Planificación TFC / Diseño y población semiautomática de ontologías.

Autor: Daniel Fiel Cortes
Consultor: Sinuhe Arroyo Gomez

1. Introducción. 3
1.1. Objetivos. 3
1.2. Requisitos funcionales. 3
1.2.1. Conceptos Básicos. *4*
1.3. Requisitos Técnicos. 5
1.4. Planificación. 6
2. Web Semántica. 8
2.1. Un poco de Historia 8
3. Tecnologías Asociadas. 10
3.1. XML y XML Schema 10
3.2. RDF y RDF Schema 13
3.3. OWL 17
3.4. URI 19
4. Ontologías. 20
4.1. Definición 20
4.2. Métodos de Creación. 21
4.3. Conceptos Básicos Relacionados 27
5. DBpedia. 29
5.1. Historia. 29
5.2. DataSet 30
5.3. Consultas Sparql 30
6. Modelado de una Ontología. 33
6.1. Selección Dominio. 33
6.2. Proceso de modelado. 34
7. Herramienta. 41
7.1. Alcance. 41
7.2. Modulo de Importación. 42
7.3. Modulo de Navegación. 43
7.4. Demostración y Explicación Técnica de la Herramienta. 43
8. Bibliografia. 49
9. Anexo1. 50

1. Introducción.

En el curso y ejecución de este trabajo, ahondare en el concepto de web semántica, una realidad cada vez más tangente, que bajo el acrónimo de web 3.0 supondrá el relevo del actual modelo web.

Al tratarse de un campo de aplicación muy extenso, centraremos la temática en el diseño y población semiautomática de ontologías, siendo estas ultimas una pieza clave en el desarrollo y el éxito potencial de las tecnologías semánticas.

1.1. Objetivos.

A continuación enumeraré cuales son los principales objetivos a alcanzar con este trabajo.

- Estudio y comprensión del modelo planteado por el concepto de web semántica.
- Conocer las principales líneas de trabajo entorno a la web 3.0, así como comprender la historia y evolución de la misma.
- Trabajar en el concepto de ontología.
- Desarrollar una aplicación que permita la instanciacion de una ontología previamente creada, así como su navegación.

1.2. Requisitos funcionales.

Para alcanzar los objetivos anteriormente citados necesitamos definir una serie de requisitos funcionales.

En primer lugar a nivel conceptual definiré cuales son los conceptos básicos de obligados estudio y compresión con los que me tendré que familiarizar antes de abordar los desarrollos:

1.2.1. Conceptos Básicos.

En la definición de objetivos he definido como uno de los objetivos principales, el estudio y compresión de la web semántica. Dentro de este dominio prestare especial atención a:

Tecnologías relacionadas

- XML
- RDF y RDFS
- OWL
- URI
- DATASET
- SPARQL

Así mismo hare especial hincapié en DBPEDIA, al ser un elemento clave en nuestra aplicación. Es muy interesante analizar la plataforma, ya que el proyecto DBPEDIA a una varios de los conceptos críticos a considerar a la hora de abordar mi proyecto. En dicha herramienta se abordan temas como:

- Extracción estructurada de información.
- Marcado RDF
- Consultas SPARQL
- Extracción datos con SCALA
- Población de Infoboxes Wikipedia
- etc.…

Volviendo a los requisitos funcionales nuestra aplicación, soportara 2 funcionalidades principales:

- Importación. Que nos permitirá instanciar nuestra ontología, usando como origen de información, DBPedia.

- Navegación. Una vez poblada nuestra ontología, estableceremos los mecanismos necesarios para su correcta navegación. La idea es implementar un browser que através de links, nos permita explorar el contenido de la ontología.

Para abordarlo seguiremos un ciclo de vida en cascada:

- Análisis y Especificación.
- Diseño
- Desarrollo
- Pruebas

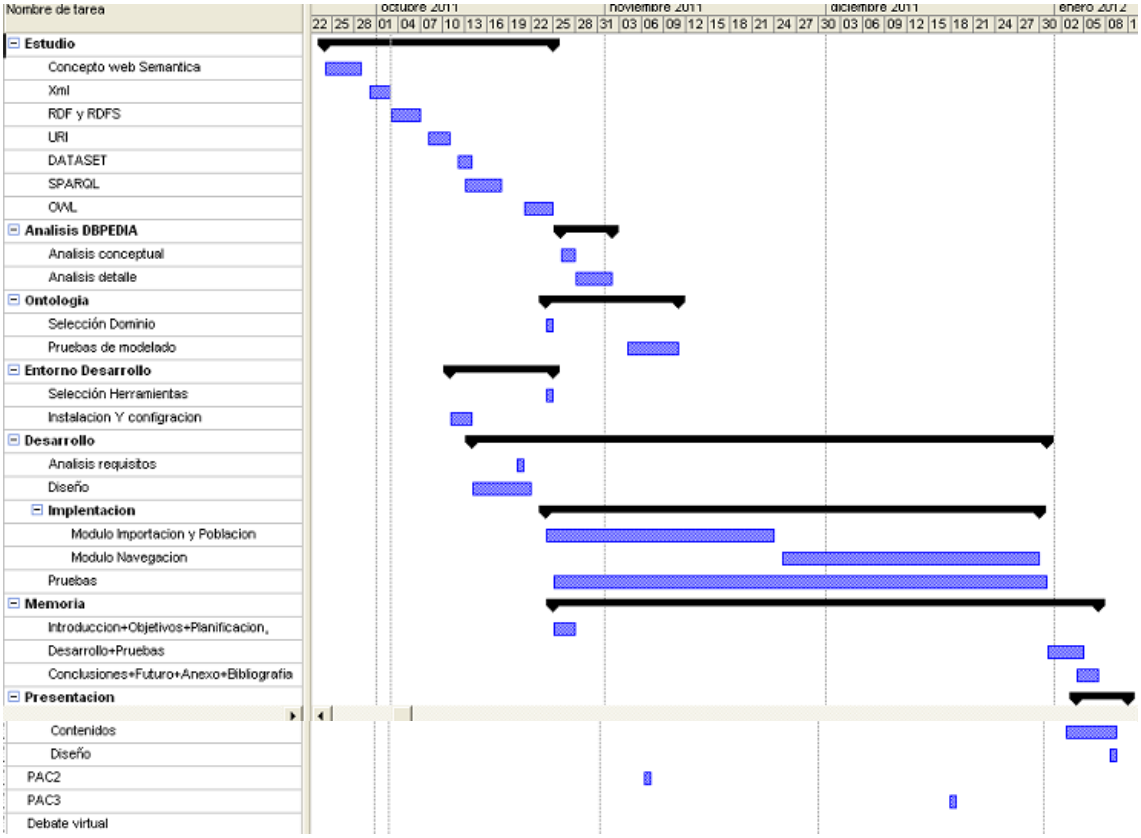
1.3. Requisitos Técnicos.

Con la premisa de utilizar herramientas opensource, necesitaremos:

- JDK java
- NetBeans
- API de la versión 3.x de Protege
- Jena 2.6.4 Library Plugin.
- ARQ 2.8.8

1.4. Planificación.

Nombre de tarea	Duración	Comienzo	Fin
Estudio	**25 días?**	**sáb 24/09/11**	**lun 24/10/11**
Concepto web Semantica	4 días?	sáb 24/09/11	mié 28/09/11
Xml	2 días?	vie 30/09/11	dom 02/10/11
RDF y RDFS	4 días?	lun 03/10/11	jue 06/10/11
URI	2 días?	sáb 08/10/11	lun 10/10/11
DATASET	2 días?	mié 12/10/11	jue 13/10/11
SPARQL	3 días?	jue 13/10/11	lun 17/10/11
OWL	3 días?	vie 21/10/11	lun 24/10/11
Analisis DBPEDIA	**6 días?**	**mié 26/10/11**	**mar 01/11/11**
Analisis conceptual	2 días	mié 26/10/11	jue 27/10/11
Analisis detalle	4 días?	vie 28/10/11	mar 01/11/11
Ontologia	**15 días?**	**lun 24/10/11**	**jue 10/11/11**
Selección Dominio	1 día?	lun 24/10/11	lun 24/10/11
Pruebas de modelado	5 días?	vie 04/11/11	jue 10/11/11
Entorno Desarrollo	**11 días?**	**mar 11/10/11**	**lun 24/10/11**
Selección Herramientas	1 día?	lun 24/10/11	lun 24/10/11
Instalacion Y configracion	3 días?	mar 11/10/11	jue 13/10/11
Desarrollo	**59 días?**	**vie 14/10/11**	**vie 30/12/11**
Analisis requisitos	1 día?	jue 20/10/11	jue 20/10/11
Diseño	6 días?	vie 14/10/11	vie 21/10/11
Implentacion	**51 días?**	**lun 24/10/11**	**jue 29/12/11**
Modulo Importacion y Poblacion	25 días?	lun 24/10/11	mié 23/11/11
Modulo Navegacion	25 días?	vie 25/11/11	jue 29/12/11
Pruebas	51 días?	mar 25/10/11	vie 30/12/11
Memoria	**57 días**	**mar 25/10/11**	**vie 06/01/12**
Introduccion+Objetivos+Planificacion,	3 días	mar 25/10/11	jue 27/10/11
Desarrollo+Pruebas	4 días	sáb 31/12/11	mié 04/01/12
Conclusiones+Futuro+Anexo+Bibliografia	3 días	mié 04/01/12	vie 06/01/12
Presentacion	**5 días?**	**mié 04/01/12**	**mar 10/01/12**
Contenidos	5 días	mié 04/01/12	mar 10/01/12
Diseño	1 día?	mar 10/01/12	mar 10/01/12
PAC2	1 día?	lun 07/11/11	lun 07/11/11
PAC3	1 día?	lun 19/12/11	lun 19/12/11
Debate virtual	1 día?	jue 26/01/12	jue 26/01/12


Nombre de tarea
octubre 2011
noviembre 2011
diciembre 2011
enero 2012
Estudio
Concepto web Semantica
Xml
RDF y RDFS
URI
DATASET
SPARQL
OWL
Analisis DBPEDIA
Analisis conceptual
Analisis detalle
Ontologia
Selección Dominio
Pruebas de modelado
Entorno Desarrollo
Selección Herramientas
Instalacion Y configracion
Desarrollo
Analisis requisitos
Diseño
Implentacion
Modulo Importacion y Poblacion
Modulo Navegacion
Pruebas
Memoria
Introduccion+Objetivos+Planificacion,
Desarrollo+Pruebas
Conclusiones+Futuro+Anexo+Bibliografia
Presentacion
Contenidos
Diseño
PAC2
PAC3
Debate virtual

2. Web Semántica.

«No es una Web separada sino una extensión de la actual, donde la información está dotada de un significado bien definido, los ordenadores están mejor capacitados y las personas trabajan en colaboración».

Estas palabras con las que Tim Berners-Lee, padre del concepto, define la web semántica, reúnen los fundamentos de este paradigma: Una evolución de la web actual tal y como la conocemos, hacia un nuevo modelo donde los ordenadores serán capaces de interpretar documentos dotados de contenido semántica, y el trabajo colaborativo será una realidad tangible.

Pese a ser una tecnología incipiente, y no consolidada en el mundo web, ríos de tinta han documentado en innumerables artículos las características o usos potenciales que se le atribuyen a la web semántica, y como esta representa un nuevo ciclo o etapa llena de posibilidades en la experiencia web.

El objetivo de este trabajo no es analizar minuciosamente el concepto teórico de la web semántica, sino tratar de incidir en establecer posible soluciones para una de sus debilidades, esto es la carencia de métodos automáticos de anotación, y la consecuente falta de contenidos semánticas en la web.

En torno a esto último centraremos el trabajo, pero es necesario hacer un breve acercamiento a la historia de la web semántica, que me permita contextualizar mi propuesta.

2.1. Un poco de Historia

Tal y como comente, al principio de este apartado, la Web Semántica es una propuesta de Tim Berners-Lee , quien a su vez fue el precursor del HTML y fundador de la W3C (Consorcio de la World Wide Web.)

Berners-Lee , publico en Septiembre de 1998 dos artículos, “Semantic Web Road Map” y “What the Semantic Web can represent” , que sirvieron para sentar las bases de lo que a la postre entendemos como tecnología Web semántica.

La intención original era clara, se pretendía mantener los principios fundamentales de la web actual como la descentralización y comparición, a la vez que se pretendía dotar a la misma de significado, propiciando un entorno mucho mas eficiente y mas adaptado a las necesidades reales de los usuarios.

En un principio el ideal semántica fue confundido o entremezclado con otros ámbitos científicos como la inteligencia artificial, aunque realmente , el paradigma de

documentos compresibles por ordenadores, no se basa en los métodos clásicos de ‘inteligencia artificial’, sino en una organización estructurada de la información.

Lo que conocemos como Web 2.0, e incluso la predecesora de esta misa, la Web 1.0, consiste básicamente en un conjunto enorme de páginas que contienen texto no estructurado. Los esfuerzos se han centralizado en la visualización de contenidos, independientemente de su estructura.

Hemos sido espectadores y participes de la mayor revolución tecnológica de la historia de la humanidad. Internet y la web, han pasado a ser elementos cotidianos e imprescindibles en la vida de muchas personas. Sin embargo este crecimiento desorbitado, no ha hecho sino recalcar las carencias en lo que a estructura de la información se refiere.

Comienzan a aparecer nuevos términos como Infoxicacion, y la sensación de poco criterio o acierto a la hora de realizar búsquedas en la web, es tangible.

No estamos preparados para manejar toda la información generada, y mucho menos para hacerlo con la carencia de estructura actual.

Los tan conocidos y socorridos buscadores, cuentan con algoritmos de búsquedas poco precisos y exhaustivos, por lo que resulta demasiado frecuente obtener resultados de búsquedas muy distintos a los esperados.

De esta manera palabras con varios significados pueden derivar en resultados no esperados, de la misma manera que la falta de control sobre sinónimos, pueden sesgar información importante en los resultados a obtener, al no incluir resultados asociados solo a una de las acepciones de la temática expuesta.

Para terminar de enumerar algunas de las pegas, no puedo dejar de mencionar otro de los puntos débiles a la hora de buscar información por la red, la fiabilidad. Encontramos mucha información , pero la fiabilidad de las fuentes no suele estar garantizada, por lo que consecuentemente la integridad de la información ofrecida como resultado de esas búsquedas puede no ser valida con todas las connotaciones que ello implica. Se trata de un tema delicado , pero se debe tener claro que el juicio sobre la veracidad y fiabilidad de la información no debería recaer en el usuario, a quien se le debería garantizar la veracidad de la misma.

Como solución a estos problemas, Tim Berners-Lee, idealiza un entorno donde através de un procesamiento de información previamente estructurada, las aplicaciones sean capaces de por ejemplo de discernir los distintos significados de las palabras, detectar relaciones de sinonimia entre palabras en un contexto temático concreto, recuperar información relacionada asociada a términos no indicados de manera explicita en la búsqueda, aplicar cierta lógica a los resultados a mostrar o clasificar la información según su veracidad. Es decir una web capaz de comprender y asimilar el contenido de la información y de proporcionar de manera automática resultados mas ricos favoreciendo la experiencia de navegación del usuario.

Para ello se define un sistema de etiquetas, que otorgan un valor semántica a la interpretación del contenido. Se establecen una serie de reglas para el marcado y anotación de la información, que implican la tan necesaria estructuración de los contenidos.
Estas definiciones y marcados siguiendo las normas establecidas, dan lugar a ontologías, que son la piedra angular de la web semántica.

3. Tecnologías Asociadas.

Existen una serie de estándares, frameworks y tecnologías asociadas a la creación, modelización y explotación de la web semánticas, que a continuación detallare:

3.1. XML y XML Schema

Se trata como su propio nombre indica (eXtensible Markup Language), de un lenguaje de marcado de datos, o metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).
Como lenguaje de marcado, su funcionalidad pasa por describir la estructura y atributos de un documento, para que este pueda ser interpretado por una maquina. Surge a partir del SGML (Standard generalizad markup lenguaje).
A nivel semántico, aporta la sintaxis superficial para los documentos estructurados, pero sin dotarles de ninguna restricción sobre el significado.

Entre sus bondades debo destacar la portabilidad y flexibilidad de los documentos generados.

Las etiquetas de las que se compone XML se denominan elementos, cada elemento se compone de una etiqueta de apertura “<nombre_tag>” y de una etiqueta de cierre “</nombre_tag>” y el contenido entre ambas es el valor del elemento.

Los documentos XML suelen comenzar con un prólogo, en el que se define la versión de XML y la codificación.

<?xml version="1.0" encoding="UTF-8"?>

Todos los documentos XML, sin excepción, deben estar bien estructurados. Esto significa que deben cumplirse varias reglas:

- XML debe ser utilizable sobre Internet.
- XML debe soportar variedad de aplicaciones.
- XML debe ser compatible con SGML
- Debe ser fácil la escritura de programas que soporten XML.
- Mínimo numero de características opcionales en XML.
- Los documentos XML deben ser legibles y claros.
- XML debe presentar un diseño formal y conciso.
- XML debe estar formalizado de manera simple.
- XML debe ser fácil de crear.
- La brevedad de las marcas es de minima importancia.

La flexibilidad de XML permite diferentes representaciones de la misma información, sin ambigüedad y bien representada. En consecuencia, es posible que la misma información tenga diferentes formatos XML lo que implica que XML no es adecuado para incluir semántica por si mismo, hay que construir por encima de él (como veremos con RDF y OWL).

Xml, puede asociar una gramática para definir un modelo de estructura de documento (DTD). Es aquí donde entra en juego XML Schema.
Xml Schema es un lenguaje de esquema que se emplea para describir la estructura y las restricciones de los contenidos de los documentos XML a un mayor nivel de abstracción.

Los documentos '.xsd' (XML Schema Definition) suponen una alternativa a las DTD, siendo más complejas, y poseyendo más capacidades a la hora de definir estructuras para documentos XML.
El principal aporte de XML Schema es el gran número de tipos de datos que incorpora. De esta manera, XML Schema aumenta las posibilidades y funcionalidades de aplicaciones de procesado de datos, incluyendo tipos de datos complejos como fechas, números y strings.

A modo de ejemplo podríamos tener

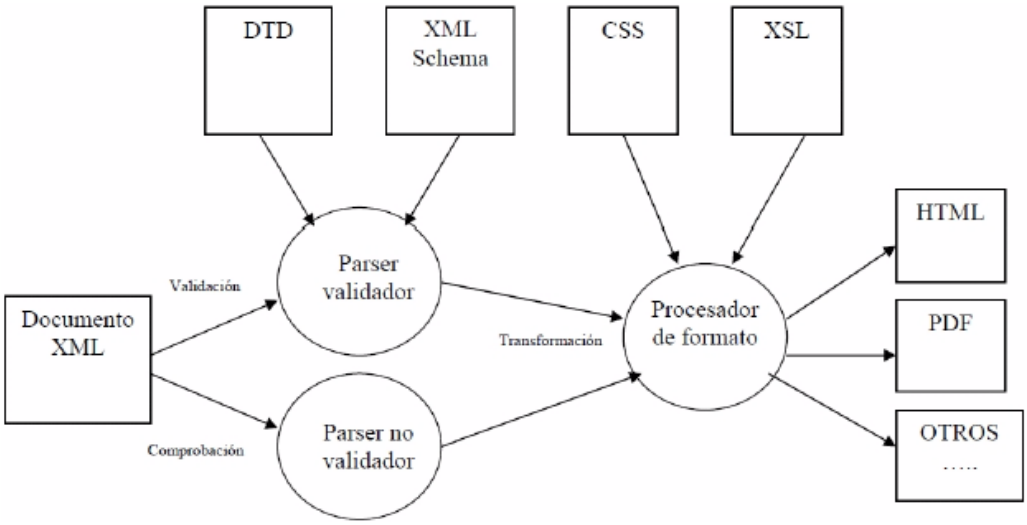
Ejemplo.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<coche codigo="1458"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=".\ejemplo.xsd">
<piloto nombre="Dani" conduce="Si"/>
</asignatura>
```

Ejemplo.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xs:element name="piloto">
<xs:complexType>
<xs:attribute name="nombre" type="xs:string"
use="required"/>
<xs:attribute name="conduce"
type="xs:boolean" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="coche">
<xs:complexType>
<xs:sequence>
<xs:element ref="piloto"/>
</xs:sequence>
<xs:attribute name="codigo"
type="xs:short" use="required"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

Para finalizar añado una imagen donde se detalla las diferente etapas del ciclo de vida de un archivo Xml, desde su análisis hasta su procesamiento y formateo.



3.2. RDF y RDF Schema

Por definición RDF, se trata de 'Resource Description Framework', o 'Marco de Descripción de Recursos'.

Según Wikipedia, es un framework para metadatos en la World Wide Web (WWW), desarrollado por el World Wide Web Consortium (W3C).

Se trata de un lenguaje de objetivo general para representar la información en la Web. Un modelo de datos para los recursos y las relaciones que se puedan establecer entre ellos. Aporta una semántica básica para este modelo de datos que puede representarse mediante XML.

RDF fue diseñado para definir un mecanismo que permitiese describir recursos. Para ello emplea, un mecanismo conocido como tripleta, estableciendo un ámbito semántico muy simple, basado en tres términos:

- Sujeto o recurso
- Propiedad o predicado
- Objeto o literal.

Todas lo descrito por expresiones RDF seria considerado como ***'sujeto o recurso'***.

Estos sujetos o recursos pueden ser un sitio web completo, una página web o una parte de ella. También puede representar objetos físicos como un vehiculo. Los recursos o sujetos se designan mediante URI. (Ver 3.4)

Una ***'propiedad o predicado'*** es una característica, aspecto, atributo o relación empleada para describir un recurso.
Para cada propiedad se ha de definir una serie de valores posibles, indicar que recursos pueden describir y establecer las relaciones con el resto de propiedades.

El '***objeto o literal'*** de una sentencia es el valor de la propiedad.
Puede ser otro recurso o un tipos de datos primitivos definidos por XML, como por ejemplo un string.

Un recurso específico junto con una propiedad y el valor de esta última determinan una ***sentencia RDF***.

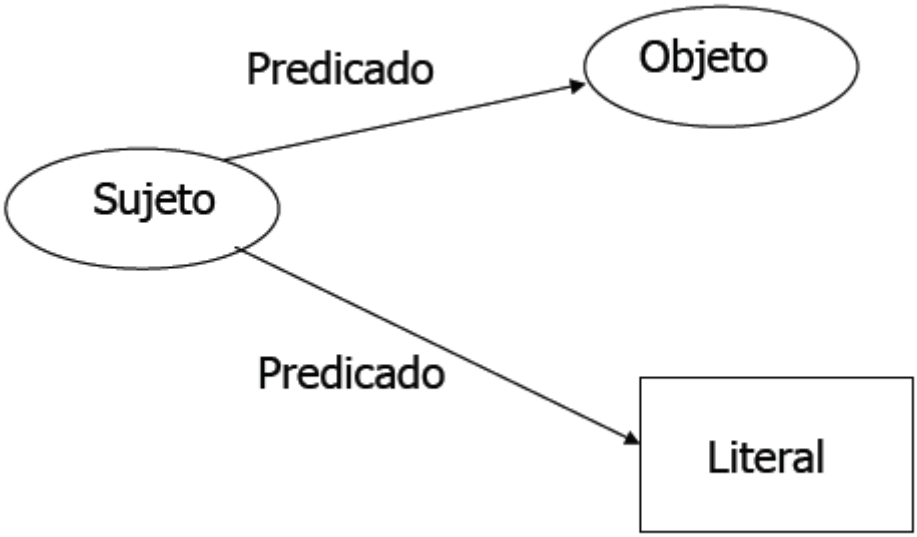
En sintaxis RDF se encuentra definida en http://www.w3.org/TR/rdf-syntax-grammar/

A modo de ejemplo podríamos pensar en una sintaxis muy básica donde indicamos que Daniel Fiel, con email daniel.fiel@mimail.com es el creador de la web *http://www.w3.org/home/danifiel*

Y podría escribirse en RDF/XML de la forma:

```
<rdf:RDF>
<rdf:Description about="http://www.w3.org/home/danifiel">
<s:creator rdf:resource="http://www.w3.org/staffId/2828″ />
</rdf:Description>
<rdf:Description about="http://www.w3.org/staffId/2828″>
<v:name>Daniel Fiel</v:name>
<v:email> daniel.fiel@mimail.com </v:name>
</rdf:Description>
</rdf:RDF>
```

El modelo de datos representado por RDF se resume en esta imagen



Cuando hablamos de RDF, es importante hablar de contenedores.

RDF define tres tipos de contenedores:

- **Bag**: Una lista no ordenada de recursos o literales. Se utiliza para declarar que una propiedad tiene múltiples valores y que el orden no es considerado. Permite realizar repeticiones.

- **Sequence**: Se trata de una lista ordenada de recursos o literales. Se emplea para indicar la posibilidad de tener múltiples valores por parte de una propiedad. En ellas el orden si cobra relevancia, y se pueden tener múltiples valores. Al igual que la anterior, permite repeticiones.

- **Alternative**: Este contenedor define una lista de recursos o literales que representa alternativas para los valores de una propiedad. Un Alternative puede utilizarse para proveer traducciones del título de un trabajo.

La sintaxis de un contenedor es

Container ::= Sequence | Bag | Alternative
Sequence ::= ‘<rdf:Seq’ idAttr? ‘>’ member* ‘</rdf:Seq>’
Bag ::= ‘<rdf:Bag’ idAttr? ‘>’ member* ‘</rdf:Bag>’
Alternative::= ‘<rdf:Alt’ idAttr? ‘>’ member* ‘</rdf:Alt>’
Member::= referencedItem | inlineItem
referencedItem::= ‘<rdf:li’ resourceAtt ‘/>’
inlineItem::= ‘<rdf:li>’ Value ‘/rdf:li>’
RDF::= ‘<rdf:RDF>’ Obj* ‘</rdf:RDF>’
Value::= Obj| string
Obj::= Description|Container

Podemos hablar también de Reification, o afirmaciones sobre afirmaciones. Las Reification son afirmaciones de alto nivel que permiten expresar conocimiento sobre otras afirmaciones, tratando a una afirmación como un recurso, representando explícitamente al sujeto, predicado, objeto y tipo de la afirmación.

En cuanto a **RDF Schema** o **Esquema RDF** , comentare que se trata de una extensión semántica de RDF que proporciona los elementos básicos para la descripción de vocabularios.

RDF provee una forma para expresar enunciados simples a cerca de los recursos usando propiedades y valores. No obstante en ocasiones se necesitaba indicar que lo que estaban describiendo eran tipos o clases específicas de recursos.

RDF por si sólo no proporciona tal vocabulario, por lo que las clases y propiedades se describen en RDF Vocabulary cuya definición se encuentra en RDFS.

RDF schema proporciona mecanismos para especificar que ciertas clases y propiedades son parte de un vocabulario y cómo establecerán la relación entre ellas. Además permite definir a los recursos como instancias de una o más clases, y jerarquizar estas de manera ordenada.

Por lo tanto, RDF schema extiende a RDF para incluir un amplio vocabulario con un significado adicional.

Las clases definidas por RDFS pueden representar casi cualquier categoría de cosas, como páginas web, gente, tipos de documentos, conceptos abstractos, etc.

Las clases son descritas por recursos rdfs:Class y rdf:Resource, y por propiedades como rdf:type y rdfs:subClassOf.

Los recursos que pertenecen a una clase son llamados instancias.

Así que en RDF schema, una clase es cualquier recurso que tenga un tipo (rdf:type) como propiedad cuyo valor sea un recurso rdfs:Class.

Entenderemos por la etiqueta "rdf:type" que todo recurso que la ocupe deberá tener las características que distinguen al recurso que se encuentre del lado derecho de la etiqueta.

3.3. OWL

OWL es el acrónimo del inglés Web Ontology Language, un lenguaje de marcado para publicar y compartir datos usando ontologías en la WWW. Se trata de una extensión de XML(S) RDF(S), y es un lenguaje inspirado en DAML+OIL

OWL proporciona más vocabulario que RDFS para describir propiedades y clases tal como:

- Relaciones entre clases
- Cardinalidad
- Igualdad, más tipos para las propiedades,
- Características de las propiedades
- Clases enumeradas.

OWL esta formado por 3 sublenguajes:

- OWL Lite, Posibilidades de clasificación en la jerarquía de conceptos (clases) de la ontología y restricciones simples.

- OWL DL (Description Logic) es el lenguaje indicado para los usuarios que requieren el máximo de expresividad pero exigiendo completitud computacional (se garantiza que todas las conclusiones son computables) y resolubilidad (todos los cálculos acaban en un tiempo finito). Incluye todos los constructores de OWL, pero sólo se pueden usar con restricciones. Usa descripciones lógicas

- OWL Full se dirige a aquellos usuarios que necesitan la máxima expresividad y la libertad sintáctica de RDF pero sin garantía computacionales. Permite, por ejemplo, aumentar el significado de vocabulario predefinido. Improbable que ningún software de razonamiento sea capaz de soportar razonamiento para cualquier característica de OWL Full.

OWL Full se puede ver como una extensión de RDFS, mientras que OWL Lite y OWL DL se pueden ver como extensiones de una vista restringida de RDF. Cualquier documento OWL (Lite, DL, Full) es un RDF documento, y cualquier documento RDF es un documento OWL Full pero sólo algunos documentos RDF serán documentos OWL Lite o OWL DL legales. Por esta razón, hay que tener cuidado cuando se desea migrar un documento RDF a OWL.

Centraremos especial atención pues en OWL DL.

Anteriormente, hemos visto como mientras XML(S) aporta la sintaxis para crear documentos bien estructurados, pero sin semántica , RDF(S) proporcionan un modelo de datos para cualquier recurso y las relaciones que se puedan establecer entre ellos, incorporando definición de clases y propiedades, con algunos mecanismos para establecer jerarquías de generalización entre dichas propiedades y clases.

Este escenario no es completo ya que necesitamos solventar las deficiencias que no cubren los requisitos semánticos deseados. Es aquí donde OWL Dl cobra relevancia, aportando:

- Restricciones de rango, por propiedad perteneciente a cada clase.
- Propiedades Transitivas de orden, simetría e inversa.
- Tratamiento de clases disjuntas
- Restricciones de cardinalidad.

3.4. URI

A menudo se emplea la expresión URL para hacer referencia a todas las direcciones Web, aunque lo correcto seria utilizar la expresión URI en su lugar.
Los URLs (Uniform Resource Locator) son identificadores que permiten acceder a paginas web., y la tendencia popular es utilizarlas para referencia a todas las direcciones Web. Sin embargo, existe un identificador más completo y adecuado para referenciar a estas direcciones web.
Se trata de URI (Uniform Resource Identifier) que a diferencia de un URL convencionalen que permite incluir en la dirección una subdirección, determinada por un "fragmento".
La Estructura de un URI seria así:
ESQUEMA : PARTE JERARQUICA ? SOLICITUD # FRAGMENTO
Ejemplo:
http : //www.miweb.com/ ? pagina=2 # inicio
El esquema identifica el protocolo que se va a usar.(http en nuestro ejemplo).
La parte jerárquica de la dirección contiene la información del dominio o IP para acceder al servidor y la ruta en el servidor para acceder al recurso.
La solicitud referencia a variables que se pasan a la página Web. Está separada de la ruta mediante el signo de interrogación y termina donde empieza el fragmento delimitado por el carácter numeral (#) si lo hubiere.

Para terminar el Fragmento posibilita indicar una subdirección dentro del recurso al que apunta la dirección. Esta delimitado por el carácter numeral (#) y se extiende hasta donde se termina el URI. Este fragmento establece la diferencia entre un URL y un URI.
Los URLs no identifican fragmentos y por esto se recomienda utilizar la expresión URI cuando se hable de direcciones completas.

4. Ontologías.

4.1. Definición

Podríamos pensar en una ontología, como una especificación de una conceptualización o, en una representación conceptual que establece u otorga un esquema de reconocimiento universal en un dominio concreto.

Aunque existen muy diversas definiciones del concepto de ontología, una de las más Ampliamente aceptadas es la siguiente de Thomas Gruber:

"Una Ontología es una especificación formal y explícita de una conceptualización compartida"

Con el l término "**conceptualización**" hace referencia al hecho de que toda ontología representa un modelo abstracto del dominio através de conceptos, atributos, valores y relaciones.

La "**especificación explícita**" subraya la capacidad de una ontología para la descripción y representación de un dominio concreto mediante conceptos, atributos, valores, relaciones, funciones, etc., definidas explícitamente..

Por su parte "**formal**" ensalza el compromiso que alude al hecho de que cualquier representación, debe ser expresada en una ontología mediante un formalismo siempre idéntico, de manera que pueda ser reutilizada y leída por cualquier máquina independientemente del lugar o de la plataforma o idioma del sistema que lo emplee.

Por ultimo aunque no menos importante "**compartida**". Ya que una ontología cobra pleno sentido solo cuando es adoptada por diferentes usuarios para su explotación. Solo cuando todos los actores de un posible dominio utilizan con éxito una ontología podemos elogiar el papel de la misma en su dominio.

4.2. Métodos de Creación.

La propia complejidad asociada al termino ontología, junto con un vacío procedimental en la creación de las mismas, ha propiciado que no exista una corriente clara a la hora de metodizar la construcción de ontologías.
Como en cualquier campo científico, la falta de un método o procedimiento claro y preciso para el modelado de soluciones puede suponer una importante traba hacia el camino del éxito y la normalización.

En el terreno semántica, no contar con una metologia puede suponer que la ontología creada:

- No refleje las necesidades de la aplicación debido a un nivel superior de subjetividad del esperado.
- De la misma manera se dificulta la integración con otras ontologías.
- Se dificulta su entendimiento dentro del ecosistema semántico.
- Se limita su campo de aplicación al entorno para el cual fue concebida.

Estas limitaciones son lo suficientemente criticas, como para justificar un esfuerzo en la búsqueda de soluciones estandarizadas que puedan minimizar su impacto.

Al tratarse de una temática relativamente reciente, no existe un paradigma único que de pautas empíricas para la creación de ontologías, aunque si se pueden enumerar algunos principios comunes o características a esperar de una potencial metologia:

- Debe ser resolutiva. Orientando a la resolución de problemas mediante la construcción de recursos.
- Debe garantizar una estructura conceptual. Proporcionando un entorno de creación estandarizado y bajo un patrón definido.
- Debe minimizar el nivel de Subjetividad. Evitando interpretaciones erróneas.
- Debe estar documentada. Estableciendo formas de documentar tanto los elementos obtenidos y como los procesos en los que se obtienen.
- Debe asegurar la autoria. Como medio de garantía que permita organizar el conocimiento.
- Debe asegurar la reprentabilidad. Es necesario establecer mecanismo para poder difundir la información generada y comprobar la integridad de la misma en un momento dado.

Por su parte, Thomas Gruber, introduce las siguientes bases para el desarrollo de ontologías:

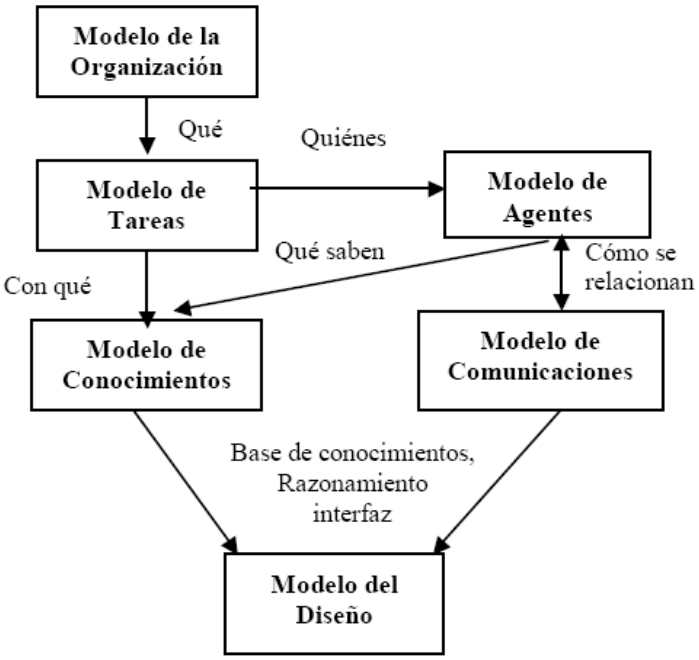
- Claridad. Una ontología debe comunicar efectivamente el significado de sus Conceptos.
- Coherencia. Una ontología solo debe permitir inferencias o deducciones que son consistentes con las definiciones.
- Expansibilidad. Una ontología debe ser diseñada para anticipar cualquier uso que se pueda hacer de ella.
- Mínimos requisitos de codificación. La ontología debe ser representada en el nivel de conocimiento.
- Mínimo compromiso ontológico. La ontología debe estar lo mas separada de las características del mundo físico y lo objetos que la componen.

Con todo las principales metodologías del panorama actual son:

- **CommonKADS**. Denominada Ingeniería del Conocimiento , fue creada en 1993 por Schreiber, Wielinga y Brecker. Esta basada en 2 hipótesis:
    - La adquisición de conocimiento es una actividad de modelado.
    - La actividad de modelado se sitúa a nivel de conocimiento.

  En resumen, esta metodología postula la objetividad del dominio como elemento diferenciador y de poder sobre la propia experiencia y establece el nivel conceptual como nivel mas importante de la resolución de problemas.

  La siguiente figura muestra un esquema de esta metodología:



Como podemos ver se establecen diferentes capas de modelos que permiten describir el conocimiento necesario para resolver un problema, dentro de un domino particular, usando niveles de abstracción que posibilitan detallar el proceso de solución en una forma independiente del dominio. Siendo el modelo de conocimiento la piedra angular de dicho esquema.

- **Gruniger y Fox**. Informalmente conocida como 'Competency Questions' , fue formulada en 1995 por Gruniger y Fox, después de construir la ontología TOVE (Toronto Virtual Enterprise). Los autores establecen que la metodología debe seguir estas fases o etapas.

  - *Escenarios de motivación* (Motivating scenarios). Los escenarios de motivación están representados por el conjunto de problemas que la ontología intenta resolver. Los escenarios generan la idea semántica informal de los objetos que después serán parte de la ontología.

  - *Preguntas de competencia informales* (Informal competency questions). Son un conjunto de consultas que conforman los requisitos que una ontología debe ser capaz de responder.
    La designación 'informales', debe su nombre al hecho de no estar aun expresadas en un lenguaje formal.
    Dentro de esta fase, através de las consultas, se definirá si se necesita una nueva ontología o si bien podemos reutilizar alguna ontología existente. Además, sirve también para evaluar la semántica informal definida en el paso anterior.

- *Especificación.* A través de la lógica de predicados de primer orden de los diferentes términos del dominio.
  En esta fase se definen los objetos atributos y demás objetos de la ontología. Estos elementos serán representados por constantes y variables de un lenguaje formal.

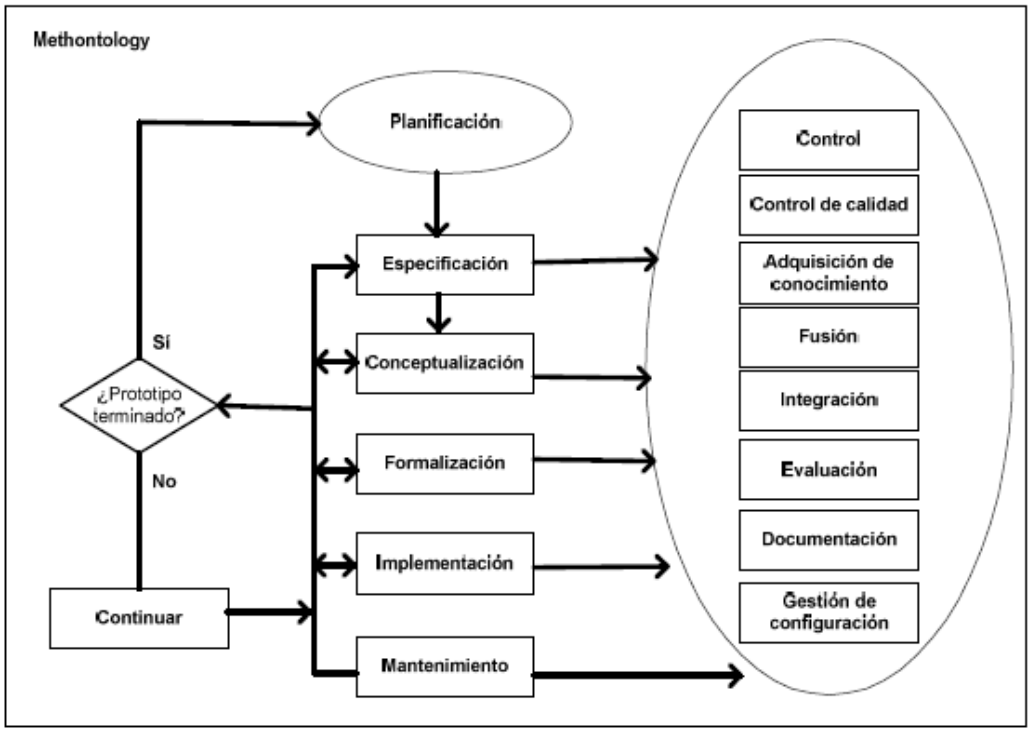
- *Preguntas de competencia formal* (Formal competency questions). Cada consulta informal denotada por Q es representada en un lenguaje formal, de tal manera que dado el conjunto de axiomas de la ontología denotado por TOntology y el conjunto de átomos o predicados atómicos denotado por TGround se pueda determinar que ontología es adecuada.

- *Especificación de la definición de los conceptos en lógica de predicados de primer orden*. Este paso no es sino el criterio uno de Gruber. Todo concepto de la ontología debe estar definido por un conjunto de axiomas, En realidad estas condiciones determinan la definición de un concepto en base a su pertenencia a un objeto o conjunto de objetos, además de incluir restricciones adicionales.

- *Teoremas de completitud*. En esta ultima fase se definen las condiciones bajo las cuales las competencias formales son completas.

- **Usholod y Gruniger** . Fue formulada en 1996 por Gruniger y Fox, d Los autores establecen que la metodología debe seguir estas fases o etapas.
    - *Identificación del propósito y definición del alcance de la ontología.*
    - *Construcción*
        - Captura
        - Codificación
        - Integración
    - *Evaluación*
    - *Documentación*
    - *Pautas*

- **Methontology** . fue formulada en 1997 por Fernández, Gómez Pérez y Juristoy rediseñada en 2004 por Gómez Pérez, Fernández López y Corcho, adaptando IEEE a.
    - *Gestión de la ontología.* Contemplando actividades de planificación y control.
    - *Desarrollo de la ontología.* Cubre Pre y Post desarrollo además del propio desarrollo en si.
    - *Soporte de la ontología*. Simultaneas a la fase anterior, están formada por las subactividades:
        - *Adquisicion de conocimiento*
        - *Evaluación*
        - *Integración*
        - *Fusión*
        - *Alineación*
        - *Documentación*
        - *Gestión de Configuración*



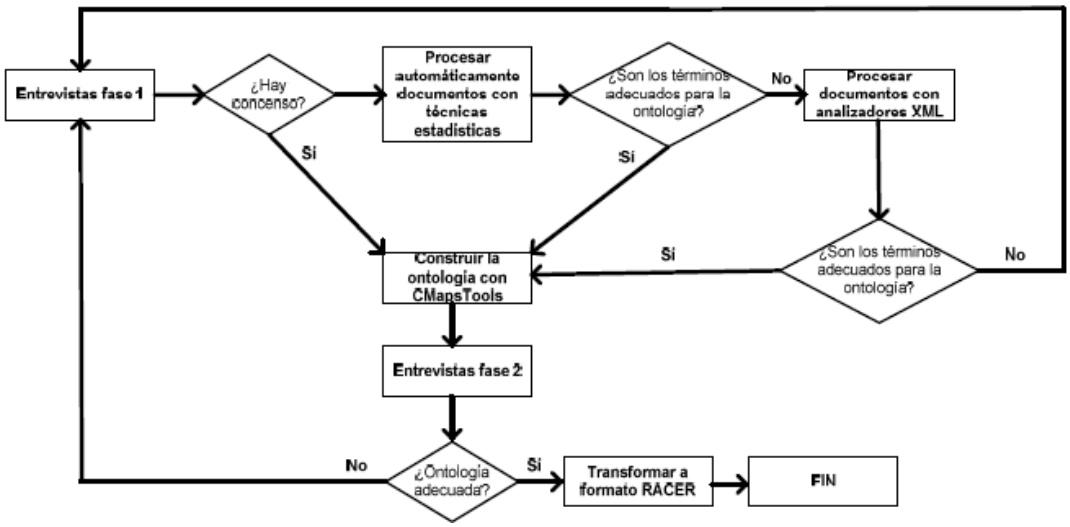
- **Gómez-Gauchía, Díaz-Agudo y González-Calero.** También conocida como On To Knowledge, fue formulada en 2004 y esta pensada para la creación de ontologías lightweight

    - *Entrevistas.* Se trata de obtener mediante entrevistas, los conceptos mínimos necesarios para crear la ontología.

    - *Procesamiento automático de los documentos.* Mediante métodos estadísticos y analizadores como DOM.

    - *Construcción manual simplificada.* Fase de construcción de la metodología

En esta figura podemos ver resumida la metodología:



4.3. Conceptos Básicos Relacionados

Es conveniente disponer cuales son los términos de uso frecuente y de aprendizaje básico para trabajar con ontologías. De esta manera he creado un pequeño glosario para dejar constancia del léxico a utilizar.

- **Clase**: Descripción formal de una entidad a representar. La clase es la piedra angular para una correcta estructuración de la realidad a representar.
  Una clase puede emplear subclases para representan conceptos mas específicos a los definidos por si misma.

- **Clase abstracta:** tipo de clase que no permite instanciacion. Su uso se limita a tareas de agrupación de conceptos, o definición de orden en la jerarquía.

- **Instancia:** Se trata de un objeto concreto dentro de un dominio, que hace referencia a una clase. La colección de instancias constituye la base de hechos también denominada base de datos o base de conocimiento.

- **Instancia indirecta:** Es una contraposición a instancia directa, donde no existen clases intermedias.

- **Propiedad (Atributo, Slot):** Son elementos que definen o detallan a las clases y las instancias. Los valores de las propiedades o atributos pueden ser tipos básicos como cadenas de caracteres o números, pero también pueden ser otras clases o instancias

- **Propiedad (Atributo, Slot):** Son elementos que definen o detallan a las clases y las instancias. Los valores de las propiedades o atributos pueden ser tipos básicos como cadenas de caracteres o números, pero también pueden ser otras clases o instancias

- **Faceta .** Se trata de una Restricción sobre las propiedades.

- **Relación:** Forma de interactuar de los diferentes elementos. Establecen el tipo de asociación entre ellos como por ejemplo subclase de, parte de, es un, etc. Las relaciones más simples se modelizan mediante una propiedad de una clase cuyo valor es una instancia de otro concepto**.**

- **Axioma:** Regla asociada a una ontología para definir el comportamiento de los conceptos o clases. Se establecen a partir de valores específicos de las propiedades.

- **Anotación:** Proceso de instanciacion a partir de texto libre. Puede realizarse mediante la inclusión de etiquetas semánticas dentro del texto que se está procesando, o rellenando instancias directamente en el modelo.

- **Herencia:** Propiedad de la relación 'es_un' que permite que las clases heredadas cuenten con los atributos de la clase padre.

- **Herencia múltiple:** Ocurre cuando se produce una herencia desde 2 clases propiedades de dos clases padre con las que establece dos relaciones del tipo 'es_un'.

- **Derivación: Se trata de una o**rganización de las clases en un árbol de jerarquía mediante sucesivas relaciones 'es_un' con la propiedad de herencia.

5. DBpedia.

5.1. Historia.

Fruto de la colaboración entre la universidad de Universidad de Leipzig, la Universidad Libre de Berlín y la compañía OpenLink Software surge el proyecto Dbpedia, basado en la extracción de los datos de Wikipedia con el fin de crear una versión semántica de la misma.

La idea original pasa por extraer información estructurada de Wikipedia y por hacerla accesible desde la Web. De ese modo se consigue que el conocimiento recogido pueda ser aprovechado computacionalmente bajo el paradigma de la Web semántica.

Pronto, el proyecto suscita interés de numerosos estudios y proyectos que buscan aprovechar el potencial de DBpedia En la actualidad existe un gran número de aplicaciones que enlazan con DBpedia, lo que posibilita establecer vocabularios comunes entre ellas y facilita su interoperabilidad. Como afirman en la propia Wikipedia DBpedia está interconectada con GeoNames, Musicbrainz, CIA World Factbook, Proyecto Gutenberg, Eurostatentre otros.

En la base de datos se describen 3.640.000 entidades, entre ellas al menos 416.000 personas, 526.000 lugares, 106.000 álbumes de música y 60.000 películas y contiene 2,724,000 enlaces a imágenes, 6,300,000 enlaces a páginas externas, 6,200,000 enlaces a datasets externos y 740,000 categorías Wikipedia.

El contenido de la base de datos está disponible bajo licencia CC-BY-SA 3.0 y GFDL

DBpedia permite ejecutar consultas sofisticadas contra Wikipedia, así como vincular los conjuntos de datos abiertos en la Web con los datos de Wikipedia. La aplicación permite asimismo guardar los modelos creados por estudiantes, profesores y otros expertos, formando así un repositorio de modelos. Futuros estudiantes pueden aprovechar el conocimiento de dichos modelos para mejorar los suyos propios. Para ello, el modelo del estudiante se compara de forma automática con otro modelo perteneciente al mismo dominio y que es usado aquí como referencia.

5.2. DataSet

EL tamaño actual del dataset de DBpedia es de 3.64 millones, una cifra hasta hace poco impensable en lo que a información con contenido semántico se refiere. Se combina información estructurada en 97 idiomas.

El dataset esta formado por 1 billón de piezas de información (en tripletas RDF) de las cuales 385 millones han sido extraídas de la versión inglesa de Wikipedia, y 665 millones de otras ediciones datasets externos.

A nivel tecnológico emplea RDF, para la modelización y publicación de datos, mientras que se recomienda utilizar SPARQL como lenguaje de consulta.

En el siguiente apartado analizaremos

5.3. Consultas Sparql

SPARQL, hace referencia a 'Sparql Protocol And rdf Query Languaje', y consiste en realidad en un lenguaje para consultas sobre RDF normalizado. Al igual que sucede con SQL, es necesario distinguir entre el lenguaje de consulta y el motor para el almacenamiento y recuperación de los datos. Por este motivo, existen múltiples implementaciones de SPARQL, generalmente ligados a entornos de desarrollo y plataforma tecnológicas. En su version estándar SPARQL únicamente incorpora funciones para la recuperación sentencias RDF. Sin embargo, algunas propuestas también incluyen operaciones para creación, modificación y borrado de información. Mas allá sus propias capacidades intrínsecas como lenguaje de consulta RDF, sobre el que modelizar DBpedia, SPARQL, que es un lenguaje estándar para recuperación de información en W3C. La importancia de SPARQL radica en gran parte en su capacidad de expresar sus consultas en un lenguaje de alto nivel sin importar el formato en que estos datos han sido almacenados.

Tim Berners Lee lo explica así:
"Pretender usar la Web Semántica sin SPARQL es como pretender usar una base de datos relacional sin SQL", "SPARQL hace posible consultar información desde bases de datos y otros orígenes de datos en sus estados primitivos a trabes de la Web".
Utilizar el lenguaje de consultas SPARQL, nos permite realizar consultas orientadas puramente a contenidos semánticos, y no basándonos en la teoría de conjuntos como el modelo relacional, aunque disminuya la organización de la base de datos.

SPARQL tiene especificaciones que explican diferentes partes de su funcionalidad; en general, consiste en un lenguaje de consulta, un formato para las respuestas, y un medio para el transporte de consultas y respuestas:

- **SPARQL Query Language.** Componente principal de SPARQL. Explica la sintaxis para la composición de sentencias y su concordancia
- **SPARQL Protocol for RDF.** Formato utilizado para la devolución de los resultados de las búsquedas (queries SELECT o ASK), a partir de un esquema XML.
- **SPARQL Query Results XML Format**: Describe el acceso remoto de datos y la transmisión de consultas de los clientes a los procesadores. Utiliza WSDL para definir protocolos remotos para la consulta de bases de datos basadas en RDF

En cuanto a la estructura de una consulta SPARQL, podemos decir que el lenguaje de consulta esta basado en comparación de patrones gráficos. Los patrones gráficos contienen patrones triples.
Los patrones triples son como las tripletas RDF, pero con la opción de una variable consulta en lugar de un termino RDF en las posiciones del sujeto, predicado u objeto. Combinando los patrones triples obtenemos un patrón grafico básico, donde es necesario una comparación exacta entre gráficos.

La sintaxis básica de una consulta SPARQL

Sintaxis básica de una consulta SPARQL	
Prologue (optional)	BASE <iri> PREFIX prefix: <iri> (repeatable)
Query Result forms (required, pick 1)	SELECT (DISTINCT)sequence of ?variable SELECT (DISTINCT)* DESCRIBE sequence of ?variable or <iri> DESCRIBE * CONSTRUCT { graph pattern } ASK
Query Dataset Sources (optional)	Add triples to the background graph (repeatable): FROM <iri> Add a named graph (repeatable): FROM NAMED <iri>
Graph Pattern (optional, required for ASK)	WHERE { graph pattern z}

Query Results Ordering (optional)	ORDER BY ...
Query Results Selection (optional)	LIMIT n, OFFSET m

La función de la palabra clave PREFIX es equivalente a la declaración de namespaces en XML, es decir asocia una URI a una etiqueta, que se usará mas adelante para describir el namespace. Pueden incluirse varias de estas etiquetas en una misma consulta.

Todo comienzo de una consulta SPARQL queda marcada por la palabra clave SELECT, similar a su uso en SQL1, sirve para definir los datos que deben ser devueltos en la respuesta. La palabra clave FROM identifica los datos sobre los que se ejecutará la consulta, es necesario indicar que una consulta puede incluir varios FROM. La palabra clave WHERE indica el patrón sobre el que se filtrarán los tripletes del RDF.

A modo de ejemplo de una consulta SPARQL presento un ejemplo tomado de W3C, donde se trata de encontrar el titulo de un libro de un grafico de datos dado. La consulta consiste de dos partes:

- La cláusula SELECT que identifica las variables que aparecen en los resultados de la consulta y
- La clausura WHERE que proporciona el patrón grafico básico a comparar con el grafico de datos.

Esta simple consulta, nos devolvería un listado de actores de televisión.

```
SELECT DISTINCT *
WHERE {
 ?Peronas skos:subset
<http://dbpedia.org/resource/Category:Television_actor>
```

Esta Otra mas compleja nos devolvería una relación de escritores españoles, su lugar de nacimiento, sus obras, su página Web y una fotografía.

```
SELECT DISTINCT ?escritor ?nombre ?lugar ?obra ?website ?foto
WHERE {
?escritor skos:subject
<http://dbpedia.org/resource/Category:Spanish_writers> .
?escritor rdfs:label ?nombre .
?escritor foaf:depiction ?foto .
?escritor <http://dbpedia.org/property/birthPlace> ?lugar .
OPTIONAL{
?escritor foaf:homepage ?website .
}
OPTIONAL{
?obra dbpedia2:writer ?escritor .
}
FILTER (LANG(?nombre) = 'es') .
}
```

6. Modelado de una Ontología.

Una vez adentrados en el universo semántico, procederemos a crear una ontología simple que permita describir el proceso de modelado de la misma.

Sin grandes pretensiones, en lo que a nivel de detalle de se refiere, nuestro trabajo abordara la fase de modelado, empleando para ello la herramienta 'Protege' en su versión 3.3. Esta herramienta, de código abierto y disponible en http://protege.stanford.edu/, esta considerada un referente en el proceso de construcción de ontologías sobre RDFS, OWL y XML Schema.

6.1. Selección Dominio.

Aunque a priori, y debido al carácter explicativo que posee nuestra ontología, el proceso de selección de dominio pudiera parecer algo trivial, hemos querido seleccionar un dominio, que al margen de la propia creación de la ontología nos sirva para reflexionar sobre las posibilidades que arroja la misma.

Hemos pensado en el mundo del deporte, y mas concretamente el futbol, por ser un deporte de masas con mucho arraigo y gran aceptación en la sociedad actual.

Entiendo que la potencialidad que se le presupone al entorno semántico, necesita de contenidos, y de la interacción con el gran publico. Dominios como el que presento podrían ser empleados en Casual Games en la red, de forma atractiva para los usuarios.

No hay que olvidar además, que una de las principales aportaciones de la semántica es proporcionar búsquedas mas eficientes, y poder interactuar de forma mas precisa con la información.

En el mundo del futbol la estadística y la historia juegan un papel relevante y tanto periodistas, como aficionados o presiónales acuden a ellas de forma recurrente . Una ontología de este alcance podría llegar a devolver información tan enrevesada como una lista de jugadores que hayan debutado antes de los 19 años en estadios de futbol con capacidad para mas de 60.000 espectadores.

6.2. Proceso de modelado.

Una vez seleccionado el dominio de nuestra ontología, y con Protege instalado nuestro equipo, podemos comenzar a modelar.

En primer lugar crearemos las clases y subclases de nuestra ontología.

Hemos definido 4 clases principales con sus respectivas subclases, que a continuación paso a enumerar

- **Equipo**. Esta clase representara a los diferentes equipos de futbol profesionales que compiten en las diferentes categorías del futbol español. Para diferenciar este ultimo aspecto, hemos creado a su vez dos subclases, dependientes de esta primera.
    - *Primera División.* Representa la máxima competición nacional de futbol.
    - *Segunda División*. Hace referencia a la división de plata del futbol español.

- **Persona**. A través de esta clase haremos referencia a las diferentes personas que de una manera u otra interactúan o forman parte de nuestro dominio. Para una mayor especialización y detalle hemos incluido las siguiente subclases, que

aportan un nivel de definición necesario para una clase tan genérica como es persona.

- *Directivo.* Hace referencia al personal ejecutivo de un club, desde presidentes a responsables de diversa índole en lo que a gestión del club se refiere.
- *Jugador*. Hacen referencia a las diferentes personas que conforman la plantilla deportiva de un club.
- *Técnico*. Representa la figura de entrenador o cuerpo técnico asociada a un club.

- **Instalación** . Esta clase nos servirá para referirnos a las dependencias físicas del club. Se trata de espacios explotados por el equipo para diversos fines. Para una mayor especialización he creado las siguientes subclases

  - *Campo de entrenamiento.* Sirve para referirse a las instalaciones deportivas en las que se ejercita un club.
  - *Estadio*. Esta subclase recogerá los diferentes terrenos de juego oficiales en los que cada equipo disputa sus partidos.

- **Equipación** . Utilizaremos esta clase para referenciar a las diferentes prendas deportivas empleadas tanto por los jugadores como por el cuerpo técnico durante sus actividades oficiales y de entrenamiento. Para ello hemos creado las diferentes subclases

  - *Entrenamiento.* Hace referencia a la indumentaria empleada para las labores de entrenamiento diarias. .
  - *Uniforme*. Representa la equipacion oficial de los diferentes club cada temporada… Esta subclase cuenta además con varias subclases que permiten diferenciar las prendas vestidas por cada jugador:
    - *Camiseta.*
    - *Medias*.
    - *Pantalón.*

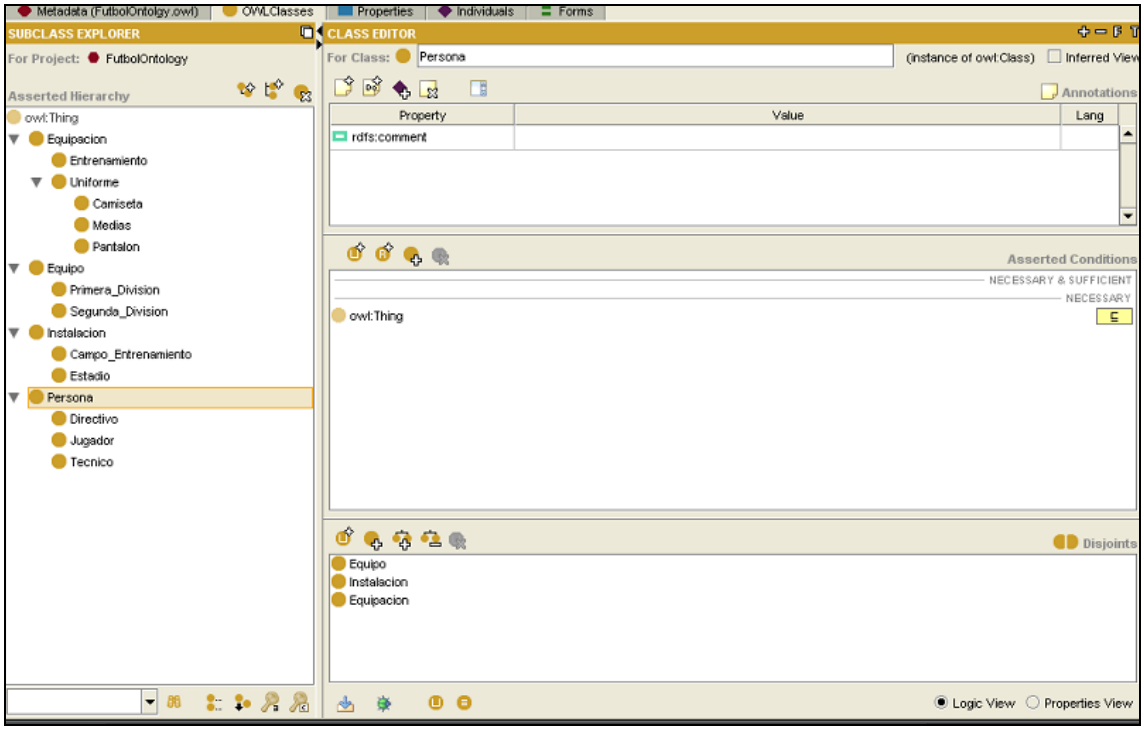
Con todo ello nuestro árbol de clases queda definido tal y como se muestra en la figura siguiente

El siguiente paso es definir la relación entre clases, para las entidades creadas. Para ello definimos las siguientes clases disjuntas.

Una relación disjunta de clases, indica que una nueva instancia no podrá ser de ambas clases disjuntas al mismo tiempo. Así por ejemplo si declaramos primera división disjunta de segunda división, ningún elemento podrá pertenecer ambas clase a la vez.

En nuestra ontología hemos creado todas las clases principales disjuntas entre si, de la misma manera que dentro de cada clase, las subclases también han sido declaradas disjuntas.

De esta manera y por ilustrar mediante un ejemplo, mostramos como una vez creadas las relaciones entre clases, al seleccionar la clase, Protege nos mostrara en un area especifica las clases disjuntas a la clase seleccionada.



En ella podemos ver como la clase ‘ Persona’ es disjunta de las clases ‘Equipo’, ‘Instalación’ y ‘Equipamiento’.

El siguiente paso será crear las propiedades, dentro de las propiedad es diferenciamos entre las propiedades:

- **Objeto**. Empleadas para definir la relación entre clases, dotaran a nuestro sistema de semántica, ya que anotan las relaciones que una clase u objeto tiene respecto a otra.

  Para definir este tipo de propiedades, debemos de crear la propiedad, y después seleccionar a que clases relaciona, estableciendo la clase dominio y la clase rango.

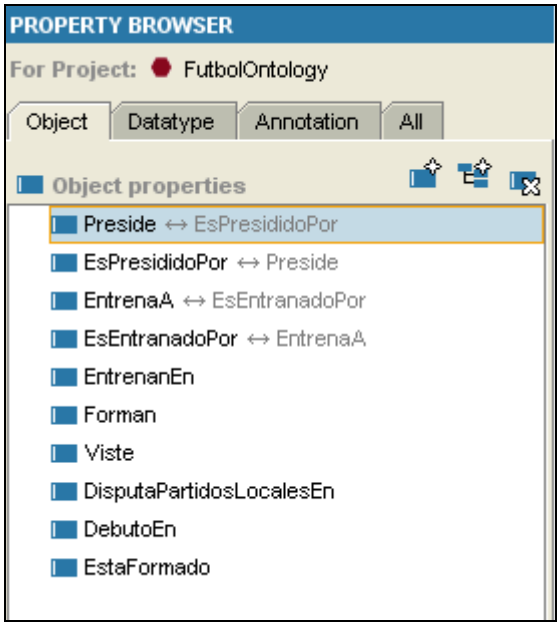
  Adicionalmente, cuando creamos una relación de este tipo , podemos indicar si se trata de una relación funcional, simétrica, transitiva o inversa.

  Las relaciones inversas, generan una relación opuesta entra las clases seleccionadas.

Las propiedades de tipo objeto creadas para mi ontología son:

Clase Dominio	Clase Range	Propiedad
Directivo	Equipo	'Preside'
Equipo	Directivo	'Es Presidida por'
Técnico	Equipo	'EntrenaA'
Equipo	Técnico	'EsEntrenadoPor'
Equipo	Campo_Entrenamiento	'EntrenanEn'
Jugador	Equipo	'Forman'
Jugador	Equipacion	'Viste'
Equipo	Estadio	'DisputaPartidosLocalesEn'
Jugador	Equipo, Estadio	'DebutoEn'
Equipo	Jugador	'EstaFormado'

En el Protege, queda así definido

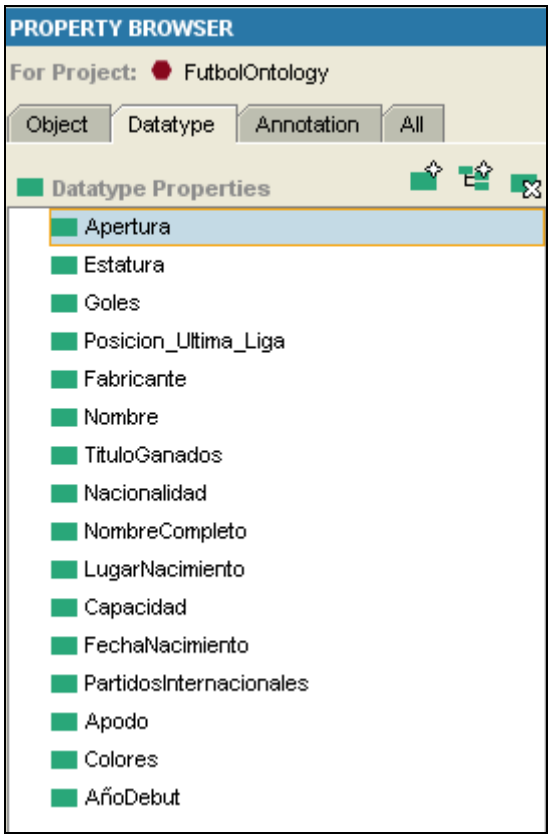


- **Dato simple**. Son propiedades asignadas directamente a una o varias clases. Empleadas para definir características propias de la clase bajo un patrón de formato , o tipo de dato concreto.

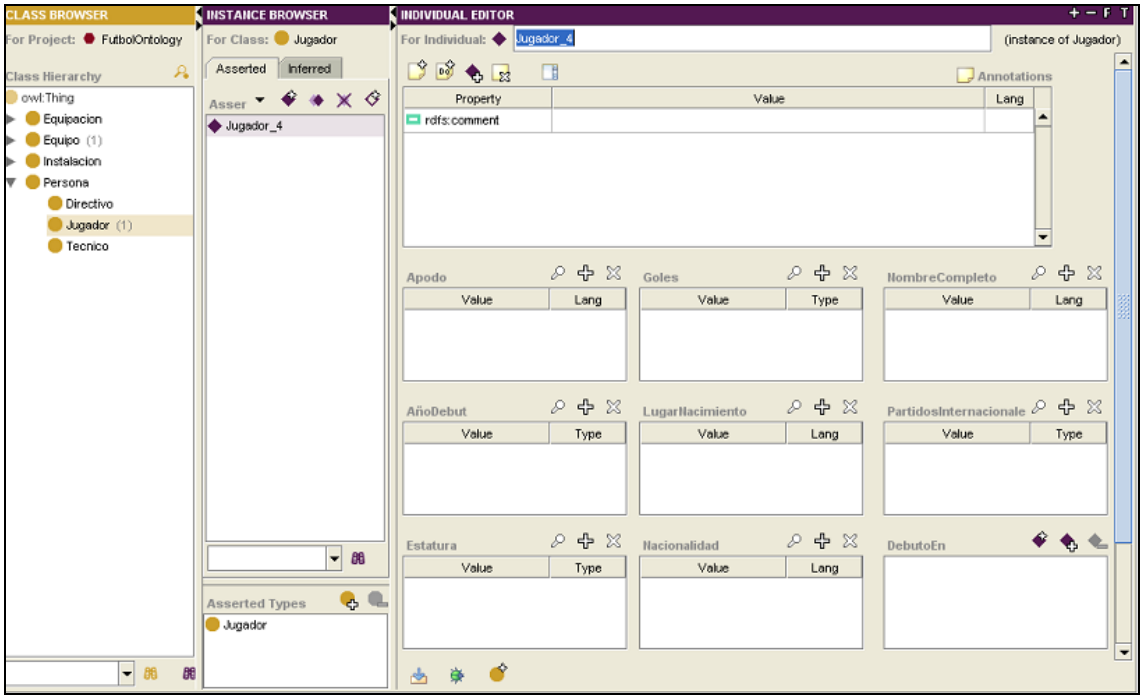
Para modelar mi ontología he empleado los siguientes datos simples:

Clase Dominio	Propiedad	Tipo de dato
Apertura	Estadio	Integer
Estatura	Jugador	Integer
Goles	Jugador	Integer
Posicion_Ultima_Liga	Equipo	Integer
Fabricante	Equipacion	String
Nombre	Jugador	String
TituloGanados	Equipo	Integer
Nacionalidad	Jugador	String
NombreCompleto	Persona,Estadio	String
Capacidad	Estadio	Integer
FechaNacimiento	Jugador	String
PartidosInternacionales	Jugador	Integer
Apodo	Jugador, Equipo	String
Colores	Medias,Pantalon;Camiseta	String
AñoDebut	Jugador	Integer

En el Protege, queda así definido



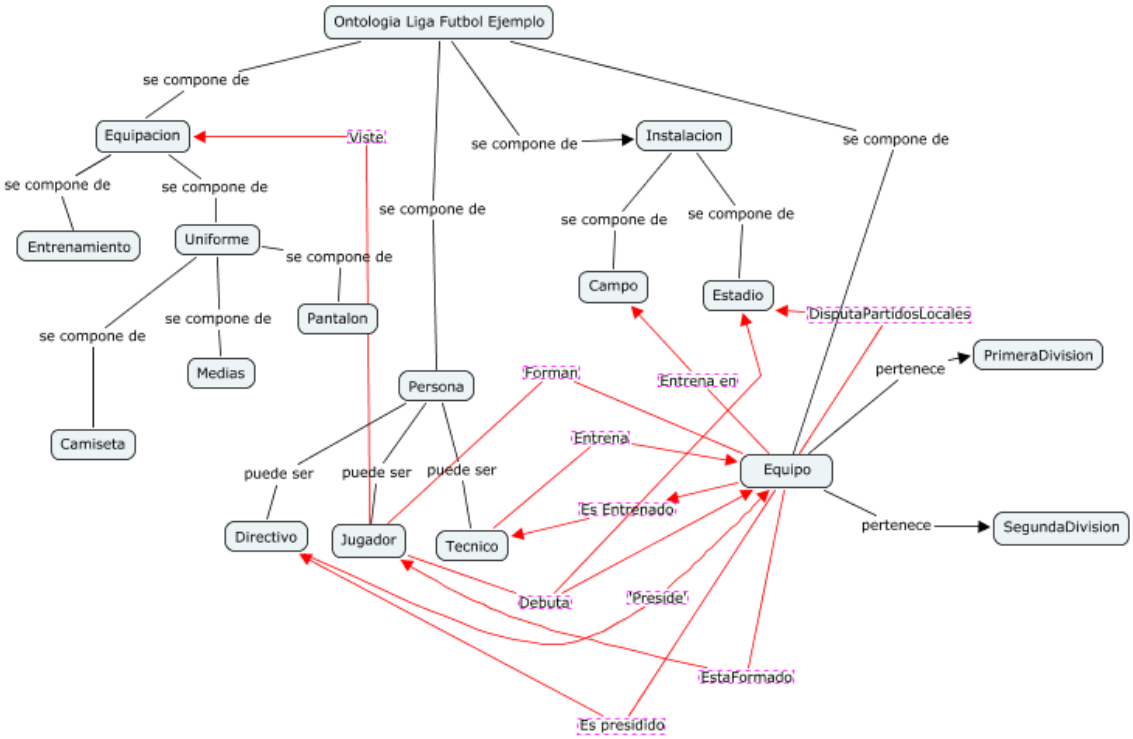
Nuestra Ontología estaría lista para ser instanciada, a través del propio Protege podría hacer anotaciones guiadas mediante la interfaz grafica que ofrece el propio sistema para ello.



Como se ve en la figura, al seleccionar la subclase jugador e intentar instanciarla, el sistema nos muestra todas las opciones creadas para esta clase, y nos facilita su introducción.

De cualquier modo, nosotros no ejecutaremos esta funcionalidad através del propio programa sino que tal y como veremos en el siguiente apartado, prodeceremos a crear anotaciones semiautomáticas a través de URI externas.

Por ultimo, veremos un esquema de la ontología creada:



En rojo se muestran las relacione tipo objeto entre las clases.

En el Anexo1, se describe el contenido del archivo OWL generado por Protege.

7. Herramienta.

7.1. Alcance.

El objetivo de la herramienta que he desarrollado, es experimentar con el tratamiento de ontologías programáticamente.

Con la mirada puesta en lograr realizar una población semiautomática de una ontología y basándome en la información contenida en la anteriormente estudiada DBPedia, la herramienta servirá de ejemplo o prototipo que demuestre como obtener un valor de una propiedad definida en un fichero RDF, mediante consultas Sparql, y asignar este a un individuo de una ontología creada por nosotros mismos. Así mismo permitirá recorrer la ontología realizando un browser que recorra su contenido.

7.2. Modulo de Importación.

Como hemos comentado en el alcancé, el propósito general de la herramienta, consiste en importar información de manera semiautomática, es por ello que será necesaria realizar tareas manuales como la selección y almacenamiento de las URI origen.

El modulo de importación nos permitirá seleccionar un archivo RDF, previamente guardado en nuestro PC o en un disco accesible, y a través de una consulta sparql, obtener la propiedad solicitada.

Al tratarse de un trabajo experimental, no he considerado una recuperación muy ambiciosa, y me he limitado a demostrar con una única propiedad, como poder lograr nuestro objetivo.

Es por ello que el modulo de Importación, centrara su funcionalidad en obtener la propiedad 'name' del archivo RDF seleccionado, volcando los resultados obtenidos a una tabla visible por el usuario.

El modelo creado podría soportar hipotéticas ampliaciones, ya que simplemente modificando la consulta sparql definida, podríamos retornar cualquier propiedad. En la versión actual la consulta esta definida programáticamente, pero se podría parametrizar para que el propio usuario que la ejecuta sea capaz de definirla a través de la interfaz.

A nivel técnico, el modulo hace uso de las siguientes clases:

- **GestionModelo**. Clase que define el propio modelo, y proporciona métodos para la construcción, establecimiento y ejecución de consultas, así como obtención de resultados.

- **ControladorModelo**. Nos habilita los métodos necesarios para seleccionar y tratar el archivo RDF origen. Emplea los métodos definidos en la clase anterior para ejecutar la consulta y mostrar los resultados.

- **JFramePrincipal**. Se tata de la interfaz grafica que habilita la interacción del usuario con el sistema. De manera intuitiva nos guiará por las diferentes funcionalidades del mismo.

Veremos mas detalles técnicos en el capitulo de demostración de la herramienta.

7.3. Modulo de Navegación.

Una vez realizada la instanciación y anotación en nuestra ontología, el sistema contara con la opción de poder ver el contenido de la ontología creada con los nuevos valores importados.

Al igual que he comentado en el modulo de importación, en el siguiente apartado detallare técnicamente el funcionamiento de la navegación.

7.4. Demostración y Explicación Técnica de la Herramienta.

Sirviéndome de una demostración guiada de la herramienta, en este apartado intentare dejar constancia de la potencialidad del sistema desarrollado, así como dar las pautas necesarias para su ejecución y/o modificación para aquellas personas que les pueda que puedan encontrase con ella y la consideren una línea de trabajo para sus pruebas.

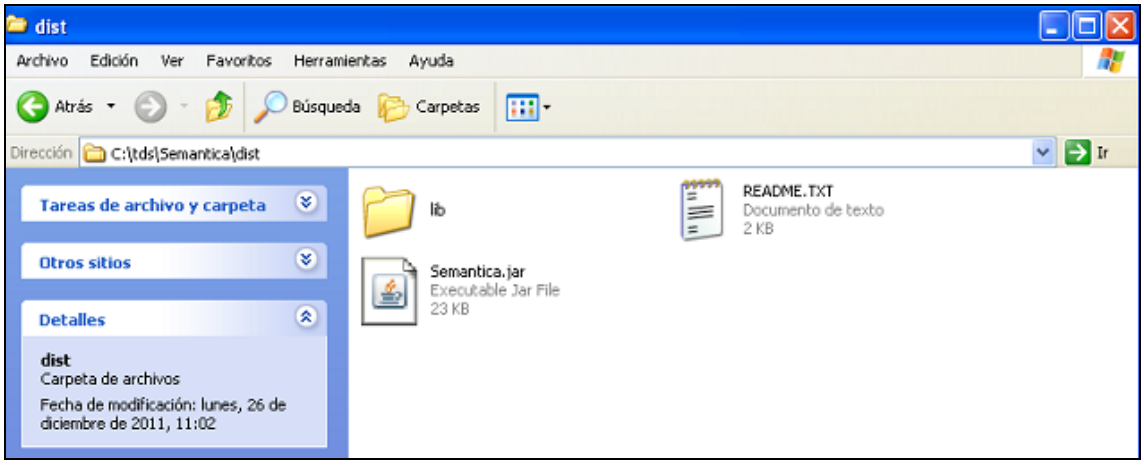
Contemplamos los siguientes apartados para ello:

- **Configuración Manual**.

Comenzaremos preparando los archivos externos necesarios. En nuestro caso hemos depositado en el directorio raíz del proyecto, el siguiente archivo “Zinedine_Zidane.rdf” , extraído de la URI http://dbpedia.org/data/Zinedine_Zidane.rdf.

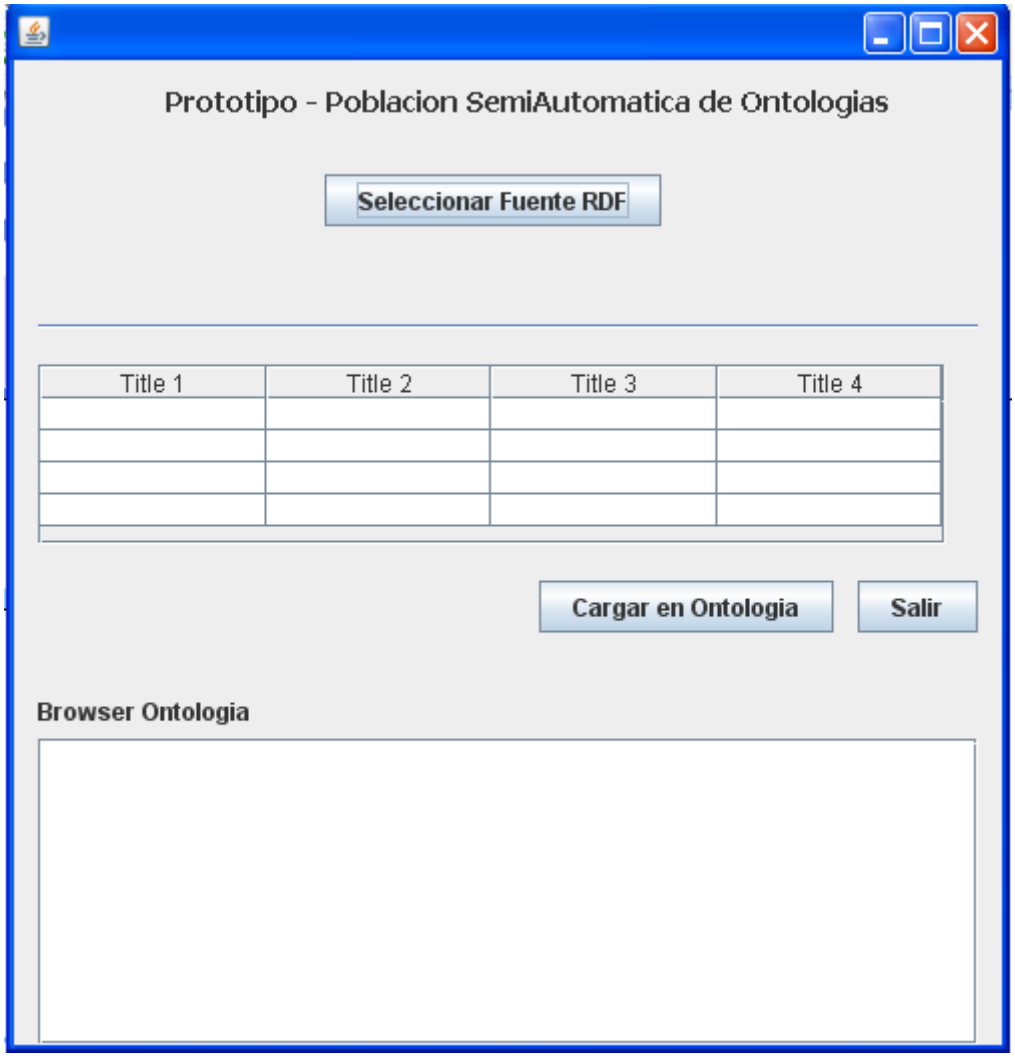
- **Arranque**.

Con todo nuestra herramienta esta lista para ser ejecutada. Para ello accedemos al directorio ‘dist’, donde encontraremos el archivo Semantica.jar, que será el ejecutable de la aplicación.



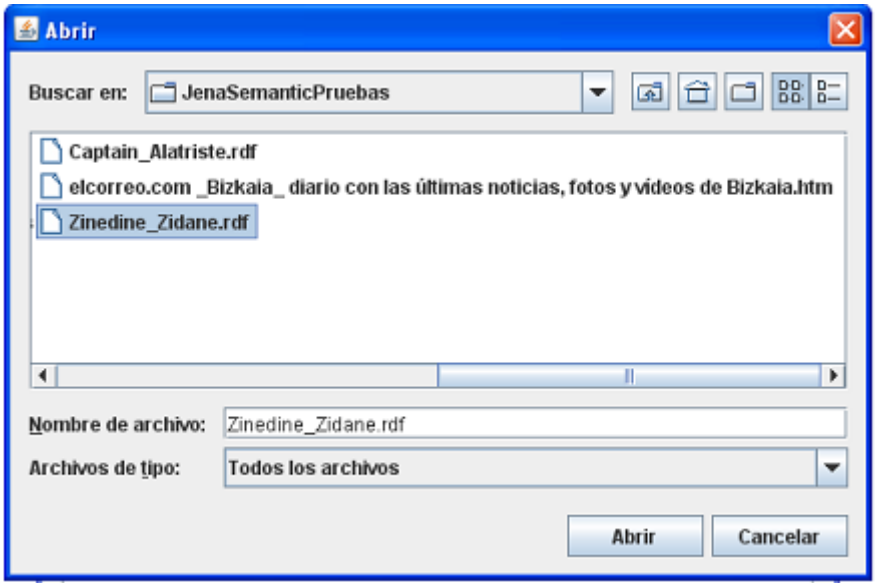
- **Ejecución**.

Al acceder a la aplicación nos encontramos con la siguiente pantalla.



Lo primero que haremos será seleccionar el archivo rdf que empleáremos como fuente, para ello pulsaremos el botón 'Seleccionar Fuente RDF'.

Aparecera un cuadro de dialogo, atraves de l cual navegaremos hasta el directorio que contiene el RFD a importar y lo seleccionaremos



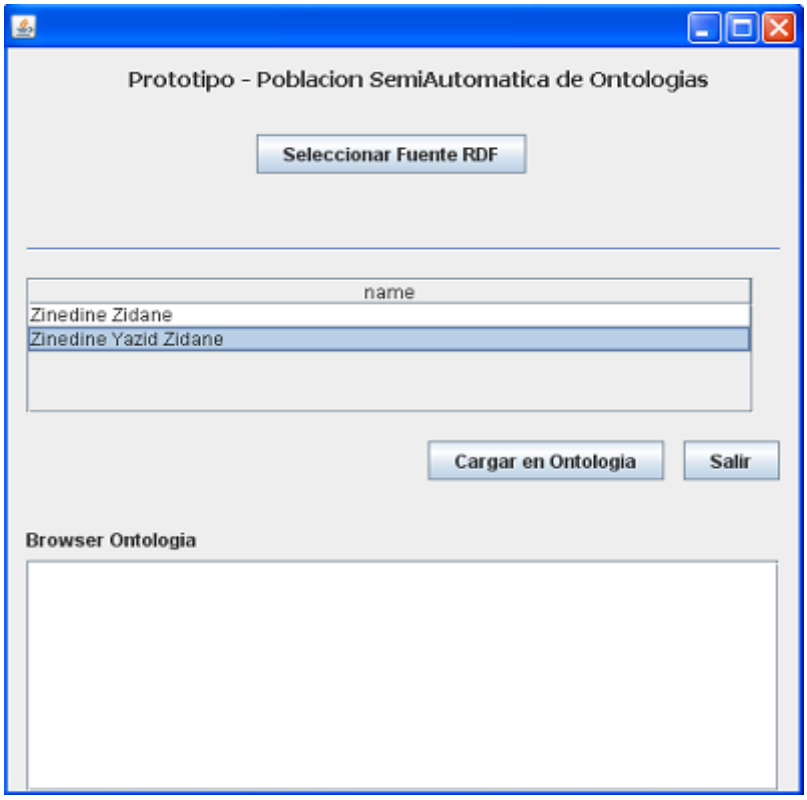
Al realizar esta operación, lo que estamos haciendo es recuperar el archivo rdf y crear un modelo con el, empleando los métodos definidos en la herramienta.

Una vez obtenido el modelo el programa procede a lanzar internamente una consulta sparql predefinida y pensada para obtener la propiedad name. Como he comentado esta consulta podría ser ampliada y dispuesta en la interfaz para dotarla de mas flexibilidad y permitir la importación de cualquier otro dato, pero me limito al ejemplo.

La consulta que se ha ejecutado es:

"SELECT ?name WHERE { <http://dbpedia.org/resource/Zinedine_Zidane> <http://xmlns.com/foaf/0.1/name> ?name.} ";

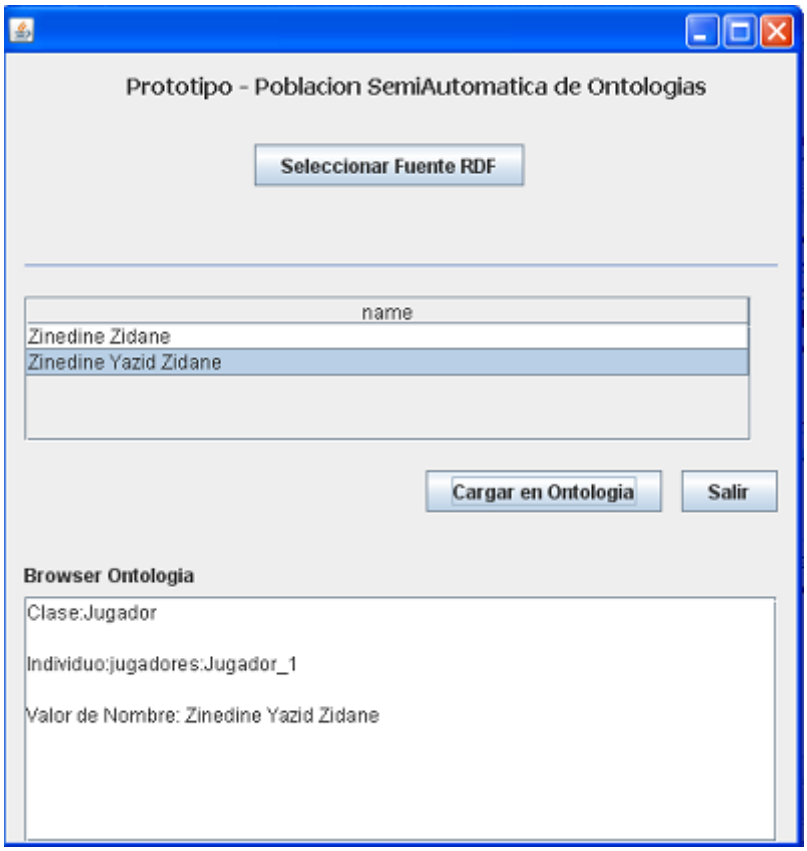
El resultado de esta operación de apertura, creación de modelo y ejecución de consulta, se puede observa en la tabla generada



En ella vemos como se devuelven los resultados coincidentes a la consulta.
El siguiente paso será cargar el valor que seleccionemos en una ontología. Para simplificar el proceso, he diseñado el sistema para que al pulsar el botón ‘ Cargar en Ontología’, se cree un nuevo modelo, al que programáticamente le asigno una clases y una propiedad para posteriormente realizar la anotación automática asignando el valor seleccionado de los resultados de la consulta lanzada sobre la URI de la DBPedia que anteriormente habíamos guardado como RDF.

A modo de comentario si se quisiera realizar esta operación sobre otra ontología , bastaría con sustituir en el código fuente, la sentencia de creación , por una de recuperación, haciendo por ejemplo un ‘model.read()’ sobre el archivo owl que deseemos.

Volviendo al ejemplo, al pulsar ‘ Cargar en Ontología’, como he comentado se disparan una serie de acciones que soportaran los procesos necesarios para cubrir la funcionalidad total implementada desde la creación de modelo hasta la navegación de la ontología creada. Todo la funcionalidad se resume en este botón.



Vemos como se ha creado un nuevo modelo para el que se ha instanciado un Individuo llamado ‘Jugador_1’, dentro de la clase ‘Jugador’, y al que le hemos asignado a su propiedad ‘name’, un valor obtenido de la consulta Sparql contra DBPedia.

De esta manera hemos logrado incluir en una nueva ontología de nuestra propiedad, una anotación de una URI externa de manera semiautomática, por lo que nuestro objetivo queda cumplido.

8. Bibliografia.

http://www.w3c.es/divulgacion/guiasbreves/websemantica.

http://semanticweb.org/wiki/Main_Page.

http://dbpedia.org/About.

http://protege.stanford.edu/overview/protege-owl.html

http://jena.sourceforge.net/

http://www.w3.org/TR/rdf-sparql-query/

http://playence.com/

http://www.isoco.com/

9. Anexo1.


```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns="http://www.owl-
ontologies.com/null//www.semanticweb.org/ontologies/2011/11/8/FutbolOn
tolgy.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://www.owl-
ontologies.com/null//www.semanticweb.org/ontologies/2011/11/8/FutbolOn
tolgy.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Campo_Entrenamiento">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Instalacion"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Estadio">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Instalacion"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Uniforme">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Equipacion"/>
    </rdfs:subClassOf>
    <owl:disjointWith>
      <owl:Class rdf:ID="Entrenamiento"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Persona">
    <owl:disjointWith>
      <owl:Class rdf:about="#Equipacion"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Instalacion"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="Equipo"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Equipo">
    <owl:disjointWith rdf:resource="#Persona"/>
    <owl:disjointWith>
      <owl:Class rdf:about="#Instalacion"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#Equipacion"/>
    </owl:disjointWith>
```

```
</owl:Class>
<owl:Class rdf:about="#Entrenamiento">
  <owl:disjointWith rdf:resource="#Uniforme"/>
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Equipacion"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Segunda_Division">
  <owl:disjointWith>
    <owl:Class rdf:ID="Primera_Division"/>
  </owl:disjointWith>
  <rdfs:subClassOf rdf:resource="#Equipo"/>
</owl:Class>
<owl:Class rdf:about="#Instalacion">
  <owl:disjointWith rdf:resource="#Persona"/>
  <owl:disjointWith>
    <owl:Class rdf:about="#Equipacion"/>
  </owl:disjointWith>
  <owl:disjointWith rdf:resource="#Equipo"/>
</owl:Class>
<owl:Class rdf:ID="Directivo">
  <rdfs:subClassOf rdf:resource="#Persona"/>
</owl:Class>
<owl:Class rdf:about="#Primera_Division">
  <rdfs:subClassOf rdf:resource="#Equipo"/>
  <owl:disjointWith rdf:resource="#Segunda_Division"/>
</owl:Class>
<owl:Class rdf:ID="Tecnico">
  <rdfs:subClassOf rdf:resource="#Persona"/>
</owl:Class>
<owl:Class rdf:ID="Pantalon">
  <rdfs:subClassOf rdf:resource="#Uniforme"/>
</owl:Class>
<owl:Class rdf:ID="Medias">
  <owl:disjointWith rdf:resource="#Pantalon"/>
  <rdfs:subClassOf rdf:resource="#Uniforme"/>
</owl:Class>
<owl:Class rdf:about="#Equipacion">
  <owl:disjointWith rdf:resource="#Persona"/>
  <owl:disjointWith rdf:resource="#Instalacion"/>
  <owl:disjointWith rdf:resource="#Equipo"/>
</owl:Class>
<owl:Class rdf:ID="Jugador">
  <rdfs:subClassOf rdf:resource="#Persona"/>
</owl:Class>
<owl:Class rdf:ID="Camiseta">
  <rdfs:subClassOf rdf:resource="#Uniforme"/>
  <owl:disjointWith rdf:resource="#Pantalon"/>
  <owl:disjointWith rdf:resource="#Medias"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="Preside">
  <rdfs:range rdf:resource="#Equipo"/>
  <rdfs:domain rdf:resource="#Directivo"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:ID="EsPresididoPor"/>
  </owl:inverseOf>
</owl:ObjectProperty>
```

```
  <owl:ObjectProperty rdf:ID="EntrenaA">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:ID="EsEntranadoPor"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#Tecnico"/>
    <rdfs:range rdf:resource="#Equipo"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="EntrenanEn">
    <rdfs:domain rdf:resource="#Equipo"/>
    <rdfs:range rdf:resource="#Campo_Entrenamiento"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#EsPresididoPor">
    <owl:inverseOf rdf:resource="#Preside"/>
    <rdfs:range rdf:resource="#Directivo"/>
    <rdfs:domain rdf:resource="#Equipo"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="Forman">
    <rdfs:domain rdf:resource="#Jugador"/>
    <rdfs:range rdf:resource="#Equipo"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="Viste">
    <rdfs:range rdf:resource="#Equipacion"/>
    <rdfs:domain rdf:resource="#Equipo"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="DisputaPartidosLocalesEn">
    <rdfs:domain rdf:resource="#Equipo"/>
    <rdfs:range rdf:resource="#Estadio"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:about="#EsEntranadoPor">
    <owl:inverseOf rdf:resource="#EntrenaA"/>
    <rdfs:range rdf:resource="#Tecnico"/>
    <rdfs:domain rdf:resource="#Equipo"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="DebutoEn">
    <rdfs:domain rdf:resource="#Jugador"/>
    <rdfs:range>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Equipo"/>
          <owl:Class rdf:about="#Estadio"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:range>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="EstaFormado">
    <rdfs:domain rdf:resource="#Equipo"/>
    <rdfs:range rdf:resource="#Jugador"/>
  </owl:ObjectProperty>
  <owl:DatatypeProperty rdf:ID="Apertura">
    <rdfs:domain rdf:resource="#Estadio"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Estatura">
    <rdfs:domain rdf:resource="#Jugador"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#float"/>
  </owl:DatatypeProperty>
```

```
  <owl:DatatypeProperty rdf:ID="Goles">
    <rdfs:domain rdf:resource="#Jugador"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Posicion_Ultima_Liga">
    <rdfs:domain rdf:resource="#Equipo"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Fabricante">
    <rdfs:domain rdf:resource="#Equipacion"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Nombre">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Jugador"/>
          <owl:Class rdf:about="#Estadio"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="TituloGanados">
    <rdfs:domain rdf:resource="#Equipo"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Nacionalidad">
    <rdfs:domain rdf:resource="#Jugador"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="NombreCompleto">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Equipo"/>
          <owl:Class rdf:about="#Persona"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="LugarNacimiento">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Jugador"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Capacidad">
    <rdfs:domain rdf:resource="#Estadio"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="FechaNacimiento">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
```

```
    <rdfs:domain rdf:resource="#Jugador"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="PartidosInternacionales">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Jugador"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Apodo">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Jugador"/>
          <owl:Class rdf:about="#Equipo"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Colores">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#Medias"/>
          <owl:Class rdf:about="#Pantalon"/>
          <owl:Class rdf:about="#Camiseta"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="AÃ±oDebut">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain rdf:resource="#Jugador"/>
  </owl:DatatypeProperty>
  <rdf:Statement rdf:ID="Fundacion">
    <rdf:subject rdf:resource="#Colores"/>
    <rdf:predicate
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </rdf:Statement>
  <Jugador rdf:ID="Jugador_4">
    <Apodo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    ></Apodo>
  </Jugador>
  <Equipo rdf:ID="Equipo_3"/>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.3, Build 418)
http://protege.stanford.edu -->
```