UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA
UNAN-LEON
FACULTAD DE CIENCIAS
DEPARTAMENTO DE COMPUTACION

INDEXADOR SEMANTICO DE DOCUMENTOS DIGITALES (PDF, DOC, MP3, EXIF, AVI, XLS, WMA, PPT)

Maestría en Computación con énfasis en Gestión de Información

2006-2008

Autor:

Raúl Dávila Menis

Director de Tesis:

Dr. David Fernández Barrero.

Julio 2008

RESUMEN

La rápida evolución de Internet junto a las necesidades cambiantes y cada día más exigentes por parte de los usuarios, ha hecho que las representaciones en que se basa la Web actual no sean suficientes para ayudar a evitar gran parte del trabajo manual, tedioso y difícil que realizan dichos usuarios cuando navegan e interactúan con Internet.

Es indudable que las ventajas que ofrece Internet a la hora de buscar información son inigualables, pero adolece de un mecanismo para encontrar información de forma precisa además que las máquinas puedan hacer algún tipo de procesamiento semi-automático de la misma.

Es con el objetivo de aliviar este problema que aparece la denominada "Web Semántica" como una extensión de la Web actual que pretende dotar de significado explícito la información disponible en Internet de tal manera que sea procesable por máquinas. Este procesamiento avanzado de información no es tan sencillo de lograr y se aboga por la utilización de metadatos para la descripción explícita de estos recursos lo cual entre otras cosas vendrá a mejorar los procesos de recuperación de información.

El formato de especificación de estos metadatos debe ser tan general, o estándar, como sea posible para que otras aplicaciones o personas sean capaces de entenderlos. Es aquí donde las especificaciones del W3C (XML, RDF, RDFS y OWL) proporcionan herramientas prácticas donde iniciativas como Dublín Core, para la especificación de distintos tipos de metadatos, transversales entre dominios de conocimiento son realmente útiles.

El presente trabajo rescata estas técnicas de la Web semántica y las utiliza para representar la metainformación extraída de los documentos digitales permitiendo de esta forma realizar búsquedas transparentes sobre los distintos formatos en que puedan estar dichos documentos.

1. INTRODUCCION

El presente escrito documenta el desarrollo de un indexador de documentos digitales incorporando tecnologías de la Web Semántica el cual pretende permitir algo más que las búsquedas sintácticas tradicionales sobre palabras claves que suelen hacer la mayoría de las aplicaciones informáticas para la búsqueda de información actualmente.

En este punto la Web semántica aboga por dotar de significado explícito la información a través de metadatos por ejemplo, y representar esta información de tal manera que las máquinas la puedan procesar; es aquí donde el World Wide Web Consortium (W3C) ha dedicado esfuerzos para la creación y promoción de varios estándares que permitan a las máquinas establecer relaciones y realizar inferencias acerca de entidades (significados) representados en la información compartida.

En los documentos digitales podemos encontrar una serie de campos que describen el documento mismo (metadato), los cuales varían de acuerdo al formato del documento, pero a pesar de esto se pueden encontrar en todos ellos algunos campos generales como nombre, tipo MIME del archivo, tamaño, etc. Al trasladar el espíritu de la Web Semántica a este escenario se puede ver claramente que esta metainformación es la que habla de forma explícita sobre los recursos (documentos digitales), así que será esta metainformación la que manipulemos y representemos utilizando técnicas de la Web Semántica.

Además de la base teórica sobre dichas técnicas las cuales se presentan en la siguiente sección, se exponen brevemente algunas herramientas similares al indexador que se desarrolló, de las cuales se tomó como base el "Indexador semántico de paquetes Unix" sobre el cual se hace un resumen de las herramientas que se utilizaron para su desarrollo, además se describe la forma en que éste ultimo opera, seguidamente se realiza una abstracción de los metadatos encontrados en los diferentes tipos de documentos a indexar, para luego exponer un poco las herramientas que se usaron en este indexador como tal.

En cuanto al diseño e Implementación de la herramienta; se inicia con la elaboración de la ontología sobre documentos digitales donde se modela la jerarquía de conceptos bajo la cual se representará la información extraída de los documentos digitales. Una vez hecho esto, se procede a comentar el funcionamiento de los módulos modificados y/o desarrollados para luego cerrar la sección con la presentación de un ejemplo de indexación y búsqueda sobre dichos documentos.

Para concluir el documento se presentan las conclusiones a las que se ha llegado después de la experiencia con el mismo, además de algunas recomendaciones en Pro del mejoramiento de la herramienta.

2. MARCO TEORICO

2.1 Visión General de la Web

La Web actual es un medio extraordinariamente flexible y económico para la comunicación, el comercio, los negocios, ocio, entretenimiento, acceso a información y muchos otros servicios, pero existe una característica muy importante en dicha Web y es que todo lo que existe en ella está orientado al consumo humano, lo cual agudiza el problema sobre la incapacidad de las representaciones en que se basa la Web actual para expresar significados.

2.2 ¿Qué es la Web Semántica?

Según el inventor de la Web de hoy Tim Berners-Lee, La Web semántica[1] [1] propone superar las limitaciones de la Web actual mediante la introducción de descripciones explícitas del significado (metadatos[2]), la estructura interna y global de los contenidos además de los servicios disponibles en la WWW; lo cual significa que los datos serán "definidos y enlazados en una forma en que puedan ser utilizados por máquinas, no sólo con el propósito de mostrarlos, sino para automatización, integración y reutilización de los datos a través de varias aplicaciones" [2].

La Web Semántica proporciona un marco común de trabajo que permite compartir y reutilizar datos a través de diferentes aplicaciones, organizaciones y comunidades. Es un esfuerzo colaborativo liderado por el consorcio de la World Wide Web (World Wide Web Consortium, W3C) con la participación de un gran número de investigadores y socios industriales.

La Web semántica mantiene los principios que han hecho un éxito de la Web actual, como es la descentralización, compartición, compatibilidad, máxima facilidad de acceso y

[1] http://www.semanticweb.org/
[2] Metadatos: Datos estructurados que describen las características de un recurso de información.

contribución. En este contexto un problema clave es alcanzar un entendimiento entre las partes que han de intervenir en la construcción y explotación de la Web: usuarios, desarrolladores y programas de muy diverso perfil. Para cumplir este objetivo, la Web semántica rescata del campo de la inteligencia artificial la noción de Ontología la cual utiliza como vehículo para lograr dicho objetivo.

2.3 Ontologías

El termino ontología proviene de la filosofía, pero en inteligencia artificial tiene diferentes connotaciones. La definición declarativa mas consolidada es la propuesta por [3] el cual afirma que es "una especificación explicita y formal sobre una conceptualización compartida".

Una ontología es una taxonomía de conceptos con atributos y relaciones que proporciona un vocabulario para definir redes semánticas de unidades de información interrelacionadas. Concretamente, estará formada por una taxonomía relacional de conceptos y por un conjunto de axiomas o reglas de inferencia mediante los cuales se podrá inferir nuevo conocimiento.

Siendo las ontologías un pilar fundamental de la Web semántica, en los últimos años se han desarrollado diversos lenguajes y estándares para la definición de las mismas, también se han puesto en marcha las tecnologías que permiten a las computadoras realizar un manejo de la información enfocada en su significado.

2.3.1 Dublín Core

El estándar de metainformación Dublín Core (DC) es un conjunto de términos encaminados a describir recursos, de los cuales se define una semántica independientemente de la sintaxis y del medio final en el que se expresen dichos términos. Dicha semántica ha sido establecida por un equipo internacional y multidisciplinar de profesionales en muy diversas materias. El resultado es un estándar en la metainformación sobre recursos muy general, flexible y fácilmente utilizable en muy diversos ámbitos.

Dublín Core está dividido en simple y cualificado, el primero es un conjunto de quince términos que define la semántica básica utilizada por DC y que cubre la mayoría de los aspectos relevantes de un recurso, en cambio DC cualificado extiende DC simple el cual agrega un nuevo grupo de cualificadotes que restringen la semántica de los términos elementales de DC simple logrando de esta manera aportar información más precisa respecto a un recurso.

Principios Básico de Dublín Core

El principal objetivo de Dublín Core es su tamaño y simplicidad, para que indistintamente del área donde se utilizaran se pudiera utilizar sin mayores problemas. Por otra parte se quiso hacer de DC un estándar general, que abarcara una amplia gama de intereses, tanto de un usuario de Internet como de un bibliotecario o un músico, así que fue este requisito el que obligó a buscar una semántica universal en la que pudieran ponerse de acuerdo individuos de muy diversa formación e intereses. Con este esfuerzo de generalización se ha conseguido que los términos definidos en DC tengan sentido en un amplio rango de disciplinas diferentes, y por lo tanto pueda ser utilizado en una diversidad de ámbitos considerable.

Elementos del Dublín Core

La base de DC se encuentra en los quince elementos que define DC simple, y que proporciona la semántica básica para describir un recurso. En la especificación formal de estos términos [4] además de listar los términos con sus respectivas descripciones, se proporcionan elementos complementarios sobre dichos términos, como una URI asociada, una etiqueta y toda una serie de metainformación que complementa la definición básica.

Contenido	**Propiedad Intelectual**	**Instancia**
Coverage	Contributor	Date
Description	Creador	Format
Type	Publisher	Identifier
Relation	Rights	Language
Source		
Subject		
Title		

Cuadro 2.1: Clasificación de los elementos del Dublín Core.

En el cuadro 2.1 se presenta un resumen de los 15 términos del DC agrupados por tipo de atributo del recurso al que describen, los cuales se han clasificado de la siguiente manera, contenido, propiedad intelectual e instanciación.

2.3.2 Pasos para la elaboración de una Ontología.

1. Determinación del dominio y alcance de la ontología.

Cabe señalar que cuando se va a elaborar una ontología no tiene por qué cubrir toda la información del dominio, su generalización o especialización dependerá de las necesidades del entorno donde va a ser utilizada. En el caso que nos ataña, se pretende definir una ontología que aborde la información sobre documentos digitales con distintos formatos relevante para el caso de estudio.

2. Considerar la reutilización de ontologías existentes.

Casi siempre vale la pena considerar lo que otra persona ha hecho y verificar si podemos refinar y extender recursos existentes para nuestro dominio y tarea particular. Reusar ontologías existentes puede ser un requerimiento si nuestro sistema necesita interactuar con otras aplicaciones que ya se han dedicado a ontologías particulares o vocabularios controlados. Muchas ontologías ya están disponibles en forma electrónica y pueden ser importadas dentro de un entorno de desarrollo de ontologías que se esté usando.

3. Enumerar términos importantes para la ontología.

En este punto es importante enumerar todos los posibles términos que necesitaría la ontología que estemos elaborando para hacer enunciados o responder preguntas de un usuario, todo esto en el dominio y alcance para lo cual fue desarrollada.

Cabe mencionar que los siguientes dos pasos (desarrollo de la jerarquía de clases y definiendo las propiedades de los conceptos (slots)) están estrechamente relacionados. Es difícil hacer primero uno de ellos y luego hacer el otro.

Nomenclatura para Nombres utilizados en la ontología

Llegada la hora de definir las clases y las propiedades, se debe establecer una nomenclatura sobre los nombres que serán utilizados para hacer la ontología más fácil de entender.

La convención de nombres adoptada para el desarrollo de nuestra ontología es la siguiente:

- Se distinguirán mayúsculas y minúsculas.
- No se utilizarán caracteres especiales en los nombres, como espacios, comas o asteriscos.

- El nombre de una clase representa una colección de objetos, y como tal puede ser singular o plural. En nuestra ontología todos los nombres de clase serán en singular.
- Los nombres de las clases comenzarán con mayúscula.
- Los nombres de las propiedades comenzarán con minúsculas.
- En caso que el nombre contenga mas de una palabra, será utilizado el carácter "_" para separarlos.
- Los nombres de conceptos no incluirán las palabras clase, class, propiedad, property, etc.
- Se evitarán las abreviaturas en la medida de lo posible.
- Los nombres de las subclases incluirán el nombre de su superclase.

4. Definir las clases y la jerarquía de clases.

A la hora de definir las clases y organizarlas en una jerarquía, existen tres aproximaciones:

- De arriba abajo (top-down): comenzando por los conceptos más generales.
- De abajo a arriba (bottom-up): comenzando por los conceptos mas específicos.
- Mixta: definiendo los conceptos más destacables y después generalizándolos o especificándolos.

Cualquiera que sea la elección, los pasos a seguir se describen a continuación:

1. Definir las clases: elegir de nuestra lista de términos aquellos que representan objetos con existencia independiente, y descartando aquellos que los describen.

2. Organizar las clases en una taxonomía jerárquica: definir qué clases son subclases de otras.

Algo muy importante que se plantea a la hora de definir las clases es cuándo un concepto debe ser considerado una clase, y cuando una instancia. En el caso de los conceptos que formen una jerarquía natural, deberán ser representados como clases.

Las instancias no se pueden organizar en una jerarquía, pues son los conceptos más específicos que pueden representarse en una base de conocimiento.

Además, se debe tomar en cuenta que una clase a la que pertenecen las instancias individuales no debe cambiar muy a menudo, pues haría la ontología difícilmente mantenible.

Es conveniente señalar que son las clases y no las palabras, las que representan conceptos. Esto quiere decir dos cosas:

- El nombre elegido para una clase puede cambiar si cambia la terminología, pero la clase seguirá representando una realidad objetiva.

- Sinónimos de un mismo concepto no representan clases diferentes.

Seguidamente, se debe definir la jerarquía que deben tener los conceptos (clases), en este punto debemos tomar en cuenta lo siguiente:

- Relación “es un tipo de”: una subclase de una clase representa un concepto que es “un tipo de” el concepto que representa la superclase.

- Las relaciones jerárquicas son transitivas: si B es subclase de A y C es subclase de B, entonces C es subclase de A, Aquí se diría que B es subclase directa de A, y C subclase directa de B.

- No pueden existir ciclos de clases: Si A es superclase de B y B es superclase de A, entonces A y B son la misma clase.

- Clases hermanas en la jerarquía: las clases que son subclases directas de una misma clase se llaman hermanas (siblings). Todas las clases hermanas en la jerarquía (excepto las clases raíz) deben representar conceptos con el mismo nivel de generalidad.

- Número de subclases directas de una clase: una clase no debe tener una única subclase directa, ni tampoco demasiadas (por ejemplo, más de una docena). Si fuere esto último el caso, deberíamos plantear el hecho de introducir clases intermedias.

- Sobre políticas para añadir una subclase: generalmente las subclases presentan características como las siguientes:
  - Tienen propiedades que no tiene la superclase, o
  - Tiene restricciones diferentes de las de la superclase, o
  - Participan en relaciones distintas que la superclase.

  Aunque en ocasiones puede resultar útil definir una subclase sin que se cumpla ninguno de estos requisitos, simplemente porque representa la realidad de esa manera.

5. Definir propiedades de las clases.

En torno a este procedimiento, existen algunas consideraciones a tomar en cuenta para la asignación de las propiedades a las clases.

- Todas las subclases de una clase, heredan las propiedades de esa clase.

- Una propiedad debe asignarse a la clase más general que pueda tener dicha propiedad.

6. Definición de Restricciones de Propiedades.

Las propiedades pueden tener distintas restricciones como las que describen el tipo de valor, valores admitidos, el número de los valores (cardinalidad), entre otras características que las propiedades pueden tomar.

Cardinalidad

La cardinalidad es la que indica cuantos valores puede tener una determinada propiedad. Mientras algunos sistemas permiten definir la cardinalidad como simple o múltiple, otros permiten indicar el número de apariciones mínimo o máximo.

En el caso de las subclases la cardinalidad nos sirve para indicar si esta última hereda o no alguna propiedad de su superclase. En caso de no heredarse, esto se indicaría definiendo su cardinalidad máxima igual a 0.

Tipo de valor

Otra restricción que se pueden definir sobre las propiedades es el tipo de valor permitido, entre los valores más comunes que pueden adoptar las distintas propiedades.

- Cadena de caracteres (string)
- Fecha y Tiempo (datetime)
- Numero (number). Para enteros (integer) y reales (float)
- Enumerado(enumerated)
- Instancia (instance). Permite definir relaciones entre las instancias de dos clases.

Dominio y Rango

El dominio y rango de las propiedades son restricciones que se deben definir obligatoriamente.

El dominio representa el conjunto de clases que puede describir una determinada propiedad, mientras que el rango de la misma representa el conjunto de valores que puede tomar la propiedad.

En las propiedades cuyo valor es un tipo de datos (DatatypeProperty), el rango viene representado precisamente por ese tipo de datos, sin embargo en las propiedades cuyo valor es una instancia (propiedades de objeto), el rango es el conjunto de clases a las cuales puede pertenecer dicha instancia.

7. Crear Instancias

El último paso consiste en crear instancias individuales de clases en la jerarquía, esto se refiere a los valores que pueden tomar las distintas propiedades correspondientes a una clase.

2.3.3 Tecnologías para la Web Semántica.

Las tecnologías para hacer posible la Web semántica incluyen lenguajes para la representación de ontologías, parsers, lenguajes de consulta, entornos de desarrollo, módulos de gestión (almacenamiento, acceso, actualización) de ontologías, módulos de visualización, conversión de ontologías, etc.

El primer lenguaje para la construcción de la Web semántica fue SHOE[3], creado por Jim Hendler en la Universidad de Maryland en 1997. Desde entonces se han definido otros lenguajes y estándares con finalidad similar, como XML, RDF[4], DAML+OIL[5], y mas recientemente OWL[6], por citar los mas importantes.

[3] http://www.cs.umd.edu/projects/plus/SHOE/
[4] http://www.w3.org/RDF/
[5] http://www.w3.org/TR/daml+oil-reference/
[6] http://www.w3.org/2001/sw/WebOnt/

2.3.3.1 XML

En 1998, el eXtensible Markup Language (XML)[7], se convirtió en una recomendación del W3C para estructurar y compartir datos (describiéndolos). XML es una convención notacional que permite especificar lenguajes como HTML, pero que no está atada a ninguna semántica particular. No es un lenguaje completo, sino un medio para hacer lenguajes (metalenguaje), pues los usuarios pueden definir la semántica de las etiquetas. A través de los años, XML se ha convertido en un estándar de facto para el intercambio de información.

Es de hacer notar, que esta estandarización en el formato de intercambio de información es un gran paso en la dirección correcta, pero no por publicar la información en XML, esta podrá ser procesada de manera automática por las computadoras. XML es útil, primordialmente, para estructurar y documentar la propia información, pues tiene la bondad y el problema de que el creador del documento es quien escoge el nombre de las etiquetas, lo cual supone que al momento de intercambiar información debe existir un acuerdo previo sobre el significado explícito de las mismas.

El modelo de datos XML consiste en un árbol que no distingue entre objetos y relaciones, ni tiene noción de jerarquía de clases.

2.3.3.2 RDF

En 1999 se publicó la primera versión de RDF (Resource Description Framework o Marco de Descripción de Recursos), un lenguaje para la definición de ontologías y metadatos en la Web. RDF es hoy el estándar más popular y extendido en la comunidad de la Web semántica.

[7] "eXtensible Markup Language", 2006-05-22. http://www.w3.org/XML

Podríamos decir que el objetivo principal de RDF es definir un mecanismo que describa los recursos de una forma no concreta, es decir, que se especifique un esquema que sea posible utilizar en distintos dominios de aplicación.

Las ventajas que ofrece RDF frente a otra forma de representar información son:

- La posibilidad de reutilización de las definiciones de los metadatos.
- Puede ser extensible de forma incremental, después de haber sido definido un esquema.
- La posibilidad de crear objetos que intercalen distintos tipos de metadatos para hacerlos más completos (Esto se consigue basándose en esquemas distintos: RDF, Friend Of a Friend (FOAF), Dublín Core (DC), OWL, etc.).

Se pueden destacar tres aspectos de la semántica funcional del modelo: un modelo de datos, una sintaxis y un esquema.

El modelo de datos básico consiste en tres tipos de objetos, que constituyen la base sintáctica del lenguaje:

- Propiedades

  Son los atributos, características o relaciones utilizadas para describir un recurso. Cada propiedad tiene un significado determinado, define cuales son sus valores permitidos, los tipos de recursos que puede describir y sus relaciones con otras propiedades.

- Recursos

  Un objeto de información o recurso se describe a través de un conjunto de propiedades denominadas "Descripción RDF" (<*rdf:Description*>), la esencia de RDF es entonces, un modelo formal para la representación de las propiedades y los

valores de las mismas, que se pueden entender como atributos de los recursos y, en este sentido corresponden a los pares tradicionales de atributo-valor.

Los recursos se encuentran unívocamente identificados por Uniform Resource Identifiers(URIs), las cuales son una cadena corta de caracteres que identifica unívocamente un recurso. Las URIs están definidas en el RFC 2396.

Ejemplo simple de una URI: *http://www.unan.edu.ni/imagenes/unan.jpg*
Donde podemos distinguir varias partes:

- Esquema: Nombre que identifica el protocolo de acceso al recurso. En nuestro caso *http*. Los ":" son un separador y las dos barras son una indicación de raíz absoluta para la autoridad.
- Autoridad: Elemento que identifica quién o qué es el autor del recurso, es decir, en nuestro ejemplo de URI sería *www.unan.edu.ni*
- Ruta: Información que identifica la ruta jerárquica donde se encuentra el recurso, o también la ruta por la cual será posible determinar la situación física real del mismo. */imágenes/*

En su mayoría las URI que utiliza RDF son las URIref siendo estas una especificación de dichas URIs, las cuales están conformadas por la URI, junto a un identificador al final de la misma (generalmente un símbolo #).

Ejemplo:
La URIref definida por *http://www.w3.org/1999/22-rdf-syntax-ns#Description*, hace referencia al elemento *Description* definido en la URI *http://www.w3.org/1999/22-rdf-syntax-ns*.

- Enunciados

  También llamados RDF statement. La estructura de un enunciado o declaración consta de tres partes: un sujeto, un predicado y un objeto. Como podemos observar, es la misma estructura de una oración simple en el lenguaje natural. Dado que se trata de describir un recurso, las declaraciones especifican valores para las propiedades de dichos recursos, compuestas de la siguiente manera:

  - Sujeto: el recurso que se va a describir.
  - Predicado: propiedad del recurso.
  - Objeto: valor de la propiedad del recurso.

Una oración en lenguaje natural podría ser la siguiente:
Ejemplo 1.

Da'Vinci es el creador de la Monalisa.

Sujeto(Recurso)	Monalisa	*http://www.examplemuseum.net/pinturas/monalisa*
Predicado(Propiedad)	Creador	*http://purl.org/dc/elements/1.1/creator*
Objeto(Literal)	Da'Vinci	*http://www.examplemuseum.net/pintores/Da'Vinci*

Tabla 2.1

En la figura 2.1 se diagrama esta sentencia RDF usando un grafo dirigido y rotulado. En estos diagramas los nodos (dibujados como óvalos) representan recursos y los arcos representan propiedades con nombres (el rotulo del arco es el nombre de la propiedad). Los nodos literales se dibujan como rectángulos.

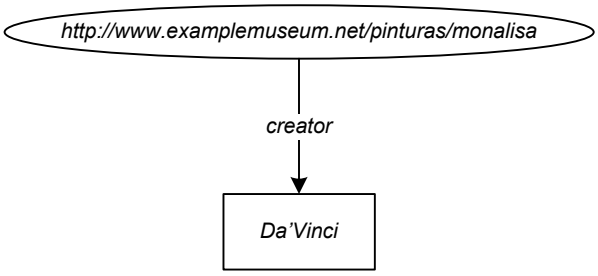

Figura 2.1.

La dirección del arco es importante, siempre empieza en el sujeto y termina en el objeto de la sentencia.

Los arcos de un grafo RDF, pueden ser etiquetados con URIs, literales o un valor en blanco (estos últimos se conocen como nodos en blanco), los cuales son utilizados para representar información referente a un recurso no identificado.

Ejemplo 2:

El individuo cuyo nombres es Da'Vinci, nacionalidad italiana, es el creador de la Monalisa.

La intención de esta frase es precisar el valor de la propiedad Creator[Creador] como una entidad estructurada. En RDF tal entidad se representa como otro recurso. La sentencia anterior no proporciona un nombre a ese recurso; es anónimo, por ello el gráfico a continuación lo representa como un óvalo vacío.

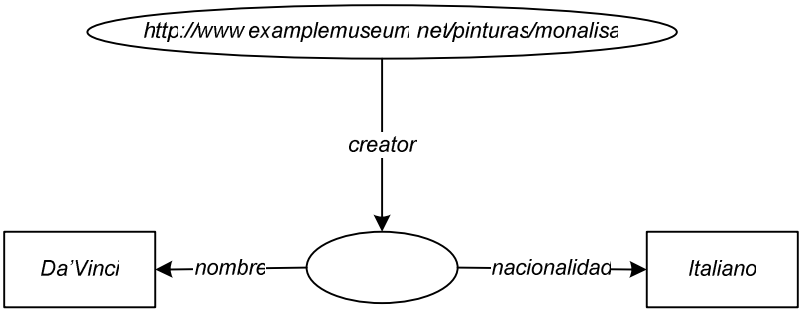

Figura 2.2: Propiedad con valor estructurado.

Contenedores RDF

Frecuentemente es necesario referirse a una colección de recursos, por ejemplo para describir los autores de un trabajo; para estas colecciones de datos RDF ofrece tres tipos de contenedores que permiten manejar listas de recursos o literales.

- *Bag*: utilizado para representar una lista de recursos o literales sin orden. Los valores duplicados son permitidos.

- *Sequence*: Una secuencia a diferencia de un *bag*, es una lista ordenada de recursos o literales. Este tipo de contenedor es usado cuando se necesita establecer algún orden en los recursos o literales que se pretenden enumerar.

- *Alternative*: Es una secuencia la cual define los posibles valores que pueden ser tomados por ejemplo para el valor de una propiedad.

Un contenedor RDF se representa como un recurso, que puede ser un nodo en blanco o un recurso identificado por una URIref. A este recurso se le define una propiedad *rdf:type* cuyo valor puede ser *rdf:Bag*, *rdf:Seq o rdf:Alt*.

Si estuviéramos hablando de un dominio en particular como el de rones producidos en Nicaragua e hiciéramos la representación mostrada en la figura 2.3 utilizando grafos RDF.

Según el grafo mencionado el orden en que aparecen los rones no nos interesa, por lo tanto el contenedor empleado para listarlos es un *rdf:Bag* cuya única función es describir un conjunto de datos.

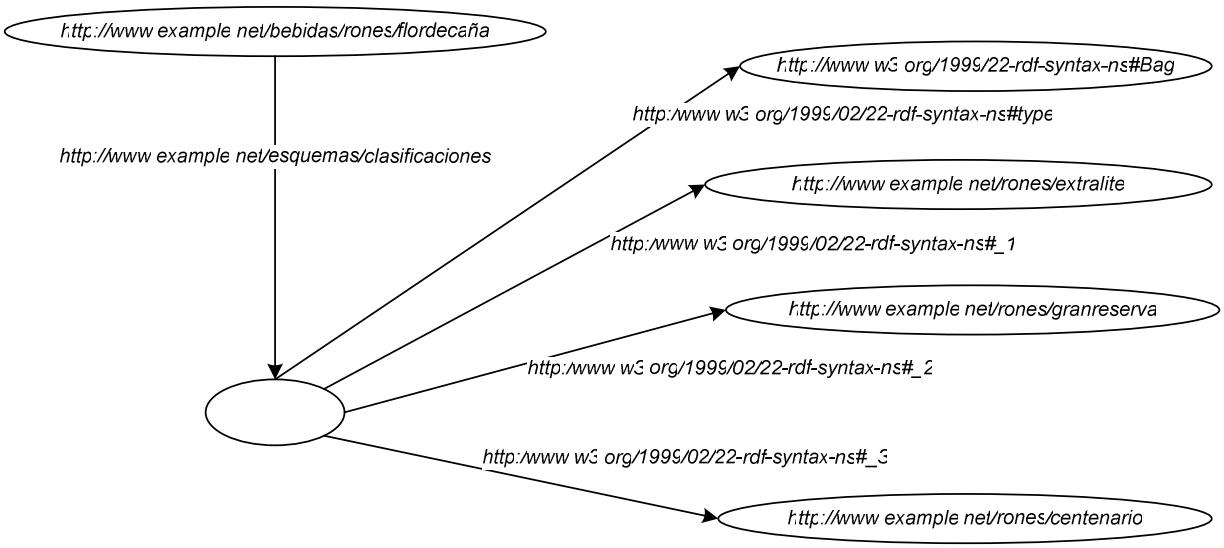

Figura 2.3: Grafo utilizando un contenedor.

Namespaces

Un espacio de nombres XML es un conjunto de nombres identificados por una referencia URI que se utilizan en documentos XML como tipos de elementos y nombres de atributo[8]. El concepto de "namespace" surge de la necesidad para combinar diferentes vocabularios en XML; es una forma de agrupar distintos elementos para poder ser utilizados en un solo documento.

Siendo RDF un marco de trabajo abierto y mundial que permite a personas de diferentes áreas hacer declaraciones sobre cualquier recurso, hace que la diversidad de intereses de las distintas comunidades que hacen uso de estos vocabularios ó recursos, finalmente se puede revertir en una disparidad terminológica, la cual se soluciona haciendo uso de los "namespaces".

[8] http://html.conclase.net/w3c/xml-names-es/#Philosophy

RDF/XML

La representación computacional del modelo RDF se realiza a través de un vocabulario XML, de esta manera se obtienen los beneficios que ofrece XML como tecnología para estructurar información en la Web.

Sintaxis

La sintaxis básica como se ha comentado anteriormente, es XML versión 1.0. Dentro de este contexto se permite la utilización de dos tipos de sintaxis para realizar la codificación del modelo de datos. Por un lado está la serializada que aporta un conjunto reducido de expresiones para llevar a cabo la descripción del modelo y la abreviada que incluye construcciones adicionales que hacen que se pueda representar de una manera más compacta el modelo de datos. En un documento rdf es posible encontrar una combinación de estos dos tipos de sintaxis.

Definiendo un Documento RDF

La serialización de un grafo RDF en XML suele estar contenida en un elemento *rdf:RDF*, el cual será el elemento raíz del documento XML. En este elemento se definen todos los *namespaces* que serán referenciados por las descripciones de los recursos que se están abordando.

```
<rdf:RDF
        xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc = "http://purl.org/dc/elements/1.1/">
        *
        *
</rdf:RDF>
```

Un documento RDF es, al fin una serie de descripciones sobre recursos. Cada declaración o sentencia es una "descripción sobre" algo. Por lo tanto en RDF/XML se

representa con un elemento *rdf:Description* y un atributo *rdf:about* el cual es el encargado de identificar el recurso que se va a describir, lo cual lo convierte en el sujeto de la declaración.

Para representar los predicados o propiedades el elemento *rdf:Description* se vale de otro elemento que puede contener una de estas formas:

a) <propName>value</propName>
b) <propName resourceAttr/>

Donde:

propName:	Qname
Qname:	NamespacePrefix:name
NamespacePrefix:	Cualquier espacio de nombres permitido en XML.
name:	Cualquier nombre permitido en XML.
resourceAttr:	resource = URI-reference

El objeto o valor del enunciado RDF puede ser tratado de dos maneras dependiendo de lo que represente:

- Valor literal: el valor es representado entre las etiquetas de la propiedad
- Recurso: se utiliza la opción b) de la representación donde la URI apunta al recurso en cuestión y como se puede ver, se hace uso de un elemento de etiqueta vacía (sin etiqueta de cierre).

Ejemplo concreto de sintaxis RDF para el grafo de la figura 2.1 afirmando que *creator* es el elemento definido en Dublín core.

```
1.  <rdf:RDF
2.      xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.      xmlns:dc = "http://purl.org/dc/elements/1.1/">
```

```
4.              <rdf:Description rdf:about = “http://examplemuseum.net/pinturas/monalisa”>
5.                      <dc:creator>Da’Vinci</dc:creator>
6.              </rdf:Description>
7.      </rdf:RDF>
```

Literales

RDF proporciona una manera de tipar los literales (decir si es un entero, una fecha, etc), aunque este modelo no interpreta los tipos de datos por si mismo, permite a las aplicaciones que utilizan dichos datos, tratarlos de una manera adecuada.

Sintácticamente, esto se traduce en agregar al elemento propiedad un atributo *rdf:datatype* seguido de la URIref del tipo de dato.

Por ejemplo si tenemos la siguiente declaración: “DaVinci nace el 15/04/1452”, y queremos especificar que el literal “15/02/1452” es una fecha, el documento *rdf* debe rezar algo así:

```
1. <rdf:RDF
2.      xmlns:rdf = “http://www.w3.org/1999/02/22-rdf-syntax-ns#”
3.      xmlns:ejemplo = “http://www.example.org/esquema/”>
4.      <rdf:Description about = “http://examplemuseum.net/pintores/DaVinci”>
5.              <ejemplo:nace rdf:datatype =
6               “http://www.w3.org/2001/XMLSchema#date”>15/04/1452</ejemplo:nace>
7       </rdf:Description>
8</rdf:RDF>
```

2.3.3.3 Esquema RDF (RDF Schema o RDFs)

RDFS al igual que RDF es un lenguaje basado en XML para la especificación de vocabularios RDF; de hecho es una extensión del mismo el cual provee mecanismos para describir grupos de recursos relacionados y las relaciones entre estos recursos. Los recursos

en RDFS son divididos en grupos llamados clases. Los miembros de una clase son conocidos como instancias de la clase, las clases mismas son recursos, identificados por una URI y descritas usando propiedades RDF.

El esquema RDF es útil para definir jerarquías de clases de recursos, especificando las propiedades y relaciones admitidas entre ellas. En este esquema existe elementos como la herencia de propiedades de una clase a sus subclases, cabe señalar que RDFS es quien contiene las definiciones que posteriormente se usan en archivos RDF; los archivos RDF son la información con sus metadatos, mientas que el esquema RDF contiene las definiciones de dichos metadatos.

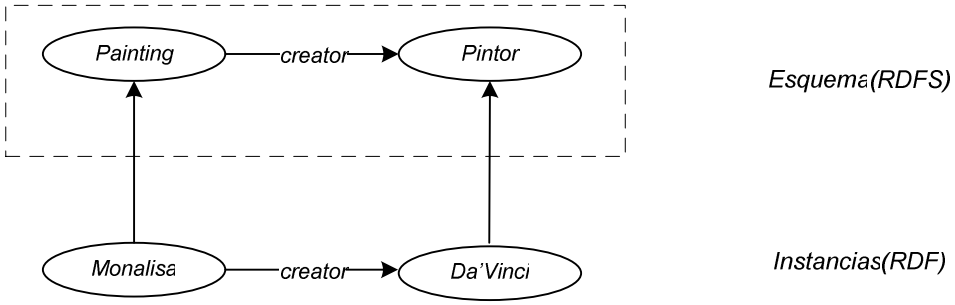

Figura 2.4. Ejemplo de un Esquema RDF e instancias asociadas.

Aquí se puede ver que tenemos un pequeño esquema RDF el cual define dos clases relacionadas a través de una propiedad *creator*. Como podemos observar la relación de las instancias está dada por el Qname *creator* donde este elemento podría ser el definido en Dublín Core para lo cual le antepondríamos el espacio de nombres dc (Dublín Core) y el nombre local creador (dc:creator), se puede notar que en este pequeño esquema la instancia de pintor se trata como un recurso ya que está representada con un óvalo.

2.3.3.4 OWL

Han existido diversos lenguajes para el desarrollo de ontologías, pero eran específicos para determinados ámbitos de conocimiento y no aplicables a cualquier

ontología. La necesidad de escribir ontologías que pudieran ser interpretadas y procesadas por las aplicaciones de una forma inequívoca hizo necesaria la aparición del estándar OWL.

El OWL (Ontology Web Language), está basado en un lenguaje anterior denominado DAML+ÓIL. Este proviene de la conjunción de DARPA[9] Agent Markup Language (DAML, en español lenguaje DARPA de marcado), y de Ontology Interchange Language (OIL, en español Lenguaje de Intercambio de Ontologías).

Este lenguaje de etiquetado semántico extiende RDFS para permitir la expresión de relaciones complejas entre diferentes clases RDFS, y mayor precisión en las restricciones de clases y propiedades específicas.

OWL es una de las principales tecnologías que implementan la Web Semántica debido al ámbito tan amplio en el que trabaja y la gran capacidad de definición que tiene.

Sublenguajes de OWL

OWL proporciona tres sublenguajes que pueden ser utilizados por los usuarios o implementadores en función de sus necesidades. Estos sublenguajes están clasificados por su expresividad creciente, y son:

- OWL Lite.
- OWL DL.
- OWL Full

Cada sublenguaje es una extensión del anterior; tanto en lo que se puede expresar de manera legal, como en las conclusiones válidas que se pueden obtener.

[9] DARPA, the Defense Advanced Research Projects Agency, es una agencia del Departamento de Defensa de los Estados Unidos responsable del desarrollo de nuevas tecnologías para uso militar.

Estructura de una Ontología OWL

Dado que un documento OWL está basado en RDF, debe comenzar con un elemento *rdf:RDF* que además servirá para incluir el conjunto de *namespaces* XML que se van a utilizar durante la definición de dicho documento.

Un ejemplo de definición de namespaces sería el siguiente:

```
1.      <rdf:RDF
2.         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.         xmlns:exif="http://www.w3.org/2003/12/exif/ns#"
4.         xmlns="http://localhost/documentos.owl#"
5.         xmlns:owl="http://www.w3.org/2002/07/owl#"
6.         xmlns:dc="http://purl.org/dc/elements/1.1/#"
7.         xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
8.         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
9        xml:base="http://localhost/documentos.owl">
```

Como podemos ver en la línea 4 se define el espacio de nombres por defecto, para indicar que los nombres cualificados sin prefijo se refieren a la ontología actual.

En la línea 9, *xml:base* identifica la URI base para el documento, que puede utilizarse para definir la URI base de un elemento como una distinta a la URI del documento. El resto de declaraciones no tienen ningún significado en especial, siguen las mismas convenciones que los espacios de nombres comunes.

Seguidamente aparece un elemento *owl:Ontology*, que se utiliza para definir algunos metadatos propios de la ontología.

```
    <owl:Ontology rdf:about=””/>
```

El atributo *rdf:about* indica un nombre o referencia para la ontología. El valor es “”, indicando que el nombre de la ontología es la URI base del elemento *owl:Ontology*, que típicamente será la URI del documento que contiene la ontología o la definida en el elemento *rdf:RDF* con la declaración *xml:base*. Al finalizar todas las declaraciones que involucran a la ontología vendrá el elemento de cierre de RDF.

Definición de conceptos elementales

A continuación mencionaremos algunos conceptos básicos sobre la definición de clases, propiedades, restricciones e instancias con OWL.

Clases: En OWL las clases se construyen a partir de descripciones que especifican las condiciones que deben ser satisfechas para que un individuo sea miembro de una clase.

Cada clase en OWL se define como un elemento *owl:Class*, y su nombre se indica con un atributo *rdf:ID*. Por ejemplo si quisiéramos definir las clases utilizadas en los grafos anteriores:

```
<owl:Class rdf:ID=”Pintor”/>
<owl:Class rdf:ID=”Pinturas”/>
```

Las clases *raíz* son aquellas que, en el dominio de la ontología, no son subclase de ninguna otra clase. En OWL, estas clases son implícitamente subclases de una clase predefinida *owl:Thing*. Las clases que hemos definido en el ejemplo podrían ser clases raíz de la ontología de Pintores.

El atributo *rdf:ID* permite hacer referencia a las clases con *rdf:resource*. Así, por ejemplo, para hacer referencia a la clase pintor se utilizaría el atributo *rdf:resource = “#pintor”*.

Las jerarquías de clases en OWL se crean utilizando la etiqueta *rdfs:subclassOf*. Si deseáramos expresar que la clase "pintorclasico" es subclase de pintor, lo escribiéramos de la siguiente manera en OWL:

```
<owl:Class rdf:ID="pintorclasico">
        <rdfs:subClassOf rdf:resource = "#pintor"/>
</owl:Class>
```

Instancias:

Una instancia es una representación individual concreta de una clase. En OWL se declaran como miembros de la clase a la que representan:

```
<pintor rdf:ID = "Da'Vinci"/>
```

Propiedades: El alcance de definición de una ontología no debe quedarse en la taxonomía de clases, sino que se debe poder hacer afirmaciones o restricciones sobre los miembros que componen una clase. Estas afirmaciones o restricciones pueden hacerse gracias a las propiedades.

Existen 2 tipos de propiedades:

- Propiedades de tipos de datos (datatype properties): relacionan las instancias de las clases con literales RDF y tipos de datos XML Schema.

- Propiedades de objeto: permiten relacionar instancias de dos clases.

Tomando en cuenta que las propiedades son relaciones binarias, 1 a 1. Estas relaciones se pueden restringir especificando un dominio y un rango. Tomando como referencia el modelo RDF, el dominio representa la clase a la que puede pertenecer el sujeto de la declaración, y el rango la clase a la que puede pertenecer el objeto.

En el caso particular de las propiedades de tipos de datos, el rango haría referencia al tipo de datos XML Schema al que puede pertenecer el valor literal de la propiedad. Por ejemplo, podríamos, definir la siguiente propiedad para pintor:

```
<owl:DatatypeProperty rdf:ID=”edad”>
        <rdfs:domain rdf:resource=”#pintor”>
        <rdfs:range rdf:resource=”&xsd;integer”/>
<owl:DatatypeProperty>
```

Entre muchas otras cosas que permite OWL es posible definir algunas características sobre las propiedades facilitando un mayor poder de razonamiento sobre ellas, por ejemplo la característica de propiedad denominada *FunctionalProperty* indica que una instancia del dominio sobre el que se aplica la propiedad puede estar asociada, como mucho, con un valor del rango de la misma.

Como ejemplo, a continuación se presenta una serialización de una pequeña ontología para mostrar la sintaxis del lenguaje OWL.

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://localhost/dublin.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://localhost/dublin.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Document"/>
  <owl:DatatypeProperty
rdf:about="http://purl.org/dc/elements/1.1/creator">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty
rdf:about="http://purl.org/dc/elements/1.1/description">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty
rdf:about="http://purl.org/dc/elements/1.1/language">
    <rdfs:domain rdf:resource="#Document"/>
```

```
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="link">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://purl.org/dc/elements/1.1/date">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty
rdf:about="http://purl.org/dc/elements/1.1/publisher">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Document"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty
rdf:about="http://purl.org/dc/elements/1.1/format">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Document"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty
rdf:about="http://purl.org/dc/elements/1.1/title">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.3.1, Build 430)
http://protege.stanford.edu -->
```

¿Que es Indexar?

En informática el término indexar se refiere a la creación de índices sobre algún tipo de información, y en este punto tienen el mismo objetivo que el índice de un libro, el cual es agilizar las búsquedas de los usuarios cuando se intenta encontrar información sobre algún determinado tema.

De aquí el proceso de indexación sobre algo, se concibe como la creación de formas más eficientes y eficaces (índices) de encontrar información en algún medio en particular, específicamente en nuestro caso se trata sobre documentos digitales que pueden estar disponibles en diferentes formatos.

2.4 Documentos Digitales:

Se le llama documentos digitales a aquellos en los cuales la información está registrada en formato electrónico, y que requiere de dispositivos informáticos para poder consultarlos.

2.4.1 Formato pdf.

Del inglés (Portable Document Format o Formato de Documento Portátil) es un formato de almacenamiento de documentos desarrollado por la empresa Adobe Systems a inicio de los 90's. Fue diseñado para ser totalmente independiente del hardware, software y sistema operativo, característica que le permitió ganar mucha popularidad a través de los años principalmente en Internet, convirtiéndose finalmente en un estándar ISO[10] 32000 a inicios del presente año.

2.4.2 Formato doc.

Formato nativo muy utilizado y cerrado (del Procesador de texto, Microsoft Word), se ha convertido en un estándar de facto con el que pueden transferirse textos con formato o sin formato, inclusive imágenes, siendo preferido por muchos usuarios antes que otras opciones como el texto plano para el texto sin formato. Posee la desventaja de tener un mayor tamaño comparado con otros formatos similares.

Nota: En el momento de edición de este documente existe una gran polémica en torno a que Microsoft consiguió que ISO concediera la certificación a su formato para documentos intercambiables Office Open XML (IS-29500)[11] el cual aplica tanto a procesadores de texto como a hojas de calculo.

[10] ISO (Organización Internacional Para la Estandarización) Es el organismo encargado de promover el desarrollo de normas de fabricación, comercio y comunicación para todas las ramas industriales a excepción de la eléctrica y la electrónica.
[11] http://www.iso.org/iso/pressrelease.htm?refid=Ref1123

2.4.3 Formato mp3.

MPEG-1 Audio Layer 3, también conocido como MP3, es un formato de audio digital comprimido con pérdida desarrollado por el Moving Picture Experts Group (MPEG) para formar parte de la versión 1 y posteriormente también presente en la versión 2 del formato de video.

2.4.4 Formato Exif.

Exchangeable image file format (Exif), es una especificación para formatos de imagen usado por las cámaras digitales. Fue creado por Japan Electronic Industry Development Association (JEIDA). La especificación usa los formatos de archivos existentes como JPEG, TIFF Rev. 6.0 y RIFF.

Las etiquetas de metadatos definidas en el estándar Exif cubren un amplio espectro y se pueden clasificar de la siguiente manera:

- Información sobre Tiempo
- Configuración de la cámara
- Información sobre localización.
- Descripción sobre derechos de autor.

2.4.5 Formato AVI.

El formato *AVI* (Audio and Video Interleave) fue diseñado por Microsoft para su tecnología *Video for Windows* en 1992. Este formato permite almacenar simultáneamente un flujo de datos de video y varios flujos de audio. El formato concreto de estos flujos no es el objeto del formato *AVI* y es interpretado por un programa externo denominado *códec*. Es decir, que el audio y el video contenidos en el *AVI*, pueden estar en cualquier formato, por esto se le considera un formato contenedor.

2.4.6 Formato WMA.

Windows Media Audio o WMA es un formato de compresión de audio propiedad de Microsoft, al igual que el mp3 se incluye en muchos reproductores portátiles, pero no es tan popular como este último.

2.4.7 Formato XLS.

Microsoft Excel es una aplicación de hoja de cálculo (propietaria) escrita y distribuida por Microsoft para los sistemas operativos *Microsoft Windows* y *Mac OS X.* realmente los libros de Microsoft Excel son guardados con el formato BIFF (Binary Interchange File Format), en la siguiente tabla se puede ver las versiones del formato.

Versión BIFF	**Microsoft Office Excel Versión**
BIFF5	Microsoft Excel Version 5.0 (XL5)
BIFF7	Microsoft Excel Version 7.0 (XL7)
BIFF8	Microsoft Excel 97 (XL8), Microsoft Excel 2000(XL9), Microsoft Excel 2002(XL10), Microsoft Excel 2003 (XL11), Micr. Excel 2007 (XL12).

Microsoft Excel es muy utilizado sobre todo en entornos financieros o procesos empresariales que requieren una alta manipulación de números y fórmulas, ó incluso debido a su versatilidad muchas veces se llevan pequeños registros sobre alguna información simulando en parte a las bases de datos.

2.4.8 Formato PPT.

Esta es la extensión utilizada para los archivos de presentación elaborados con el software propietario de Microsoft para presentaciones incluido en el suite de office (Microsoft Power Point); disponible tanto para el sistema operativo Microsoft Windows como para Mac OS.

3. ANALISIS DEL SISTEMA

3.1 Herramientas Similares Disponibles (Libre Distribución)

Actualmente existen herramientas para la indexación y búsqueda de documentos digitales tales como:

3.1.1 Simple Web Indexing System for Human – Enhanced (SWISH-E)

Esta es una herramienta que fue mejorada y actualmente utilizada por el boletín oficial del Estado (España)[12] para búsqueda y recuperación de contenidos completos en el cual se requerían las siguientes características:

- Alto rendimiento en búsqueda e indexación ante el volumen esperado de consultas.
- Disponibilidad de librería de programación (API).
- Ordenación de resultados por diversos criterios (campos).
- Indexación de caracteres nacionales (ISO-8859) y soporte de tablas de conversiones.
- Utilización de filtros externos para indexar diferentes tipos de contenidos (texto, html, pdf, etc)

Según pruebas de rendimiento sobre este software funciona muy bien, pero con todo y esto su enfoque es más sintáctico que semántico.

[12] http://www.boe.es/

3.1.2 Lucene

Este software no se trata de una aplicación completa, sino que es un API (Application Programming Interface) de desarrollo para indexación y búsqueda el cual es muy eficiente, versátil y robusto escrito en Java pero también disponibles en C++ y otros lenguajes.

Entre las características de Lucene podemos mencionar las siguientes:

- Multiplataforma.
- Alto rendimiento, escalable.
- Algoritmos de búsqueda, potentes, fiables y eficientes.
    - Permite ordenar resultados por relevancia
    - Lenguaje de consulta muy potente (frases, comodines, proximidad, rango…)
    - Búsqueda por campos
    - Búsqueda por rangos de fechas
    - Ordenación por cualquier campo
    - Búsqueda multi-índice y combinación de resultados

Se puede consultar información adicional en http://lucene.apache.org/

3.1.3 Indexador de Paquetes Unix

Indexador Semántico de paquetes Unix, es una herramienta desarrollada como trabajo de fin de carrera de la carrera de Ingeniería de Telecomunicación en la Universidad de Alcalá Henares. Debido a su naturaleza este trabajo se explicará un poco más ya que el trabajo desarrollado lo toma como base.

3.2 Base de la Herramienta

Como hemos mencionado anteriormente esta herramienta toma como base la herramienta de Indexación de paquetes Unix por lo cual se abordarán brevemente las herramientas utilizadas en el desarrollo de este último, si se desea mas detalles sobre el desarrollo de la misma consulte [5].

Enumeración de Herramientas utilizadas:

- Plataforma de Desarrollo: Linux
- Tecnologías Web Semántica: XML, RDF y OWL.
- Herramienta gráfica para desarrollar ontologías (Protegé)
- Base de datos Semántica (Sesame)
- Servidor de Aplicaciones (Tomcat)
- Base de datos Relacional (Mysql)
- Lenguaje de Consulta (SeRQL)
- Lenguaje de desarrollo de módulos del indexador (Perl)
- Servidor Web (Apache).
- Lenguaje de desarrollo de la aplicación de cara al usuario (PHP).

3.2.1 Arquitectura del Sistema

Esta herramienta se divide en tres bloques principales que se listan a continuación:

- Herramienta de indexación de documentos digitales.
- Sistema de almacenamiento de la metainformación.
- Aplicación de búsqueda de documentos.

En la figura 3.1 se muestra la arquitectura general del sistema, donde se pueden ver estos bloques y los elementos con los que el mismo interactúa.

La arquitectura presentada en combinación con un sistema de almacenamiento (Sesame) accesible por HTTP, permiten que el sistema se pueda distribuir para agilizar las operaciones. Siguiendo este modelo; en una red de computadores, el sistema de almacenamiento cuyas demandas de hardware son mayores podría alojarse en la

computadora que presente mejores prestaciones, siendo también el mismo de responder las consultas sobre RDF (a través de SeRQL) hechas por los usuarios.

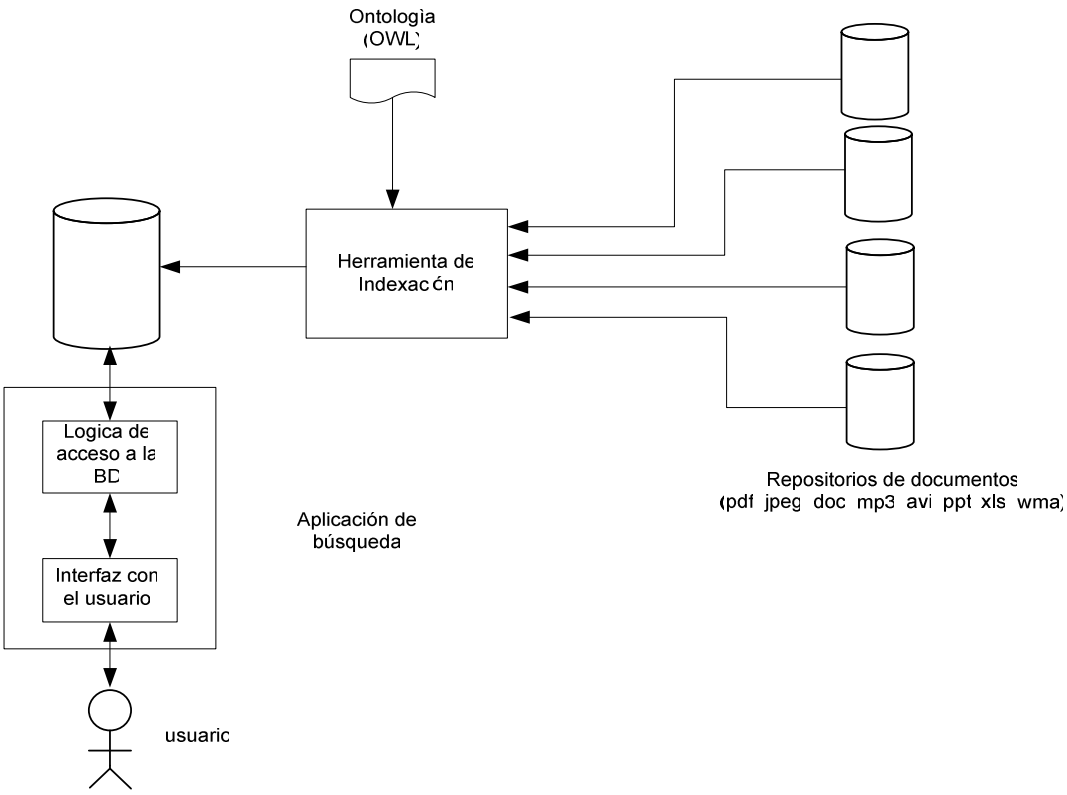

Figura 3.1: Arquitectura general de la herramienta de Indexación.

En este caso hipotético, el resto de ordenadores los cuales probablemente tengan menos prestaciones que el elegido para el sistema de almacenamiento, pueden ser ordenadores no dedicados exclusivamente a la tarea de indexación, sino que tendrían algunos repositorios asignados los cuales indexarían en determinado momento.

La idea detrás de este enfoque, es que en un momento dado cada uno de los ordenadores que tiene instalada la herramienta de indexación comenzaría a extraer la metainformación de los archivos que se les hayan asignado, donde el hecho de distribuir la carga de indexación supondría un buen ahorro de tiempo, hecho esto, se conectarían utilizando HTTP al ordenador donde yace el sistema de almacenamiento y descargarían ahí la información indexada.

También la aplicación de búsqueda (PHP) puede estar en un ordenador diferente de los indexadores y del sistema de almacenamiento, al que se conectaría al igual que las máquinas de indexación remota, utilizando HTTP.

Como se ha comentado anteriormente como sistema de almacenamiento se utilizará una base de datos RDF ya implementada (Sesame), por lo que sólo se hará un breve comentario del mismo.

A continuación se explicará en detalle las distintas facetas del indexador y de la aplicación de búsqueda.

3.2.2 Archivos de Configuración

Tratando de separar lo máximo posible el código de cualquier parámetro que pueda variar (parámetros de conexión a la base de datos RDF), se han utilizado archivos de configuración, los cuales se pueden clasificar en internos y externos.

- **Archivos de configuración externos:** son aquellos que el administrador del sistema debe modificar para echar a andar el indexador en algún equipo determinado (Indexer.conf).

- **Archivos de configuración internos:** son aquellos que el administrador no tendría por qué modificar, a no ser que quisiera cambiar determinados valores por defecto, añadir nuevos *plugins*, modificar la ontología, etc.

Archivos de configuración internos	
Nombre	**Propósito**
Plugins.conf	indica ubicación y archivo de conf. utilizado por cada plugin del sistema (procesador de ontologías, extractores o actualizadores de datos).
Ontology.conf	indica ruta de la ontología que debe procesar

Extractors.conf	indica que plugin utilizar para cada extensión de archivo.
Archivos de equivalencia.	Contiene pares de equivalencia entre términos extraídos de los metadatos y los correspondientes a la ontología.
Sesame.conf, Rdf.conf	Contienen parámetros tales como credenciales de conexión utilizados por Sesame para acceder a Mysql.

Cuadro 3.1: Resumen de archivos de configuración utilizados por el Indexador de Paquetes Unix.

Por cuestiones de seguridad, todos los archivos de configuración deben tener permisos de lectura y escritura únicamente para el propietario del sistema de indexación o, como máximo para el grupo al que pertenece. En la figura 3.3 se puede ver la arquitectura mostrada en la figura 3.2 pero de una manera mas detallada.

En la Figura 3.2 se muestra un esquema de la arquitectura general del indexador, donde se puede apreciar su carácter modular, junto con los elementos externos con los que interactúa.

3.2.3 Arquitectura del Indexador

Una característica importante del indexador es su diseño modular, debido al cual es posible ampliar las fronteras del mismo en cuanto a la información a indexar. Se debe entonces a este diseño, que se tienen módulos para el procesamiento de la ontología, para la extracción de metadatos y la actualización de los datos semánticos, todos ellos gestionados por un modulo central.

Los plugins con lo que cuenta el indexador son los siguientes:

- **Plugin para procesamiento de la ontología:** se considera plugin porque se podría querer procesar ontologías expresadas con distintas sintaxis (DAML+OIL,XML…).

- **Plugin para extracción de metadatos de archivos:** Podrían diseñarse plugins que puedan extraer metadatos de otro formato de documento (ejemplo videos flv, mpeg, etc).

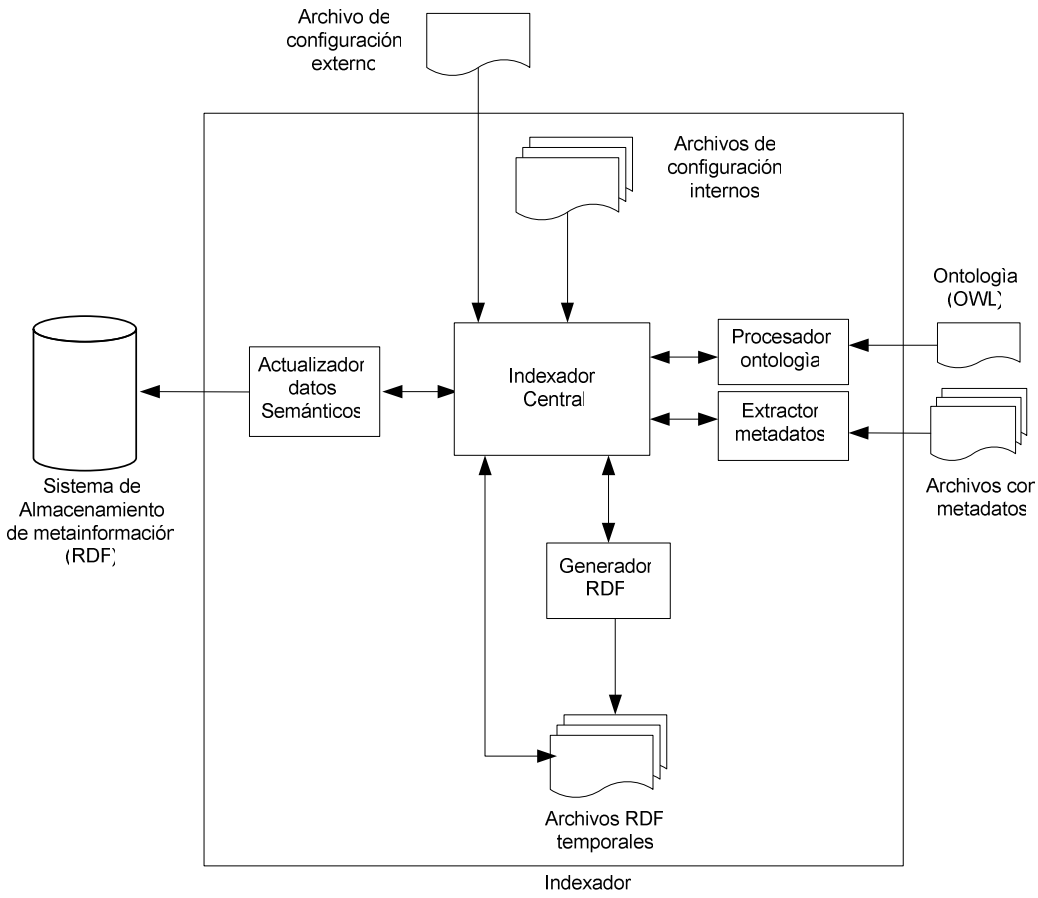

Figura 3.2: Arquitectura general del indexador.

- **Plugin para actualización de datos semánticos:** Podrían diseñarse plugins para almacenar los datos RDF en diversos sistemas o formatos (ejemplo, otros bases de datos sobre RDF, bases de datos relacionales, etc.).

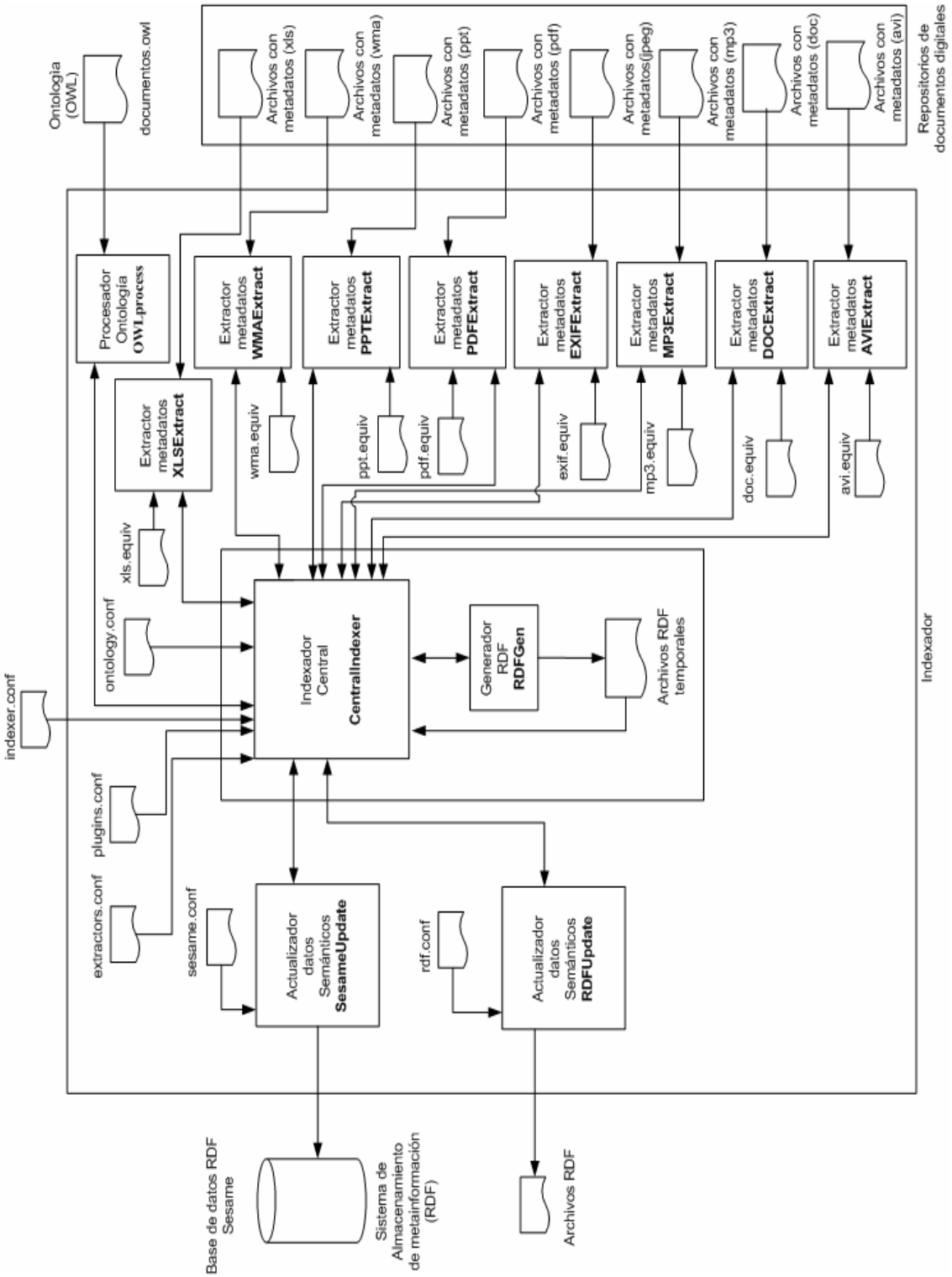

Figura 3.3: Arquitectura general del indexador, con los módulos desarrollados y/o modificados.

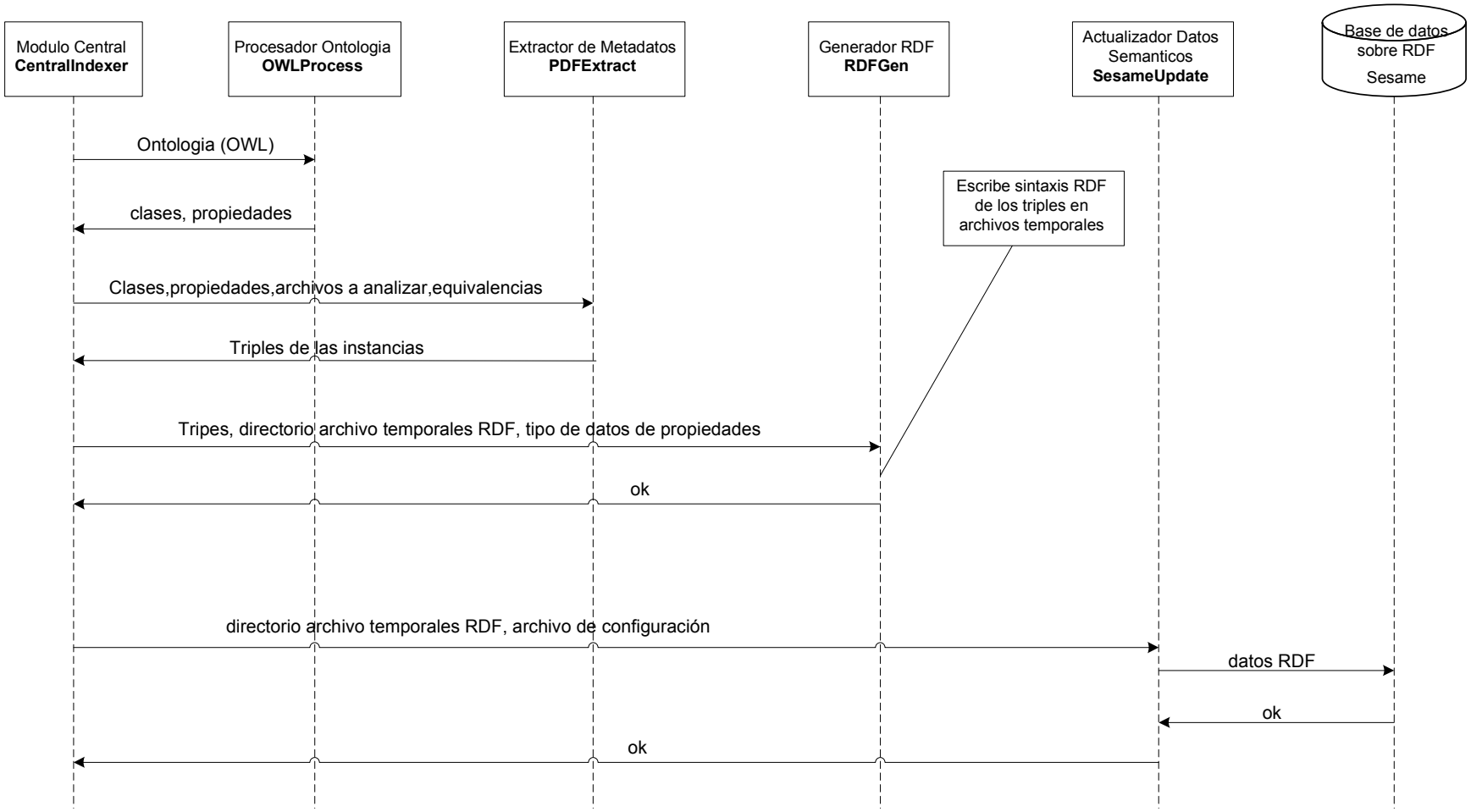

Figura 3.5: Diagrama de Secuencia de las operaciones del Indexador.

3.3 Campos de Aplicación del Indexador.

Una de las aplicaciones que se le puede dar al indexador es el de ofrecer a la comunidad estudiantil una vía de acceso a los trabajos de fin de curso elaborados en las distintas facultades y departamentos de la universidad, evitando de esta manera que dichos alumnos tengan que ir a las bibliotecas de los departamentos a buscar información sobre algún tema en particular, ya sea por curiosidad o por que están haciendo trabajos de fin de curso y necesitan revisar datos sobre otras monografías que tengan mucho que ver con el tema que están buscando.

3.4 Meta-información disponible en los documentos digitales.

A efectos de poder manipular la información que sea recopilada por el indexador utilizando estrategias de la Web Semántica, se hace necesaria alguna forma de describir las relaciones existentes entre los distintos conceptos que puedan aparecer en el dominio de estudio. En este caso en particular, necesitamos crear una ontología que defina el esquema bajo el cual se relacionarán los conceptos que describen los distintos documentos digitales.

Con este fin, necesitamos entonces tratar con la información acerca de estos documentos digitales (con sus metadatos) que pueda ser deducida de dichos documentos.

3.4.1 Metadatos comunes

A vista de la metainformación obtenida a partir de los documentos digitales se puede observar que aparecen elementos comunes ya sea en nombres o en conceptos. Estos conceptos nos resultarán muy útiles para la homogeneidad que debe permitir la ontología.

El siguiente cuadro muestra los conceptos comunes incluyendo el nombre a al menos dos tipos de documentos.

3.5 Herramientas Utilizadas

3.5.1 Herramienta Gráfica para el Diseño de la Ontología.

Para llevar a cabo la elaboración de la ontología se ha utilizado Protegé, una herramienta grafica desarrollada por la Escuela de Medicina de la Universidad de Standford. Esta herramienta tiene muchas bondades, como la exportación de las ontologías a diversos formatos tales como RDFS, OWL o XML Schema entre otros.

En la figura 3.6 y 3.7 se muestran dos capturas del entorno donde fue diseñada la ontología en Protégé.

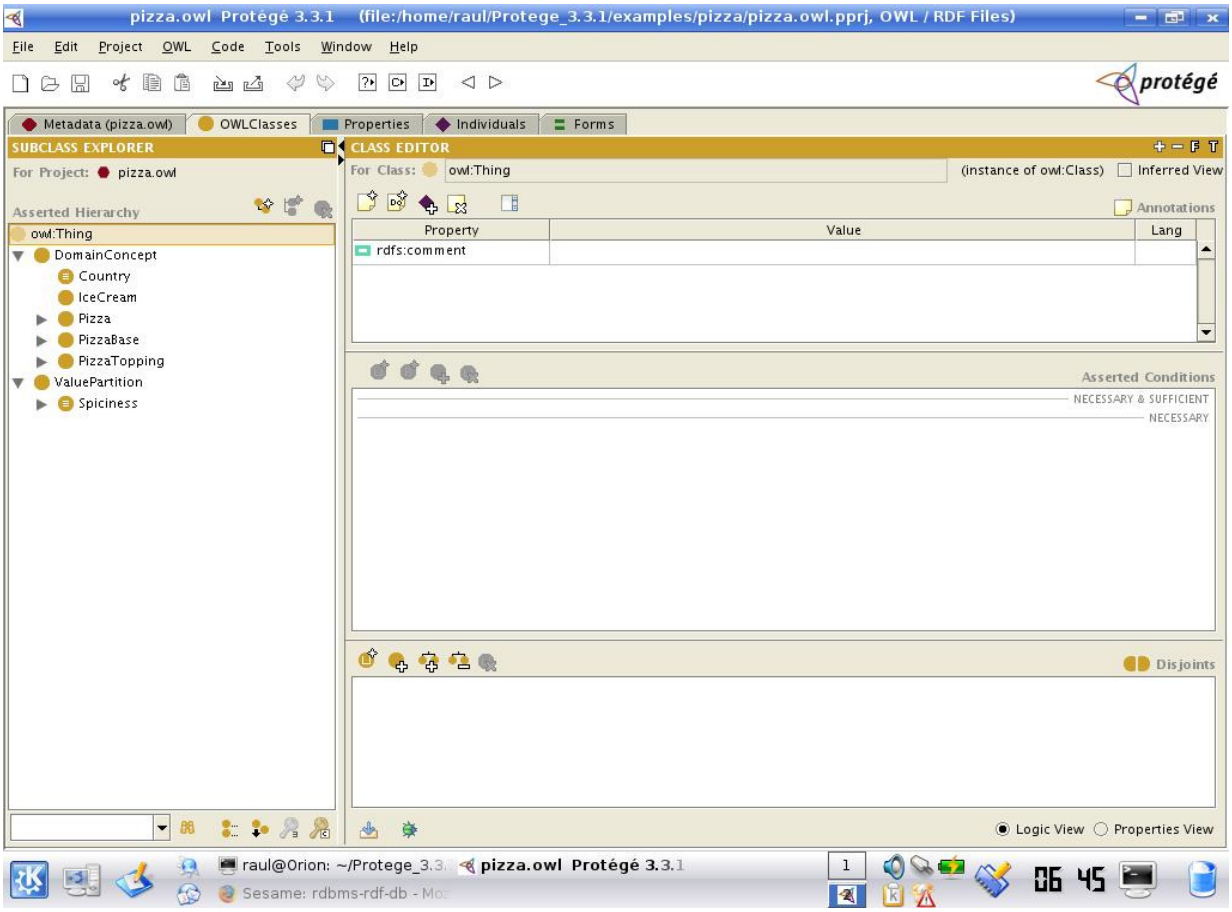

Figura 3.6: Entorno de desarrollo de ontologías.

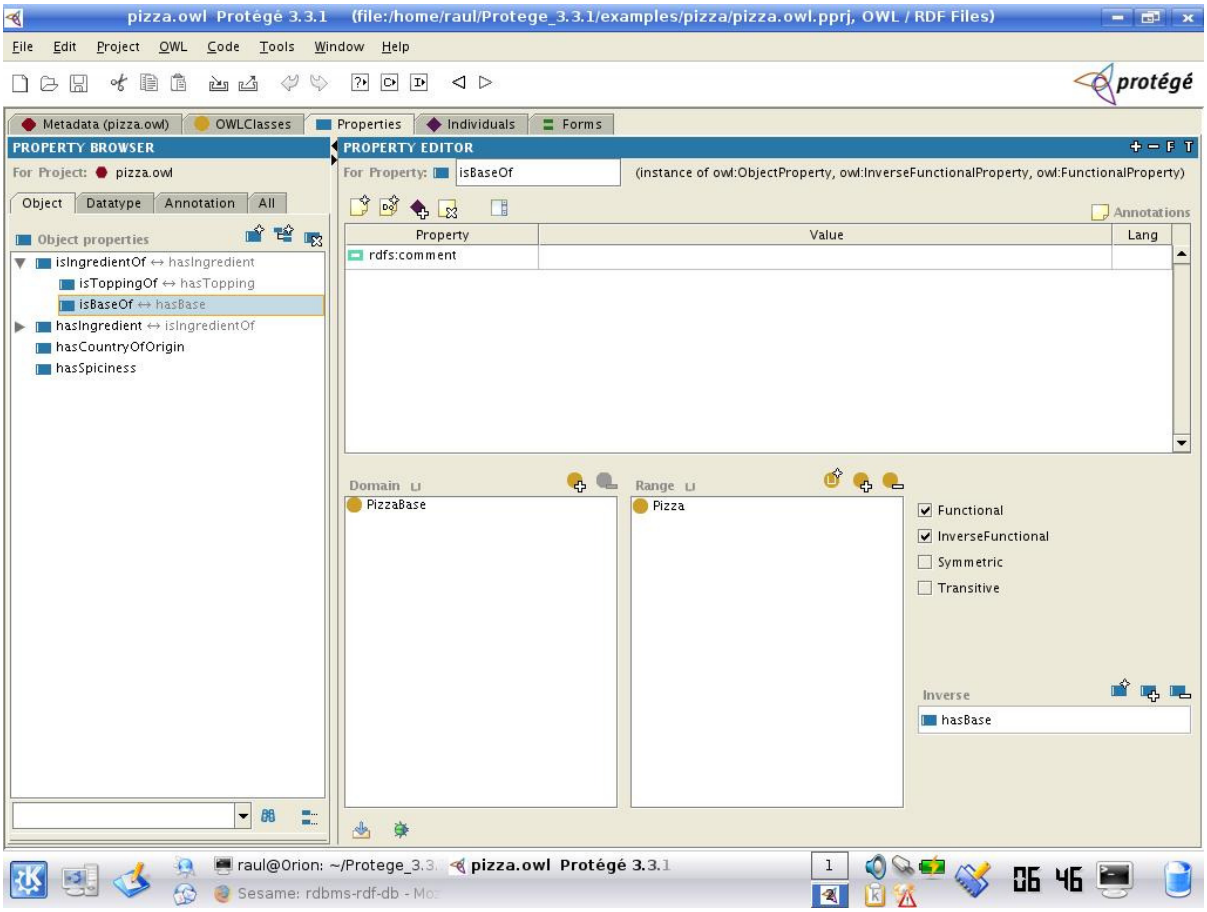

Figura 3.7: Entorno de desarrollo de ontologías.

3.5.2 Lenguaje para desarrollo de los Extractores: Perl

Por defecto se ha utilizado este lenguaje para desarrollar los módulos de extracción de metadatos que se escribieron.

- Módulos para extracción de metadatos de los documentos digitales (Imagen::ExifTool).

La Versión de Perl utilizada para el desarrollo de la herramienta es la 5.8.8.

Metadato abstraido	PDF	JPEG	MP3	DOC	AVI	PPT	XLS	WMA
name	Filename	Filename	Filename	Filename	Filename	Filename	Filename	Filename
date	CreateDate	CreateDate	-	CreateDate	-	CreateDate	CreateDate	CreationDate
author	Autor	FileSource	Artist	Autor	-	Author	Author	AlbumArtist
type	FileType	FileType	FileType	FileType	FileType	FileType	FileType	FileType
modifydate	ModifyDate	ModifyDate	FileModifyDate	ModifyDate	FileModifyDate	FileModifyDate	ModifyDate	FileModifyDate
filesize	FileSize	FileSize	FileSize	FileSize	FileSize	FileSize	FileSize	FileSize
creator	Creador	Make	-	Company	-	Company	Company	Composer
mimetype	MIMEType	MIMEType	MIMEType	MIMEType	MIMEType	MIMEType	MIMEType	MIMEType
title	Title	-	Title	Title	-	Title	-	-
pagecount	PageCount	-	-	PageCount	-	Slides	-	-
Duration	-	-	Duration	-	Duration	-	-	PlayDuration
totaledittime	-	-	-	Totaledittime	-	Totaledittime	-	-
paragraphs	-	-	-	Paragraphs		Paragraphs	-	-
wordcount	-	-	-	Wordcount	-	Wordcount	-	-

Cuadro 3.2: Metadatos comunes al menos dos tipos de documentos.

4. DISEÑO E IMPLEMENTACIÓN DE LA HERRAMIENTA

4.1 Diseño de una Ontología sobre Documentos Digitales

Como ya hemos visto, una ontología está formada por clases, que representan conceptos, propiedades que tienen esas clases y restricciones sobre dichas propiedades. Además si añadimos instancias (representaciones concretas de las clases), tendremos una base de conocimiento.

La herramienta de indexación que estamos diseñando extrae metainformación de los documentos digitales y realiza una representación semántica de la misma. Para esto, necesita basarse en una ontología sobre documentos digitales, cuyo diseño se expone a continuación.

La metodología seguida para el desarrollo de la ontología sobre documentos digitales es la recomendada por Noy y McGuinness (Universidad de Standford) en su documento Ontology Development 101: A Guide to Creating Your First Ontology[6]

Cabe señalar que los resultados mostrados en cada paso de la metodología son los definitivos para el modelado de la ontología, a los que se llegó después de iterar varias veces el proceso.

4.1.1 Determinación de dominio y ámbito de la ontología.

Lo primero que debemos hacer para crear una ontología es determinar el dominio que va cubrir su ámbito y para qué se va a utilizar.

Tomando en cuenta que dicha ontología será utilizada por una herramienta de indexación basada en la metainformación proporcionada por los documentos, por lo que la información que no pueda ser deducida de sus metadatos no tiene por qué ser considerada

en la ontología a excepción de algunas propiedades que se le pueden anexar a los archivos que puede ser de mucha utilidad.

Preguntas de capacitación

- ¿Cual es el nombre, autor, tamaño, tipo de archivo, tipo mime, fecha de creación ó modificación de un documento?
- ¿Cual es la orientación, ancho, alto, calidad de una imagen, cámara utilizada para tomarla?
- ¿Que cantidad de paginas contiene un documentos doc o pdf?
- ¿A que facultad y carrera puede pertenecer un documento?
- ¿Cuál es el género, artista, año o álbum de un determinado archivo de sonido?

A partir de este pequeño cuestionario ya podemos deducir basado en los metadatos disponible en cada tipo de documento, cual es la información que será utilizada para el desarrollo de la ontología.

4.1.2 Considerar la reutilización de ontologías existentes.

Dublín Core como se mencionó anteriormente es un vocabulario compuesto por quince elementos el cual se ha convertido en un estándar de facto a la hora de catalogar recursos.

En esta parte, cabe señalar que la ontología en cuestión utilizará un subconjunto de los metadatos Dublín core entre los que se pueden destacar los siguientes:

- Title
- Creador
- Date
- Format
- Identifier
- Contributor
- Publisher

4.1.3 Enumeración de términos importantes.

Una vez modeladas el tipo de preguntas que deben responderse basándose en la ontología, debemos buscar que términos (propiedades) se vuelven relevantes los cuales posteriormente se agruparan para formando conceptos llamados clases.

Los términos que se utilizarán el diseño de esta ontología corresponden a los metadatos extraídos de los documentos, los cuales en mas de un documento puede variar su nombre pero no su concepto, para este tipo de términos se abstraerán en uno solo.

Debemos tomar en cuenta, que si bien es cierto la mayoría de los metadatos de los documentos en su plenitud sintáctica no son iguales, existen ciertos términos comunes al menos a dos tipos de documentos, por lo cual se buscará un nombre común que los abstraiga.

Además de estos términos mencionados anteriormente, existen otros conceptos relacionados con la ontología, tales como:

- Document
- PDF Document
- DOC Document
- Exif Document
- MP3 Document
- AVI Document
- WMA Document
- XLS Document
- PPT Document

En la tabla 3.2 presentada anteriormente se puede ver los metadatos comunes al menos a dos tipos de documentos.

4.1.4 Definición de Clases y su Taxonomía.

De acuerdo a este análisis, los términos expuestos anteriormente que describen objetos con existencia independiente tenemos los siguientes:

- *Document*: representa los documentos.
- *PDF Document*: representa los documentos formato *pdf*.
- *DOC Document*: representa documentos con extensión *doc*.
- *MP3 Document*: representa documentos de sonido.
- *EXIF Document*. Representa los archivos de imagen.
- AVI Document: Representa archivos de video con extensión AVI.
- WMA Document: Representa archivos de sonidos de Window Media xxxx.
- XLS Document: Archivos de Excel.
- PPT Document: Archivos de Presentaciones PowerPoint.

Por lo tanto, estas serán las clases existentes en nuestra ontología, escribiendo sus nombres de acuerdo a la convención establecida por la nomenclatura anteriormente establecida.

Teniendo en cuenta estas aseveraciones, la manera de organizar la jerarquía en nuestra ontología es la siguiente:

- Document (única clase raíz existente en nuestra ontología).
  - PDF_Document
  - DOC_Document
  - EXIF_Document
  - MP3_Document
  - AVI_Document
  - PPT_Document
  - XLS_Document
  - WMA_Document

4.1.5 Propiedades de las clases.

Una vez definidas las clases, el siguiente paso es la definición de propiedades sobre esas clases. Para esto, se observarán los términos de la lista y se determinará cuales pueden ser propiedades, asignándolas a la clase que describen.

A partir de los términos que se eligieron para agruparse en las clases como propiedades se muestran tantos las propiedades tanto de la clase raíz (cuadro 4.1, también se incluye el espacio de nombres para diferenciar los elementos utilizados del Dublín core), así como de las subclases (cuadro 4.2).

Clase *Document*	**Espacio de Nombres**
autor	http://localhost/documentos.owl
creator	http://purl.org/dc/elements/1.1/
date	http://purl.org/dc/elements/1.1/
modifydate	http://localhost/documentos.owl
name	http://purl.org/dc/elements/1.1/
format	http://purl.org/dc/elements/1.1/
filesize	http://localhost/documentos.owl
title	http://purl.org/dc/elements/1.1/
pagecount	http://localhost/documentos.owl
contributor	http://purl.org/dc/elements/1.1/
publisher	http://purl.org/dc/elements/1.1/
filetype	http://localhost/documentos.owl
duration	http://localhost/documentos.owl
totaledittime	http://localhost/documentos.owl
paragraphs	http://localhost/documentos.owl
wordcount	http://localhost/documentos.owl

Cuadro 4.1: Propiedades de la clase raíz de la ontología sobre documentos digitales.

4.1.6 Definición de restricciones de las propiedades.

Después de la asignación de propiedades a cada una de las clases, empezaremos a definir las restricciones sobre dichas propiedades.

Concerniente a nuestra ontología la cardinalidad de las propiedades podrá ser sencilla, múltiple o igual a 0(solo en subclases).

PDF_Document	***DOC_Document***	***MP3_Document***	***EXIF_Document***	***AVI_Document***	***PPT_Document***
Pdfversion	charcount lines	Genre year album audiobitrate	orientation quality width height	Compression Encoding	CurrentUser

Cuadro 4.2 Propiedades de las subclases de la clase *Document* de la ontología sobre documentos digitales.

Propiedad	**Cardinalidad**	**Tipo de valor**	**Dominio**	**Rango**
autor	Multiple	cadena de caracteres	*Document*	string
creador	Multiple	cadena de caracteres	*Document*	string
date	Simple	fecha y hora	*Document*	datetime
modifydate	Simple	fecha y hora	*Document*	datetime
name	Simple	cadena de caracteres	*Document*	string
format	Simple	cadena de caracteres	*Document*	string
filesize	Simple	cadena de caracteres	*Document*	string
title	Simple	cadena de caracteres	*Document*	string
pagecount	Simple	numero entero	*Document*	integer
contributor	Simple	cadena de caracteres	*Document*	string
publisher	Simple	cadena de caracteres	*Document*	string
totaledittime	Simple	cadena de caracteres	*Document*	string
paragraphs	Simple	numero entero	*Document*	integer
wordcount	Simple	numero entero	*Document*	integer

Cuadro 4.3: Restricciones sobre las propiedades de la clase raíz de la ontología de documentos digitales.

Propiedad	Cardinalidad	Tipo de valor	Dominio	Rango
Propiedades heredadas de la clase *Document*				
duration	0	-	-	-
Propiedades especificas de la subclase				
Charcount	Simple	número entero	*DOC_Document*	integer
Lines	Simple	número entero	*DOC_Document*	integer

Cuadro 4.4: Restricciones sobre las propiedades de la clase DOC_Document, subclase de *Document*.

Propiedad	Cardinalidad	Tipo de valor	Dominio	Rango
Propiedades heredadas de la clase *Document*				
Date pagecount creator totaledittime wordcount	0	-	-	-
Propiedades específicas de la subclase				
Year	simple	numero entero	*MP3_Document*	Integer
Genre	simple	cadena de caracteres	*MP3_Document*	String
Album	simple	cadena de caracteres	*MP3_Document*	String

Cuadro 4.5: Restricciones sobre las propiedades de la clase *MP3_Document*, subclase de *Document*.

Propiedad	Cardinalidad	Tipo de valor	Dominio	Rango
Propiedades heredadas de la clase *Document*				
title pagecount duration wordcount	0	-	-	-

paragraphs Totaledittime				
Propiedades específicas de la subclase				
orientation	simple	cadena de caracteres	*EXIF_Document*	string
width	simple	numero entero	*EXIF_Document*	integer
height	simple	numero entero	*EXIF_Document*	integer

Cuadro 4.6: Restricciones sobre las propiedades de la clase *EXIF_Document*, subclase de *Document*.

Propiedad	**Cardinalidad**	**Tipo de valor**	**Dominio**	**Rango**
Propiedades heredadas de la clase *Document*				
totaledittime paragraphs wordcount duration	0	-	-	-
Propiedades específicas de la subclase				
pdfversion	simple	cadena de caracteres	*PDF_Document*	string

Cuadro 4.7 Restricciones sobre las propiedades de la clase *PDF_Document*, subclase de Document.

4.1.7 Definición de instancias.

Una vez modelada la ontología, sólo faltarían las instancias para formar una base de conocimiento. En la definición de una instancia seguimos los tres pasos que se describen a continuación:

1. Elegir la clase a la que va a pertenecer la instancia.
2. Crear una instancia individual de esa clase.
3. Rellenar los valores de las propiedades.

Algunas veces es deseable impedir que ciertas clases puedan tener instancias. Este tipo de clases se denomina clases abstractas. En el caso nuestro, la clase *Document*, será una clase abstracta, ya que obligaremos que cualquier instancia pertenezca siempre a una de las ocho subclases de la ontología de documentos (*PDF_Document*, *DOC_Document*, *MP3_Document*, *EXIF_Document, AVI_Document, WMA_Document, XLS_Document, PPT_Document*), de esta manera un documento nunca será un documento no tipado.

A continuación se muestra una instancia que representa un documento *pdf*. Es una instancia de la clase *PDF_Document* donde se pueden ver los valores que pueden tomar las propiedades de dicha clase para esta instancia en particular.

Propiedad	**Tipo de valor**	**Valor**
Name	*String*	*Advanced_sql_injection.pdf*
Autor	*String*	*Chris Anley*
Creador	*String*	*Chris Anley*
Title	*String*	*Advanced SQL Injection in SQL Server Applications*
Date	*Datetime*	*2002:01:31 15:20:13Z*
Modifydate	*Datetime*	*2002:01:31 15:22:18Z*
Filetype	*String*	*PDF*
Mimetype	*String*	*application/pdf*
Filesize	*String*	*291 kb.*
Pagecount	*Integer*	*25*
Pdfversion	*String*	*1.4*

Cuadro : Ejemplo de una instancia perteneciente a la clase PDF_Document.

4.1.8 Ontología sobre documentos Digitales.

En la figura 4.1 se muestra un diagrama de la ontología sobre documentos digitales modelada, en la cual se representan las clases y sus propiedades. En las subclases únicamente se presentan las propiedades específicas, no las heredadas.

Como paso final en la elaboración de la ontología, se hace necesario representarla en un lenguaje como OWL, así la herramienta de indexación será capaz de trabajar con ella. Para ello se ha utilizado la herramienta Protegé desarrollada por la universidad de Standford de la cual ya se habló anteriormente.

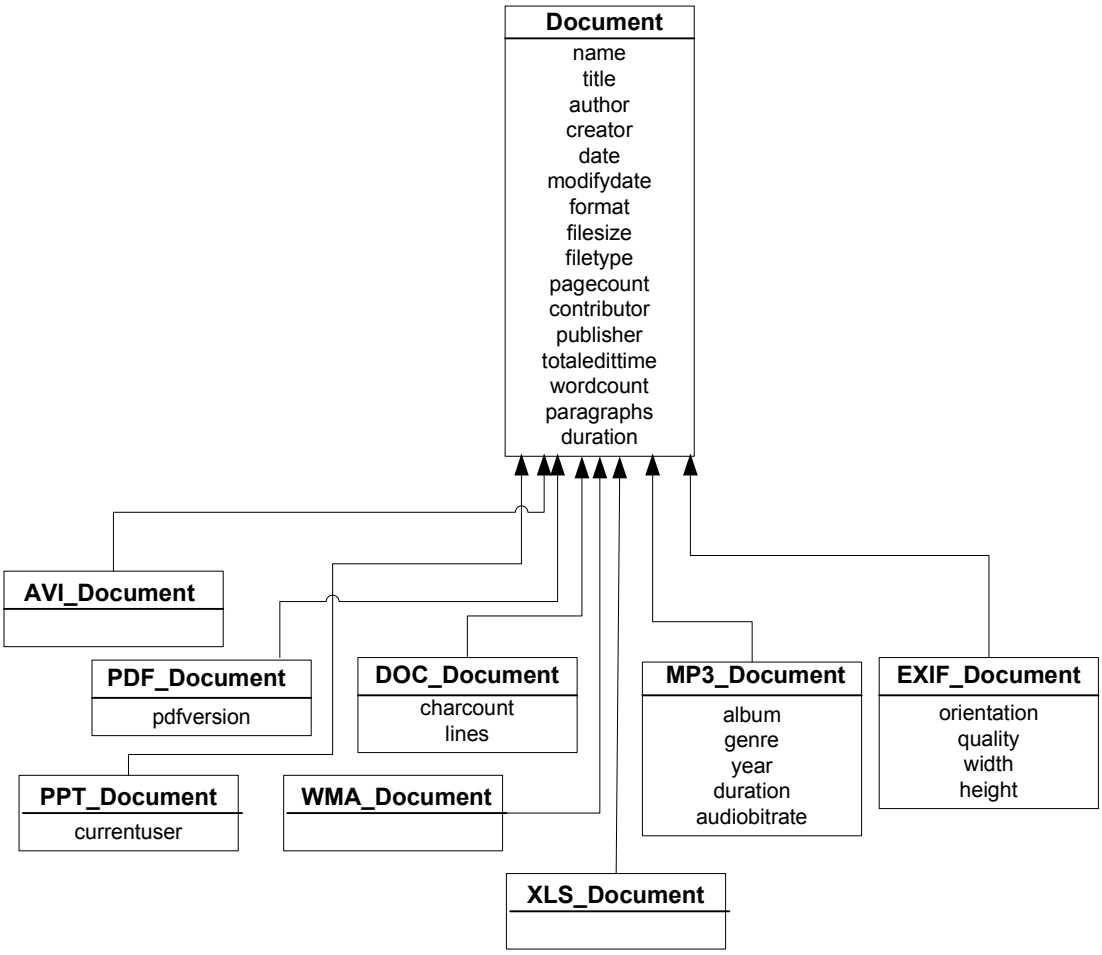

Figura 4.1: Diagrama de la ontología sobre documentos digitales. Clases y propiedades.

4.2 Módulos desarrollados y/o modificados.

Gracias a la modularidad del ya existente indexador de paquetes unix a partir del cual evoluciona este proyecto, no se necesito ningún tipo de cambio en los siguientes módulos:

- Modulo central: CentralIndexer. Encargado de centralizar todas las operaciones llevadas a cabo por el resto de módulos del indexador.

- Plugins de actualización de datos semánticos:

  - SesameUpdate. Actualiza los datos semánticos en la base de datos sobre RDF Sesame.

  - RDFUpdate. Actualiza los datos semánticos en un repositorio de archivos RDF.

Los módulos concretos desarrollados y/o modificados e implementados son los siguientes:

- Modulo generador de código RDF: *RDFGen.*

- Plugin procesador de ontologías: *OWLProcessor.*

- Plugins extractores de metadatos:

  - *PDFExtract*. Extrae la metainformación de documentos pdf.

  - *EXIFExtract*. Extrae la metainformación de imágenes jpeg.

  - *MP3Extract*. Extrae metadatos presentes en archivos de sonido mp3.

  - *DOCExtract*. Extrae la metainformación de archivos con extensión doc.

  - *AVIExtract*: Extrae metadatos de videos .avi

- *WMAExtract*: Extrae metainformación de archivos con extensión WMA

- *XLSExtract*: Extractor de metainformación en archivos de Excel.

- *PPTExtract*: Extrae metainformación de archivos de presentación power point.

A continuación se explicarán con más detalle cada uno de los módulos desarrollados. Cabe recordar que el lenguaje utilizado para la implementación de dichos módulos ha sido *Perl*. Particularmente los módulos desarrollados para extracción de metadatos se han apoyado en librerías previamente elaboradas las cuales se comentarán en determinado momento.

4.3 Módulo central: CentralIndexer

Como se comentaba previamente, el modulo central es el encargado de coordinar las labores de todos los demás módulos. Este módulo para determinar los plugins con que va a interactuar dispone del archivo de configuración denominado plugins.conf, el cual contiene una entrada por cada plugin, donde se puede destacar la siguiente información:

- Nombre que identifica el plugin (name).
- Tipo de plugin (class). Los posibles valores:
  - Ontology: indica que el plugin es un procesador de ontologías.
  - Extractor: indica que el plugin es un extractor de metadatos.
  - Updater: indica que el plugin es un actualizador de datos semánticos.
- Ruta del archivo de configuración que debe utilizar el plugin (config).
- Ruta donde se encuentra el plugin (path).

En el cuadro 6.10 se muestra el contenido del archivo de configuración plugins.conf para los plugins concretos desarrollados para la herramienta.

```
# Ontology Processor

plugin = Indexer::Plugin::Ontology::OWLProcess
class = ontology
name = OWL Processor
config = ontology.conf


# Metadata Extractors

plugin = Indexer::Plugin::MDExtraction::PDFExtract
class = extractor
name = PDF Documents Metadata Extractor
config = Extractors/TermEquivalences/pdf.equiv


plugin = Indexer::Plugin::MDExtraction::EXIFExtract
class = extractor
name = EXIF Documents Metadata Extractor
config = Extractors/TermEquivalences/exif.equiv


plugin = Indexer::Plugin::MDExtraction::MP3Extract
class = extractor
name = MP3 Documents Metadata Extractor
config = Extractors/TermEquivalences/mp3.equiv

plugin = Indexer::Plugin::MDExtraction::DOCExtract
class = extractor
name = DOC Documents Metadata Extractor
config = Extractors/TermEquivalences/doc.equiv

# Semantic Data Updaters

plugin = Indexer::Plugin::SDUpdate::RDFUpdate
class = updater
name = Pure RDF updater
config = SemanticStorage/rdf.conf
```

```
plugin = Indexer::Plugin::SDUpdate::SesameUpdate
class = updater
name = Sesame RDF database updater
config = SemanticStorage/sesame.conf
```

Cuadro 4.9: Porción del Archivo de configuración del indexador *plugins.conf*

Para que el sistema sea capaz de trabajar con *plugins* se ha utilizado el modulo de Perl *Module::Pluggable* (versión 3.5), que permite que todos los módulos en un determinado directorio puedan ser considerados como *plugins*.

La ventaja que esto implica, es el hecho de que pueden invocarse las rutinas de los módulos de manera dinámica. Los nombres de los módulos se pueden almacenar en variables tras extraerlos del archivo de configuración, y así evitamos escribirlos explícitamente en el código.

Esto permite ampliar la cantidad y nombre de los módulos tanto como se quiera, ya que bastará con tenerlos en un directorio específico y registrarlos en el archivo de configuración, sin necesidad de modificar el código del indexador.

Las operaciones que lleva a cabo el módulo central son las siguientes:

1. **Procesado de la Ontología**

    a. Extrae la ubicación de la ontología del archivo de configuración *ontology.conf* la pasa al plugin *OWLProcess* para su respectivo procesamiento.

    b. Obtiene además de la URI base de la ontología, las clases de la misma con sus respectivas propiedades.

2. Procesado de metadatos de los documentos.

a. Analiza el archivo de configuración *indexer.conf* y para cada repositorio de documentos extrae lo siguiente:
   - Nombre del repositorio (dirname).
   - Ruta del repositorio (path)
   - Enlace base del repositorio, a partir del cual se generará el enlace para la descarga del documento (link).
   - Opciones para sobreescritura de metadatos. Es posible especificar, utilizando los términos de la ontología, ciertos datos comunes a todos los documentos de un repositorio. En ese caso, el metadato extraído de los documentos se ignoraría, y se tomaría el especificado en este archivo de configuración.

   En el cuadro 4.10 se puede ver una porción del archivo de configuración *indexer.conf*, correspondiente a un repositorio de ejemplo.

b. Analiza el archivo de configuración *extractors.conf*, para saber qué *plugin* se debe utilizar para cada extensión de archivo. En el cuadro 6.12 se muestra el contenido de este archivo para los módulos desarrollados.

c. Por cada directorio y por cada documento
   - Obtiene la metainformacion del documento, proporcionando al *plugin* correspondiente la información sobre clases y propiedades extraída de la ontología e indicándole su archivo de equivalencias. Esta metainformación se obtiene en forma de pares propiedad-valor.
   - Genera un archivo RDF temporal con los triples que describen la metainformación obtenida, a través del módulo *RDFGen*.

3. **Actualización de datos semánticos.** Invoca cada *plugin* de actualización, proporcionándole la ubicación de su archivo de configuración y el directorio bajo el que se encuentran los archivos RDF temporales.

4. Elimina los archivos RDF temporales.

```
dirname = PDF example repository
path = /home/raul/documentos/pdf
link = ftp://localhost/pdf
```

Cuadro 4.10: Opciones para un repositorio de ejemplo en el archivo de configuración del indexador *indexer.conf*.

```
pdf = Indexer::Plugin::MDExtraction::PDFExtract

jpg = Indexer::Plugin::MDExtraction::EXIFExtract

mp3 = Indexer::Plugin::MDExtraction::MP3Extract

doc = Indexer::Plugin::MDExtraction::DOCExtract
```

Cuadro 4.11: Archivo de configuración del indexador *extractors.conf*

4.4 Modulo Procesador de la Ontología en OWL: *OWLProcess*

Este módulo es el encargo de procesar la ontología y recibe del modulo central la ubicación de la misma.

Dado que el archivo *owl* está expresado en RDF, este se apoya en un módulo ya implementado llamado *RDF::Simple::Parser*, el cual permite de una manera mas cómoda analizar la ontología obteniendo lo siguiente:

- Las clases.
- Las subclases de otras clases.

- Las propiedades asignadas a cada clase y subclases.
- El tipo de valor de cada propiedad.

A partir de esta información se devuelve al modulo central lo siguiente:

- Clases hoja, clases que no tienen ninguna subclase.
- Propiedades y su tipo de valor.

Es entonces debido al uso de este modulo procesador de la ontología, el cual analiza una ontología almacenada en un archivo externo, no es necesario modificar el código del indexador si la ontología es modificada, y además el indexador es capaz de basar su funcionamiento en cualquier ontología sencilla.

4.5 Módulos Extractores de metadatos.

Los módulos de extracción de metadatos reciben del módulo central la información proporcionada por el procesador de la ontología y la ubicación de su archivo de equivalencias.

Dicho archivo de equivalencias contienes pares de términos que indican la correspondencia entre un término de la ontología y un término de la metainformación extraída del documento. También indica cuando un término puede extraerse directamente de los metadatos o requiere de alguna operación adicional. De este modo si cambio la ontología, no es necesario modificar el código del modulo extractor, bastaría con modificar los términos del archivo de equivalencias.

Los módulos de extracción contienen tres tipos de rutinas:

- Rutinas comunes a todos los extractores de archivos.
- Rutinas específicas del tipo de archivo.
- Rutinas específicas de la ontología.

En nuestro caso, ya que tomamos como base un módulo cualquiera de los ya desarrollados utilizado para la indexación de paquetes Unix, los dos últimos ítems fueron cambiados para adaptarlo a nuestras necesidades.

El trabajo de un extractor de metadatos consiste en crear una instancia del archivo que está siendo analizado creando triples que los describen. Para esto, las tareas llevadas a cabo son las siguientes:

1. Extracción de metadatos del archivo. Rutinas dependientes del tipo de archivo.
2. Obtención de las propiedades directas e indirectas del archivo de equivalencias. Rutinas comunes a todos los extractores.
3. Obtención de los valores para cada propiedad del archivo en cuestión:

    - Propiedades directas: se generan los triples simplemente analizando los metadatos en bruto extraídos. Rutinas comunes a todos los extractores.

    - Propiedades indirectas: una rutina común a todos los extractores va invocando subrutinas cuyo nombre depende del tipo de archivo, y llevan a cabo operaciones específicas sobre determinado datos para obtener el valor de la propiedad.

Si se ha indicado en el archivo de configuración *indexer.conf* que alguna propiedad del archivo debe ser sobrescrita, se ignora el valor obtenido para esa propiedad en este paso y se toma el proporcionado por el archivo de configuración.

Todos los módulos extractores de metadatos se apoyan en un modulo de *Perl* ya desarrollado llamado *Image::ExifTool* que hace que la extracción de los mismos sea transparente al tipo de archivo.

ExifTool es una poderosa herramienta para manipular metadatos desarrollada por Phil Harvey, además de ser independiente de la plataforma, también está disponible como

una librería de *Perl* y una aplicación de la línea de comandos. Puede ser descargada desde *CPAN*[13] como un módulo *Perl* o como una aplicación de línea de comandos que entre otros sitios se encuentra disponible aquí[14].

Entre los tipos de archivos mas comunes que *ExifTool* puede analizar además de los que ya se exponen aquí se encuentran las extensiones MPEG, MPG, PNG, MP4, MOV, WAV, WMV, HTML, HTM, XHTML, FLV, DIVX Y BMP entre otros.

4.6 Módulo generador de RDF: *RDFGen*

Este modulo es el encargado de generar la serialización RDF/XML de la metainformación extraída de los documentos. En este modulo se hicieron ajustes para que fuera capaz de tratar con los espacios de nombres contenidos en la definición de la ontología sobre todo al utilizar elementos importados de ontologías que no fuera la base del documento OWL que se estuviera analizando.

El comportamiento descrito a continuación no ha sido alterado en forma alguna:

RDFGen es invocado una vez por cada archivo procesado, los parámetros que recibe del modulo central son los siguientes:

- URI base de la ontología, para incluirla en la definición de prefijos de *namespaces* de la cabecera RDF.

- Ruta del archivo temporal donde debe almacenarse la representación semántica del archivo. Con el propósito de no saturar un determinado directorio con una enorme cantidad de archivos y agilizar el proceso de actualización de datos semánticos, los archivos temporales se almacenan en directorios creados dinámicamente de la siguiente manera:

[13] http://www.cpan.org/authors/id/E/EX/EXIFTOOL/Image-ExifTool-7.30.tar.gz
[14] http://www.sno.phy.queensu.ca/~phil/exiftool/

- Un directorio por cada repositorio de archivos que analice.

- Dentro del mismo, un directorio por cada letra inicial de nombre del archivo.

De este modo, los archivos RDF temporales de todos los archivos de un mismo repositorio que empiezan por una misma letra son almacenados en el mismo directorio.

- Triples extraídos del archivo, que describen la instancia del archivo.

- Correspondencia de propiedades y sus tipos, en caso de existir alguna.

Las tareas llevadas a cabo por el modulo generador de RDF se detallan a continuación:

1. Creación del archivo temporal.

2. Escribe las cabeceras RDF del archivo temporal, incluyendo entre las definiciones de *namespaces* además de la URI base de la ontología utilizada, algún otro espacio presente de elementos reutilizados en la ontología.

3. Por cada instancia descrita en los triples recibidos:
    - Abre la descripción de la instancia, indicando la clase a la que pertenece.

      <rdf: Description rdf:about="http://purl.org/dc/elements/1.1/xml">

    - Por cada propiedad asociada a dicha instancia, crea la línea correspondiente a dicha propiedad.

      <name>xml</name>

    - Cierra la descripción de la instancia.

</rdf:Description>

4. Cierra el documento RDF.
   </rdf:RDF>

4.7 Módulos actualizadores de datos semánticos

Una vez generados los triples que describen los documentos indexados, el indexador recurre a la descarga de esa información sobre la base de datos RDF utilizando haciendo uso de los módulos actualizadores de RDF.

Las tareas generales que debe llevar a cabo un módulo actualizador de datos semánticos son las siguientes:

1. Borrar datos del repositorio existente. Esto se hace para evitar inconsistencias de datos, por si ha habido archivos que se han eliminado desde la última vez que se actualizó.

2. Recorrer el directorio de archivos temporales copiando los datos RDF de cada archivo al sistema de almacenamiento correspondiente.

La estructura y la forma de operar internamente de estos módulos dependen fuertemente del sistema de almacenamiento que se esté utilizando. A continuación se exponen dichos módulos con más detalle.

4.7.1 Módulo actualizador de un repositorio de archivos RDF: *RDFUpdate*.

El modelo *RDFUpdate* copia los archivos RDF temporales a un directorio permanente dado, para que estén disponibles por si se les quiere dar algún otro uso.

Su modo de funcionamiento es muy sencillo. El archivo de configuración le proporciona el directorio permanente bajo el cual se deben copiar los archivos. A partir de ahí:

1. Borra todos los archivos y subdirectorios que existan bajo el directorio objetivo.

2. Recorre uno a uno los directorios y archivos RDF que se encuentran bajo el directorio temporal, y los copia al directorio objetivo respetando la misma estructura de directorios, que se comentó al hablar del módulo RDFGen.

4.7.2 Modulo actualizador de la base de datos sobre RDF de Sesame: *SesameUpdate*.

Las tareas llevadas a cabo por el modulo actualizador de la base de datos RDF de Sesame son un poco mas complejas que las del otro plugin de actualización.

En el archivo de configuración de este módulos están especificados los parámetros utilizados para la conexión con la base de datos sobre RDF.

- Ubicación de Sesame (uri). Por defecto, http://localhost:8080/sesame.
- Nombre de usuario (username). Por defecto, sesame.
- Contraseña del usuario (password). Por defecto, sesame.
- Repositorio de sesame donde se almacenan los datos RDF (repository). Por defecto, rdbms-rdf-db, correspondiente a una base de datos relacional.
- Formato de serialización de los datos RDF (format): rdfxml.

- Opción de verificación de los datos antes de la actualización (verify): yes.

En el cuadro 4.16 se muestra el contenido de este archivo de configuración.

```
# This is the configuration file for the indexer plugin
# that updates the Sesame RDF database
# Sesame location (URI).
uri = http://localhost:8080/sesame

# Sesame username
# if not specified, only public repositories will be available
# username = sesame

username = sesame

# Sesame password to log into the server
# password = sesame

password = sesame

# Sesame repository that must be updated
# repository = rddms-rdf-db

repository = rdbms-rdf-db

# The following are some options about the uploading of RDF data
# please do not edit them

# RDF format: the format of the RDF file to be uploaded
# availablre options: rdfxml, ntriples, turtle
# format = rdfxml

format = rdfxml

# Verify data before uploading?
# available options: yes, no

verify = yes
```

Cuadro 4.12: Archivo de configuracion *sesame.conf*

Los parámetros presentados en este archivo, tales como ubicación, nombre de usuario, contraseña y repositorio deben ser modificados si se modifica la configuración de Sesame.

En cuanto al formato de serialización, aunque Sesame permite otros (*Ntriples*, Turtle), en este caso no se puede modificar, ya que los datos que se van a actualizar son los generados por el módulo *RDFGen*, que los escribe en *RDF/XML*.

Estos parámetros son imprescindibles para que este modulo sea capaz de comunicarse con Sesame a través de *HTTP* utilizados por un módulo *Perl* llamado *RDF::Sesame*. Básicamente las operaciones que lleva acabo *SesameUpdate* son las siguientes:

1. Establecer una conexión con Sesame, especificando la URI, usuario y contraseña.
2. Abrir el repositorio especificado.
3. Borrar toda la información del repositorio especificado.
4. Guardar los nuevos datos RDF en el repositorio (Sube los archivos RDF temporales).
5. Liberar la conexión con Sesame.

Todas estas operaciones se llevan a cabo utilizando el protocolo HTTP, lo que permite que la base de datos de Sesame se encuentre en un ordenador distinto al que aloja el indexador, como ya se había comentado.

4.8 Aplicación de búsqueda de documentos.

El último bloque de la herramienta de indexación es la aplicación de búsqueda, cuya misión consiste en ofrecer al usuario una interfaz que será la encargada de traducir las consultas del mismo a algún tipo de consulta en el lenguaje SeRQL.

En esta parte se ha reutiliza una interfaz ya desarrollada para el indexador de paquetes Unix mencionado anteriormente, a la cual simplemente se le han hecho algunos cambios de configuraciones.

Esta interfaz desarrollada en PHP fue cuidadosamente desarrollada separando en la medida de lo posible la vista de usuario y la funcionalidad de la aplicación, así que las funciones PHP han sido agrupadas por funcionalidad en distintos módulos PHP como se describen a continuación:

- ontology.php: funciones relacionadas con la ontología.
- forms.php: funciones que muestran los formularios con lo que el usuario realiza las búsquedas.
- search.php: encargo de traducir las búsquedas de usuario a consultas para la base de datos RDF.
- sesame.php: implementa acceso a la base de datos RDF.
- showresults.php: encargadas de mostrar los resultados obtenidos al usuario.

Existen además otros dos módulos, index.php y details.php, que implementan la vista inicial y de los detalles de los resultados respectivamente.

La bondad de esta aplicación, donde la mayoría de los parámetros utilizados son definidos en archivos de configuración, nos permitió que con pequeños ajustes se pudiera reutilizar la interfaz para la búsqueda de documentos digitales.

4.9 Funcionamiento de la Herramienta

A continuación se explica el funcionamiento de la herramienta mediante unos ejemplos, se verá la ejecución del indexador y por otro lado la interfaz utilizada para realizar las búsquedas sobre los documentos digitales.

4.9.1 Funcionamiento del Indexador.

La distribución de los directorios donde residen todos los módulos y archivos de configuración necesarios para su funcionamiento se describe en el cuadro 4.13.

```
|   indexer.conf
|
+---EndRDF
+---Indexer
|   |   CentralIndexer.pm
|   |   RDFGen.pm
|   |
|   +---Config
|   |   |   ontology.conf
|   |   |   plugins.conf
|   |   |
|   |   +---Extractors
|   |   |   |   extractors.conf
|   |   |   |
|   |   |   \---TermEquivalences
|   |   |           avi.equiv
|   |   |           doc.equiv
|   |   |           exif.equiv
|   |   |           mp3.equiv
|   |   |           pdf.equiv
|   |   |           ppt.equiv
|   |   |           wma.equiv
|   |   |           xls.equiv
|   |   |
|   |   \---SemanticStorage
|   |           rdf.conf
|   |           sesame.conf
|   |
|   +---Files
|   |   \---RDF
|   \---Plugin
|       +---MDExtraction
|       |       AVIExtract.pm
|       |       DOCExtract.pm
|       |       EXIFExtract.pm
|       |       MP3Extract.pm
|       |       PDFExtract.pm
|       |       PPTExtract.pm
|       |       WMAExtract.pm
|       |       XLSExtract.pm
|       |
|       +---Ontology
|       |       OWLProcess.pm
|       |       OWLProcess1.pm
|       |
|       \---SDUpdate
|               RDFUpdate.pm
|               SesameUpdate.pm
|
\---Ontology
        documentos.owl
```

Cuadro 4.13: Estructuras de directorios y archivos utilizados por el indexador

En el directorio Indexer se encuentran los módulos del indexador y los archivos de configuración, estos últimos se encuentra bajo el directorio *Config* y los módulos bajo *Plugin*, excepto el módulo central y el generador de RDF que se encuentran en la raíz del mismo.

Los archivos RDF temporales se almacenan en Files/RDF hasta que son eliminados por el indexador, la ontología por su parte se encuentra bajo el directorio Ontology mientras que en el directorio *EndRDF* se almacenan los archivos generados por el actualizador de datos semánticos *RDFUpdate*.

Para demostrar el funcionamiento del indexador se ha creado un directorio llamado prueba que a su vez contiene 4 directorios llamados pdf, jpeg, doc, mp3 con los siguientes archivos.

- jpeg
  - ✓ apoyo.jpg
  - ✓ bandera_nicaragua.jpg
  - ✓ escudo_nicaragua.jpg
  - ✓ mapa_politico
  - ✓ Masaya.jpg
  - ✓ nicaragua01.jpg
  - ✓ nicaragua0.jpg
  - ✓ nicaragua28.jpg
  - ✓ palacio.jpeg
  - ✓ ometepe_island_on_lake-nicaragua.jpg
- Mp3
  - ✓ Aventura - El Perdedor.mp3
  - ✓ Evanesence - Lithium.mp3
  - ✓ haddaway - haddaway - what is love.mp3
  - ✓ heroes del silencio - en los brazos de la fiebre.mp3
  - ✓ Perrozompopo - Entre Remolinos.mp3

- ✓ Within Temptation - 02 - See Who I Am.mp3
- ✓ Within Temptation - A Dangerous Mind.mp3
- ✓ Within Temptation - All I Need.mp3
- ✓ Within Temptation - Final destination.mp3
- ✓ Within Temptation - Forsaken.mp3
- ✓ Within Temptation - Mother Earth.mp3
- ✓ Within Temptation - Neverending Story.mp3

- Pdf
  - ✓ Linux Complete Command Reference.pdf
  - ✓ Mastering Active Directory For Windows Server 2003 (Sybex).pdf
  - ✓ VB.NET - Introduction.pdf
  - ✓ Wiley.Ubuntu.Linux.Bible.Jan.2007.pdf
- Doc
  - ✓ documentos migraciones sql server.doc
  - ✓ funcion select.doc
  - ✓ librerias.doc
  - ✓ replicacion.doc
  - ✓ salida.txt
  - ✓ sQLInjection.doc
  - ✓ Trucos Linux.doc

Una vez definidos los directorios que el indexador procesará, el siguiente paso es modificar el archivo de configuración del mismo (Cuadro 4.14), para indicarle donde se encuentran dichos directorios, y de esta manera también se pueden establecer algunas propiedades que serán sobrescritas o agregadas a los archivos.

```
# Example:
dirname = directorio1
path = /home/raul/pruebas/pdf
link = ftp://165.98.8.7/pub/documentos/pdf
contributor=ciencias
publisher=informatica

dirname = directorio2
path = /home/raul/pruebas/doc
```

```
link = ftp://165.98.8.7/pub/documentos/doc
contributor=ciencias
publisher=computacion

dirname = directorio3
path = /home/raul/pruebas/jpeg
link = ftp://165.98.8.7/pub/documentos/jpeg


dirname = directorio4
path = /home/raul/pruebas/mp3
link = ftp://165.98.8.7/pub/documentos/mp3
```

Cuadro 4.14: archivo de configuración modificado para que procese los archivos del ejemplo.

Una vez concluido este paso, entonces no queda mas que iniciar el indexador para lo cual ubicados en el directorio Indexer tecleamos los siguiente:

perl CentralIndexer.pm

Esto producirá una salida donde se va imprimiendo los pasos que el indexador va ejecutando en este caso para la configuración de ejemplo, como se puede ver en el cuadro 4.15

```
Processing ontology at /usr/local/indexador/Ontology/documentos.owl...
Ontology successfully processed

Starting metadata extraction

Processing files in /home/raul/pruebas/doc

Processing file /home/raul/pruebas/doc/librerias.doc
Processing file /home/raul/pruebas/doc/funcion select.doc
[…..]
Processing file /home/raul/pruebas/doc/documentos migraciones sql server.doc
Processing files in /home/raul/pruebas/mp3
Processing file /home/raul/pruebas/mp3/Within Temptation - Neverending Story.mp3
Processing file /home/raul/pruebas/mp3/Within Temptation - Mother Earth.mp3
Processing file /home/raul/pruebas/mp3/Perrozompopo - Entre Remolinos.mp3
```

```
Processing file /home/raul/pruebas/mp3/Aventura - El Perdedor.mp3
Processing file /home/raul/pruebas/mp3/Evanesence - Lithium.mp3
[……]
Processing files in /home/raul/pruebas/pdf
Processing file /home/raul/pruebas/pdf/Linux Complete Command Reference.pdf
[……]
Processing files in /home/raul/pruebas/jpeg
Processing file /home/raul/pruebas/jpeg/escudo_nicaragua.jpg
Processing file /home/raul/pruebas/jpeg/nicaragua28.jpg
Starting Sesame database update
Connecting to: http://localhost:8080/sesame
1...
Connected
Clearing repository
1...
Repository successfully cleared
Uploading data
Uploading VB.NET - Introduction.pdf.rdf RDF
1...
Successful upload: 13 triples uploaded
Uploading Wiley.Ubuntu.Linux.Bible.Jan.2007.pdf.rdf RDF
1...
Successful upload: 6 triples uploaded
Uploading Linux Complete Command Reference.pdf.rdf RDF
1...
Successful upload: 11 triples uploaded
Uploading Mastering Active Directory For Windows Server 2003 (Sybex).pdf.rdf RDF
1...
Successful upload: 14 triples uploaded
Uploading Within Temptation - Mother Earth.mp3.rdf RDF
1...
Successful upload: 14 triples uploaded
Uploading Within Temptation - All I Need.mp3.rdf RDF
1...
Successful upload: 15 triples uploaded
Uploading Within Temptation - Final destination.mp3.rdf RDF
1...
Successful upload: 15 triples uploaded
```

```
[…..]
Disconnecting from http://localhost:8080/sesame
Disconnected

Starting Pure RDF update in /usr/local/indexador/EndRDF

Clearing repository

Repository successfully cleared

Storing RDF data

Storing RDF on file /usr/local/indexador/EndRDF/pdf/v/VB.NET - Introduction.pdf.rdf
RDF stored on file /usr/local/indexador/EndRDF/pdf/v/VB.NET - Introduction.pdf.rdf
[…..]
Storing RDF on file /usr/local/indexador/EndRDF/mp3/e/Evanesence - Lithium.mp3.rdf
RDF stored on file /usr/local/indexador/EndRDF/mp3/e/Evanesence - Lithium.mp3.rdf

Pure RDF update on /usr/local/indexador/EndRDF finished

All operations finished. Bye
```

Cuadro 4.15: Operaciones realizadas por el indexador.

4.9.2 Funcionamiento de la Aplicación de búsqueda.

A Continuación se mostrarán capturas de pantalla de la interfaz gráfica las cuales no requieren de mucha explicación.

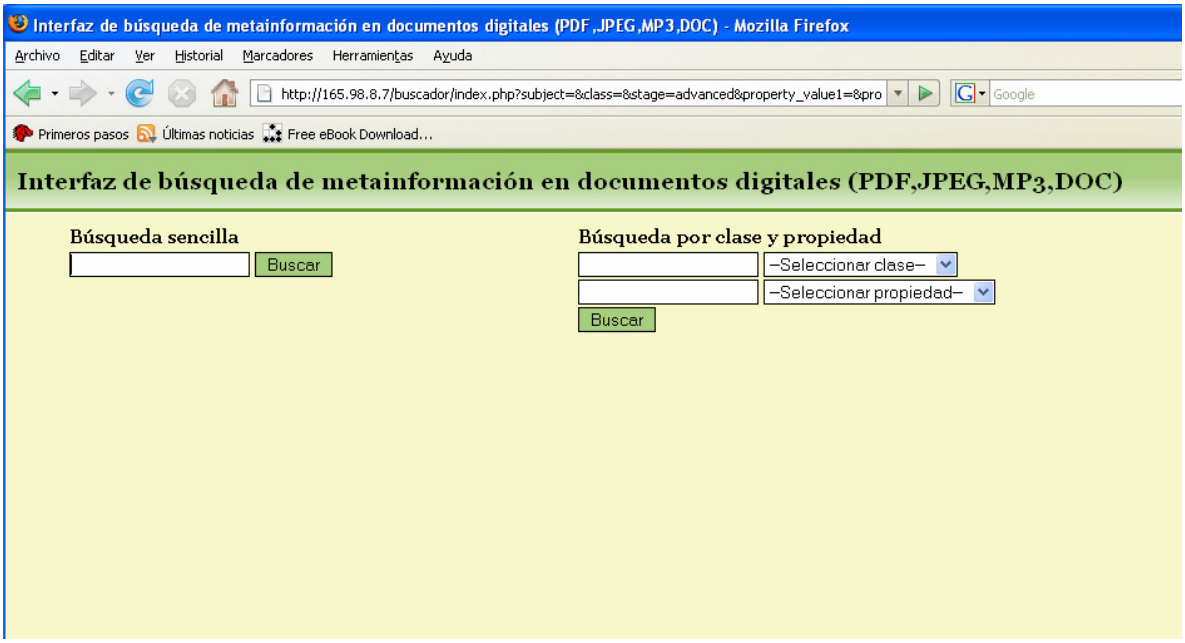

Figura 4.2: Pantalla Inicial de búsqueda sobre documentos digitales.

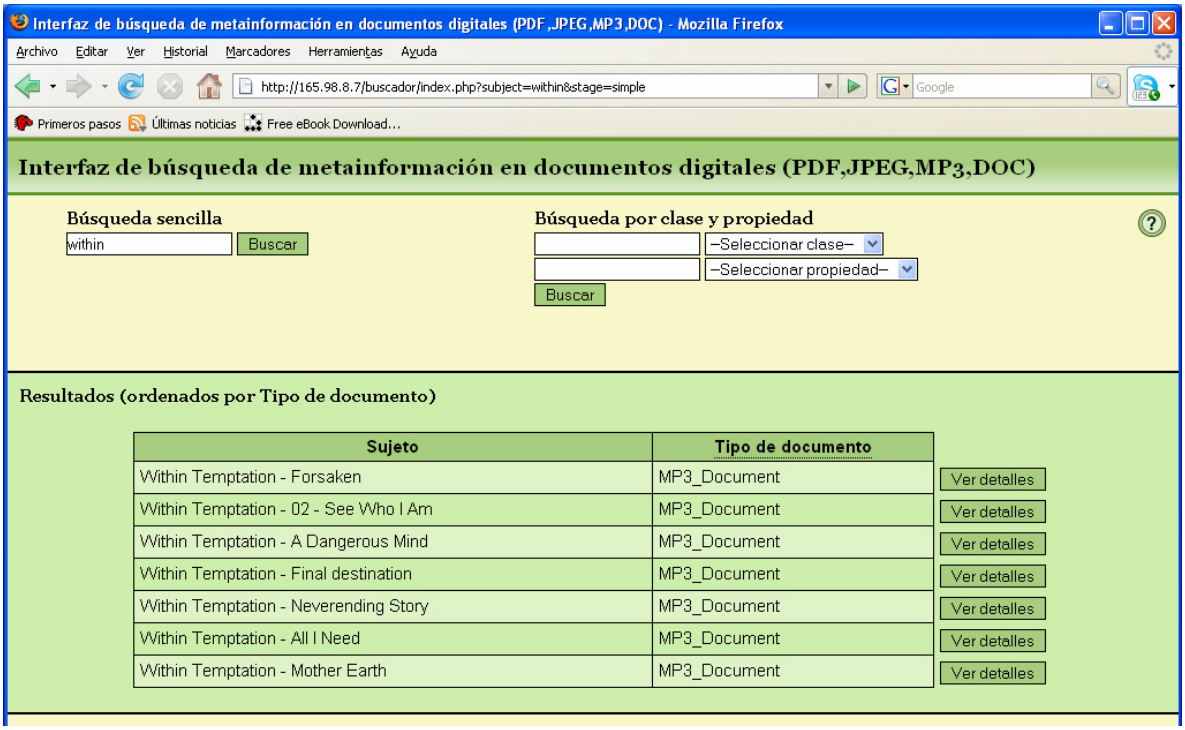

Figura 4.3: Ejemplo de los resultados de una búsqueda sencilla

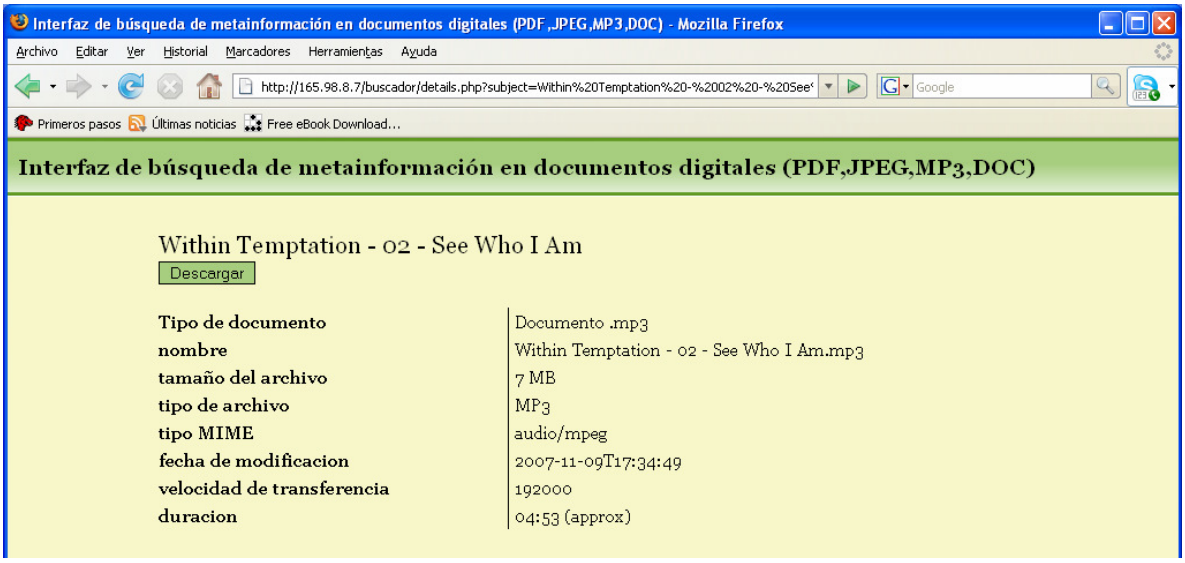

Figura 4.4: Detalles de un archivo MP3.

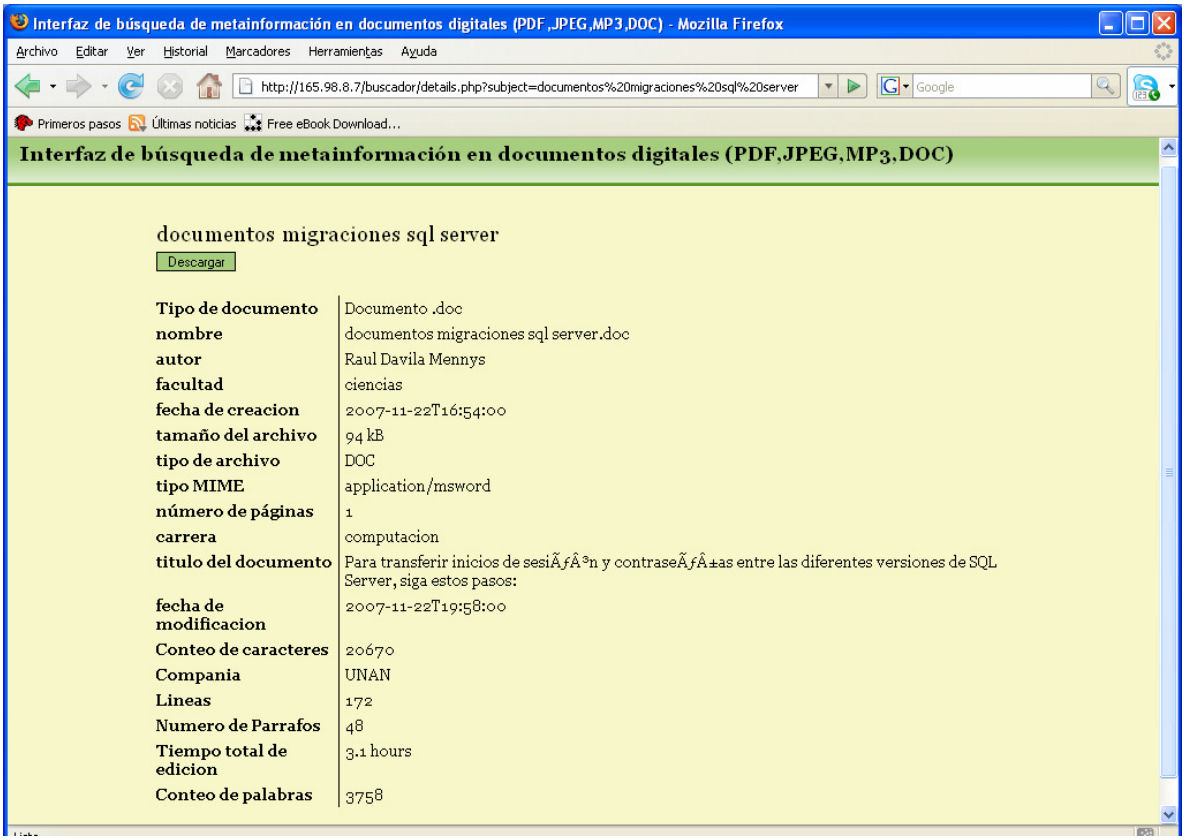

Figura 4.5: Detalles de un archivo con extensión .doc

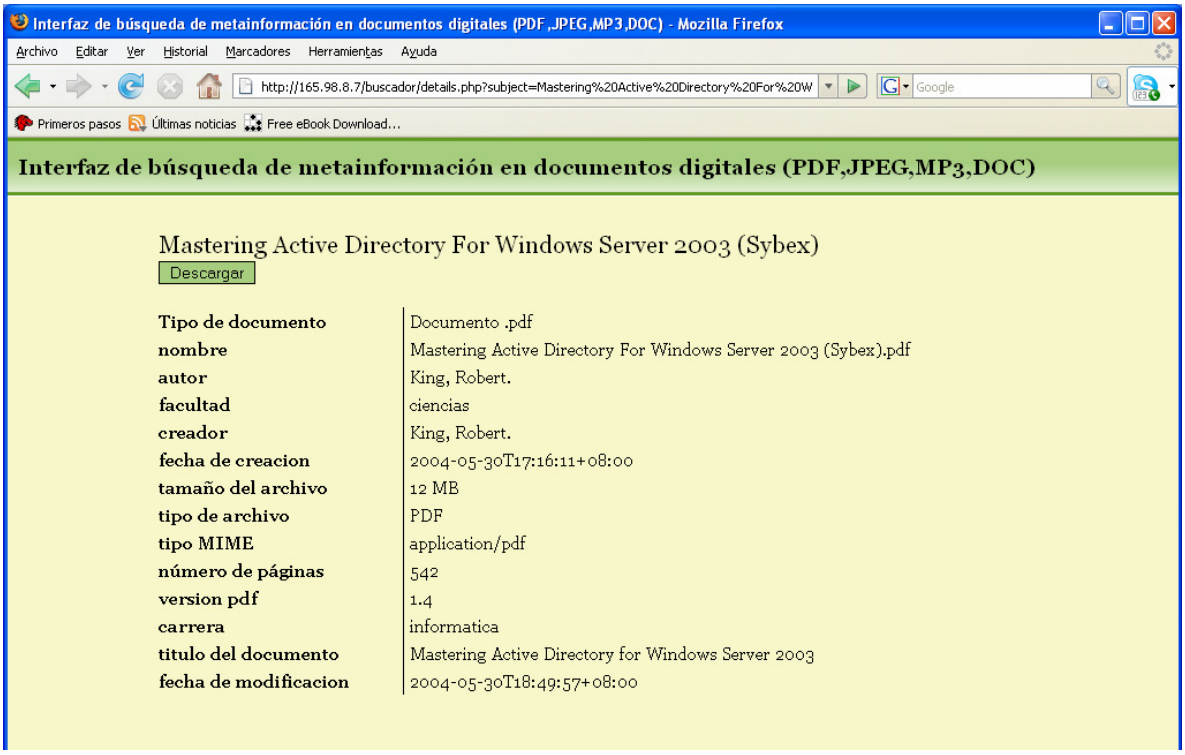

Figura 4.6: Detalles de un archivo PDF.

Figura 4.7: Detalle de un archivo JPEG.

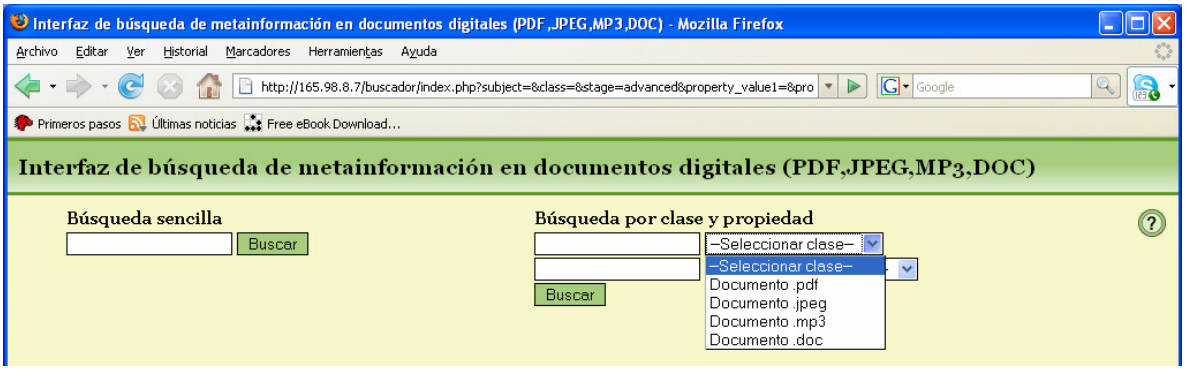

Figura 4.8: Aquí se pueden ver algunos de los tipos de documentos sobre los cuales buscar.

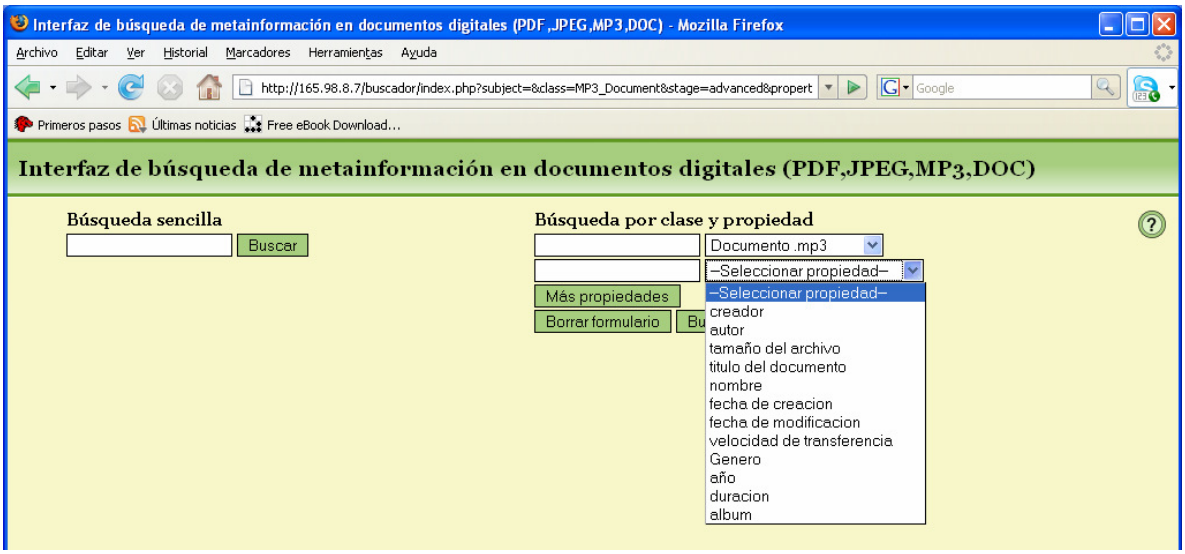

Figura 4.9: Despliegue de propiedades sobre las cuales se pueden hacer búsquedas muy específicas. En este caso para un documento mp3.

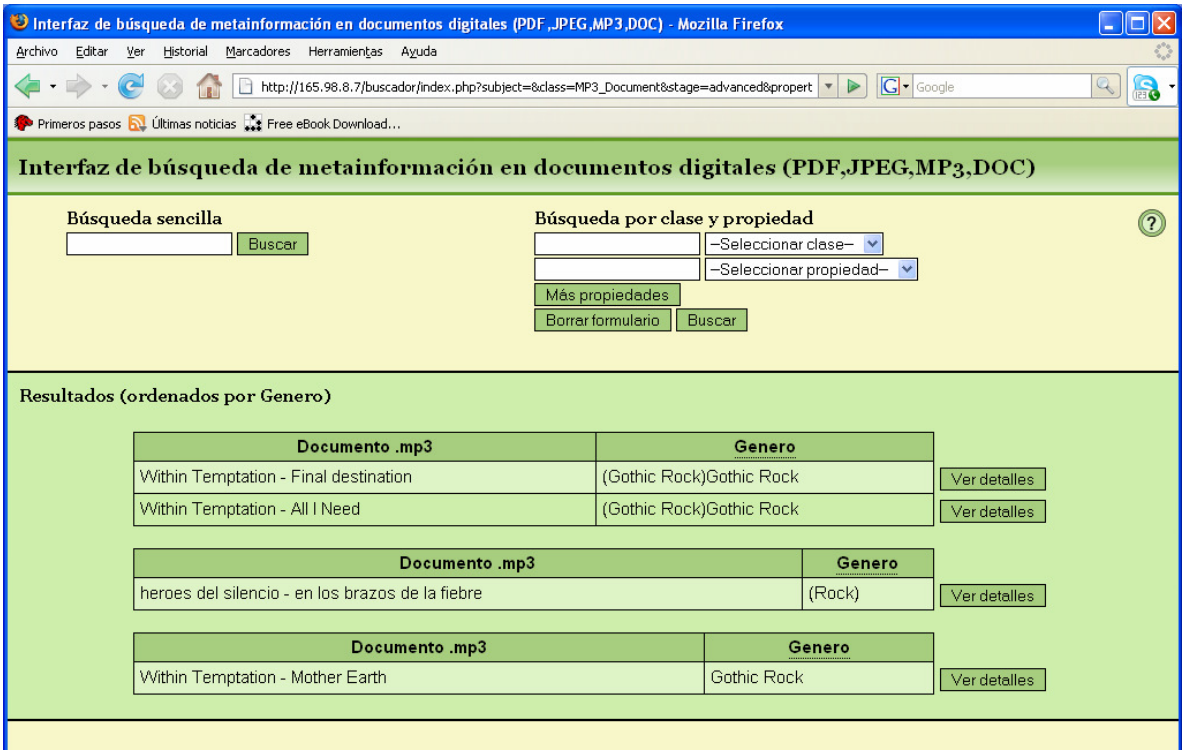

Figura 4.10: Resultado de búsqueda de cualquier documento mp3 cuyo general tenga la palabra rock. Aquí se puede ver resultados ordenados por género.

5. CONCLUSIONES Y RECOMENDACIONES.

Hoy en día, tenemos una colosal cantidad de información digital y nos agobia el hecho de no contar con herramientas que sean ágiles y eficientes a la hora de buscar información ya que la mayoría de ellas, sea cual sea la estrategia utilizada realizan búsquedas basadas en símbolos(sintáctica) no basada en significado(semántica).

Si bien es cierto existen herramientas de libre distribución muy interesantes en su manera de trabajar, su funcionamiento es completamente sintáctico y en alguna de ellas se intenta ofrecer tímidas búsquedas por algún campo que no sea el nombre, además tienen la desventaja que en su mayoría están orientados a documentos basados en texto.

En este documento se ha expuesto el desarrollo de una nueva herramienta de indexación de documentos, que permita búsquedas homogéneas independientes del formato de dicho documento. Esta homogeneidad de la representación de los metadatos disponibles en los documentos, se ha logrado utilizando técnicas de la Web Semántica.

Entre estas técnicas se destaca el uso de una ontología que describe el dominio sobre el que trabaja la herramienta, en este caso la ontología sobre documentos digitales la cual se representa mediante el estándar W3C, OWL y que viene a abstraer la información sobre el contenido de los documentos (metadatos) lo cual permite la homogeneidad que se perseguía.

Cabe mencionar que también es posible indexar la información contenida en los documentos (metadatos) utilizando una ontología ya existente como Dublín Core Simplificado, lo cual es una muestra de lo flexible que puede ser esta aplicación.

Dificultades encontradas

Quizá la mayor dificultad encontrada durante el desarrollo de esta herramienta además de la estrecha brecha de tiempo para su elaboración ha sido precisamente el modelado de la ontología sobre documentos digitales, el proceso de conceptualización de la metainformación proporcionada por los documentos, que es muy heterogénea.

Trabajo Futuro

1. Extender la indexación a los demás tipos de documentos soportados por *ExifTool* se podría hacer de manera transparente, requiriendo solamente un nuevo modelado de la ontología para incluir los metadatos nuevos que puedan aparecer en estos tipos de archivos.

2. Migración de la base de datos semántica en Sesame 1.4 a la última revisión de las versiones 2.0 las cuales cabe mencionar que son incompatibles, pero a diferencia de la versión 1.4; la revisión 2.0 y superiores de Sesame incluye soporte al lenguaje de consulta para bases de datos semánticas recomendación del W3C (SPARQL) dado que la aplicación actualmente se encuentra ligada a SeRQL, el cual es un lenguaje propio de Sésame, y por lo tanto, no es un estándar.

3. Sería interesante utilizar una ontología de alto nivel propuesta aquí[15] la cual sugiere que la mayoría de los estándares de metadatos pueden ser mapeados a una ontología de nivel superior de seis dimensiones la cual permite una integración conveniente entre una variedad de tipos heterogéneos de recursos y sus estándares de metadatos específicos. En este caso sería necesario realizar algún cambio al modulo generador de RDF.

[15] http://swe.uni-linz.ac.at/publications/pdf/TR-SE-07.05.pdf

6. Anexos

6.1 Ontologia documentos.owl

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:exif="http://www.w3.org/2003/12/exif/ns#"
    xmlns="http://localhost/documentos.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:dc="http://purl.org/dc/elements/1.1/#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://localhost/documentos.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="XLS_Document">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Document"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="AVI_Document">
    <rdfs:subClassOf rdf:resource="#Document"/>
  </owl:Class>
  <owl:Class rdf:ID="MP3_Document">
    <rdfs:subClassOf rdf:resource="#Document"/>
  </owl:Class>
  <owl:Class rdf:ID="DOC_Document">
    <rdfs:subClassOf rdf:resource="#Document"/>
  </owl:Class>
  <owl:Class rdf:ID="PPT_Document">
    <rdfs:subClassOf rdf:resource="#Document"/>
  </owl:Class>
  <owl:Class rdf:ID="WMA_Document">
    <rdfs:subClassOf rdf:resource="#Document"/>
  </owl:Class>
  <owl:Class rdf:ID="EXIF_Document">
    <rdfs:subClassOf rdf:resource="#Document"/>
```

```
  </owl:Class>
  <owl:Class rdf:ID="PDF_Document">
    <rdfs:subClassOf rdf:resource="#Document"/>
  </owl:Class>
  <owl:DatatypeProperty rdf:ID="author">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="link">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Document"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty
rdf:about="http://purl.org/dc/elements/1.1/#creator">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="company">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#DOC_Document"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty
rdf:about="http://purl.org/dc/elements/1.1/#date">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
    <rdfs:domain rdf:resource="#Document"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="modifydate">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="charcountwithspaces">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
```

```
    <rdfs:domain rdf:resource="#DOC_Document"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="duration">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Document"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="name">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="paragraphs">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Document"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty
rdf:about="http://purl.org/dc/elements/1.1/#contributor">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#PDF_Document"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="genre">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#MP3_Document"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty
rdf:about="http://www.w3.org/2003/12/exif/ns#orientation">
```

```
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#EXIF_Document"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="year">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#MP3_Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="quality">
    <rdfs:domain rdf:resource="#EXIF_Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="pdfversion">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#PDF_Document"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="totaledittime">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="format">
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="charcount">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#DOC_Document"/>
```

```
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="album">
    <rdfs:domain rdf:resource="#MP3_Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="encoding">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#AVI_Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty
rdf:about="http://purl.org/dc/elements/1.1/#publisher">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#PDF_Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty
rdf:about="http://purl.org/dc/elements/1.1/#title">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Document"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="filesize">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Document"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="audiobitrate">
    <rdfs:domain rdf:resource="#MP3_Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
```

```
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty
rdf:about="http://www.w3.org/2003/12/exif/ns#width">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#EXIF_Document"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="pagecount">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Document"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="wordcount">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="filetype">
    <rdfs:domain rdf:resource="#DOC_Document"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="compression">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#AVI_Document"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty
rdf:about="http://www.w3.org/2003/12/exif/ns#height">
    <rdfs:domain rdf:resource="#EXIF_Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
```

```
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="lines">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#DOC_Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="currentuser">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#PPT_Document"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:FunctionalProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.3.1, Build 430)
http://protege.stanford.edu -->
```

6.2 Archivos de Equivalencias.

```
# Equivalences documents ontology terms <=> pdf document terms

# is_a: indicates the class to which these properties apply

is_a: PDF_Document


name              <=>   filename
date              <=>   createdate
creator           <=>   creator
author            <=>   author
filesize          <=>   filesize
modifydate        <=>   modifydate
name              <=>   filename
filetype          <=>   filetype
format            <=>   mimetype
title             <=>   title
pagecount         <=>   pagecount
pdfversion        <=>   pdfversion
link              <=>   (indirect)link
contributor       <=>   (indirect)contributor
publisher         <=>   (indirect)publisher
```

Equivalencias entre términos de la ontología y metadatos de un documento pdf

```
# Equivalences documents ontology terms <=> doc document terms

# is_a: indicates the class to which these properties apply

is_a: DOC_Document


name              <=>   filename
date              <=>   createdate
author            <=>   author
filesize          <=>   filesize
modifydate        <=>   modifydate
filetype          <=>   filetype
format            <=>   mimetype
title             <=>   title
totaledittime     <=>   totaledittime
paragraphs        <=>   paragraphs
wordcount         <=>   wordcount
charcount         <=>   charcount
pagecount         <=>   pagecount
lines             <=>   lines
company           <=>   company
link              <=>   (indirect)link
contributor       <=>   (indirect)contributor
publisher         <=>   (indirect)publisher
```

Equivalencias entre términos de la ontología y metadatos de un documento doc

```
# Equivalences documents ontology terms <=> pdf document terms

# is_a: indicates the class to which these properties apply

is_a: MP3_Document


name              <=>   filename
creator           <=>   artist
author            <=>   artist
filesize          <=>   filesize
modifydate        <=>   filemodifydate
name              <=>   filename
filetype          <=>   filetype
format            <=>   mimetype
title             <=>   title
audiobitrate      <=>   audiobitrate
genre             <=>   genre
year              <=>   year
duration          <=>   duration
album             <=>   album
link              <=>   (indirect)link
facultad          <=>   (indirect)facultad
carrera           <=>   (indirect)carrera
```

Equivalencias entre términos de la ontología y metadatos de un documento mp3

```
# Equivalences documents ontology terms <=> jpg document terms

# is_a: indicates the class to which these properties apply

is_a: EXIF_Document


name              <=>   filename
date              <=>   createdate
creator           <=>   make
author            <=>   filesource
filesize          <=>   filesize
modifydate        <=>   modifydate
filetype          <=>   filetype
format            <=>   mimetype
title             <=>   filename
width             <=>   imagewidth
quality           <=>   quality
height            <=>   imageheight
orientation       <=>   orientation
link              <=>   (indirect)link
```

Equivalencias entre términos de la ontología y metadatos de un documento exif.

```
# Equivalences documents ontology terms <=> avi document terms

# is_a: indicates the class to which these properties apply
```

```
is_a: AVI_Document

name              <=>   filename
filesize          <=>   filesize
modifydate        <=>   filemodifydate
filetype          <=>   filetype
format            <=>   mimetype
compression       <=>   compression
encoding          <=>   encoding
link              <=>   (indirect)link
```

Equivalencias entre términos de la ontología y metadatos de un documento avi

```
# Equivalences documents ontology terms <=> wma document terms

# is_a: indicates the class to which these properties apply

is_a: WMA_Document

name              <=>   filename
date              <=>   creationdate
filesize          <=>   filesize
modifydate        <=>   filemodifydate
filetype          <=>   filetype
format            <=>   mimetype
link              <=>   (indirect)link
```

Equivalencias entre términos de la ontología y metadatos de un documento wma

```
# Equivalences documents ontology terms <=> xls document terms

# is_a: indicates the class to which these properties apply

is_a: XLS_Document

name              <=>   filename
date              <=>   createdate
author            <=>   author
filesize          <=>   filesize
modifydate        <=>   modifydate
filetype          <=>   filetype
format            <=>   mimetype
company           <=>   company
link              <=>   (indirect)link
contributor       <=>   (indirect)contributor
publisher         <=>   (indirect)publisher
```

Equivalencias entre términos de la ontología y metadatos de un documento xls

```
# Equivalences documents ontology terms <=> ppt document terms
```

```
# is_a: indicates the class to which these properties apply

is_a: PPT_Document


name              <=>   filename
date              <=>   createdate
author            <=>   author
filesize          <=>   filesize
modifydate        <=>   filemodifydate
filetype          <=>   filetype
format            <=>   mimetype
title             <=>   title
totaledittime     <=>   totaledittime
paragraphs        <=>   paragraphs
wordcount         <=>   wordcount
pagecount         <=>   slides
currentuser       <=>   currentuser
company           <=>   company
currentuser       <=>   currentuser
link              <=>   (indirect)link
contributor       <=>   (indirect)contributor
publisher         <=>   (indirect)publisher
```

Equivalencias entre términos de la ontología y metadatos de un documento ppt

REFERENCIAS

[1] Berners-Lee, T. (2001): "The Semantic Web", Scientific American.

[2] Andy Carvin, "Tim Berners-Lee: Weaving a Semantic Web", EDC Center for Media & Community, 2005-02-01. http://www.digitaldivide.net/articles/view.php?ArticleID=20

[3] Gruber T., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing" Technical Report KSL-93-04, Knowledge Systems Laboratory, Standford University, CA, 1993.

[4] DCMI, Dublin Core Metadata Elements Set. Version 1.1: Reference Description. Recomendación DCMI, Enero 2008. http://dublincore.org/documents/2008/01/14/dces/

[5] Ana Saiz García, "Indexador de Paquetes Unix", Universidad de Alcalá Henares, Septiembre 2007.

[6] Natalya F. Noy and Deborah L. McGuinness. Universidad de Standford. Ontology Development 101: A guide to Creating Your First Ontology, Última visita: 15/03/2008. http://protege.standord.edu/doc/users.html#tutorials.

[7] Ronald Maier and Johannes Sametinger. A top-level ontology for smart document access.
Ultima Visita: 15/03/2008. http://swe.uni-linz.ac.at/publications/abstract/TR-SE-07.05.html.

[8] David Fernandez Barrero. Integración del acceso a Sistemas de información mediante agentes SOAP Distribuidos. Universidad de Alcalá. 2004.