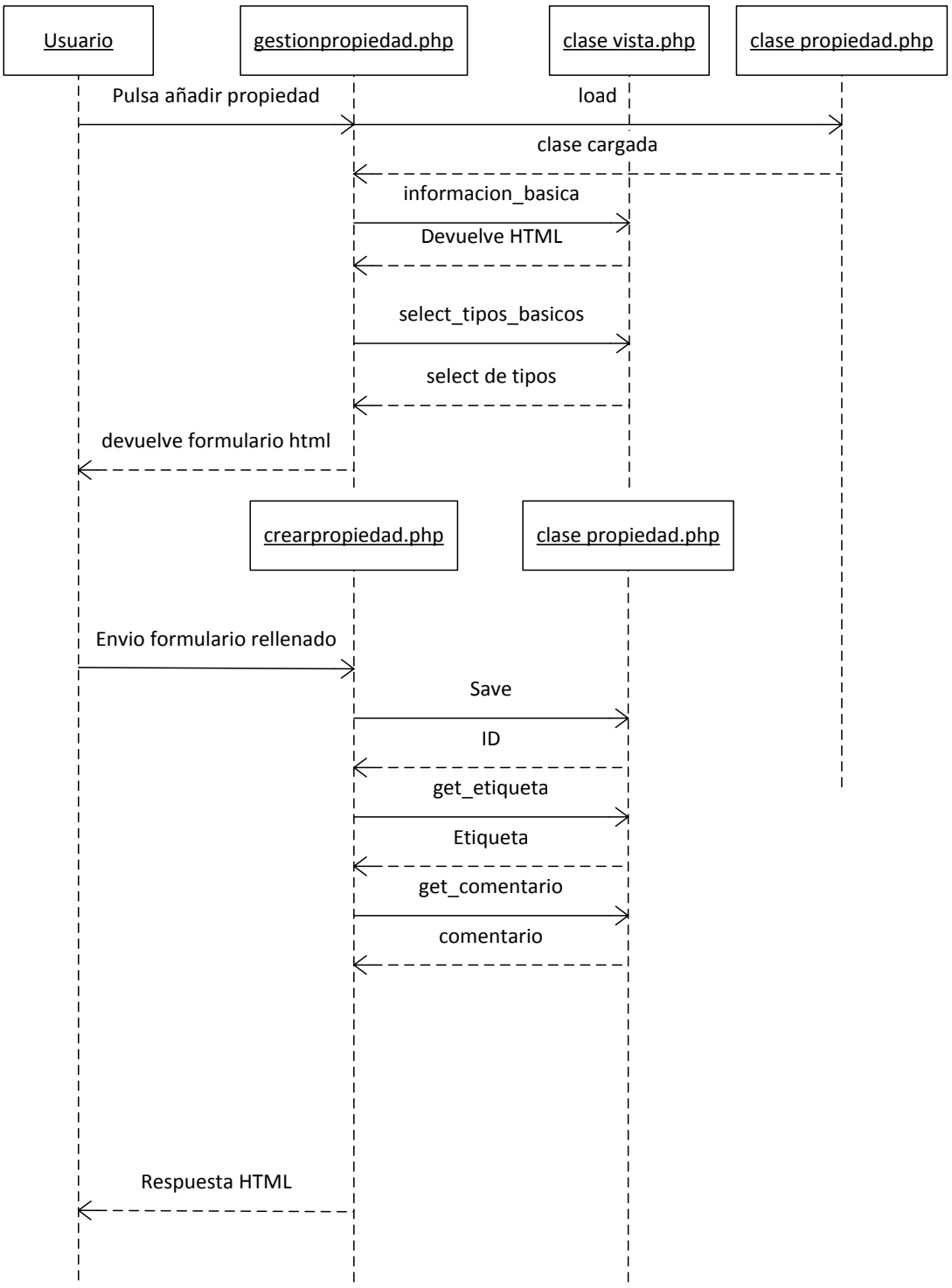
5.3.6.12 Modificar formulario



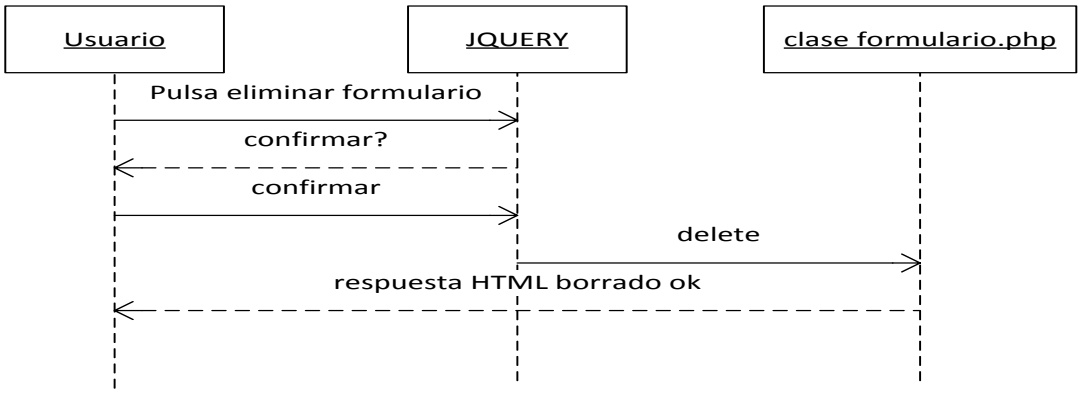
5.3.6.13 Modificar termino



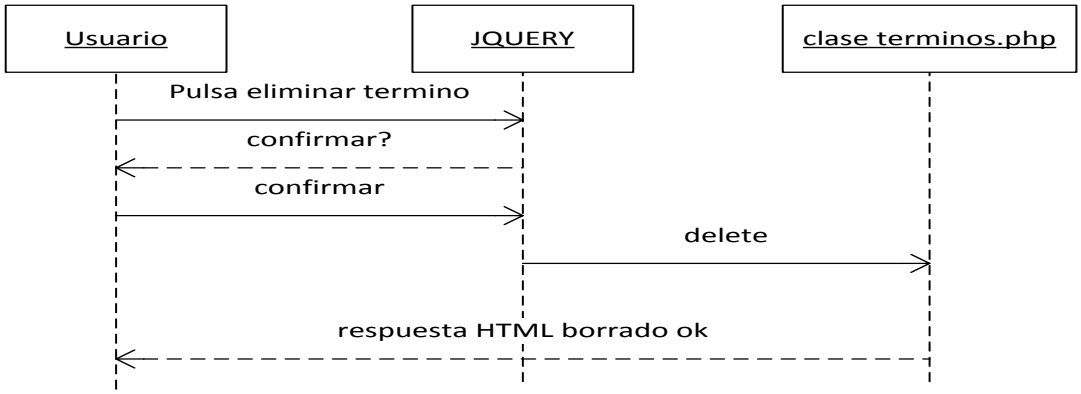
5.3.6.14 Modificar propiedad



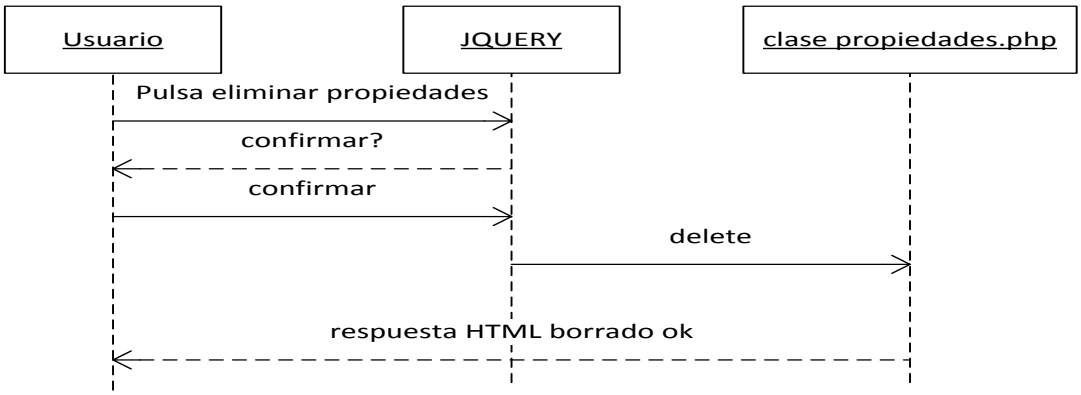
5.3.6.15 Eliminar formulario



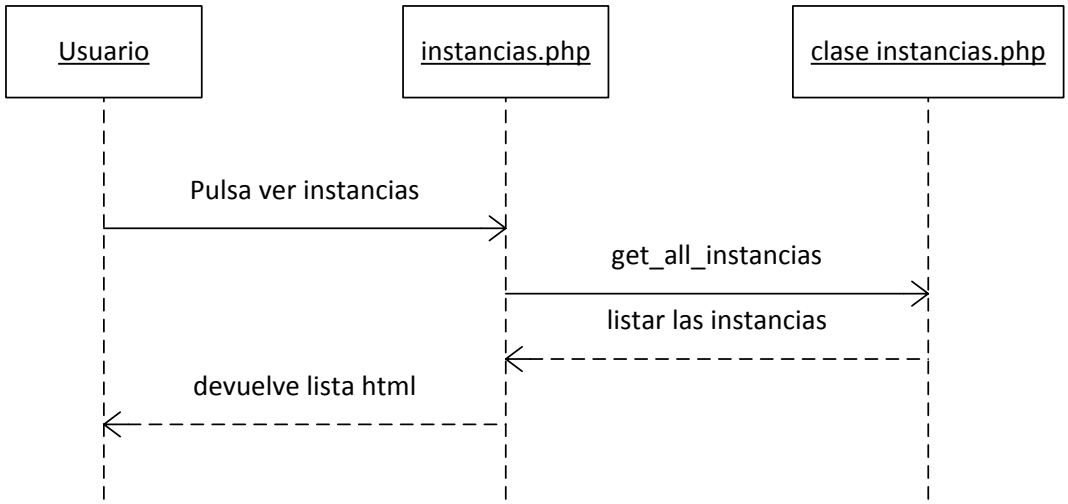
5.3.6.16 Eliminar termino



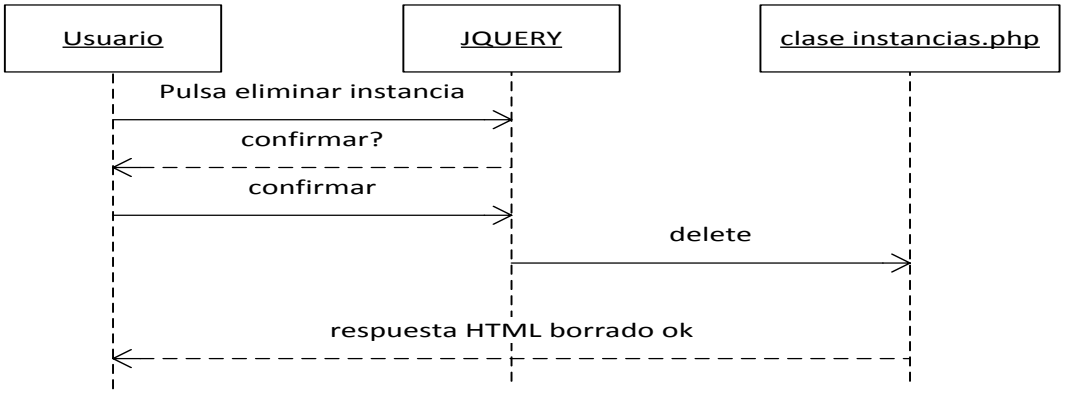
5.3.6.17 Eliminar propiedad



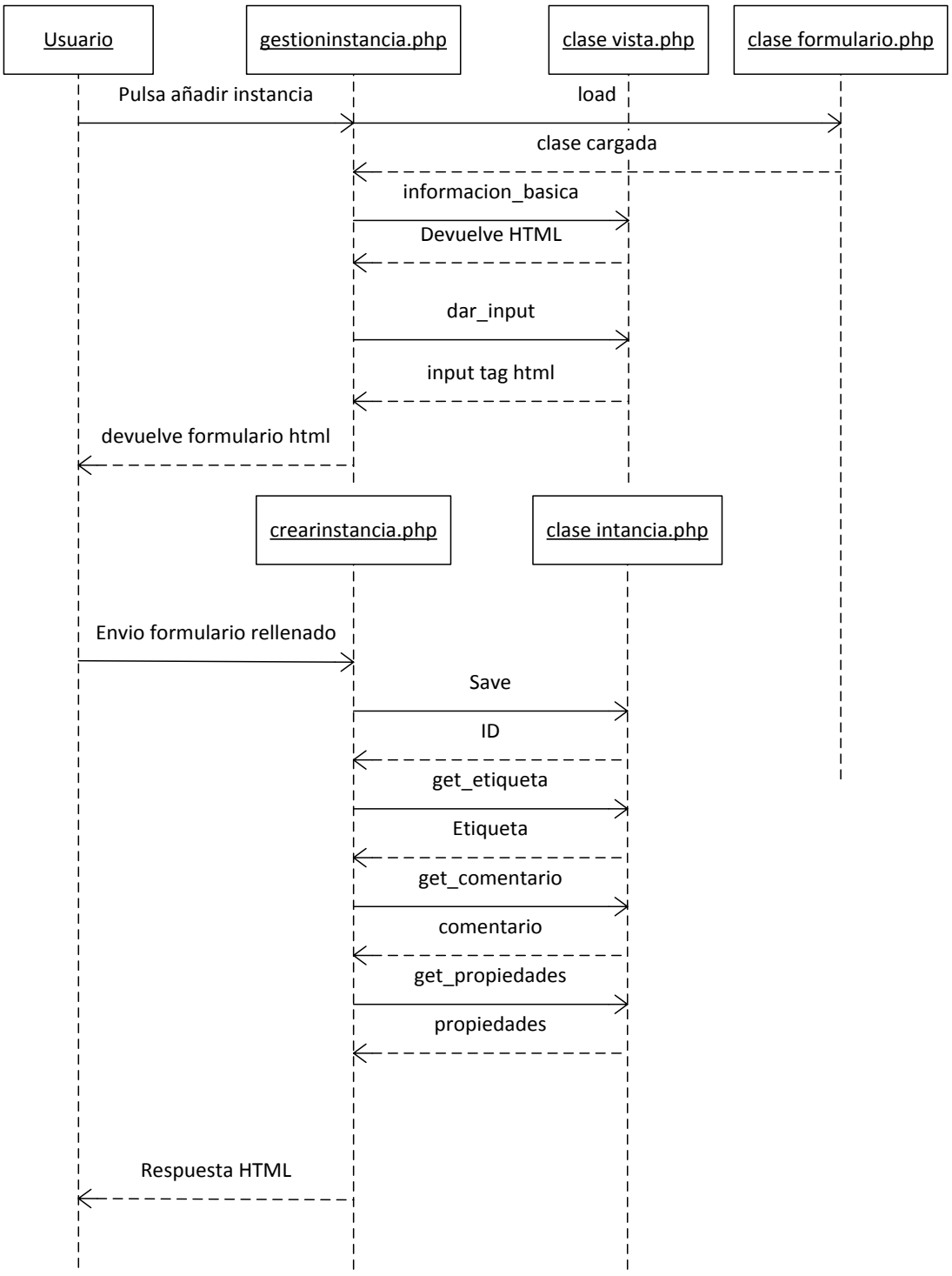
5.3.6.18 Gestión de instancias



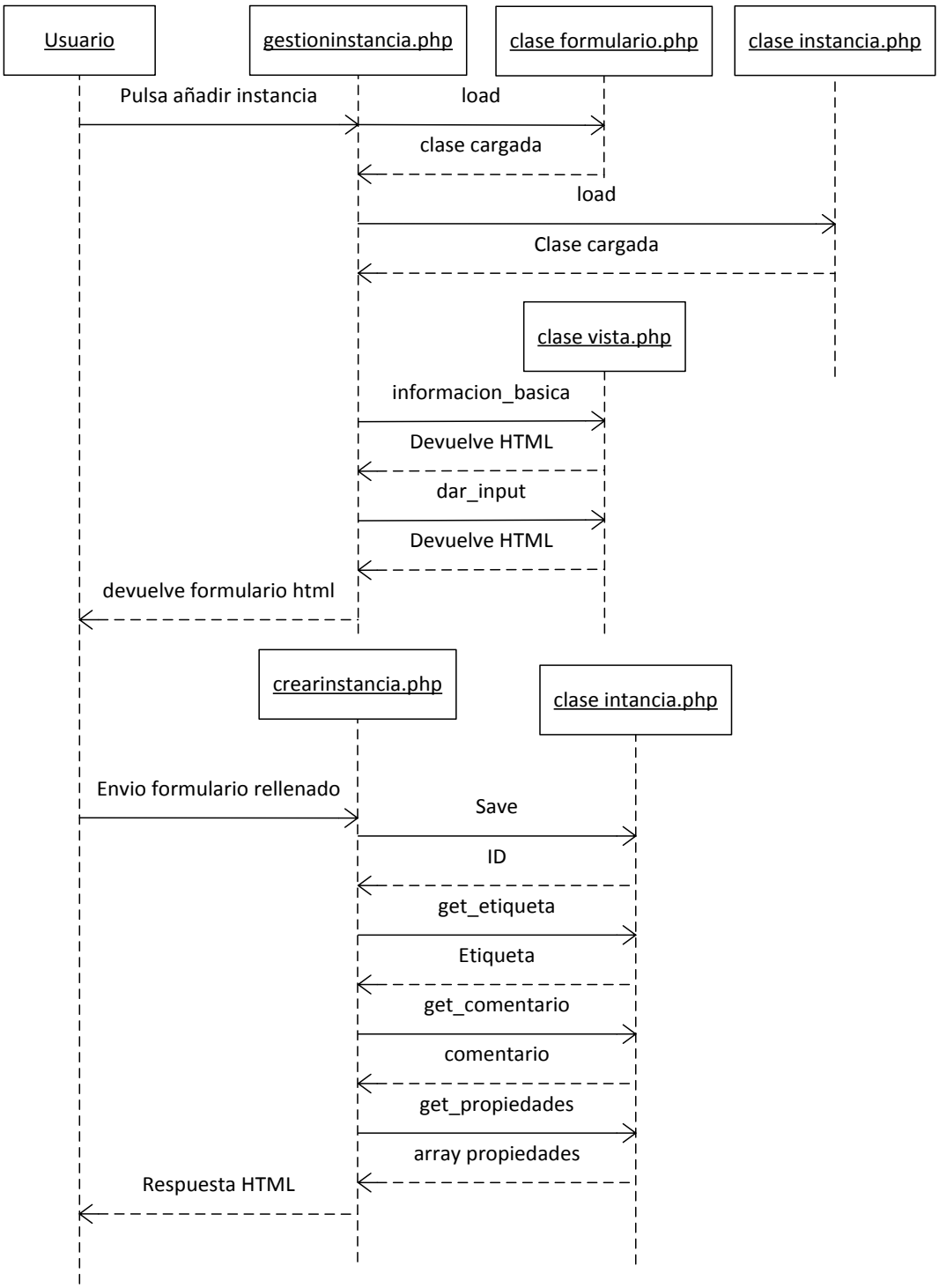
5.3.6.19 Eliminar instancia



5.3.6.20 Agregar instancia



5.3.6.21 Modificar instancia



5.3.7 Pruebas y test

En este apartado se explicara las diferentes pruebas de robustez que se han hecho a la aplicación web del proyecto Todos los las pruebas se han hecho primeramente en el sistema local WAMP y cuando se ha comprobado que el funcionamiento del sistema era robusto se ha probado lo miso en el servidor LAMP exterior, como se ha dicho varias veces durante el proyecto.

Los primeros test fue el intentar hacer los datos RDF en simples ficheros de texto, sin embargo cuando se observó que cuando se introducía un modelo de datos en RDF muy extenso PHP tardaba tiempo, por lo que después de las primeras pruebas se optó por realizar tablas en MySql para agilizar la velocidad sobre todo al hacer las listas de los diferentes elementos del proyecto

Durante el proyecto no ha habido muchos errores a nivel de programación porque al estar probando cada módulo individualmente se podían corregir casi de inmediato los problemas.

Sin embargo al utilizar usuarios sin muchos conocimientos de informática, para saber si podía gestionar o no la aplicación que era uno de los puntos que queríamos hacer en el proyecto, hubo un gran problema. Ya que la primera versión de esta aplicación web no era muy intuitiva, por lo que los usuarios se perdían dentro de la aplicación. Los usuarios que se usaron de test, no encontraban las formas de cómo gestionar los diferentes elementos, aunque si entendían la definición de los mismos, porque entendían términos como tag de búsqueda de internet. Tras el nuevo diseño sí que los usuarios no daban tantas vueltas en la aplicación. Todas estas pruebas se hicieron a nivel de back-end.

Esto hizo que se replanteara totalmente el diseño con las diferentes páginas hasta llegar a las páginas actuales, dando como resultado la doble opción de gestionar tanto por iconos como por botones al realizar una selección de los diferentes elementos.

Otro problema con el que nos encontramos es saber si al eliminar deberíamos tener integridad referencial o no en los datos, sin embargo al estudiar RDF nos dimos cuenta que no hacía falta tener tanta integridad, porque RDF soportaba que pudiera faltar un elemento. Esto es debido a que en RDF se tiene más en cuenta la semántica de los datos que los propios datos, es decir si un elemento intenta acceder a un elemento que no existe únicamente tendría que tenerlo en cuenta pero no dar un error.

Se intentó hacer la importación de documentos de texto en RDF para que los guardara en el sistema, pero se tuvo que desistir por falta de tiempo, y porque resultaba bastante complicado y un poco fuera del alcance del proyecto actual, debido que lo que verdaderamente interesaba era la exportación para las diferentes aplicaciones.

6. CONCLUSIONES

En este último apartado se expondrán tanto la concordancia de objetivos, como las conclusiones finales y líneas de futuro.

6.1 Concordancia de objetivos

Tras la realización de todo el proyecto, se ha llegado a los objetivos, sin embargo el objetivo de documentar todo el proyecto, ha quedado un poco más pobre que lo que verdaderamente se podría hablar por falta de tiempo. A nivel de programación y diseño sí que se han cumplido con los objetivos.

6.2 Líneas de futuro

Las líneas de futuro que tiene este proyecto son muy amplias debido a que RDF es enormemente ampliable y escalable. Se me ocurren varias ideas a nivel de futuro:

- Importación al sistema por medio de un rdf en texto. Una de las cosas que se hubiera querido hacer pero por falta de tiempo no se ha podido
- Se han usado solo ciertos tipos básicos de datos, pero en rdf se podría hacer que la función dar_input de la clase vista, pudiera dar más que solo esos tipos básicos. Sobre todo con aquellas ontologías que ya están expuestas en W3C.
- Asignar a cada usuario su propio entorno de trabajo, o lo que se llama segmentar el programa para diferentes usuarios. En el proyecto actual el usuario no tiene asignado sus propios formularios o términos, debido a que solamente se usan los usuarios para entrar al sistema. Sin embargo podrían darse privilegios a cada usuario, asignando privilegios lectura o escritura a cada elemento del proyecto.

6. BIBLIOGRAFIA

Toda la bibliografía se ha buscado a través de internet en los siguientes enlaces:

Página web de W3C : http://www.w3.org/

Página web de Wikipedia: https://es.wikipedia.org/

Página oficial de PHP: https://secure.php.net/manual/es/index.php

Página oficial de MySql: https://www.mysql.com/

Página oficial de Apache: http://httpd.apache.org/

Foro oficial de WAMP: http://forum.wampserver.com/list.php?2